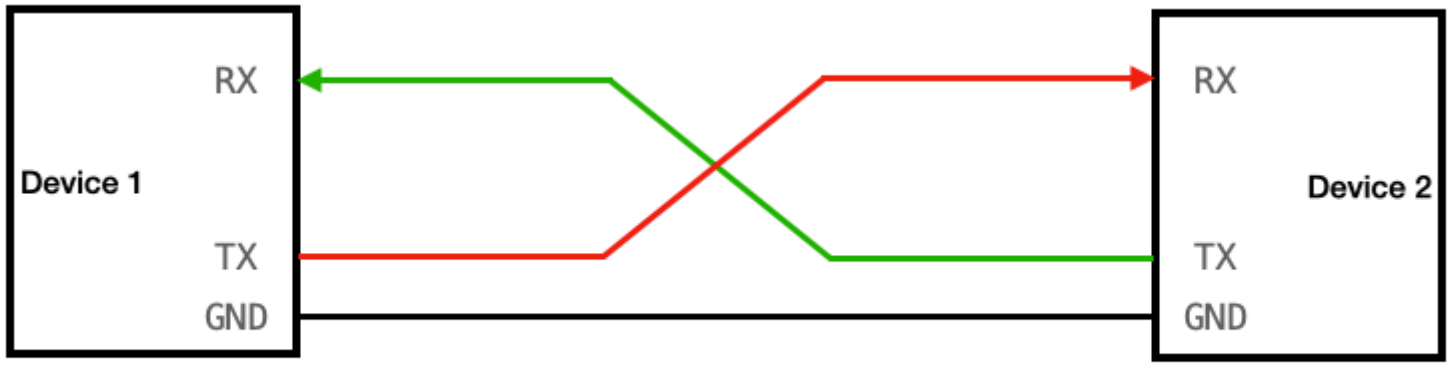
فصل چهارم : Universal synchronous asynchronous receiver transmitter (USART)

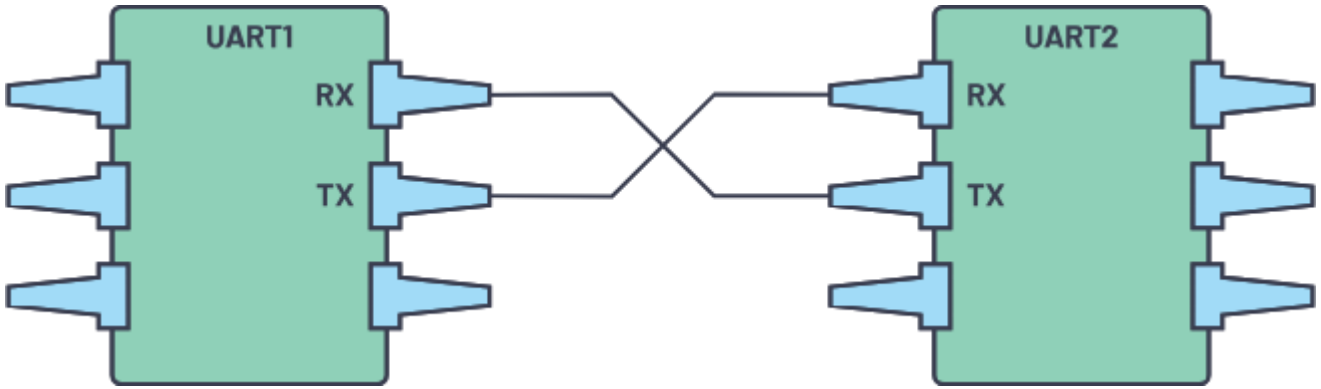
USART

- یکی از پروتکل‌های ارتباطی سریال می‌باشد که کاربرد فراوانی دارد.
- قابلیت ارتباط دو طرفه و تک طرفه را دارد.
- در این پروتکل از یک پایه یا سیم برای ارسال اطلاعات استفاده می‌شود.
- در این پروتکل از یک پایه یا سیم برای دریافت اطلاعات استفاده می‌شود.
- حتما، حتما و حتما بایستی سیگنال‌های زمین در دو سمت ارتباط **USART** به یکدیگر متصل شوند.چرا!؟



