Getting started with USART automatic baud rater detection for STM32 MCUs

Introduction

Correct USART communication requires the transmission and reception baud rates to be matched reasonably closely, otherwise communication errors may occur.

Automatic baud rate detection is useful when establishing a communication link between two devices, where the slave device is able to detect the baud rate of the master controller and self-adjust accordingly. This requires an automatic mechanism to determine the baud rate.

The USART peripheral embedded in some STM32 devices offers many features, including automatic baud rate detection hardware.

The purpose of this application note is to present the automatic baud rate detection feature of STM32 microcontrollers and to give an alternative software approach for STM32 devices that do not implement this feature in hardware.

This application note applies to the products listed in Table 1.

Table 1. Applicable products

Type	Product series
Microcontrollers	STM32C0 series, STM32F0 series, STM32F1 series, STM32F2 series, STM32F3 series, STM32F4 series, STM32F7 series, STM32H5 series, STM32H7 series, STM32L0 series, STM32L1 series, STM32L4 series, STM32U5 series, STM32G0 series, STM32G4 series, STM32L5 series, STM32WB series, STM32U0 series, STM32WBA series, STM32WL series.

1 General information

This application note applies to STM32 microcontrollers that are Arm® based devices.

Note: Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

2 Hardware automatic baud rate detection

2.1 Feature overview

Automatic baud rate detection (ABR) allows the receiving device to accept data from a variety of transmitting devices operating at different speeds without needing to establish data rates in advance.

In some STM32 products, the USART is able to automatically determine the baud rate using dedicated hardware.

Table 2 gives an overview of the STM32 series devices that support automatic baud rate detection.

Table 2. USART hardware automatic baud rate detection on STM32 series devices

Product	ABR support
Mainstream	
STM32C0	Yes
STM32F0	Yes
STM32F1	No
STM32F3	Yes
STM32G0	Yes
High performance	
STM32F2	No
STM32F4	No
STM32F7	Yes
STM32H7	Yes
STM32H5	Yes
Ultra-low power	
STM32L0	Yes
STM32L1	No
STM32L4	Yes
STM32L5	Yes
STM32U0	Yes
STM32U5	Yes
Wireless	
STM32WB	Yes
STM32WBA	Yes
STM32WL	Yes

For STM32 series devices that embed ABR, not all of the instantiated USART interfaces support the automatic baud rate feature. This constraint is detailed in Table 3, Table 4, Table 5, and Table 6.

X: Automatic baud rate detection supported
-: Automatic baud rate detection is not supported
NA: Instance not available on this product

Table 3. Hardware automatic baud rate detection on STM32 USART interfaces (mainstream)

Port	STM32C0				STM32F0										STM32F3												STM32G0											
	STM32C011x4	STM32C011x6	STM32C031x4	STM32C031x6	STM32F030x4	STM32F030x6	STM32F030x8	STM32F070x6	STM32F070xB	STM32F030xC	STM32F03x	STM32F05x	STM32F04x	STM32F07x	STM32F09x	STM32F37xx	STM32F302xB/C	STM32F302xD/E	STM32F302x6/8	STM32F303xB/C	STM32F358xC	STM32F303xD/E	STM32F398xE	STM32F303x6/8	STM32F328x8	STM32F334xx	STM32F301x6/8	STM32F318x8	STM32G030xx	STM32G050xx	STM32G070xx	STM32G0B0xx	STM32G031xx	STM32G041xx	STM32G051xx	STM32G061xx	STM32G071xx	STM32G081xx	STM32G0B1xx	STM32G0C1xx
USART 1	X				X	X	X	X	X	X	X	X	X	X	X	NA	X	X	X					X	X	X	X		X	X	X	X			X		X	X	X
USART 2	NA				NA	NA	-	-	X	X	NA	-	-	X	X	NA	X	-	X					-	-	-	NA		X	X	X	NA					X	X
UART2	-				NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA					NA	NA	NA	NA		-	NA	NA	X					NA	NA
USART 3	NA				NA	NA	NA	NA	-	X	NA	NA	NA	-	X	NA	X	-	X					-	-	-	NA		NA	NA	X	NA					NA	X
UART3	NA				NA	NA	NA	NA		-	NA	NA	NA	NA	NA	NA	NA	NA	NA					NA	NA	NA	NA		NA	-	NA	NA					-	NA
USART 4	NA				NA	NA	NA	NA	-	-	NA	NA	NA	-	-	NA	-	NA	-					NA	NA	NA	NA		NA	NA	NA	NA					NA	NA
USART 5	NA				NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	-	NA	-	NA	-					NA	NA	NA	NA		NA	NA	NA	NA					NA	NA
UART4	NA				NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA					NA	NA	NA	NA		NA	-	-	NA					-	-
UART5	NA				NA	NA	NA	NA	NA	-	NA	NA	NA	NA	NA	NA	NA	NA	NA					NA	NA	NA	NA		NA	NA	-	NA					NA	-
USART 6	NA				NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	-	NA	NA	NA	NA					NA	NA	NA	NA		NA	NA	NA	NA					NA	NA
UART6	NA				NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA					NA	NA	NA	NA		NA	NA	-	NA					NA	-
USART 7	NA				NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	-	NA	NA	NA	NA					NA	NA	NA	NA		NA	NA	NA	NA					NA	NA
USART 8	NA				NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	-	NA	NA	NA	NA					NA	NA	NA	NA		NA	NA	NA	NA					NA	NA
UART 7	NA				NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA					NA	NA	NA	NA		NA	NA	NA	NA					NA	NA
UART 8	NA				NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA					NA	NA	NA	NA		NA	NA	NA	NA					NA	NA
USART9	NA				NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA					NA	NA	NA	NA		NA	NA	NA	NA					NA	NA
UART9	NA				NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA					NA	NA	NA	NA		NA	NA	NA	NA					NA	NA
USART 10	NA				NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA					NA	NA	NA	NA		NA	NA	NA	NA					NA	NA
USART 11	NA				NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA					NA	NA	NA	NA		NA	NA	NA	NA					NA	NA
UART11	NA				NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA					NA	NA	NA	NA		NA	NA	NA	NA					NA	NA
USART12	NA				NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA					NA	NA	NA	NA		NA	NA	NA	NA					NA	NA
UART12	NA				NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA					NA	NA	NA	NA		NA	NA	NA	NA					NA	NA

