STM32 UART (USART) 13

گیرنده/فرستنده غیرهمزمان جهانی یا UART به اختصار نمایانگر مدار سخت‌افزاری است که برای ارتباط سریال استفاده می‌شود. UART به‌صورت یک مدار مجتمع مستقل (IC) یا به‌عنوان یک ماژول داخلی در میکروکنترلرها عرضه می‌شود و به صورت گیرنده/فرستنده غیرهمزمان جهانی(UART) و گیرنده/فرستنده همزمان/غیرهمزمان جهانی(USART) بکار می‌رود.

در نوع همزمان، فرستنده کلاک داده را تولید کرده و به گیرنده ارسال می‌کند که به‌طور هماهنگ عمل می‌کند. از طرف دیگر، نوع غیرهمزمان، فرستنده کلاک داده را به‌صورت داخلی تولید می‌کند و هیچ سیگنال کلاک سریالی بین فرستنده و گیرنده مبادله نمی‌شود بنابراین برای دستیابی به ارتباط صحیح بین دو طرف، هر دو باید از نرخ Baud یکسان استفاده کنند.

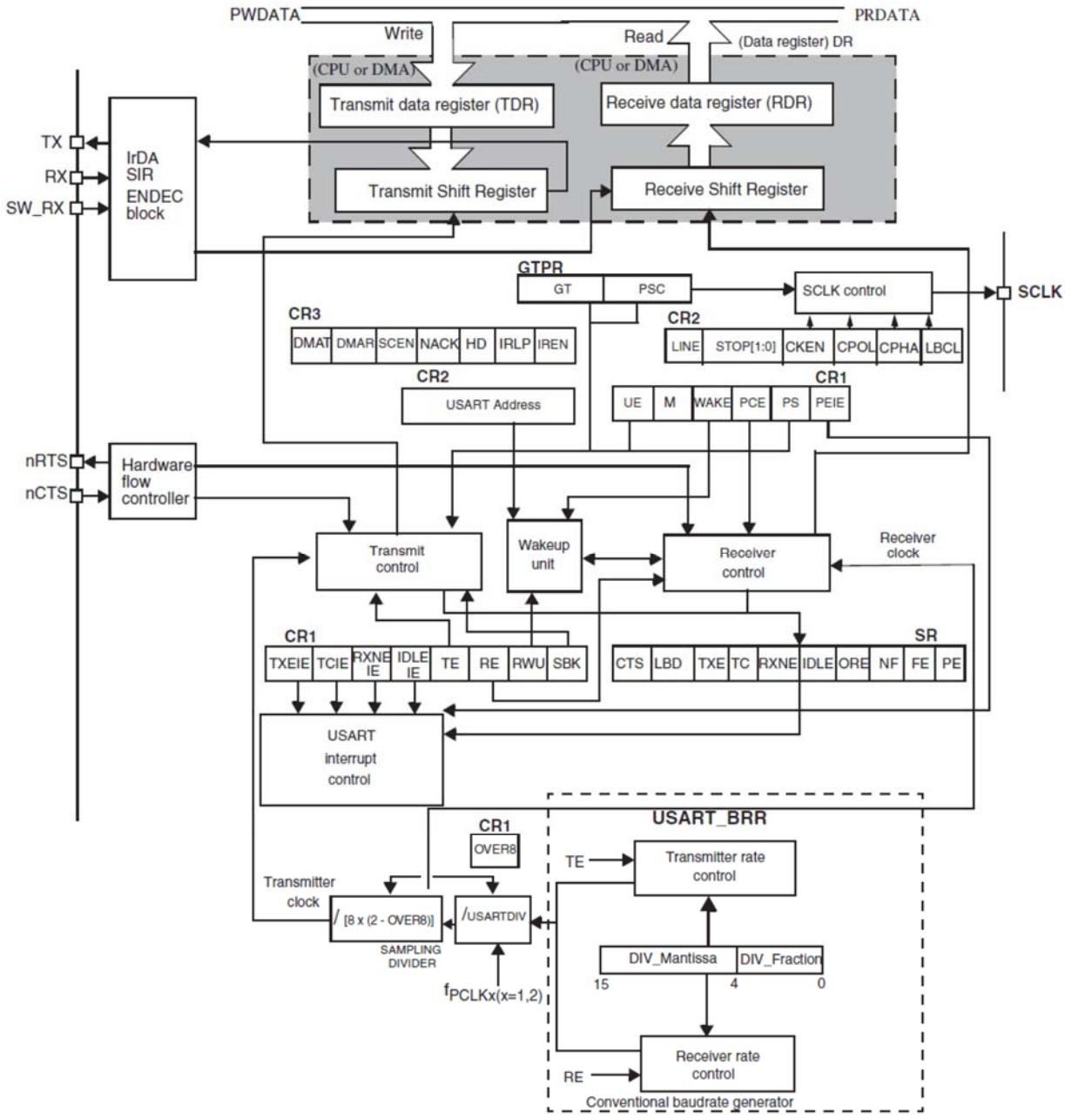
13.1 قابلیت‌های سخت‌افزاری UART STM32

در این بخش، نگاهی عمیق به سخت‌افزار ماژول USART STM32، دیاگرام بلوکی، قابلیت‌ها، تولیدکننده نرخ baud (BRG)، حالت‌های عملیاتی و دریافت/انتقال داده خواهیم داشت.

هر ارتباط دوطرفه USART به حداقل دو پین نیاز دارد: ورودی داده دریافتی (RX) و خروجی داده ارسالی (TX). از طریق این پین‌ها، داده‌های سری در حالت عادی USART منتقل و دریافت می‌شوند. پین CK برای ارتباط در حالت همزمان ضروری است. پین‌های CTS و RTS در حالت کنترل جریان سخت‌افزاری نیاز هستند.

13.1.1 دیاگرام بلوکی UART STM32

دیاگرام UART درشکل 13-1 نشان داده شده است که دارای شیفت رجیستر و بافر برای عملیات انتقال و دریافت داده دوطرفه کامل وجود دارد. کلاک تولید کننده نرخ baud (BRG) به هر دو شیفت رجیستر که داده‌ها را در حین دریافت/انتقال جابه‌جا می‌کنند، اعمال می‌شود.



شکل 13-1: نمایی از بلوک دیاگرام داخلی UART

همچنین یک رجیستر آدرس برای حالت ارتباط چندپردازنده‌ای، واحد کنترل جریان داده سخت‌افزاری برای پشتیبانی از این ویژگی موجود، یک مدار کدگشای IrDA و واحد کنترل وقفه برای تولید سیگنال‌های وقفه مختلف در رویدادهای سخت‌افزاری مختلف USART وجود دارد.

13.2 رجیسترهای UART

در میکروکنترلر STM32F407، بخش USART (Universal Synchronous/Asynchronous Receiver/Transmitter) برای برقراری ارتباط سریال استفاده می‌شود. این بخش شامل چندین رجیستر است که هرکدام وظایف خاصی را بر عهده دارند. در ادامه به توضیح اصلی‌ترین رجیسترهای USART می‌پردازیم:

13.2.1 USART_CR1 (Control Register 1)

Control register 1 (USART_CR1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OVER8	Reserved	UE	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	RWU	SBK
rw	Res.	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

بیت	نام	شرح
31:16	reserved	رزرو شده‌اند و باید در مقدار بازنشانی نگه‌داری شوند
15	OVER8	0: Oversampling با 16 1: Oversampling با 8
14	reserved	
13	UE	فعال‌سازی USART 0: غیر فعال 1: فعال
12	M	طول کلمه 0: 8 bit data 1: 9 bit data
11	WAKE	روش بیداری 0: خط Idle. 1: نشانه آدرس.

فعال‌سازی کنترل پاریتی 0: کنترل پاریتی غیرفعال. 1: کنترل پاریتی فعال.	PCE	10	
انتخاب پاریتی 0: پاریتی زوج. 1: پاریتی فرد.	PS	9	
فعال‌سازی وقفه PE 0: وقفه غیرفعال است. 1: یک وقفه USART زمانی تولید می‌شود که PE=1 در رجیستر USART_SR	PEIE	8	
فعال‌سازی وقفه TXE 0: وقفه غیرفعال است. 1: یک وقفه USART زمانی تولید می‌شود که TXE=1 در رجیستر USART_SR	TXEIE	7	
فعال‌سازی وقفه پایان انتقال 0: وقفه غیرفعال است. 1: یک وقفه USART زمانی تولید می‌شود که TC=1 در رجیستر USART_SR.	TCIE	6	
فعال‌سازی وقفه RXNE 0: وقفه غیرفعال است. 1: یک وقفه USART زمانی تولید می‌شود که ORE=1 یا RXNE=1 در رجیستر USART_SR.	RXNEIE	5	
فعال‌سازی وقفه IDLE 0: وقفه غیرفعال است. 1: یک وقفه USART زمانی تولید می‌شود که IDLE=1 در رجیستر USART_SR.	IDLEIE	4	
فعال‌سازی فرستنده 0: فرستنده غیرفعال است. 1: فرستنده فعال است.	TE	3	
فعال‌سازی گیرنده 0: گیرنده غیرفعال است. 1: گیرنده فعال است و شروع به جستجوی بیت شروع می‌کند.	RE	2	
بیداری گیرنده	RWU	1	
گیرنده در حالت فعال است.	0		
گیرنده در حالت بی‌صدا است	1		
ارسال کاراکتر Break	SBK	0	

هیچ کاراکتر break ارسال نمی‌شود.	0	
کاراکتر break ارسال خواهد شد	1	

13.2.2 USART_CR2 (Control Register 2)

Control register 2 (USART_CR2)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	LINEN	STOP[1:0]		CLKEN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	Res.	ADD[3:0]			
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

شرح	نام	بیت
رزرو شده‌اند و باید در مقدار بازنشانی نگه‌داری شوند	reserved	31:15
فعال‌سازی حالتLIN	LINEN	14
حالت LIN غیرفعال است	0	
حالت LIN فعال است.	1	
بیت‌های توقف این بیت‌ها برای برنامه‌ریزی تعداد بیت‌های توقف استفاده می‌شوند	STOP	13:12
1 بیت توقف	00	
0.5بیت توقف	01	
2بیت توقف	10	
1.5بیت توقف	11	
فعال‌سازی کلاک	CLKEN	11
پین SCLK غیرفعال است.	0	
پین SCLK فعال است.	1	
قطبش کلاک	CPOL	10

مقدار ثابت پایین بر SCLKدر	0	
مقدار ثابت بالا بر SCLKدر	1	
فاز کلاک	CPHA	9
اولین تغییر کلاک اولین لبه داده است	0	
دومین تغییر کلاک اولین لبه داده است	1	
پالس کلاک بیت آخر این بیت به کاربر اجازه می‌دهد تا انتخاب کند که آیا پالس کلاک مربوط به آخرین بیت داده منتقل شده (MSB) باید بر روی پین SCLK در حالت همزمان خروجی شود یا خیر.	LBCL	8
پالس کلاک بیت آخر به پین SCLK خروجی داده نمی‌شود.	0	
پالس کلاک بیت آخر به پین SCLK خروجی داده می‌شود	1	
	reserved	7
فعال‌سازی وقفه تشخیص Break LIN	LBDIE	6
وقفه غیرفعال است	0	
زمانی که LBD=1در رجیستر USART_SR، یک وقفه ایجاد می‌شود.	1	
طول تشخیص Break LIN	LBDL	5
تشخیص 10 Break بیتی.	0	
تشخیص 11 Break بیتی.	1	
	reserved	4
آدرس نود USART	ADD[3:0]	3:0

13.2.3 USART_CR3 (Control Register 3)

Control register 3 (USART_CR3)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				ONEBIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

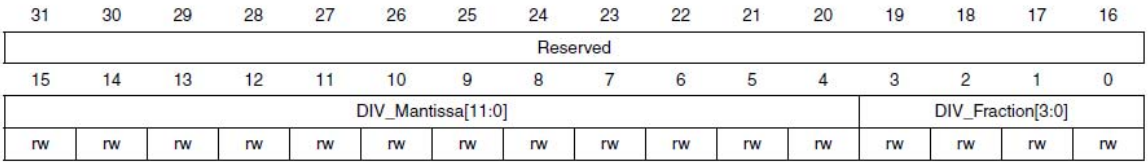
شرح	نام	بیت
رزرو شده‌اند و باید در مقدار بازنشانی نگه‌داری شوند	reserved	31:12
فعال‌سازی روش نمونه‌برداری یک بیتی	ONEBIT	11

روش سه نمونه در بیت	0	
روش یک نمونه در بیت	1	
فعال‌سازی وقفهCTS	CTSIE	10
وقفه غیرفعال است.	0	
هرگاه CTS=1در رجیستر USART_SR، یک وقفه ایجاد می‌شود.	1	
فعال‌سازیCTS	CTSE	9
کنترل جریان سخت‌افزاری CTS غیرفعال است.	0	
حالت CTS فعال است، داده فقط زمانی ارسال می‌شود که ورودی nCTSفعال (متصل به 0) باشد.	1	
فعال‌سازیRTS	RTSE	8
کنترل جریان سخت‌افزاری RTS غیرفعال است.	0	
وقفه RTS فعال است، درخواست داده تنها زمانی انجام می‌شود که در بافر دریافت فضایی وجود داشته باشد.	1	
فعال‌سازی DMA برای فرستنده	DMAT	7
DMA غیرفعال	0	
DMA فعال	1	
فعال‌سازی DMA برای گیرنده	DMAR	6
DMA غیرفعال	0	
DMA فعال	1	
فعال‌سازی حالت اسمارت‌کارت	SCEN	5
اسمارت‌کارت غیرفعال	0	
اسمارت‌کارت فعال	1	
فعال‌سازی NACK در اسمارت‌کارت	NACK	4
ارسال NACK در صورت بروز خطای پاریتی غیرفعال است	0	
ارسال NACK در هنگام بروز خطای پاریتی فعال است.	1	
انتخاب نیمه‌دوپلکس	HDSEL	3
نیمه دوبلکس فعال	0	
نیمه دوبلکس غیرفعال	1	
حالت کم‌مصرفIrDA	IRLP	2
حالت عادی	0	
حالت کم‌مصرف	1	
فعال‌سازی حالتIrDA	IREN	1
غیر فعال	0	
فعال	1	
فعال‌سازی وقفه خطا	EIE	0
غیر فعال	0	
فعال	1	

13.2.4 USART_BRR (Baud Rate Register)

- این رجیستر برای تنظیم نرخ baud ارتباط سریال استفاده می‌شود. با تغییر مقادیر این رجیستر، می‌توان نرخ انتقال داده‌ها را تنظیم کرد.