وضعیت پایه‌های USART

- به طور معمول پایه RX از نوع ورودی به صورت Pull-up می‌باشد.
- به طور معمول پایه TX از نوع خروجی و به صورت Push-Pull می‌باشد.
- در صورت اتصال اشتباه پایه‌های چه اتفاقی رخ می‌دهد!؟



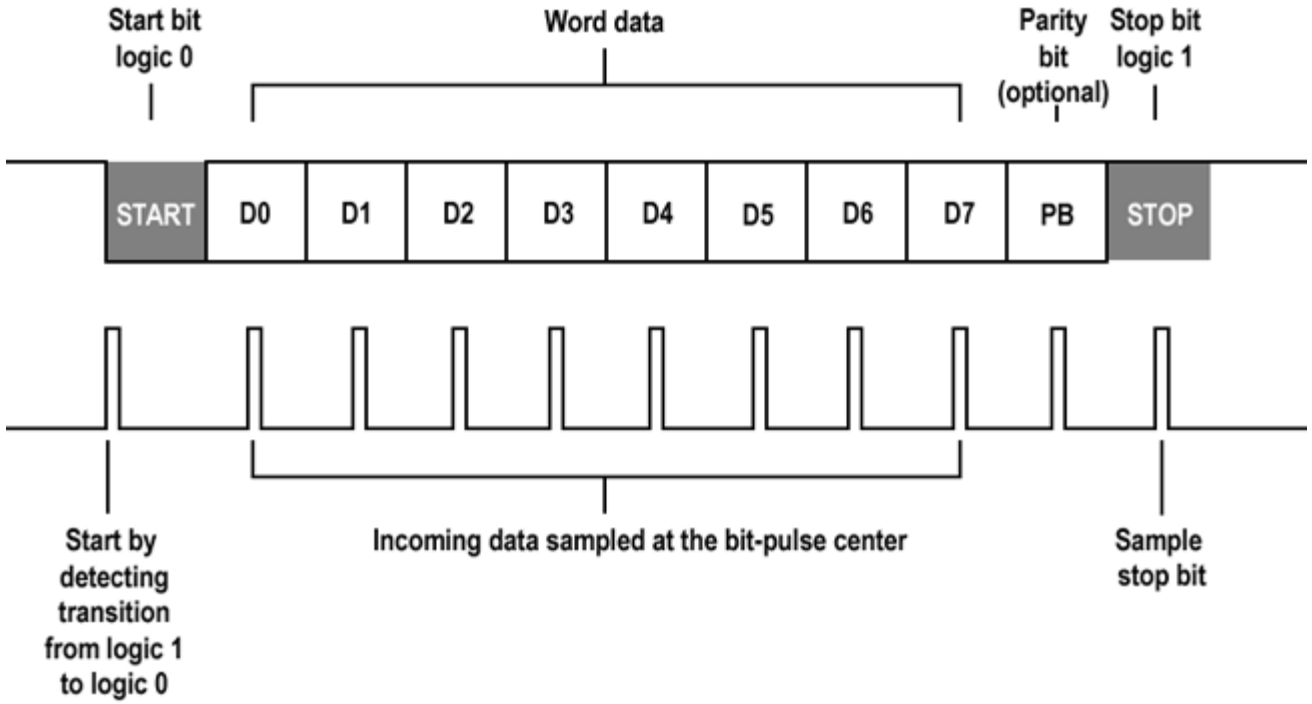
معرفی بیت‌های هر بسته داده در پروتکل USART

- در هر بسته از داده پروتکل USART، بین 5 تا 9 بیت دیتا وجود دارد (8 بیت یا 1 بایت رایج‌ترین تعداد بیت در یک بسته می‌باشد.
- یک بیت به عنوان Start بیت...
- یک یا دو بیت به عنوان Stop بیت... (تفاوت 1 با 2)!؟
- یک بیت به عنوان بیت Parity (Optional)
  - Even parity (متداول)
  - Odd Parity
- حداقل و حداکثر تعداد بیت در یک بسته داده دیتا!؟