Table 4. **Hardware automatic baud rate detection on STM32 USART interfaces (high performance)**

Port	STM32F7		STM32H5			STM32H7			STM32U5			
	STM32F745xx	STM32F756xx	STM32H503	STM32H563/H573	STM32H562	STM32H74x/75x	STM32H7A3/7B3	STM32H72x/73x	STM32U59xxx	STM32U5Axxx	STM32U59Fxx	STM32U59Gxx
USART 1	X		X	X		X	X		X		X	
USART 2	X		X	X		X	X		X		X	
UART2	NA		NA	NA		NA	NA		NA		NA	
USART 3	X		X	X		X	X		X		X	
UART3	NA		NA	NA		NA	NA		NA		NA	
USART 4	X		NA	NA		NA	NA		NA		NA	
USART 5	X		NA	NA		NA	NA		NA		NA	
UART4	NA		NA	X		X	X		X		X	
UART5	NA		NA	X		X	X		X		X	
USART 6	X		NA	X		X	X		X		X	
UART6	NA		NA	NA		NA	NA		NA		NA	
USART 7	X		NA	NA		NA	NA		NA		NA	
USART 8	X		NA	NA		NA	NA		NA		NA	
UART 7	NA		NA	X		X	X		NA		NA	
UART 8	NA		NA	X		X	X		NA		NA	
USART9	NA		NA	NA		NA	NA		NA		NA	
UART9	NA		NA	NA		NA	X		NA		NA	
USART 10	NA		NA	NA		NA	X		NA		NA	
USART 11	NA		NA	X		NA	NA		NA		NA	
UART11	NA		NA	NA		NA	NA		NA		NA	
USART12	NA		NA	NA		NA	NA		NA		NA	
UART12	NA		NA	X		NA	NA		NA		NA	

Table 5. Hardware automatic baud rate detection on STM32 USART interfaces (ultra-low power)

Port	STM32L0			STM32L4					STM32U5						STM32G0										STM32L5	STM32U0			STM32H5		
	STM32L0x1	STM32L0x2	STM32L0x3	STM32L4x1	STM32L4x2	STM32L4x3	STM32L4x5	STM32L4x6	STM32U575xx	STM32U585xx	STM32U59xxx	STM32U5Axxx	STM32U59Fxx	STM32U59Gxx	STM32G070xx	STM32G0B0xx	STM32G031xx	STM32G041xx	STM32G051xx	STM32G061xx	STM32G071xx	STM32G081xx	STM32G0B1xx	STM32G0C1xx	STM32L5x2	STM32U031xx	STM32U073	STM32U083xx	STM32H563	STM32H573	STM32H562
USART 1	X			X					X		X		X		X	X	X				X		X		X	X	X		X		
USART 2	X			X					X		X		X		X	X	NA				X		X		X	X	X		X		
UART2	NA			NA					NA		NA		NA		NA	NA	X				NA		NA		NA	NA	NA		NA		
USART 3	NA			X					X		X		X		NA	X	NA				NA		X		X	NA	NA		X		
UART3	NA			NA					NA		NA		NA		-	NA	NA				-		NA		NA	X	X		NA		
USART 4	-			NA					NA		NA		NA		NA	NA	NA				NA		NA		NA	NA	NA		NA		
USART 5	-			NA					NA		NA		NA		NA	NA	NA				NA		NA		NA	NA	NA		NA		
UART4	NA			X					X		X		X		-	-	NA				-		-		X	-	-		X		
UART5	NA			NA					X		X		X		NA	-	NA				NA		-		X	-	-		X		
USART 6	NA			NA					NA		X		X		NA	NA	NA				NA		NA		NA	NA	NA		X		
UART6	NA			NA					NA		NA		NA		NA	-	NA				NA		-		NA	NA	NA		NA		
USART 7	NA			NA					NA		NA		NA		NA	NA	NA				NA		NA		NA	NA	NA		NA		
USART 8	NA			NA					NA		NA		NA		NA	NA	NA				NA		NA		NA	NA	NA		NA		
UART 7	NA			NA					NA		NA		NA		NA	NA	NA				NA		NA		NA	NA	NA		X		
UART 8	NA			NA					NA		NA		NA		NA	NA	NA				NA		NA		NA	NA	NA		X		
USART9	NA			NA					NA		NA		NA		NA	NA	NA				NA		NA		NA	NA	NA		NA		
UART9	NA			NA					NA		NA		NA		NA	NA	NA				NA		NA		NA	NA	NA		NA		
USART 10	NA			NA					NA		NA		NA		NA	NA	NA				NA		NA		NA	NA	NA		NA		
USART 11	NA			NA					NA		NA		NA		NA	NA	NA				NA		NA		NA	NA	NA		X		
UART11	NA			NA					NA		NA		NA		NA	NA	NA				NA		NA		NA	NA	NA		NA		
USART12	NA			NA					NA		NA		NA		NA	NA	NA				NA		NA		NA	NA	NA		NA		
UART12	NA			NA					NA		NA		NA		NA	NA	NA				NA		NA		NA	NA	NA		X		