Baud rate register (USART_BRR)

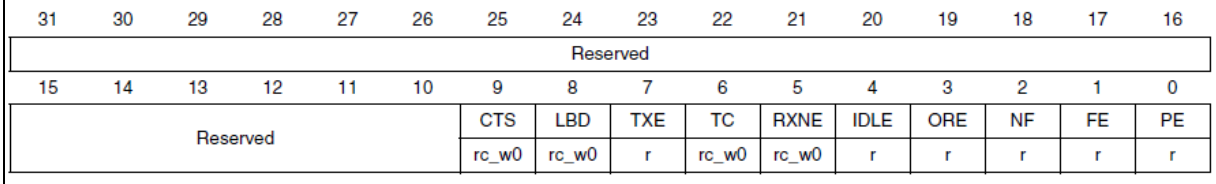


بیت	نام	شرح
31:16	reserved	
15:4	DIV_Mantissa[11:0]	بخش صحیح تقسیم کننده
3:0	DIV_Fraction[3:0]	بخش اعشاری تقسیم کننده

13.2.5 USART_SR (Status Register)

- این رجیستر وضعیت فعلی USART را نشان می‌دهد.

Status register (USART_SR)



بیت	نام	شرح
31:10	reserved	
9	CTS	پرچمCTS
	0	تغییر وضعیت در خط وضعیت nCTSرخ نداده است
	1	تغییر وضعیت در خط وضعیت nCTSرخ داده است
8	LBD	پرچم تشخیص شکستLIN
	0	شکست LIN تشخیص داده نشده است
	1	شکست LIN تشخیص داده شده است
7	TXE	بافر داده برای ارسال خالی است
	0	داده به رجیستر شیفت منتقل نشده است
	1	داده به رجیستر شیفت منتقل شده است
6	TC	انتقال کامل
	0	انتقال کامل نیست
	1	انتقال کامل است
5	RXNE	رجیستر داده خواندنی نیست
	0	داده دریافتی وجود ندارد
	1	داده دریافتی آماده خواندن است
4	IDLE	خط IDLE تشخیص داده شده است
	0	هیچ خط IDLE تشخیص داده نشده است.
	1	خط IDLE تشخیص داده شده است.
3	ORE	خطای overrun
	0	خطا رخ نداده
	1	بروز خطا
2	NF	پرچم تشخیص نویز
	0	نویز تشخیص داده نشد
	1	نویز تشخیص داده شد
1	FE	خطای فریم
	0	فریم خطا ندارد
	1	فریم خطا دارد
0	PE	خطای parity
	0	خطا ندارد
	1	خطا دارد

13.2.6 USART_DR (Data Register)

- این رجیستر برای ارسال و دریافت داده‌ها استفاده می‌شود. برای ارسال داده، باید داده مورد نظر را در این رجیستر قرار دهید و برای دریافت، داده در این رجیستر خوانده می‌شود.

13.2.7 USART_GTPR (Guard Time and Prescaler Register)

- این رجیستر برای تنظیم زمان گارد و پیش‌ساز ضریب (Prescaler) استفاده می‌شود و در ارتباطات خاص مانند LIN و یا پروتکل‌های خاص دیگر کاربرد دارد.

Guard time and prescaler register (USART_GTPR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GT[7:0]								PSC[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

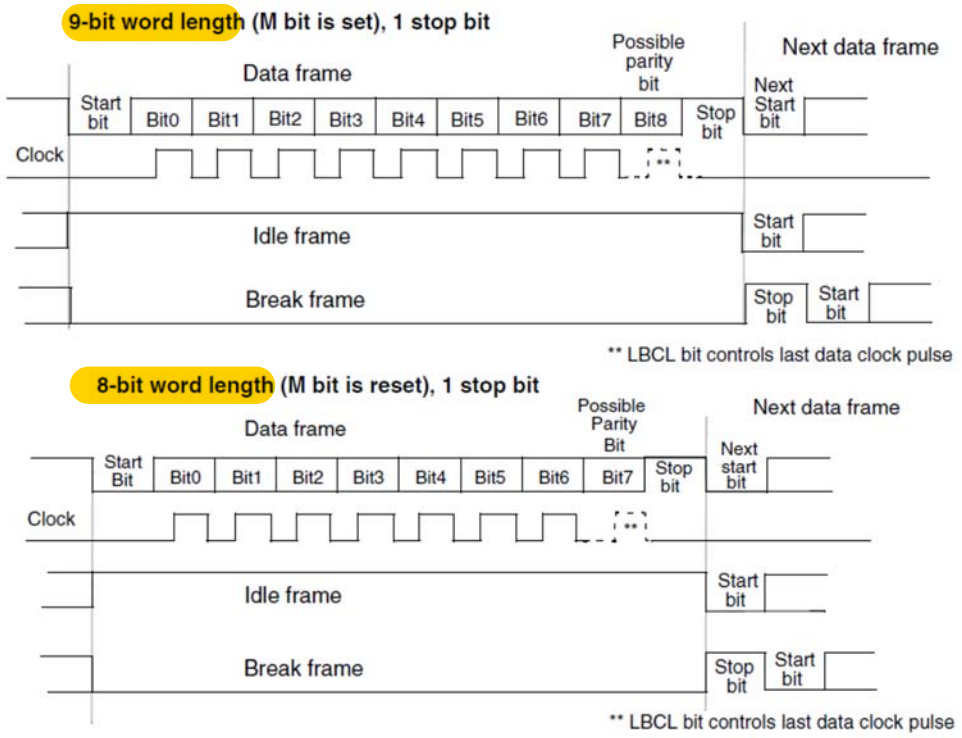
بیت	نام	شرح
31:16	reserved	
15:8	GT[7:0]	این بیت‌فیلد مقدار زمان نگهدارنده (Guard time) را به تعداد کلاک‌های (baud clocks) تعیین می‌کند. این مورد در حالت کارت هوشمند (Smartcard mode) استفاده می‌شود. پرچم انتقال کامل (Transmission Complete flag) پس از این مقدار زمان نگهدارنده تنظیم می‌شود.
7:0	PSC[7:0]	این مورد برای برنامه‌ریزی تقسیم‌کننده (prescaler) جهت تقسیم ساعت سیستم به منظور دستیابی به فرکانس کم‌مصرف استفاده می‌شود.

13.3 بسته داده UART STM32

طول بسته داده مانند شکل 13-2 می‌تواند به‌صورت ۸ یا ۹ بیت با برنامه‌ریزی بیت M در رجیستر USART_CR1 انتخاب شود. انتقال و دریافت توسط یک تولیدکننده نرخ baud مشترک انجام می‌شود. پین TX در طول بیت شروع مقدار صفر و در طول بیت توقف در مقدار 1 است.

یک کاراکتر Idle به‌عنوان یک فریم کامل از "۱"ها تفسیر می‌شود که با بیت شروع فریم حاوی داده‌ی بعدی دنبال می‌شود (تعداد "۱"ها شامل تعداد بیت‌های توقف خواهد بود).

یک کاراکتر Break به‌عنوان دریافت "۰"ها برای یک دوره فریم تفسیر می‌شود. در پایان فریم Break، فرستنده یکی یا دو بیت توقف (بیت منطقی "۱") را برای تأیید بیت شروع وارد می‌کند.



شکل 13-2: فریم داده در UART

13.3.1 ژنراتور نرخ بیت کسری (BRG) STM32

نرخ بیت برای گیرنده و فرستنده (RX و TX) هر دو به همان مقداری که در رجیستر USARTDIV برنامه‌ریزی شده‌اند، مانند شکل 13-3 تنظیم می‌شوند. USARTDIV یک عدد ثابت بدون علامت است که در رجیستر USART_BRR کدگذاری شده است. FCK ساعت ورودی به دستگاه USART است.

نرخ بیت در حالت استاندارد UART

$$\text{Tx/Rx baud} = \frac{f_{CK}}{8 \times (2 - \text{OVER8}) \times \text{USARTDIV}}$$

نرخ بیت در حالت IrDA، LIN و SmartCard

$$\text{Tx/Rx baud} = \frac{f_{CK}}{16 \times \text{USARTDIV}}$$

شکل 13-3: محاسبه نرخ بیت در UART

شمارنده‌های نرخ بیت پس از نوشتن به USART_BRR با مقدار جدید رجیسترهای نرخ بیت به‌روزرسانی می‌شوند. بنابراین، مقدار رجیستر نرخ بیت نباید در حین ارتباط تغییر کند.

لازم به ذکر است که هر چه فرکانس ساعت CPU پایین‌تر باشد، دقت برای یک نرخ بیت خاص کمتر خواهد بود. تنها USART1 با PCLK2 (حداکثر ۷۲ مگاهرتز) کلاک دهی می‌شود. سایر USARTها با PCLK1 (حداکثر ۳۶ مگاهرتز) کلاک دهی می‌شوند. در شکل 13-4، مقدار خطا (درصد) در نرخ بیت در نرخ‌های مختلف با فرکانس‌های مختلف ساعت برای مقایسه آورده شده است.

خطای (Error٪) به صورت (نرخ بیت محاسبه شده – نرخ بیت مورد نظر) / نرخ بیت مورد نظر تعریف می‌شود.

Oversampling by 8 (OVER8=1)							
Baud rate		f_{PCLK} = 42 MHz			f_{PCLK} = 84 MHz		
S.No	Desired	Actual	Value programmed in the baud rate register	% Error = (Calculated - Desired)B.Rate /Desired B.Rate	Actual	Value programmed in the baud rate register	% Error
1.	1.2 KBps	1.2 KBps	4375	0	1.2 KBps	8750	0
2.	2.4 KBps	2.4 KBps	2187.5	0	2.4 KBps	4375	0
3.	9.6 KBps	9.6 KBps	546.875	0	9.6 KBps	1093.75	0
4.	19.2 KBps	19.195 KBps	273.5	0.02	19.2 KBps	546.875	0
5.	38.4 KBps	38.391 KBps	136.75	0.02	38.391 KBps	273.5	0.02
6.	57.6 KBps	57.613 KBps	91.125	0.02	57.613 KBps	182.25	0.02
7.	115.2 KBps	115.068 KBps	45.625	0.11	115.226 KBps	91.125	0.02
8.	230.4 KBps	230.769 KBps	22.75	0.11	230.137 KBps	45.625	0.11
9.	460.8 KBps	461.538 KBps	11.375	0.16	461.538 KBps	22.75	0.16
10.	921.6 KBps	913.043 KBps	5.75	0.93	923.076 KBps	11.375	0.93
11.	1.792 MBps	1.826 MBps	2.875	1.9	1.787Mbps	5.875	0.27
12.	1.8432 MBps	1.826 MBps	2.875	0.93	1.826 MBps	5.75	0.93
13.	3.584 MBps	3.5 MBps	1.5	2.34	3.652 MBps	2.875	1.9
14.	3.6864 MBps	3.82 MBps	1.375	3.57	3.652 MBps	2.875	0.93
15.	7.168 MBps	N.A	N.A	N.A	7 MBps	1.5	2.34
16.	7.3728 MBps	N.A	N.A	N.A	7.636 MBps	1.375	3.57
18.	9 MBps	N.A	N.A	N.A	9.333 MBps	1.125	3.7
20.	10.5 MBps	N.A	N.A	N.A	10.5 MBps	1	0

شکل 13-4: خطای نرخ بیت های مختلف در UART

13.3.2 UART STM32 Parity bit

کنترل توازن(Parity) (تولید بیت توازن در ارسال و بررسی توازن در دریافت) می‌تواند با تنظیم بیت PCE در رجیستر USART_CR1 فعال شود. بسته به طول فریم که توسط بیت M تعریف شده است، فرمت‌های ممکن فریم USART درشکل 13-5 نشان داده شده است.