خطاهای پروتکل USART/USART

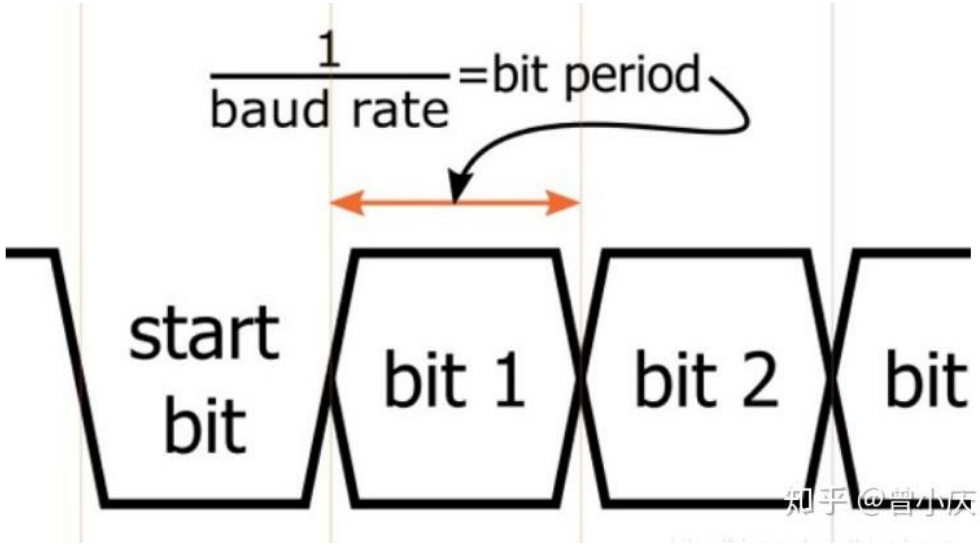
- Framing Error : این خطا هنگامی رخ می‌دهد که گیرنده در بازه زمانی که انتظار بیت Stop را دارد، آن را دریافت نمی‌کند. به عبارت دیگر بیت Stop در زمان خود توسط گیرنده دیده نمی‌شود.
- Parity Error : این خطا هنگامی رخ می‌دهد که بیت Parity مخالف با تنظیمات باشد. برای مثال اگر تنظیمات بیت Parity از نوع Even باشد، بایستی اگر تعداد بیت‌های ارسالی در یک بسته، زوج باشد، مقدار 0 را نشان دهد. در غیر اینصورت خطای Parity Error رخ می‌دهد.
- Overrun Error : این خطا هنگامی رخ می‌دهد که دیتای قبلی دریافتی توسط گیرنده، هنوز از طریق کاربر خوانده نشده است و دیتای بعدی دریافت می‌شود! (در این شرایط دیتای قبلی از بین می‌رود).

نحوه انتقال اطلاعات از مبدا به مقصد (UART)



- در حالتی که انتقال اطلاعاتی وجود ندارد، پایه TX در وضعیت ’1‘ یا High قرار دارد.
- در شروع ارسال اطلاعات، پایه TX به وضعیت ’0‘ یا Low می‌رود.
- سپس بیت‌های دیتا به صورت سریال و در فاصله زمانی معین، با شروع از بیت کم‌ارزش، ارسال می‌شوند (اگر بیت Parity استفاده شده باشد، حداکثر تعداد بیت دیتا برابر با 8 بیت می‌باشد، در غیر اینصورت، تعداد 9 بیت نیز قابلیت ارسال دارد.)
- در صورتی که بیت Parity فعال شده باشد، پس از بیت‌های دیتا، یک بیت Parity ارسال می‌شود.
- در انتها نیز 1 یا 2 بیت Stop و با منطق ’1‘ ارسال می‌گردد.
- در تمامی مراحل نمونه برداری سعی می‌شود تا اخذ دیتا در وسط هر بیت انجام شود (بجز بیت Start)