Table 6. Hardware automatic baud rate detection on STM32 USART interfaces (wireless)

Port	STM32WB						STM32WBA			STM32WL
	STM32WB10CC	STM32WB15CC	STM32WB30CE	STM32WB50CG	STM32WB55xx	STM32WB35xx	STM32WBA52xx	STM32WBA54XX	STM32WBA55XX	STM32WLEx
USART 1	X						X			X
USART 2	NA						X			X
UART2	NA						NA			NA
USART 3	NA						NA			NA
UART3	NA						NA			NA
USART 4	NA						NA			NA
USART 5	NA						NA			NA
UART4	NA						NA			NA
UART5	NA						NA			NA
USART 6	NA						NA			NA
UART6	NA						NA			NA
USART 7	NA						NA			NA
USART 8	NA						NA			NA
UART 7	NA						NA			NA
UART 8	NA						NA			NA
USART9	NA						NA			NA
UART9	NA						NA			NA
USART 10	NA						NA			NA
USART 11	NA						NA			NA
UART11	NA						NA			NA
USART12	NA						NA			NA
UART12	NA						NA			NA

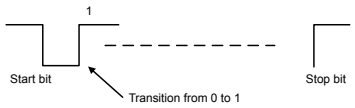
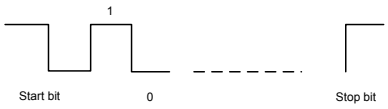
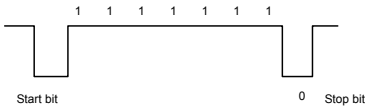
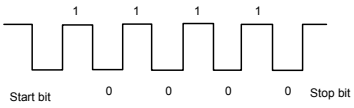
2.2 Automatic baud rate detection modes

ABR refers to the process by which a receiving device determines the speed of incoming data by examining the first character, usually a preselected sign-on character.

The automatic baud rate feature on STM32 products embeds various modes based on different character patterns:

- Any character starting with a bit at '1': Mode 0
- Any character starting with a 10xx pattern: Mode 1
- 0x7F: Mode 2
- 0x55: Mode 3

Table 7. Automatic baud rate detection modes

ABR mode	Description	Waveform
0	The received character can be a character starting with a bit at 1. In this case, the USART measures the duration of the start bit (falling edge to rising edge).	
1	Any character starting with a 10xx pattern. In this case, the USART measures the duration of the start and of the first data bit. The duration is measured from falling edge to falling edge, ensuring better accuracy in the case of slow signal slopes.	
2	0x7F character frame. In this case, the baud rate is updated first at the end of the start bit then at the end of the bit 6.	
3	A 0x55 character frame. In this case, the baud rate is updated first at the end of the start bit, then at the end of bit and finally at the end of bit 6. In parallel, another check is performed for each intermediate transition of the RX line.	

Before activating the automatic baud rate detection, one of the ABR modes must be selected through the `ABRMOD[1:0]` field in the `USARTx_CR2` register. In all ABR modes, the baud rate is measured several times during the synchronization data reception, and is compared each time to the previous measurement.

Note: In 7-bit data length mode, 0x7F and 0x55 frames detection ABR modes are not supported.

2.3 Error calculation for ABR

The USART clock source (fCK) determines the communication speed range (specifically the maximum communication speed). The receiver implements different user-configurable oversampling techniques for data recovery by discriminating between valid incoming data and noise. This allows a trade-off between the maximum communication speed and immunity to noise/clock inaccuracy.