Frame formats

M bit	PCE bit	USART frame
0	0	| SB | 8 bit data | STB |
0	1	| SB | 7-bit data | PB | STB |
1	0	| SB | 9-bit data | STB |
1	1	| SB | 8-bit data PB | STB |

شکل 13-5: تنظیمات فریم UART

بیت توازن به دو دسته توازن زوج و فرد تقسیم می‌شود.

توازن زوج: بیت توازن به‌گونه‌ای محاسبه می‌شود که تعداد "۱" ها در داخل فریم متشکل از ۷ یا ۸ بیت کم‌اهمیت (بسته به اینکه M برابر ۰ یا ۱ باشد) و بیت توازن، عددی زوج باشد.

توازن فرد: بیت توازن به‌گونه‌ای محاسبه می‌شود که تعداد "۱" ها در داخل فریم متشکل از ۷ یا ۸ بیت کم‌اهمیت (بسته به اینکه M برابر ۰ یا ۱ باشد) و بیت توازن، عددی فرد باشد.

13.4 اتصال آسنکرون در UART

13.4.1 فرستنده آسنکرون USART STM32

فرستنده USART می‌تواند داده‌های کلمه‌ای با طول ۸ یا ۹ بیت را بسته به وضعیت بیت M ارسال کند. زمانی که بیت فعال‌سازی ارسال (TE) تنظیم شود، داده‌ها در شیفت رجیستر ارسال به پین TX منتقل می‌شوند و پالس‌های کلاک مربوطه بر روی پین CK برای همزمانی ارسال می‌گردد.

در طول یک انتقال USART، داده‌ها از کم‌اهمیت‌ترین بیت اول بر روی پین TX منتقل می‌شوند. هر کاراکتر با یک بیت شروع که به مدت یک بیت در سطح منطقی پایین است، آغاز و با تعداد قابل تنظیمی از بیت‌های توقف خاتمه می‌یابد. بیت‌های توقف0/5، ۱، ۱,۵ و ۲ بیتی توسط USART پشتیبانی می‌شوند.

لازم به ذکر است که بیت TEدر طول انتقال داده، بازنشانی نشود زیرا بازنشانی بیت TE در حین انتقال باعث خراب شدن داده‌ها بر روی پین TX به سبب توقف شمارنده‌های نرخ baud میگردد و داده‌های جاری در حال انتقال از دست خواهد رفت. پس از فعال‌سازی بیت TE، یک فریم Idle ارسال خواهد شد.

مراحل انتقال داده USART به صورت گام به گام بیان شده است.

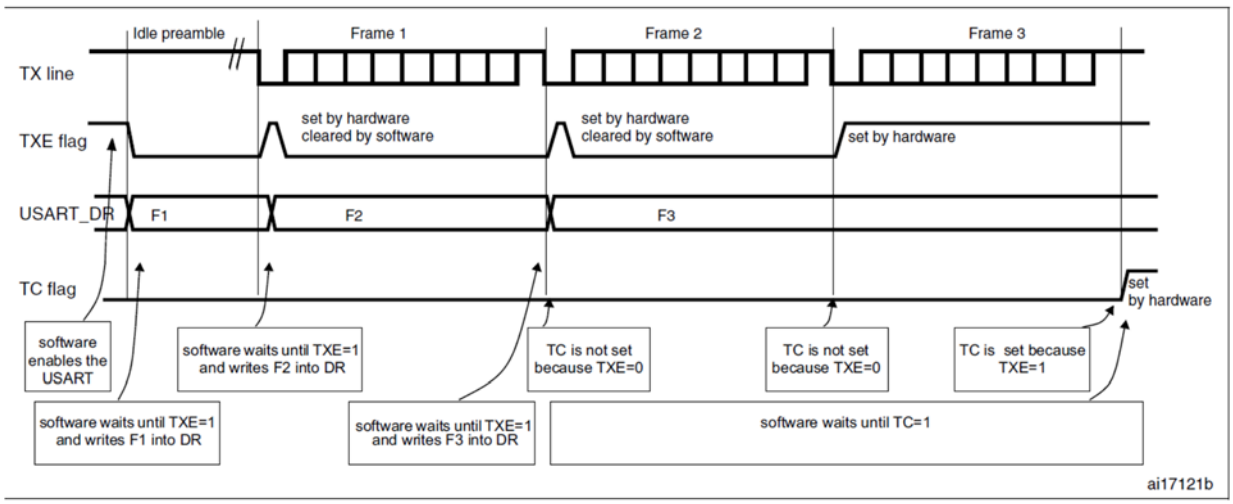
1. USART را با نوشتن مقدار 1 در بیت UE در رجیستر USART_CR1 فعال کنید.
2. بیت M را در USART_CR1 برنامه‌ریزی کنید تا طول کلمه را تعریف کنید.
3. تعداد بیت‌های توقف را در USART_CR2 برنامه‌ریزی کنید.
4. اگر قرار است ارتباط چندبافری انجام شود، DMA را در USART_CR3 فعال کنید (DMAT).
5. نرخ baud مورد نظر را با استفاده از رجیستر USART_BRR انتخاب کنید.
6. بیت TE را در USART_CR1 تنظیم کنید تا یک فریم Idle به‌عنوان اولین انتقال ارسال شود.
7. داده‌ای که باید ارسال شود را در رجیستر USART_DR بنویسید (این بیت TXE را پاک می‌کند). این کار را برای هر داده‌ای که باید منتقل شود، در صورت استفاده از یک بافر تکرار کنید.
8. پس از نوشتن آخرین داده در رجیستر USART_DR، منتظر بمانید تا TC=1 شود. این نشان می‌دهد که انتقال آخرین فریم کامل شده است. این کار برای زمانی که USART غیرفعال می‌شود یا به حالت توقف می‌رود، ضروری است تا از خراب شدن آخرین انتقال جلوگیری شود.

در هنگام انتقال داده به موارد ذیل توجه نمایید.

1. زمانی که یک انتقال در حال انجام است، یک دستور نوشتن به رجیستر USART_DR داده‌ها را در رجیستر TDR ذخیره می‌کند و این داده‌ها در انتهای انتقال جاری به شیفت رجیستر کپی می‌شوند.
2. زمانی که هیچ انتقالی در حال انجام نیست، یک دستور نوشتن به رجیستر USART_DR داده‌ها را مستقیماً در شیفت رجیستر قرار می‌دهد، انتقال داده آغاز می‌شود و بیت TXE بلافاصله تنظیم می‌شود.

3. اگر یک فریم (پس از بیت توقف) منتقل شود و بیت TXE تنظیم شده باشد، بیت TC بالا (مقدار یک) می‌رود. اگر بیت TCIE در رجیستر USART_CR1 تنظیم شده باشد، یک وقفه تولید می‌شود.
4. پس از نوشتن آخرین داده در رجیستر USART_DR، لازم است که قبل از غیرفعال کردن USART یا ورود میکروکنترلر به حالت کم‌مصرف، منتظر بمانید تا TC=1 شود.
5. بیت TC با به صورت نرم افزاری با یک خواندن از رجیستر USART_SR و به دنبال آن یک نوشتن به رجیستر USART_DR پاک می‌شود، همچنین می‌توان با نوشتن یک '۰' به بیت TC، آن را پاک کرد.

دیاگرام توالی زمانی در شکل 13-6رفتار دقیق سخت‌افزار فرستنده USART، بیت TC و بیت پرچم TXE را نشان می‌دهد.



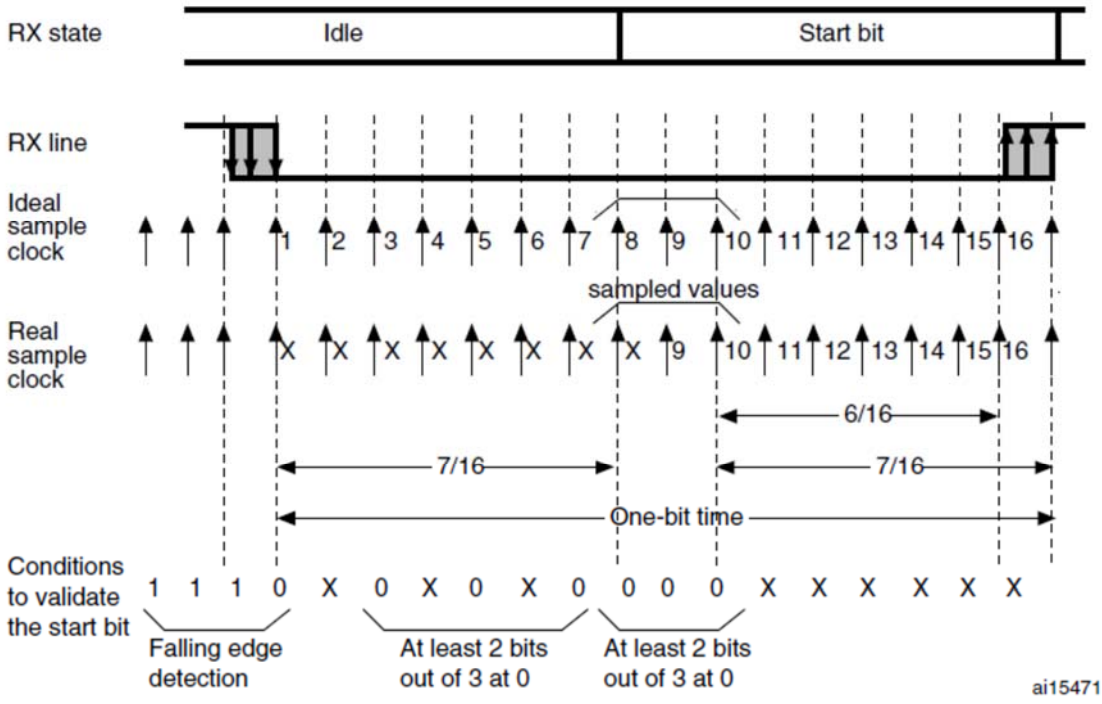
شکل 13-6 : دیاگرام توالی زمانی در ارسال داده با UART

13.4.2 گیرنده آسنکرون USART STM32

گیرنده USART می‌تواند کلمات داده‌ای با طول ۸ یا ۹ بیت دریافت کند که بسته به بیت M در رجیستر USART_CR1 است. در USART، بیت شروع زمانی شناسایی می‌شود که یک توالی خاص از نمونه‌ها تشخیص داده شود.

این توالی به صورت زیر است: 1 1 1 0 X 0 X 0 0 0 0. اگر این توالی کامل نباشد، شناسایی بیت شروع متوقف می‌شود و گیرنده به حالت بیکار (بدون تنظیم پرچم) بازمی‌گردد و منتظر یک لبه نزولی در خط RX می‌ماند.

کلاک گیرنده ۱۶ برابر سریع‌تر از کلاک فرستنده است که توسط همان ژنراتور نرخ بیت تولید می‌شود. این موضوع تضمین می‌کند که نمونه‌های بیشتری (معمولاً ۱۶عدد) در هر مدت زمان بیت وجود داشته باشد. دیاگرام شکل 13-7 شناسایی بیت شروع و شرایط دقیقی که باید برای تأیید یک بیت شروع واقعی برآورده شود، نشان می‌دهد.



شکل 13-7: دیاگرام زمانی گیرنده در ارتباط UART

در حین دریافت داده در USART، داده‌ها از طریق پین RX به ترتیب از کم‌اهمیت‌ترین بیت وارد می‌شوند. در این حالت، رجیستر USART_DR شامل یک بافر (RDR) بین باس داخلی و رجیستر شیفت دریافتی است.

مراحل دریافت داده در USART به صورت گام به گام به شرح ذیل است:

1. USART را با نوشتن بیت UE در رجیستر USART_CR1 به ۱ فعال کنید.
2. بیت M در USART_CR1 را برنامه‌ریزی کنید تا طول کلمه را تعیین کنید.
3. تعداد بیت‌های توقف را در USART_CR2 برنامه‌ریزی کنید.
4. در صورت برقراری ارتباط چندبافری، گزینه فعال‌سازی DMA (DMAR) را در USART_CR3 انتخاب کنید. رجیستر DMA را طبق توضیحات مربوط به ارتباط چندبافری پیکربندی کنید.
5. نرخ بیت مورد نظر را با استفاده از رجیستر نرخ بیت USART_BRR انتخاب کنید.
6. بیت RE را در USART_CR1 تنظیم کنید. بدین ترتیب گیرنده فعال شده و فرآیند جستجوی یک بیت شروع را آغاز می‌کند.

7. وقتی یک کاراکتر دریافت می‌شود، بیت RXNE تنظیم می‌شود. این نشان‌دهنده انتقال محتوای رجیستر شیفت به RDR است. به عبارت دیگر، داده دریافت شده و می‌تواند خوانده شود. اگر بیت RXNEIE تنظیم شده باشد، یک وقفه تولید می‌شود.

در حالت چندبافری، RXNE پس از هر بایت دریافتی تنظیم می‌شود و با خواندن DMA به رجیستر داده پاک می‌شود. در حالت بافر تک، پاک‌سازی بیت RXNE توسط خواندن نرم‌افزاری به رجیستر USART_DR انجام می‌شود. همچنین می‌توان با نوشتن صفر به آن، پرچم RXNE را پاک کرد. بیت RXNE باید قبل از پایان دریافت کاراکتر بعدی پاک شود تا از خطای Overrun جلوگیری شود.

لازم به ذکر است که بیت RE نباید در حین دریافت داده‌ها بازنشانی شود. اگر بیت RE در حین دریافت غیرفعال شود، دریافت بایت جاری متوقف خواهد شد.

13.5 حالت اتصال سنکرون

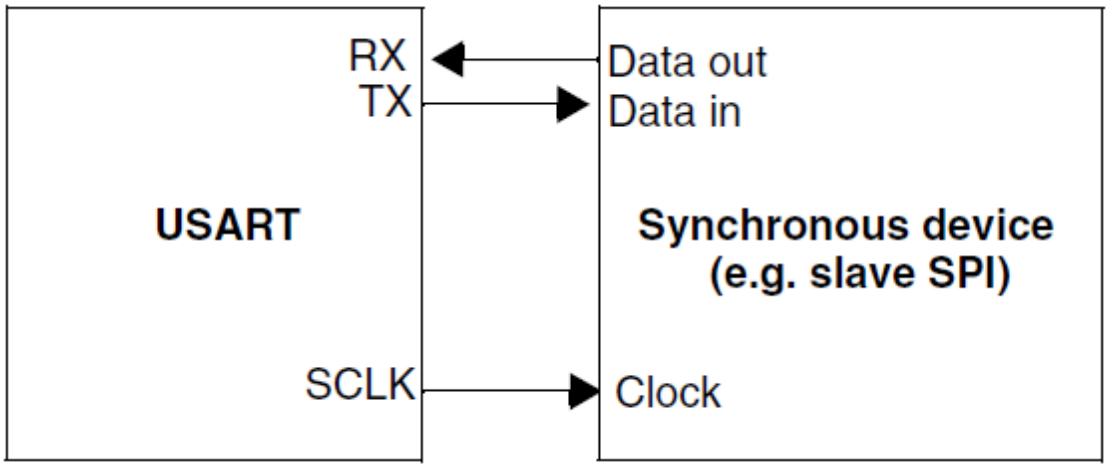
حالت همزمان با نوشتن بیت CLKEN در رجیستر USART_CR2 به ۱ انتخاب می‌شود. در حالت همزمان، باید بیت‌های زیر پاک (۰) نگه داشته شوند:

- بیت LINEN در رجیستر USART_CR2
- بیت‌های SCEN، HDSEL و IREN در رجیستر USART_CR3

USART به کاربر اجازه می‌دهد تا ارتباط سریال همزمان دوطرفه را در حالت مستر کنترل کند. نحوه اتصال در این حالت در شکل 13-8نشان داده شده است.

پین CK خروجی ساعت فرستنده USART است. هیچ پالس ساعتی در زمان بیت شروع و بیت توقف به پین CK ارسال نمی‌شود. تولید پالس ساعت در آخرین بیت داده به وضعیت بیت LBCL در رجیستر USART_CR2، بستگی دارد.

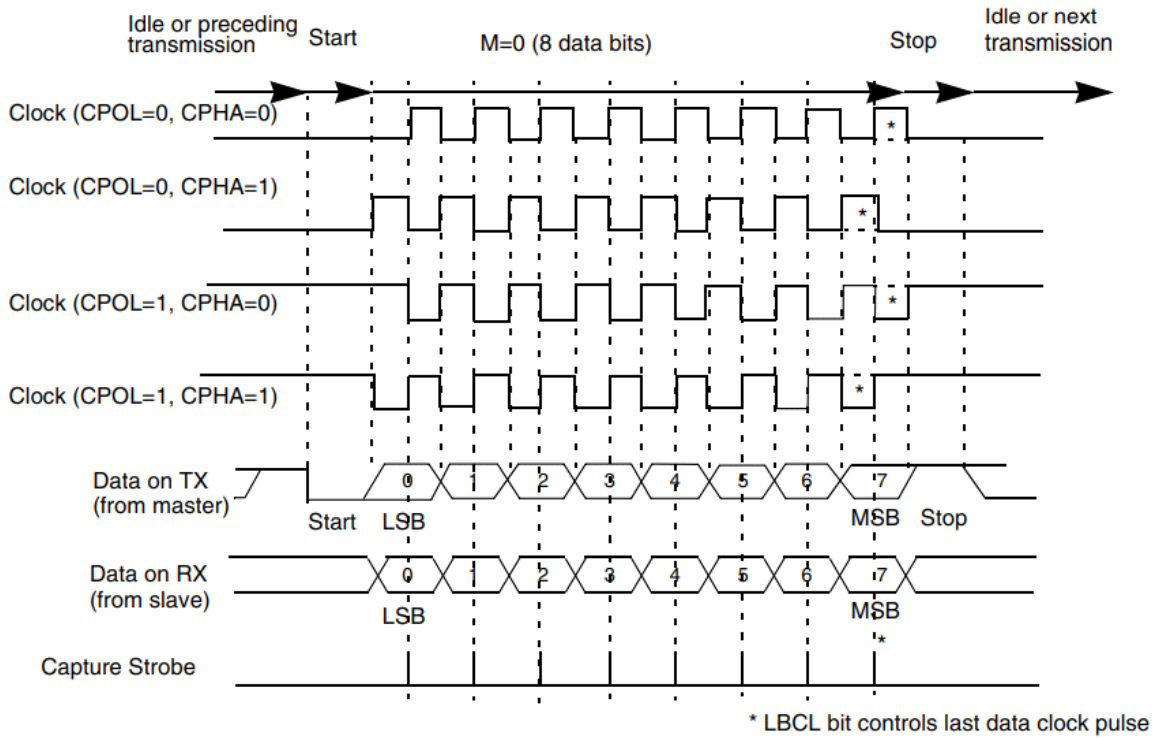
بیت CPOL در رجیستر USART_CR2 به کاربر اجازه می‌دهد تا قطبیت ساعت را انتخاب کند و بیت CPHA در رجیستر USART_CR2 به کاربر اجازه می‌دهد تا فاز ساعت خارجی را انتخاب کند.



شکل 13-8: اتصال UART در حالت سنکرون

در حالت IDLE، مقدمه و استراحت ارسال، ساعت CK خارجی فعال نمی‌شود. در حالت همزمان، فرستنده USART دقیقاً مانند حالت ناهمزمان کار می‌کند. اما از آنجا که CK با TX همزمان است (بر اساس CPOL و CPHA)، داده‌های موجود در TX همزمان هستند.

در حالت سنکرون، گیرنده USART در مقایسه با گیرنده آسنکرون به روشی متفاوت کار می کند. اگر RE=1 باشد، داده‌ها بر روی CK (لبه صعودی یا نزولی، بسته به CPOL و CPHA) مانند آنچه درشکل 13-9 نشان داده شده است، نمونه برداری می‌شود.



شکل 13-9: ارسال و دریافت در حالت سنکرون UART

پین CK به همراه پین TX کار می‌کند. بنابراین، ساعت تنها در صورتی ارائه می‌شود که فرستنده فعال باشد (TE=1) و داده در حال ارسال باشد (رجیستر داده USART_DR نوشته شده باشد). این بدان معناست که امکان دریافت داده‌های همزمان بدون ارسال داده وجود ندارد.

13.6 حالت تک‌سیم (نیم‌دوطرفه) UART STM32

حالت نیم‌دوطرفه تک‌سیم با تنظیم بیت HDSEL در رجیستر USART_CR3 انتخاب می‌شود. در این حالت، باید بیت‌های زیر پاک (۰) نگه داشته شوند:

- بیت‌های LINEN و CLKEN در رجیستر USART_CR2
- بیت‌های SCEN و IREN در رجیستر USART_CR3

USART می‌تواند به گونه‌ای پیکربندی شود که پروتکل نیم‌دوطرفه تک‌سیم را دنبال کند. در حالت نیم‌دوطرفه تک‌سیم، پین‌های TX و RX به‌طور داخلی به هم متصل هستند. انتخاب بین ارتباط نیم‌دوطرفه و دوطرفه کامل با یک بیت کنترل به نام 'HALF DUPLEX SEL' (HDSEL در USART_CR3) انجام می‌شود.

به محض اینکه HDSEL به ۱ نوشته شود:

-خطوط TX و RX به صورت داخلی متصل هستند

- RX دیگر استفاده نمی‌شود.

- TX همیشه زمانی که داده‌ای ارسال نمی‌شود آزاد است.

ارتباطات مشابه آنچه در حالت عادی USART انجام می‌شود، خواهد بود. با این تفاوت که ارسال و دریافت بایستی توسط نرم افزار مدیریت شود.

13.7 ارتباط چندپردازنده‌ای UART STM32

امکان انجام ارتباط چندپردازنده‌ای با USART وجود دارد (چندین USART که در یک شبکه متصل هستند). به عنوان مثال، یکی از USART ها می‌تواند به عنوان مستر عمل کند، خروجی TX آن به ورودی RX دیگر USART متصل می‌شود. سایر USART ها به عنوان اسلیو عمل می‌کنند و خروجی‌های TX آن‌ها به‌طور منطقی با هم AND شده و به ورودی RX مستر متصل می‌شوند.

در پیکربندی‌های چندپردازنده‌ای، فقط گیرنده مورد نظر بایستی پیام را به صورت کامل دریافت کند، و سایر گیرنده ها پیام را دریافت نکنند. لذا برای هر گیرنده آدرسی در نظر گرفته می‌شود.

دستگاه‌های غیرمخاطب می‌توانند با استفاده از تابع بی‌صدا (muting function) به حالت بی‌صدا وارد شوند. در حالت بی‌صدا:

- هیچ یک از بیت‌های وضعیت دریافت نمی‌توانند تنظیم شوند.
- تمام وقفه‌های دریافت غیرفعال هستند.

- بیت RWU در ثبت‌نام USART_CR1 به ۱ تنظیم می‌شود. RWU می‌تواند به صورت خودکار توسط سخت‌افزار کنترل شود یا تحت شرایط خاصی توسط نرم‌افزار نوشته شود.

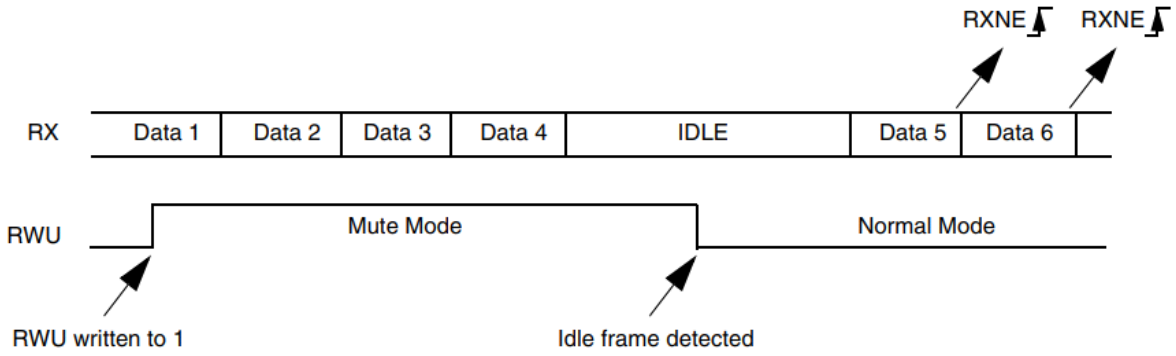
USART می‌تواند با استفاده از یکی از دو روش، بنا به بیت WAKE در رجیستر USART_CR1، به حالت بی‌صدا وارد شود یا از آن خارج شود:

- تشخیص خط بیکار (Idle Line detection) اگر بیت WAKE تنظیم نشده باشد،
- تشخیص علامت آدرس (Address Mark detection) اگر بیت WAKE تنظیم شده باشد.

13.7.1 تشخیص خط بیکار (WAKE=0)

USART زمانی به حالت بی‌صدا وارد می‌شود که بیت RWU به ۱ نوشته شود.

این دستگاه هنگامی بیدار می‌شود که یک فریم بی‌کار تشخیص داده شود. سپس بیت RWU توسط سخت‌افزار پاک می‌شود، اما بیت IDLE در رجیستر USART_SR تنظیم نمی‌شود. همچنین می‌توان بیت RWU را با نرم‌افزار به ۰ تغییر داد. در شکل 13-10 رفتار حالت بی‌صدا با استفاده از تشخیص خط بی‌کار نشان داده شده است.