The oversampling method is selected by programming the OVER8 bit in the `USARTx_CR1` register, and can be either 16 or 8 times the baud-rate clock.

The USART clock source frequency must be compatible with the expected communication speed:

- When oversampling by 16, the baud rate is between fCK/65535 and fCK/16
- When oversampling by 8, the baud rate is between fCK/65535 and fCK/8.

The baud-rate error is dependent on the USART clock source, the oversampling method, and the ABR mode.

$$\text{Error}\left(\%\right) = \left|\frac{\text{desired baud rate} - \text{actual baud rate}}{\text{desired baud rate}}\right| \cdot 100 \quad (1)$$

Where:

- The desired baud rate is fixed by the transmitter device.
- The actual baud rate is the baud rate determined by the USART receiver using the automatic baud rate detection operation.

3 Software automatic baud rate detection

When hardware auto baud rate detection is not supported, the software approach described in this section can be adopted.

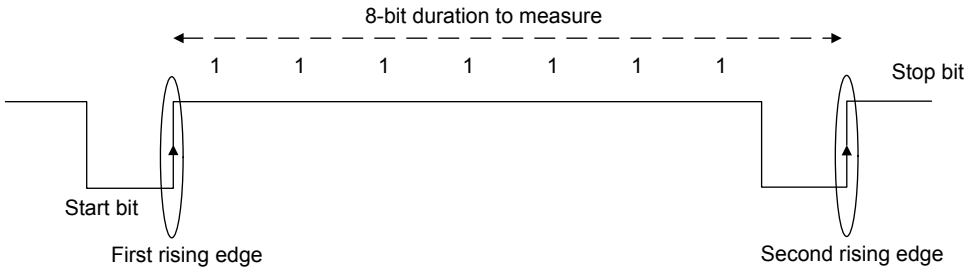
The idea of the software approach is to send a 0x7F data frame to the USARTx_RX pin. This is connected to the EXTI line, which is configured to generate an interrupt on each rising edge.

The duration of the interval between the two rising edges is measured using the SysTick timer. This duration corresponds to the duration of 8 bits, so

- Bit time = calculated duration/8
- Baud rate = 1/bit time

The `USARTx_BRR` register is then programmed, based on the calculated baud rate value.

Figure 1. Software automatic baud rate detection overview



4 Setups for software and hardware approaches

The STM32F303xD/E embedding the hardware automatic baud rate feature is used for this setup example.

The HyperTerminal PC application is used to send and receive data frames to and from the STM32F303. Consequently, standard baud rates in the range 600 bits/s to 115200 bits/s are tested. The highest baud-rate value that can be reached (9 Mbits/s) is tested using another STM32F3 device as a transmitter.

4.1 USART1 configuration example

In both examples, the STM32 USART1 is configured as follows:

```
/*##-1- Configure the UART peripheral
######################################*/
/* Put the USART peripheral in the Asynchronous mode (UART Mode) */
/* UART configured as follows:
- Word Length = 8 Bits
- Stop Bit = One Stop bit
- Parity = NONE parity
- BaudRate = 115200 baud It can be any other value as the USARTx_BRR
register will be reprogrammed
- Hardware flow control disabled (RTS and CTS signals)
- The oversampling mode is 8 or 16 (Both are tested)
*/
UartHandle.Instance = USARTx;
UartHandle.Init.BaudRate = 115200;
UartHandle.Init.WordLength = UART_WORDLENGTH_8B;
UartHandle.Init.StopBits = UART_STOPBITS_1;
UartHandle.Init.Parity = UART_PARITY_NONE;
UartHandle.Init.HwFlowCtl = UART_HWCONTROL_NONE;
UartHandle.Init.Mode = UART_MODE_TX_RX;
UartHandle.Init.OverSampling = UART_OVERSAMPLING_16;
```

Note: The USART1 clock source is a system clock at 72 MHz using the HSE PLL clock source. Some tests are made using the HSI clock as the USART1 clock source. This is to check the impact of the HSI inaccuracy on the results.

4.2 Hardware auto baud rate detection

The USART1 is configured to detect baud rate automatically. The user has to select the ABR mode in the USART1 initialization function as follows:

```
/*##-2- Configure the AutoBaudRate method */
UartHandle.AdvancedInit.AdvFeatureInit =UART_ADVFEATURE_AUTOBAUDRATE_INIT;
UartHandle.AdvancedInit.AutoBaudRateEnable =
UART_ADVFEATURE_AUTOBAUDRATE_ENABLE;
/*Uncomment your appropriate mode */
//UartHandle.AdvancedInit.AutoBaudRateMode =
UART_ADVFEATURE_AUTOBAUDRATE_ONSTARTBIT;
//UartHandle.AdvancedInit.AutoBaudRateMode =
UART_ADVFEATURE_AUTOBAUDRATE_ONFALLINGEDGE;
//UartHandle.AdvancedInit.AutoBaudRateMode =
UART_ADVFEATURE_AUTOBAUDRATE_ON0X7FFRAME;
//UartHandle.AdvancedInit.AutoBaudRateMode =
UART_ADVFEATURE_AUTOBAUDRATE_ON0X55FRAME;
if (HAL_UART_Init(&UartHandle) != HAL_OK)
{
/* Initialization Error */
Error_Handler();
}
/* Wait until Receive enable acknowledge flag is set */
while(__HAL_UART_GET_FLAG(&UartHandle,UART_FLAG_REACK) == RESET)
{}
/* Wait until Transmit enable acknowledge flag is set */
while(__HAL_UART_GET_FLAG(&UartHandle,UART_FLAG_TEACK) == RESET)
{}
/* Loop until the end of Autobaudrate phase */
```

```
while(__HAL_UART_GET_FLAG(&UartHandle,UART_FLAG_ABRF) == RESET)
{}
```