شکل 13-10: شناسایی وضعیت بیکار خط در حالتی که گیرنده در وضعیت بیصدا قرار دارد

تشخیص علامت آدرس (WAKE=1)

13.7.2 تشخیص علامت آدرس(WAKE=1)

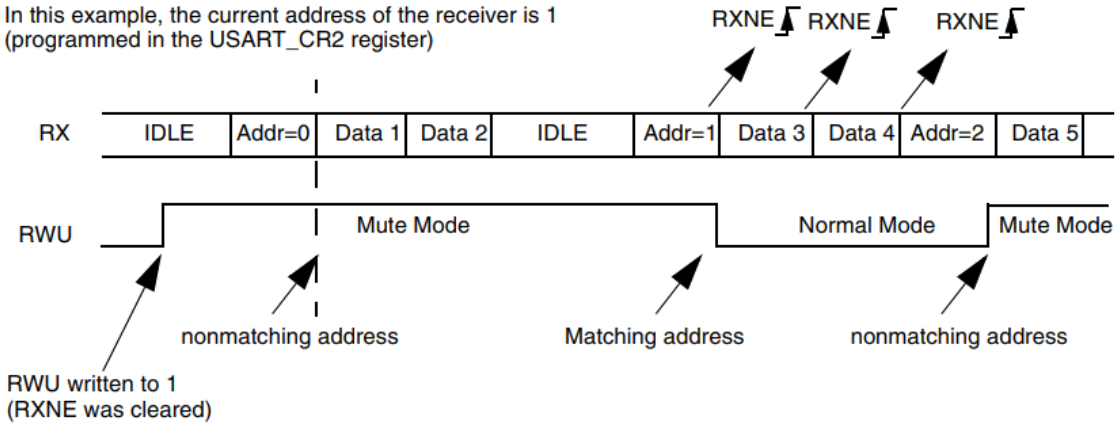
در حالت شناسایی آدرس، اگر اگر بیت MSB (بیت بیشترین ارزش) آن‌ها ۱ باشد، بایت‌ها به عنوان آدرس شناسایی می‌شوند و در غیر این صورت به عنوان داده در نظر گرفته می‌شوند. در یک بایت آدرس، آدرس گیرنده‌ی مورد نظر در ۴ بیت کم‌ارزش (LSB) قرار می‌گیرد. این کلمه ۴ بیتی توسط گیرنده با آدرس خود که در بیت‌های ADD در رجیستر USART_CR2 برنامه‌ریزی شده، مقایسه می‌شود.

و در صورت عدم مطابقت به حالت mute میرود. در این حالت، بیت RWU توسط سخت‌افزار تنظیم می‌شود. پرچم RXNE برای این بایت آدرس تنظیم نمی‌شود و هیچ وقفه‌ای یا درخواست DMA نیز صادر نمی‌شود، زیرا USART به حالت بی‌صدا وارد شده است.

این دستگاه هنگامی از حالت بی‌صدا خارج می‌شود که یک کاراکتر آدرس دریافت شود که با آدرس برنامه‌ریزی شده مطابقت داشته باشد. سپس بیت RWU پاک می‌شود و بایت‌های بعدی به طور عادی دریافت می‌شوند. بیت RXNE برای کاراکتر آدرس تنظیم می‌شود زیرا بیت RWU پاک شده است.

بیت RWU می‌تواند به ۰ یا ۱ نوشته شود زمانی که بافر دریافت کننده هیچ داده‌ای ندارد (RXNE=0 در رجیستر USART_SR). در غیر این صورت، تلاش برای نوشتن نادیده گرفته می‌شود.

مثالی از رفتار حالت بی‌صدا با استفاده از تشخیص علامت آدرس در شکل 13-11 ارائه شده است.



شکل 13-11: شناسایی آدرس در حالتی که دستگاه در حالت بیصدا باشد

13.8 سایر مدهای UART

سایر مدهای کاری شامل LIN ، USART IrDA و SmartCard نیز وجود دارند که برای توضیحات بیشتر در برگه های راهنمای STM32f407 موجود است.

برای آشنایی با اینکه چه حالت‌های پیکربندی در کدام ماژول USART در میکروکنترلر پشتیبانی می‌شود، باید به دیتاشیت آن مراجعه کنید. در شکل 13-13، حالت‌های پشتیبانی‌شده در هر ماژول USART در STM32f407را نشان داده شده است.

USART modes	USART1	USART2	USART3	UART4	UART5	USART6
Asynchronous mode	X	X	X	X	X	X
Hardware flow control	X	X	X	NA	NA	X
Multibuffer communication (DMA)	X	X	X	X	X	X
Multiprocessor communication	X	X	X	X	X	X
Synchronous	X	X	X	NA	NA	X
Smartcard	X	X	X	NA	NA	X
Half-duplex (single-wire mode)	X	X	X	X	X	X
IrDA	X	X	X	X	X	X
LIN	X	X	X	X	X	X

1. X = supported; NA = not applicable.

شکل 13-12: مدهای قابل پشتیبانی در هر یک از UART ها

13.9 خطاهای ۱،۲،۳ USART STM32

سخت‌افزار USART در میکروکنترلرهای STM32 قادر به تشخیص ۴ نوع خطای عملیاتی است. سیگنال‌های خطا به شرح زیر هستند:

13.9.1 خطای Overrun

خطای Overrun زمانی رخ می‌دهد که یک کاراکتر دریافت شود در حالی که بیت RXNE بازنشانی نشده است. داده‌ها نمی‌توانند از رجیستر شیفت به رجیستر RDR منتقل شوند تا زمانی که بیت RXNE پاک شود.

13.9.2 خطای نویز

تکنیک‌های اضافه‌نمونه‌برداری (به جز در حالت همزمان) برای بازیابی داده‌ها با تمایز بین داده‌های ورودی معتبر و نویز استفاده می‌شوند.

13.9.3 خطای فریم

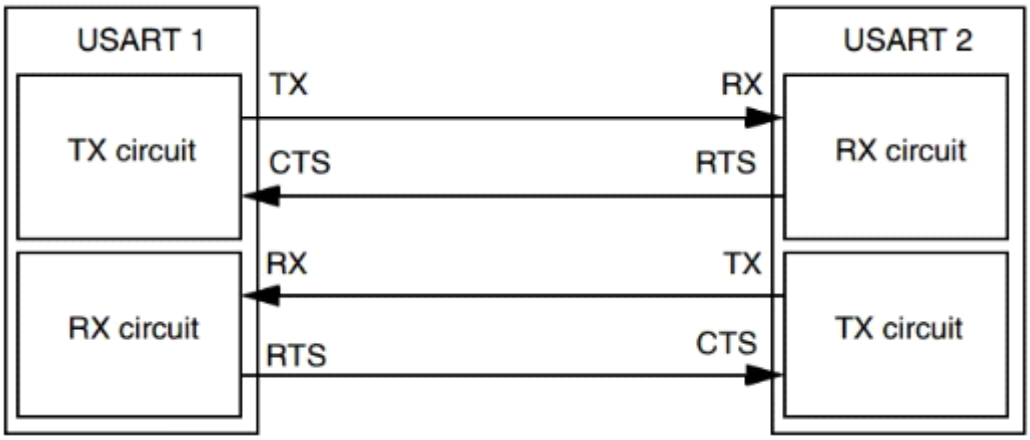
یک خطای فریم زمانی شناسایی می‌شود که: بیت توقف در زمان پیش‌بینی شده در هنگام دریافت شناسایی نشود، که ممکن است ناشی از عدم همزمانی یا نویز زیاد باشد.

13.9.4 خطای بررسی توازن

زمانی که یک خطای توازن در فریم داده دریافتی شناسایی شود، بیت PE تنظیم می‌شود و اگر فعال باشد، یک وقفه ایجاد می‌کند. این بیت می‌تواند به توازن زوج یا فرد تنظیم شود، بسته به برنامه و اینکه آیا در ارتباط پیاده‌سازی شده است یا نه.

13.10 کنترل جریان داده‌های سخت‌افزاری USART STM32

این امکان وجود دارد که جریان داده‌های سری بین دو دستگاه را با استفاده از ورودی CTS و خروجی RTS کنترل کنید. کنترل جریان RTS و CTS می‌تواند به‌طور مستقل با نوشتن بیت‌های RTSE و CTSE به ۱ (در رجیستر USART_CR3) فعال شود. شکل 13-13 نشان می‌دهد که چگونه دو دستگاه را در این حالت به هم متصل کنید.



شکل 13-13: اتصال UART با کنترل جریان داده

13.10.1 کنترل جریان RTS

اگر کنترل جریان RTS فعال باشد (RTSE=1)، سپس RTS به حالت فعال (پایین) می‌رود تا زمانی که گیرنده USART آماده دریافت داده‌های جدید باشد. وقتی که رجیستر دریافت پر شود، RTS غیرفعال می‌شود و نشان می‌دهد که انتظار می‌رود انتقال در انتهای فریم فعلی متوقف شود.

13.10.2 کنترل جریان CTS

اگر کنترل جریان CTS فعال باشد (CTSE=1)، سپس فرستنده قبل از ارسال فریم بعدی ورودی CTS را بررسی می‌کند. اگر CTS به حالت فعال (پایین) باشد، داده بعدی ارسال می‌شود (با فرض اینکه داده‌ای برای ارسال وجود داشته باشد، به عبارت دیگر، اگر TXE=0)، در غیر این صورت، انتقال انجام نمی‌شود. وقتی که CTS در حین یک انتقال غیرفعال می‌شود، انتقال جاری تکمیل می‌شود قبل از اینکه فرستنده متوقف شود.

13.11 وقفه‌های UART STM32

رویدادهای وقفه USART به همان وکتور وقفه متصل هستند. بنابراین، USART یک سیگنال وقفه واحد را بدون توجه به منبع آن ایجاد می‌کند. نرم‌افزار باید این سیگنال را شناسایی کند.

در حین انتقال:

- وقفه پایان انتقال (Transmission Complete)
- وقفه آماده ارسال (Clear to Send)
- وقفه خالی بودن رجیستر داده‌های ارسالی (Transmit Data Register empty)

در حین دریافت:

- شناسایی خط بیکار (Idle Line detection)
- خطای Overrun
- رجیستر داده‌های دریافتی غیر خالی (Receive Data register not empty)

- خطای توازن (Parity error)

- شناسایی شکست LIN (LIN break detection)

- پرچم نویز (Noise Flag) (فقط در ارتباطات چند بافر)

- خطای فریم (Framing Error) (فقط در ارتباطات چند بافر)

در شکل 13-14 لیست وقفه های UART و نگاشت وقفه ها نشان داده شده است.

Interrupt event	Event flag	Enable control bit
Transmit Data Register Empty	TXE	TXEIE
CTS flag	CTS	CTSIE
Transmission Complete	TC	TCIE
Received Data Ready to be Read	RXNE	RXNEIE
Overrun Error Detected	ORE	
Idle Line Detected	IDLE	IDLEIE
Parity Error	PE	PEIE
Break Flag	LBD	LBDIE
Noise Flag, Overrun error and Framing Error in multibuffer communication	NF or ORE or FE	EIE

13.11.1 توابع callback

توابعی ذیل به مدیریت وقفه‌های مرتبط با UART (Universal Asynchronous Receiver-Transmitter) در میکروکنترلر STM32F407 کمک می‌کنند. عملکرد هر یک از این توابع به شرح زیر است:

1. void HAL_UART_IRQHandler(UART_HandleTypeDef *huart);

این تابع به عنوان تابع برخورد با وقفه (IRQ handler) برای UART عمل می‌کند. هر زمان که یک وقفه UART رخ دهد، این تابع فراخوانی می‌شود. این تابع مسئول پردازش وقایع مختلف مانند دریافت داده‌ها، ارسال داده‌ها، و خطاها است.

2. void HAL_UART_TxCpltCallback(UART_HandleTypeDef *huart);

این تابع به عنوان یک callback برای مدیریت پایان انتقال داده‌ها (Tx Complete) عمل می‌کند. زمانی که تمام داده‌ها با موفقیت ارسال شود، این تابع فراخوانی می‌شود و می‌توان در آن اقدامات لازم مانند آزادسازی منابع یا آغاز عملیات جدید را انجام داد.

3. void HAL_UART_TxHalfCpltCallback(UART_HandleTypeDef *huart);

این تابع به عنوان callback برای مدیریت نیمه پایان انتقال (Tx Half Complete) عمل می‌کند. زمانی که نیمی از داده‌ها ارسال شده باشند، این تابع فراخوانی شده و می‌توان در آن برای پردازش‌های خاص اقدام کرد.

4. void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart);

این تابع callback برای مدیریت پایان دریافت داده‌ها (Rx Complete) است. هر زمان که داده‌ای با موفقیت دریافت شود، این تابع فراخوانی می‌شود و می‌توان از آن برای پردازش داده‌های دریافتی استفاده کرد.

5. void HAL_UART_RxHalfCpltCallback(UART_HandleTypeDef *huart);

این تابع callback برای مدیریت نیمه پایان دریافت (Rx Half Complete) عمل می‌کند. هنگامی که نیمه‌ای از داده‌ها دریافت شده باشد، این تابع فراخوانی می‌شود.

6. void HAL_UART_ErrorCallback(UART_HandleTypeDef *huart);

این تابع callback برای مدیریت خطاهای مرتبط با UART است. در صورتی که خطایی در حین انتقال یا دریافت داده‌ها رخ دهد (مثل Overrun, Parity error و ...)، این تابع فراخوانی می‌شود.