Once the whole initialization is complete, the USART waits until data is received from the HyperTerminal before launching the automatic baud rate detection phase. The end of this phase is monitored by the ABRF flag.

- If the auto baud rate operation is unsuccessful, the ABRE flag is set
- If the auto baud rate operation is completed successfully, an acknowledgment data is transmitted to the HyperTerminal.

```
/* If AutoBaudBate error occurred */
if (__HAL_UART_GET_FLAG(&UartHandle, UART_FLAG_ABRE)!= RESET)
{
Error_Handler();
}
else
{
/* Wait until RXNE flag is set */
while(__HAL_UART_GET_FLAG(&UartHandle,UART_FLAG_RXNE) == RESET)
{}
/* Send acknowledgement message*/
if (HAL_UART_Transmit_DMA(&UartHandle, (uint8_t *)aTxBuffer, TXBUFFERSIZE)
!= HAL_OK)
{
/* Transfer error in transmission process */
Error_Handler();
}
while (HAL_UART_GetState(&UartHandle) != HAL_UART_STATE_READY)
{
}
}
```

4.3 Software automatic baud rate detection

Table 8 details the software approach.

Table 8. Software automatic baud rate detection details

Action	Code
HAL library initialization. Suspend Tick increment to prevent wake-up by SysTick interrupt.	`HAL_Init();` `HAL_SuspendTick();`

Configure the system clock to 72 MHz.
The `SystemCoreClockUpdate` function can eventually be executed in the main to verify the CPU operating frequency.
	`System Clock source = PLL (HSE)`
	`PLLMUL = RCC_PLL_MUL9 (9)`
	`Flash Latency(WS) = 2`

Configure the UART peripheral.	See Section 4.1: USART1 configuration example.

Configure the USARTx RX pin to generate an interrupt on each rising edge.	`static void EXTILine1_Config(void)`
	`{`
	`GPIO_InitTypeDef GPIO_InitStructure;`
	`/* Enable GPIOE clock */`
	`__GPIOE_CLK_ENABLE();`
	`/* Configure PE1 pin as input floating */`
	`GPIO_InitStructure.Mode = GPIO_MODE_IT_RISIN G;`
	`GPIO_InitStructure.Pull = GPIO_NOPULL;`
	`GPIO_InitStructure.Pin = GPIO_PIN_1;`
	`HAL_GPIO_Init(GPIOE, &GPIO_InitStructure);`
	`/* Enable and set EXTI Line0 Interrupt to th e lowest`
	`priority */`
	`HAL_NVIC_SetPriority(EXTI1_IRQn, 2, 2);`
	`HAL_NVIC_EnableIRQ(EXTI1_IRQn);`
	`}`

Action	Code

0x7F received on RX pin, wait until the end of the interrupt.
Launch the automatic baud rate sequence described in Section 3.
	`/*Wait until the end of interrupt */`
	`while (end_interrupt_flag != 1) {`
	`BSP_LED_On(LED2);`
	`}`
	`/* Autobaudrate sequence : Update BRR regist`
	`er */`
	`Autobaudrate();`
	`/* Send acknowledgement */`
	`if (HAL_UART_Transmit_DMA(&UartHandle, (uint`
	`8_t`
	`*)aTxBuffer, TXBUFFERSIZE) != HAL_OK)`
	`{`
	`/* Transfer error in transmission process */`
	`Error_Handler();`
	`}`
	`while (HAL_UART_GetState(&UartHandle) !=`
	`HAL_UART_STATE_READY)`
	`{}`
	`/* Infinite loop */`
	`while (1)`
	`{}`

Automatic baud rate function	`static void Autobaudrate(void)`
	`{`
	`float tmp=0, elapsed;`
	`uint32_t USART1_clk=0;`
	`uint32_t start_time_val=0;`
	`uint32_t BRR=0;`
	`tmp += 0xFFFFFF - stop_time_val;`
	`tmp -= start_time_val;`
	`elapsed =(tmp/(SystemCoreClock/1000000))/8;`
	`USART1_clk=SystemCoreClock;`
	`if( (USART1->CR1 & 0x8000)== 0x8000)`
	`{`
	`/*In case of oversampling by 8*/`
	`BRR =(elapsed*((2*USART1_clk)/1000000))+1;`
	`USART1->BRR= BRR;`
	`}`
	`else`
	`{`
	`/*In case of oversampling by 16*/`
	`BRR =(elapsed* ((USART1_clk)/1000000))+1;`
	`USART1->BRR=BRR;`
	`}`
	`}`

External line 1 interrupt
request:
First rising: temp=0 start the SysTick timer
Second rising:
• Disable the SysTick counter
• Get the encoding time
• Clear the SysTick counter
	`void EXTI1_IRQHandler()`
	`{`
	`HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_1);`
	`if(temp==0)`
	`{`
	`HAL_SYSTICK_Config(0xFFFFFF);`
	`temp++;`
	`}`
	`else`
	`{`
	`SysTick->CTRL &= SysTick_Counter_Disable;`
	`/* Stop the Timer and get the encoding time`
	`*/`
	`GETMYTIME(&stop_time_val);`
	`/* Clear the SysTick Counter */`
	`SysTick->VAL = SysTick_Counter_Clear;`
	`/* Clear the temp flag*/`
	`temp=0;`
	`/*end of interrupt*/`
	`interrupt_flag=1;`
	`}`

Action	Code

Required project defines	`#define SysTick_Counter_Disable ((uint32_t)0xFFFFFFFE)`
	`#define SysTick_Counter_Enable ((uint32_t)0x00000001)`
	`#define SysTick_Counter_Clear ((uint32_t)0x00000000)`
	`#define GETMYTIME(_t) (*_t=SysTick->VAL)`

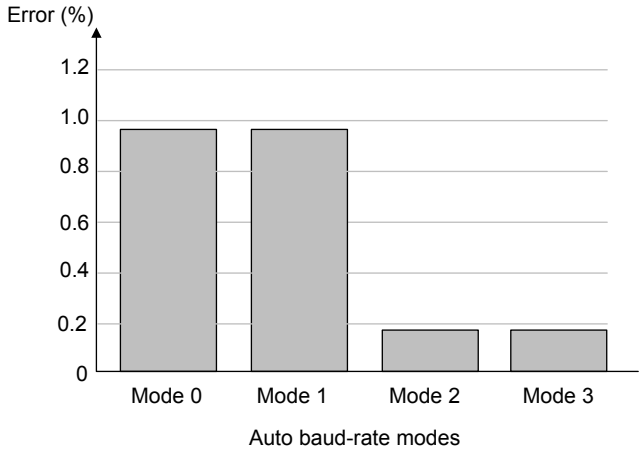
4.4 Analysis of results

4.4.1 Error calculation

Figure 2 shows that ABR modes 2 and 3 are more precise than modes 0 and 1. They provide a lower baud-rate error figure.

Nevertheless, all modes exhibit good results, as the error between the desired and actual baud rates is less than 1%.

Figure 2. Error calculation for ABR at fCK = 72 MHz, 115200 bits/s desired baud rate



4.4.2 Comparison of software and hardware approaches

Figure 3 shows that in general, when the USART is clocked by the system clock at 72 MHz (with the HSE as the PLL clock source), the results are better than when the USART clock source is used as the HSI clock. This is due to the relative inaccuracy of the HSI.

Figure 3. **ABR error comparison (fCK = HSI clock, mode 2 for hardware detection)**