7. void HAL_UART_AbortCpltCallback(UART_HandleTypeDef *huart);

این تابع زمانی فراخوانی می‌شود که یک عملیات UART (ارسال یا دریافت) به طور کامل متوقف شده باشد. این تابع به مدیریت وضعیت بعد از متوقف شدن عملیات می‌پردازد.

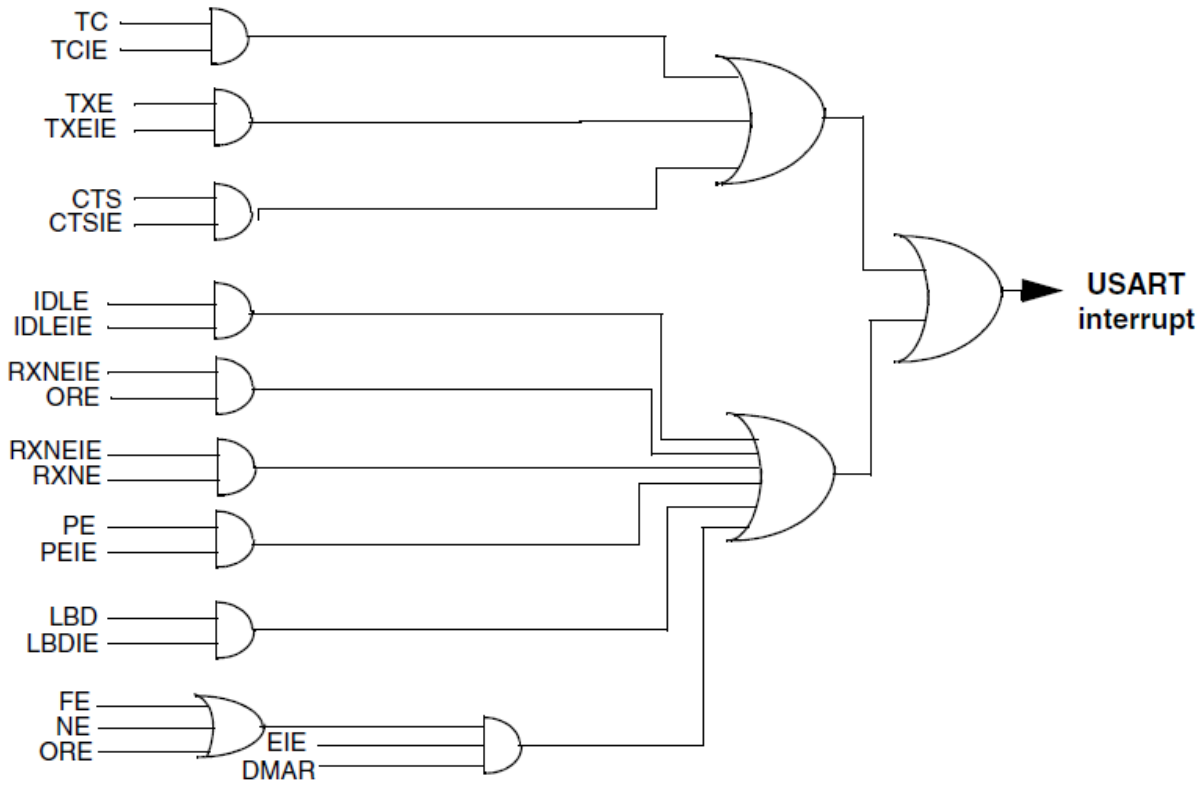
8. void HAL_UART_AbortTransmitCpltCallback(UART_HandleTypeDef *huart);

این تابع خاص برای مدیریت پایان عملیات متوقف شده ارسال (Abort Transmit) است. این تابع زمانی که انتقال داده در حال انجام متوقف می‌شود، فراخوانی می‌شود.

9. void HAL_UART_AbortReceiveCpltCallback(UART_HandleTypeDef *huart);

این تابع برای مدیریت پایان عملیات متوقف شده دریافت (Abort Receive) استفاده می‌شود. در صورتی که دریافت داده متوقف شود، این تابع فراخوانی می‌شود.

به طور کلی، این توابع برای پیاده‌سازی یک سیستم دریافت و ارسال داده‌های ارتباطی با استفاده از پروتکل UART در STM32F407 ضروری هستند و به توسعه‌دهندگان این امکان را می‌دهند که به راحتی رخدادهای مختلف UART را مدیریت کنند.



شکل 13-14: نگاشت و لیست وقفه های UART

13.12 تنظیمات UART STM32 در CubeMX

از ابزار نرم‌افزاری CubeMX برای پیکربندی سخت‌افزار USART مانند شکل 13-15 استفاده می‌گردد. در تب پیکربندی ماژول USART در CubeMX، گزینه‌های زیر تنظیم می‌گردد:

1. Baud Rate : نرخ ارسال داده، که معمولاً به صورت بیت در ثانیه (bps) تنظیم می‌شود.
2. Word Length : طول کلمه داده، که معمولاً 8 یا 9 بیت است.
3. Stop Bits : تعداد بیت‌های توقف، که می‌تواند 1 یا 2 باشد.
4. Parity : نوع توازن، که می‌تواند None، Even یا Odd باشد.
5. Mode : حالت کار، که می‌تواند TX، RX یا TX/RX باشد.
6. Hardware Flow Control : کنترل جریان سخت‌افزاری، که می‌تواند None، RTS، CTS یا RTS/CTS باشد.
7. OverSampling : تنظیمات مربوط به نمونه‌برداری اضافی، که می‌تواند 16 یا 8 باشد.

همچنین حالت‌های ممکن برای USART که می‌توان پیکربندی کرد، آورده شده است:

1. Asynchronous Mode :
   - حالت استاندارد برای ارتباطات سریال.
   - بدون کلاک خارجی، داده‌ها به صورت غیرهمزمان ارسال می‌شوند.
2. Synchronous Mode :
   - شامل کلاک داده است که به گیرنده ارسال می‌شود.
   - داده‌ها به صورت همزمان با کلاک ارسال می‌شوند.
3. Single-Wire(Half-Duplex) Mode :
   - ارتباط دوطرفه با استفاده از یک خط برای ارسال و دریافت و نه به طور همزمان.
   - مناسب برای کاهش تعداد سیم‌ها در ارتباطات.

5. IrDa

حالت IrDa (Infrared Data Association) برای انتقال داده‌ها به صورت بی‌سیم از طریق مادون قرمز است. این حالت به ویژه برای ارتباط بین دستگاه‌ها در فاصله‌های نزدیک و بدون نیاز به کابل استفاده می‌شود.

6. LIN

پروتکل LIN (Local Interconnect Network) یکی از پروتکل‌های ارتباطی اصلی در صنعت خودرو است که برای ارتباط میان واحدهای الکترونیکی (ECU) طراحی شده است. این پروتکل بر مبنای ارتباط سریالی و استفاده از پین‌های UART در میکروکنترلرها مانند STM32F407 عمل می‌کند.

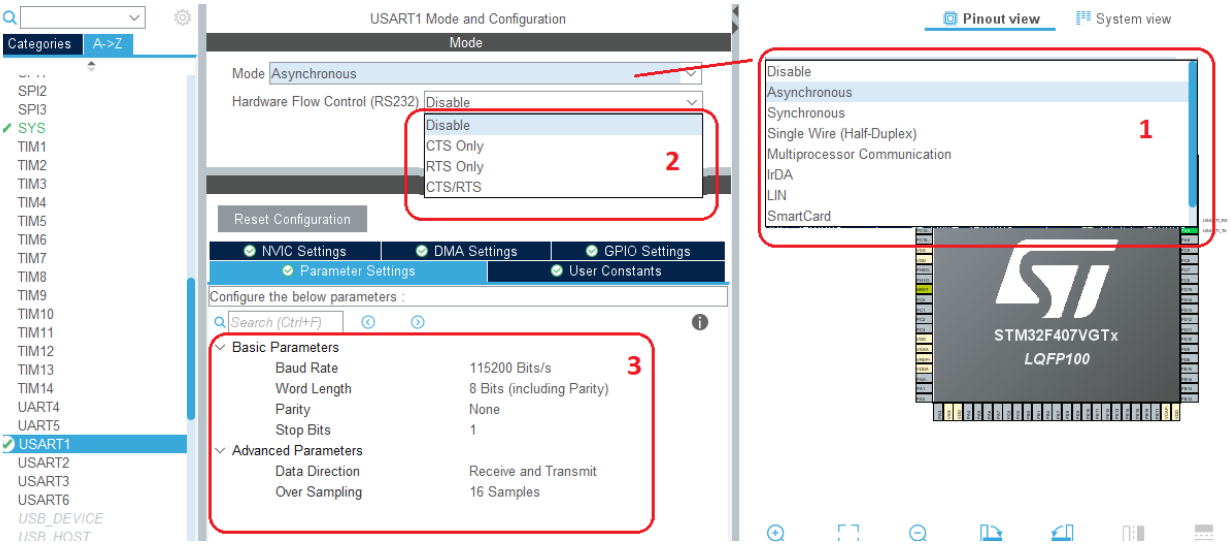
7. SmartCard

Smartcardها، کارت‌های هوشمندی هستند که معمولاً برای ذخیره و پردازش اطلاعات امن و تماس با سیستم‌ها یا دستگاه‌های دیگر استفاده می‌شوند. این قابلیت به ویژه در زمینه‌های تراکنش مالی، شناسایی و کنترل دسترسی کاربرد دارد.

8. ارتباط چندپردازنده‌ای UART STM32

امکان انجام ارتباط چندپردازنده‌ای با USART وجود دارد یکی از USART ها می‌تواند به عنوان مستر عمل کند، و سایرین به عنوان اسلیو باشند.

.



شکل 13-15: نمایی از پارامترهای UART

13.13 کتابخانه STM32 HAL برای بخش UART

کتابخانه STM32 HAL مجموعه‌ای از توابع را برای مدیریت عملیات مختلف UART (ارسال/دریافت داده، مدیریت وقفه‌ها، DMA و غیره) فراهم می‌کند. در زیر توابعی که احتمالاً در پروژه‌های خود به آن‌ها نیاز خواهید داشت، آورده شده است. با این حال، می‌توانید برای فهرست کامل تمام APIهای HAL UART به راهنمای کاربر firmware HAL مراجعه کنید.

13.13.1 توابع Polling HAL برای UART

- HAL_UART_Transmit : این تابع برای ارسال داده‌ها از طریق UART به کار می‌رود.

```
HAL_UART_Transmit(UART_HandleTypeDef * huart, const uint8_t * pData, uint16_t Size,
uint32_t Timeout);
```

پارامترها شامل:

- `huart`: اشاره‌گر به ساختار مدیریت UART.
- `pData`: اشاره‌گر به داده‌هایی که باید ارسال شوند.
- `Size`: تعداد بایت‌هایی که باید ارسال شوند.
- `Timeout`: زمان حداکثر انتظار برای اتمام عملیات ارسال.

- HAL_UART_Receive : این تابع برای دریافت داده‌ها از طریق UART استفاده می‌شود.

```
HAL_UART_Receive(UART_HandleTypeDef * huart, uint8_t * pData, uint16_t Size,
uint32_t Timeout);
```

پارامترها مشابه تابع ارسال هستند.

13.13.2 توابع Interrupt HAL برای UART

- HAL_UART_Transmit_IT : این تابع برای ارسال داده‌ها به صورت غیرهمزمان (با استفاده از وقفه) به کار می‌رود.

```
HAL_UART_Transmit_IT(UART_HandleTypeDef * huart, const uint8_t * pData, uint16_t Size);
```

- HAL_UART_Receive_IT : این تابع برای دریافت داده‌ها به صورت غیرهمزمان (با استفاده از وقفه) استفاده می‌شود.

```
HAL_UART_Receive_IT(UART_HandleTypeDef * huart, uint8_t * pData, uint16_t Size);
```

- Callbacks : این توابع برای مدیریت وقفه‌های اتمام دریافت و ارسال داده‌ها استفاده می‌شوند.

```
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
// Handle UART RX Interrupt Here!
{

void HAL_UART_TxCpltCallback(UART_HandleTypeDef *huart)
{
// Handle UART TX Interrupt Here!
{
```

13.13.3 توابع DMA HAL برای UART

- HAL_UART_Transmit_DMA : این تابع برای ارسال داده‌ها با استفاده از DMA به کار می‌رود.

HAL_UART_Transmit_DMA(UART_HandleTypeDef * huart, const uint8_t * pData, uint16_t Size);

- HAL_UART_Receive_DMA : این تابع برای دریافت داده‌ها با استفاده از DMA استفاده می‌شود.

HAL_UART_Receive_DMA(UART_HandleTypeDef * huart, uint8_t * pData, uint16_t Size);

- Callbacks : این توابع برای مدیریت وقفه‌های اتمام دریافت و ارسال داده‌ها با استفاده از DMA به کار می‌روند.

```
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
}
// Handle UART Rx Complete Interrupt Here!
{

void HAL_UART_TxCpltCallback(UART_HandleTypeDef *huart)
}
// Handle UART Tx Complete Interrupt Here!
{
```

13.14 مراحل تبدیل UART به RS-232

در باکس آزمایشگاه بخشی به نام rs232 وجود دارد که برای تبدیل UART میکروکنترلر به RS-232، نیاز به چند مرحله و برخی قطعات اضافی دارید. در زیر مراحل و تجهیزاتی که برای این تبدیل نیاز دارید، توضیح داده شده است:

1. عملکرد UART:

- UART (Universal Asynchronous Receiver/Transmitter) یک پروتکل ارتباطی سریال است که معمولاً ولتاژهای TTL (معمولاً 0 تا 5 ولت یا 0 تا 3.3 ولت) استفاده می‌کند.

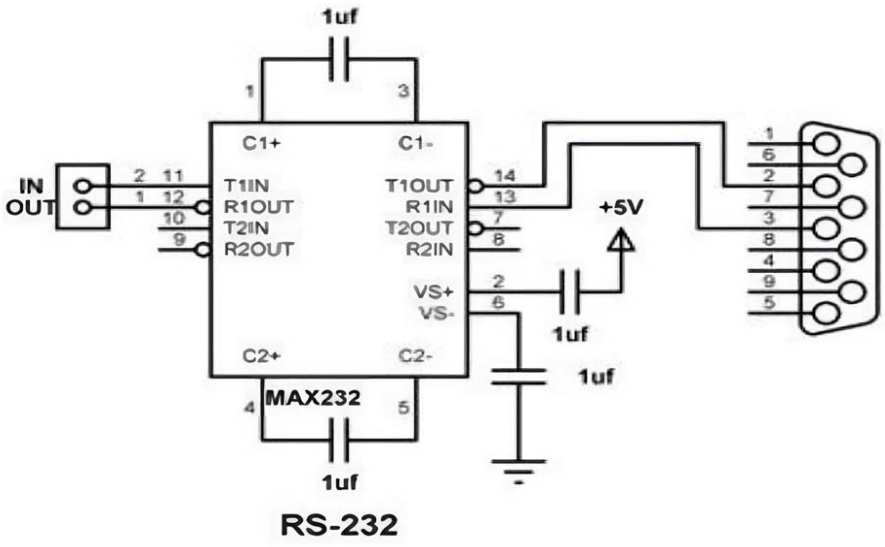
- RS-232، برعکس، از سطوح ولتاژ بالاتر (بین +3 تا +25 ولت و -3 تا -25 ولت) استفاده می‌کند.

2. استفاده از مبدل RS-232:

- برای تبدیل سیگنال‌های UART به RS-232، می‌توانید از مبدل‌هایی مانند IC های مخصوص مانند MAX232، MAX248 و یا TXS0108E مانند شکل 13-16استفاده کنید.

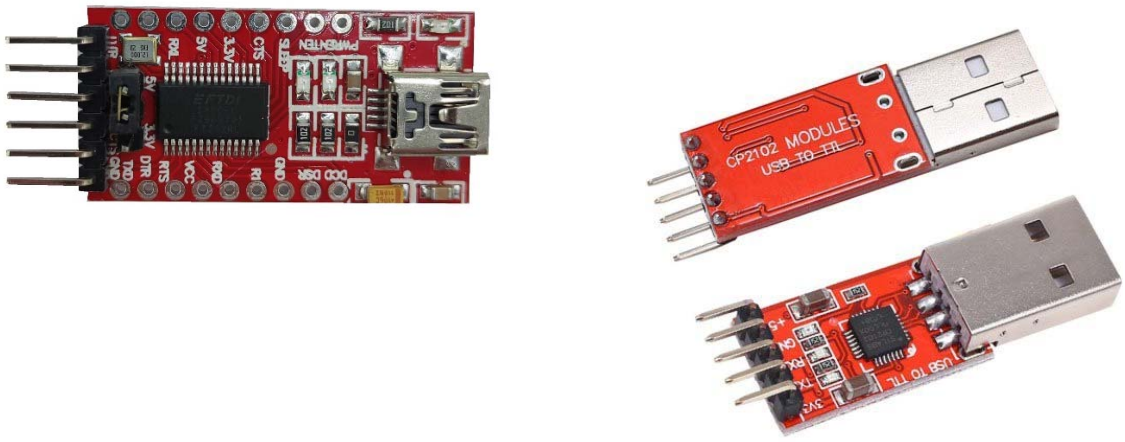
- این چیپ‌ها ولتاژهای TTL را به سطوح ولتاژ RS-232 و بالعکس تبدیل می‌کنند.

-پین‌های خروجی MAX232 را به پورت RS-232 (با کانکتور DB9 یا DB25) متصل کنید



شکل 13-16: اتصالات MAX232

برای اتصال UART به USB در میکروکنترلSTM32F407 ، شما می‌توانید از تبدیل‌کننده USB به UART مانند FTDI یا CP2102 مانند شکل 13-17 استفاده می شود. این ماژول‌ها معمولاً دارای پین‌هایTX ، RX، VCCو GND هستند. پین TX ماژول را به پین RX میکروکنترلر و پین RX ماژول را به پین TX میکروکنترلر STM32متصل کنید.

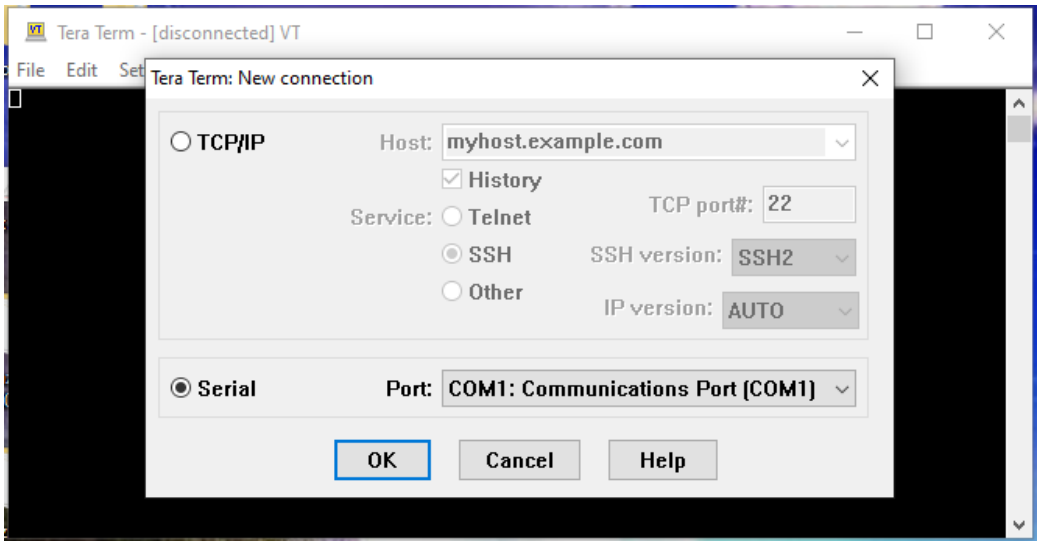


شکل 13-17: نمایی از مبدل UART به USB شامل CP2102 و FDTI232

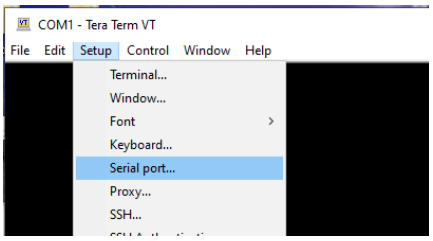
13.14.1 نرم افزار Teraterm

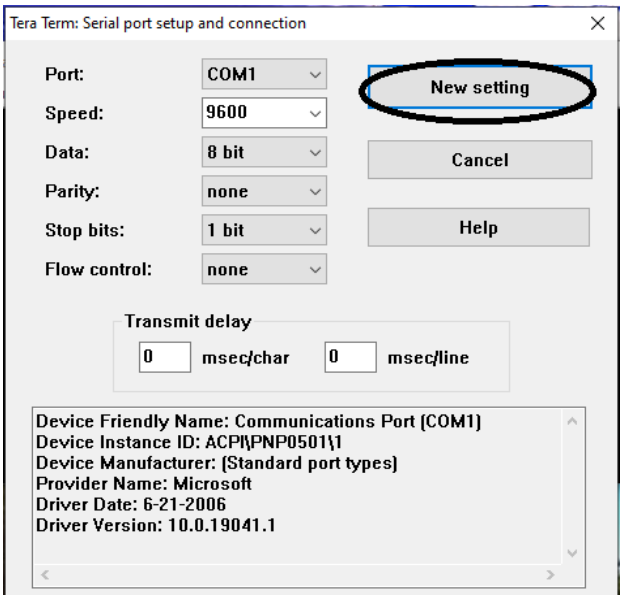
برای ارسال و دریافت داده‌ها، میتوانید با استفاده از کانکتور DB9 پکیج آموزشی را به COM کامپیوتر وصل نمایید. برای نمایش اطلاعات از محیط Teraterm و یا غیره استفاده نمایید. مراحل کار با این نرم‌افزار در شکل 13-18 نشان داده شده است. تنظیمات ارتباط سریال دراین محیط بایستی دقیقا مشابه تنضیمات میکرو شامل نرخ بیت، طول داده و بیت توازن باشد. اگر از مبدلهای UART به USB استفاده می نمایید لازم است درگاه آن را به درستی در teraterm معرفی نمایید.

الف

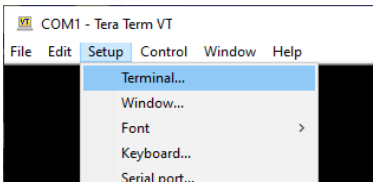


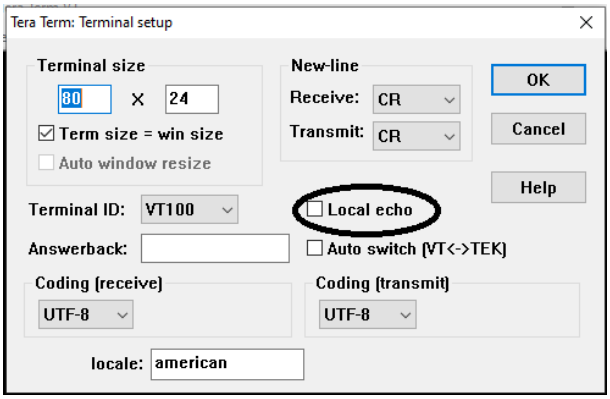
ب





ج





شکل 13-18: نمایی از نحوه کار با محیط Teraterm

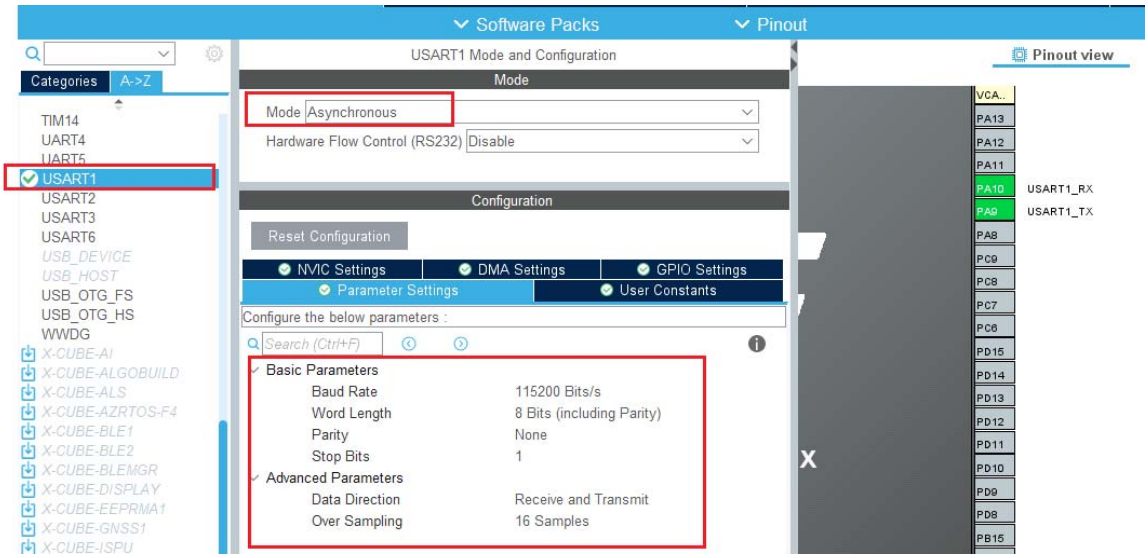
13.15 پروژه ارسال داده با UART

13.15.1 پروژه ارسال داده با UART به روش Polling

این پروژه داده‌های ورودی (۱۲ بایت) را از پورت سریال UART خوانده و آن‌ها را با استفاده از روش "polling" به ترمینال برمی‌گرداند (اکو می‌کند). برای اتصالات پایه ها در باکس آموزشی مطابق عمل نمایید.

Uart (rs232)	Stm32f407
IN	Tx
OUT	Rx

تنظیمات لازم در محیط stm32cubemx مانند شکل 13-19 انجام می شود. تنظیمات کلاک و غیره هم مانند بخش‌های قبل اجرا می‌شود.