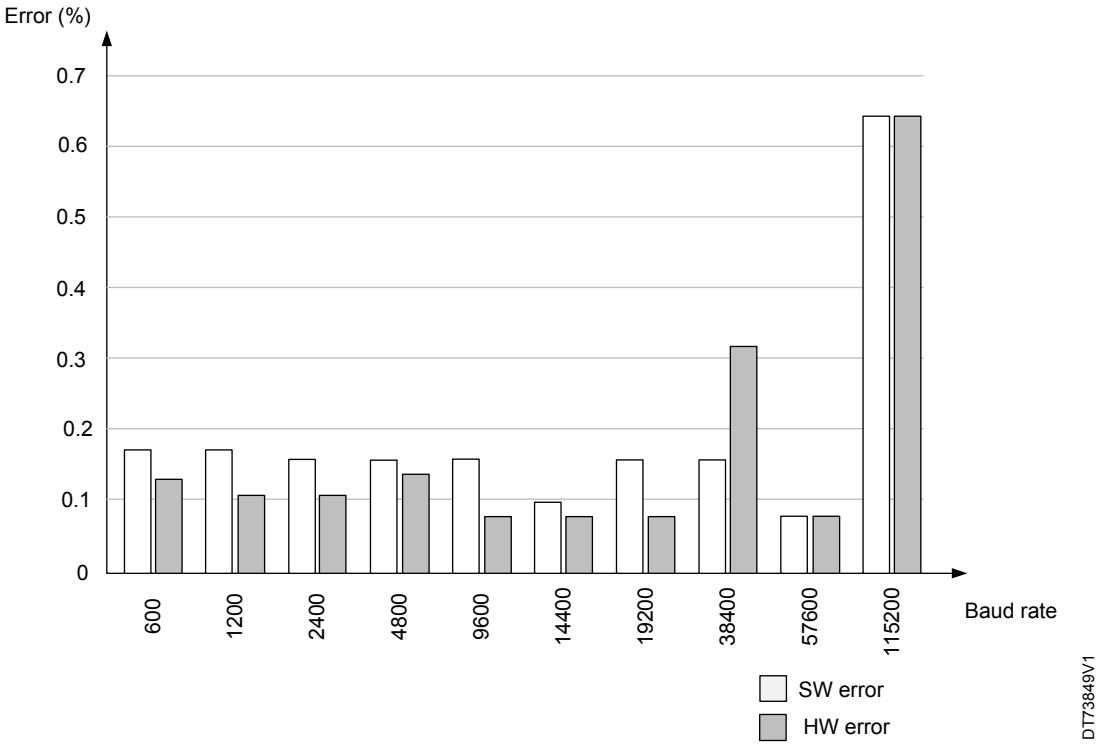


Figure 4 shows that in most cases the hardware approach provides better results than the software approach. The software approach nevertheless provides good results that are in some cases comparable to those obtained using the hardware approach.

Figure 4. **ABR error comparison (fCK = 72 MHz, mode 2 for HW detection)**

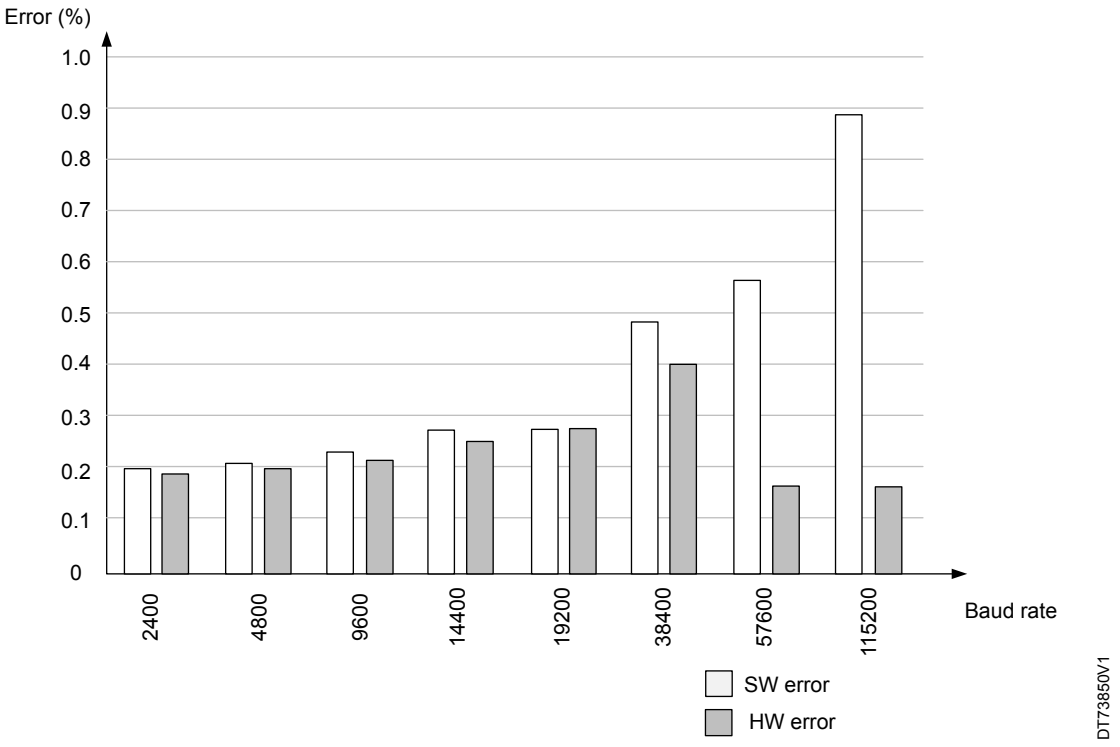
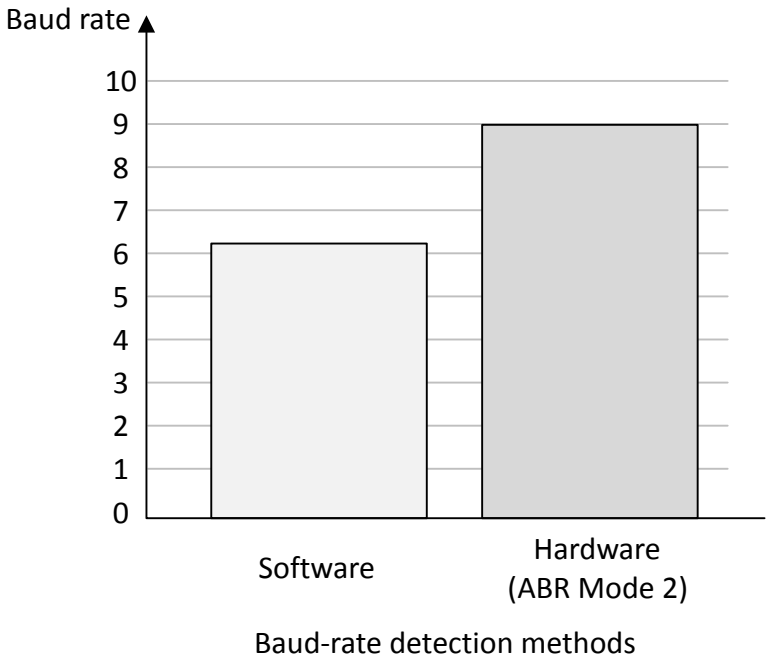