شکل 13-19: تنظیمات UART در محسط STM32cubemx در حالت polling

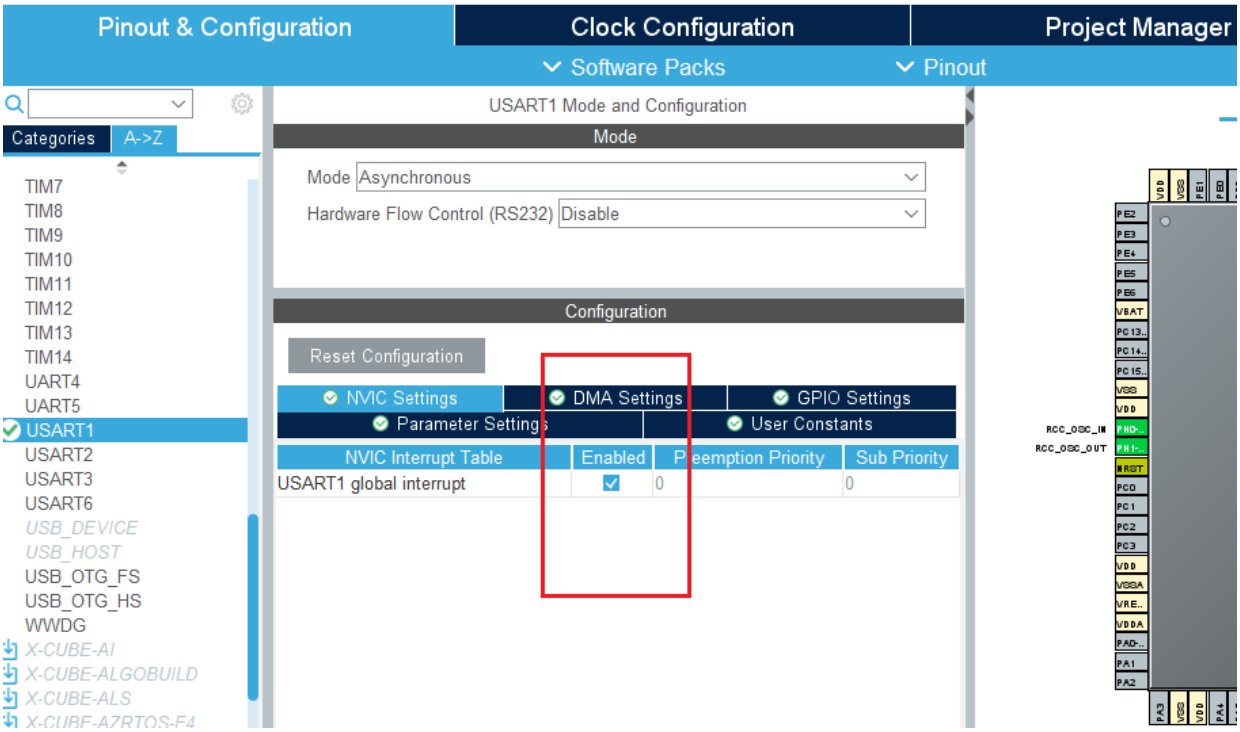
پس از تهیه کد ها تغییرات مشابه را ایجاد نموده و برنامه 13-1 را پروگرام نمایید.

```
#include "main.h"
uint8_t UART1_rxBuffer[12] = {0};

UART_HandleTypeDef huart1;

void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART1_UART_Init(void);

int main(void)
{
  HAL_Init();
 SystemClock_Config();
 MX_GPIO_Init();
 MX_USART1_UART_Init();

 while (1)
 {
  HAL_UART_Receive (&huart1, UART1_rxBuffer, 12, 5000);
  HAL_UART_Transmit(&huart1, UART1_rxBuffer, 12, 100);
 }

}
```

برنامه 13-1: نمونه کد uart برای حالت polling

13.15.2 پروژه ارسال داده با UART به روش وقفه

برای ارسال و دریافت داده با استفاده از UART از طریق وقفه، باید در STM32CubeMX مانند شکل 13-20، این وقفه را از تب کنترل‌کننده NVIC فعال کنید.



شکل 13-20: فعال کردن وقفه UART

پس از تولید کد ها می توانید تغییراتی مشابه برنامه 13-2 در کد ایجاد نمونه و پس از پروگرام کردن نتیجه آن را مشاهده نمایید.

```
#include "main.h"

uint8_t UART1_rxBuffer[12] = {0};
UART_HandleTypeDef huart1;


void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART1_UART_Init(void);

void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
  HAL_UART_Transmit(&huart1, UART1_rxBuffer, 12, 100);
  HAL_UART_Receive_IT(&huart1, UART1_rxBuffer, 12);
}

int main(void)
{
```

```
  HAL_Init();
  SystemClock_Config();

  MX_GPIO_Init();
  MX_USART1_UART_Init();
  HAL_UART_Receive_IT (&huart1, UART1_rxBuffer, 12);
  while (1)
  {

  }

}
```

برنامه 13-2: نمونه برنامه برای ارسال و دریافت داده به روش وقفه

13.15.3 پروژه ارسال داده با UART با استفاده از DMA

DMA (Direct Memory Access) در میکروکنترلرهای STM32F407 یک قابلیت بسیار کارآمد است که به انتقال داده‌ها بین دستگاه‌های مختلف (مانند UART، SPI، ADC و غیره) و حافظه بدون دخالت مستقیم پردازنده کمک می‌کند. این ویژگی باعث کاهش بار روی پردازنده، افزایش سرعت انتقال و بهبود عملکرد سیستم می‌شود.

DMA برای انتقال داده‌ها از منبع (Source) به مقصد (Destination) دائماً به اطلاعات مربوط به هر مرحله کاری نیاز دارد. این مراحل شامل موارد زیر است:

- برقراری ارتباط: انتخابی از دستگاه‌ها برای خواندن یا نوشتن.

- نقل و انتقال: حرکت داده‌ها از محل منبع به مقصد.

- مدیریت وقفه: پس از اتمام انتقال، DMA می‌تواند وقفه‌ای را برای CPU ارسال کند تا آن را آگاه نماید.

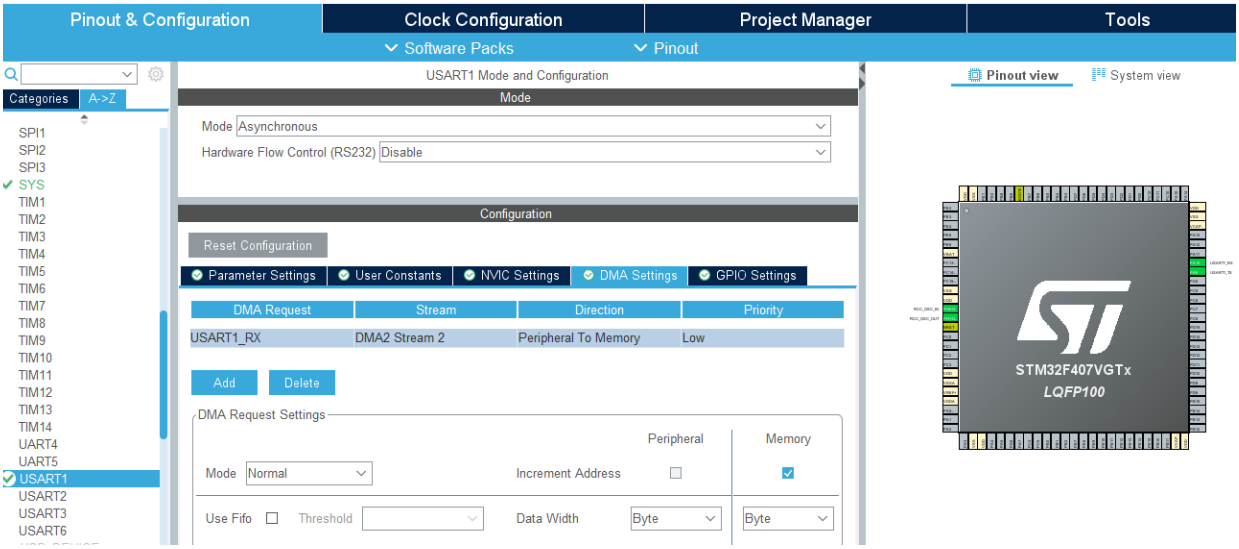
چندین کانال DMA برای نقل و انتقال داده در UART وجود دارد که در هر یک ا ز مدهای ذیل کار میکند:

- Circular Mode: برای استفاده‌های مکرر، جایی که داده‌ها به طوری مداوم منتقل می‌شوند.

- Normal Mode: پس از اتمام یک پروسه انتقال متوقف می‌شود.

تنظیمات لازم برای فعال نمودن DMa در نشان داده شده است.

در STM32cubemx، پروژه جدید ایجاد و تنظیمات مربوط به کلاک و فعال سازی Uart را انجام دهید سپس مطابق شکل 13-21 ذیل کانال DMA را برای ارسال و دریافت فعال نمایید.



شکل 13-21: فعال نمودن DMA در انتقال داده UART

پس از تهیه کدهای پروژه تغییراتی مشابه برنامه 7-1در برنامه ایجاد نموده و نهایتا برنامه را اجرا نمایید.

```
#include "main.h"
#include "stdio.h"
uint8_t UART1_rxBuffer[12] = {0};

UART_HandleTypeDef huart1;
DMA_HandleTypeDef hdma_usart1_rx;


void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_DMA_Init(void);
static void MX_USART1_UART_Init(void);

void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
   HAL_UART_Transmit(&huart1, UART1_rxBuffer, 12, 100);
   HAL_UART_Receive_DMA(&huart1, UART1_rxBuffer, 12);
}

int main(void)
{

  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();
  MX_DMA_Init();
  MX_USART1_UART_Init();
char str[15];
sprintf(str,"Microlab %d\r\n ",1403);
```

```
HAL_UART_Transmit(&huart1, str, 15, 100);

  HAL_UART_Receive_DMA (&huart1, UART1_rxBuffer, 12);

  while (1)
  {

  }

}
```

برنامه 13-3: نمونه برنامه uart با استفاده از DMA در دریافت داده

13.16 توابع کاربردی برای کار با رشته ها

در زبان C، کتابخانه‌های متعددی مانند `string.h` و `stdio.h` برای کار با رشته‌ها (string) و ورودی/خروجی داده‌ها وجود دارند. در ادامه، چندین تابع کاربردی از این کتابخانه‌ها برای کار با رشته‌ها معرفی شده است:

توابع مفید در `string.h`

1. strlen()

- شرح: اندازه (تعداد کاراکترها) یک رشته را برمی‌گرداند.

- نحوه استفاده:

```
size_t len = strlen("Hello, world!"); // len برابر با 13 خواهد بود
```

2. strcpy()

- شرح: یک رشته را به رشته دیگری کپی می‌کند.

- نحوه استفاده:

```
char dest;[50]

strcpy(dest, "Hello"); // dest" اکنون Helloاست "
```

3. strcat()

- شرح: یک رشته را به انتهای رشته دیگر می‌چسباند.

- نحوه استفاده:

```
char str1[50] = "Hello;" ,

strcat(str1, "world!"); // str1" اکنون Hello, worldاست "!
```

4. strcmp()

- شرح: دو رشته را مقایسه می‌کند و نتیجه را برمی‌گرداند.

- نحوه استفاده:

```
int result = strcmp("Hello", "World"); // result" منفی است (زیرا "Hello" کمتر از "World(است "
```

5. strstr()

- شرح: وجود یک زیررشته در یک رشته را جستجو می‌کند و اشاره‌گری به اولین وقوع آن برمی‌گرداند.

- نحوه استفاده:

```
char *result = strstr("Hello, world!", "world"); // result" به worldاشاره می‌کند "!
```

توابع مفید در `stdio.h`

1. printf()

- شرح: رشته‌های فرمت شده را به خروجی نمایش می‌دهد.

- نحوه استفاده:

```
printf("Value: %d\n", 42); // Output: Value: 42
```

2. scanf()

- شرح: ورودی کاربر را با فرمت مشخص شده دریافت می‌کند.

- نحوه استفاده:

```
int num;
scanf("%d", &num) و // کاربر عددی را وارد می‌کند ;numمقدار آن را دریافت می‌کند
```

3. fgets()

- شرح: یک خط رشته را از یک فایل یا ورودی استاندارد می‌خواند.

- نحوه استفاده:

```
char buffer;[100]
```

fgets(buffer, sizeof(buffer), stdin)داده‌ها از ورودی استاندارد خوانده می‌شوند// ;

4. fputs()

- شرح: یک رشته را به یک فایل یا خروجی استاندارد می‌نویسد.

- نحوه استفاده:

fputs("Hello, world!\n", stdout)رشته به خروجی نمایش داده می‌شود// ;

5. sprintf()` و `snprintf()

- شرح: برای فرمت‌دهی و نوشتن رشته به یک آرایه کاراکتری استفاده می‌شود.

- نحوه استفاده:

```
char buffer;[50]
sprintf(buffer, "Value: %d", 42); // sprintf
snprintf(buffer, sizeof(buffer), "Value: %d", 42); // snprintf
```