Figure 5 shows that:

- With the hardware approach, the maximum baud rate reach 9 Mbits/s with 0% error.
- With the software approach, the error at maximum baud rate is about 30%, which is explained by the CPU cycles spent executing the interrupt handler.

Figure 5. Baud rate comparison (fCK = 72 MHz, desired baud rate = 9 Mbits/s, mode 2 for hardware detection)



5 Conclusion

This application note describes the hardware automatic baud rate detection feature embedded in some STM32 devices. It also provides a technique for implementing this feature in software, as a solution for STM32 devices not implementing this feature in hardware.

Although the automatic baud rate detection is applied at the beginning of the examples, it could be extended and used whenever the transmitter and receiver devices detect communication errors. This allows a robust application where the host varies its baud rate between communications.

Revision history

Table 9. **Document revision history**

Date	Version	Changes
15-Nov-2016	1	Initial release.
01-Oct-2020	2	The following tables now scope STM32H7 series products: • Table 1: Applicable products • Table 2: USART hardware automatic baud rate detection on STM32 Series devices • Table 3: Hardware automatic baud rate detection on STM32 USART interfaces.
12-Jan-2023	3	Added STM32C0 and STM32U5 series: • Updated Table 1: Applicable products, Table 2: USART hardware automatic baud rate detection on STM32 Series devices and Table 3: Hardware automatic baud rate detection on STM32 USART interfaces • Modified STM32H7 series products in Table 3: Hardware automatic baud rate detection on STM32 USART interfaces
08-Mar-2024	4	The following tables now scope STM32G0, STM32H5, STM32L4, STM32L5, STM32U0, STM32WB STM32WBA, STM32WBL series products: • Table 1. Applicable products • Table 2. USART hardware automatic baud rate detection on STM32 series devices • Table 3. Hardware automatic baud rate detection on STM32 USART interfaces (mainstream) • Table 4. Hardware automatic baud rate detection on STM32 USART interfaces (high performance) • Table 5. Hardware automatic baud rate detection on STM32 USART interfaces (ultra-low power) • Table 6. Hardware automatic baud rate detection on STM32 USART interfaces (wireless)

Contents

1 General information 2
2 Hardware automatic baud rate detection 3
2.1 Feature overview 3
2.2 Automatic baud rate detection modes 8
2.3 Error calculation for ABR 8
3 Software automatic baud rate detection 10
4 Setups for software and hardware approaches 11
4.1 USART1 configuration example 11
4.2 Hardware auto baud rate detection 11
4.3 Software automatic baud rate detection 12
4.4 Analysis of results 14
4.4.1 Error calculation 14
4.4.2 Comparison of software and hardware approaches 14
5 Conclusion 17
Revision history 18
List of tables 20
List of figures 21

List of tables

Table 1. Applicable products . . . 1
Table 2. USART hardware automatic baud rate detection on STM32 series devices . . . 3
Table 3. Hardware automatic baud rate detection on STM32 USART interfaces (mainstream) . . . 4
Table 4. Hardware automatic baud rate detection on STM32 USART interfaces (high performance) . . . 5
Table 5. Hardware automatic baud rate detection on STM32 USART interfaces (ultra-low power) . . . 6
Table 6. Hardware automatic baud rate detection on STM32 USART interfaces (wireless) . . . 7
Table 7. Automatic baud rate detection modes . . . 8
Table 8. Software automatic baud rate detection details . . . 12
Table 9. Document revision history . . . 18

List of figures

Figure 1. Software automatic baud rate detection overview 10
Figure 2. Error calculation for ABR at fCK = 72 MHz, 115200 bits/s desired baud rate 14
Figure 3. ABR error comparison (fCK = HSI clock, mode 2 for hardware detection) 15
Figure 4. ABR error comparison (fCK = 72 MHz, mode 2 for HW detection) 15
Figure 5. Baud rate comparison (fCK = 72 MHz, desired baud rate = 9 Mbits/s, mode 2 for hardware detection) 16

IMPORTANT NOTICE – READ CAREFULLY

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgment.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of purchasers’ products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. For additional information about ST trademarks, refer to www.st.com/trademarks. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2024 STMicroelectronics – All rights reserved