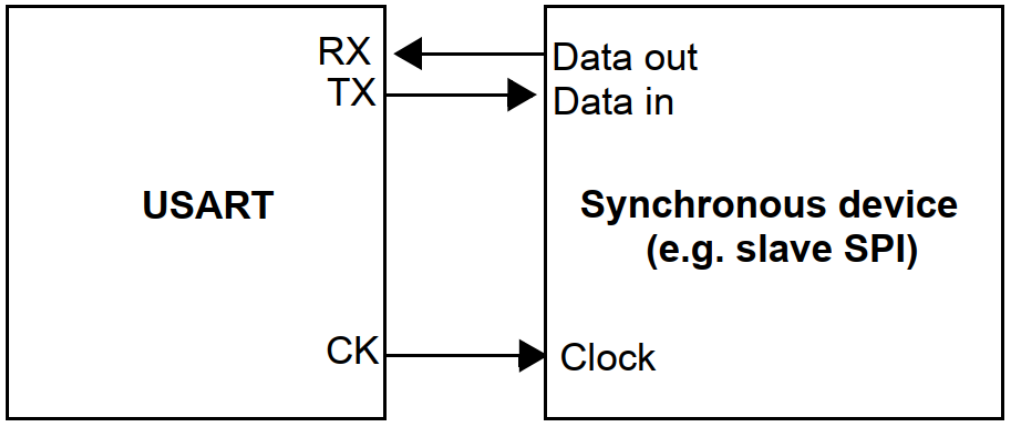
نحوه انتقال اطلاعات از مبدا به مقصد (UART)

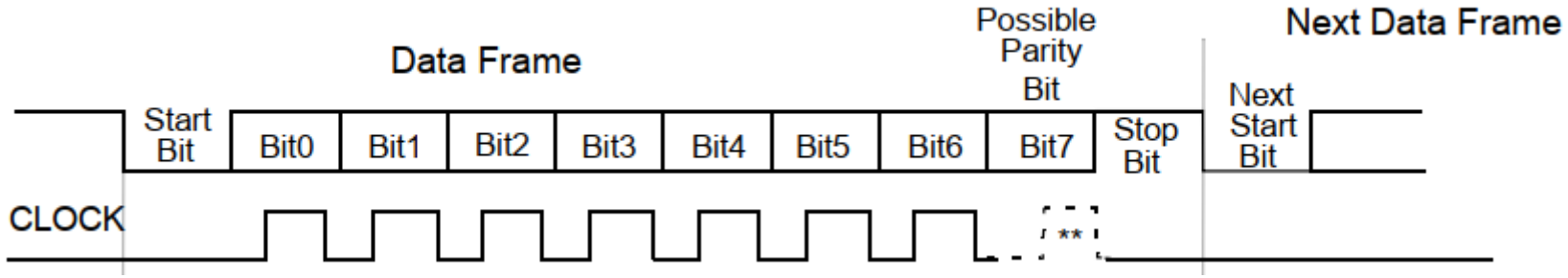


- عدم وجود سیگنال Clock در ساختار UART
- به منظور ارسال و دریافت صحیح اطلاعات، بایستی بازه زمانی ’0‘ یا ’1‘ بودن هر بیت برای طرفین رابطه مشخص باشد.
- به تعداد بیت‌های انتقالی در 1ثانیه، Baud rate گفته می‌شود که بیانگر سرعت انتقال اطلاعات نیز می‌باشد.
- Baud rate های رایج عبارتند از 4800, 9600, 115200 و ...
- در پردازنده‌های مختلف قابلیت تعریف Baud rate های خاص و اعشاری نیز وجود دارد (نه لزوما).
- معمولا در پردازنده‌های مختلف، از تکنیک Over sampling به منظور افزایش دقت در تشخیص داده‌ها استفاده می‌شود.

نحوه انتقال اطلاعات از مبدا به مقصد (USART)

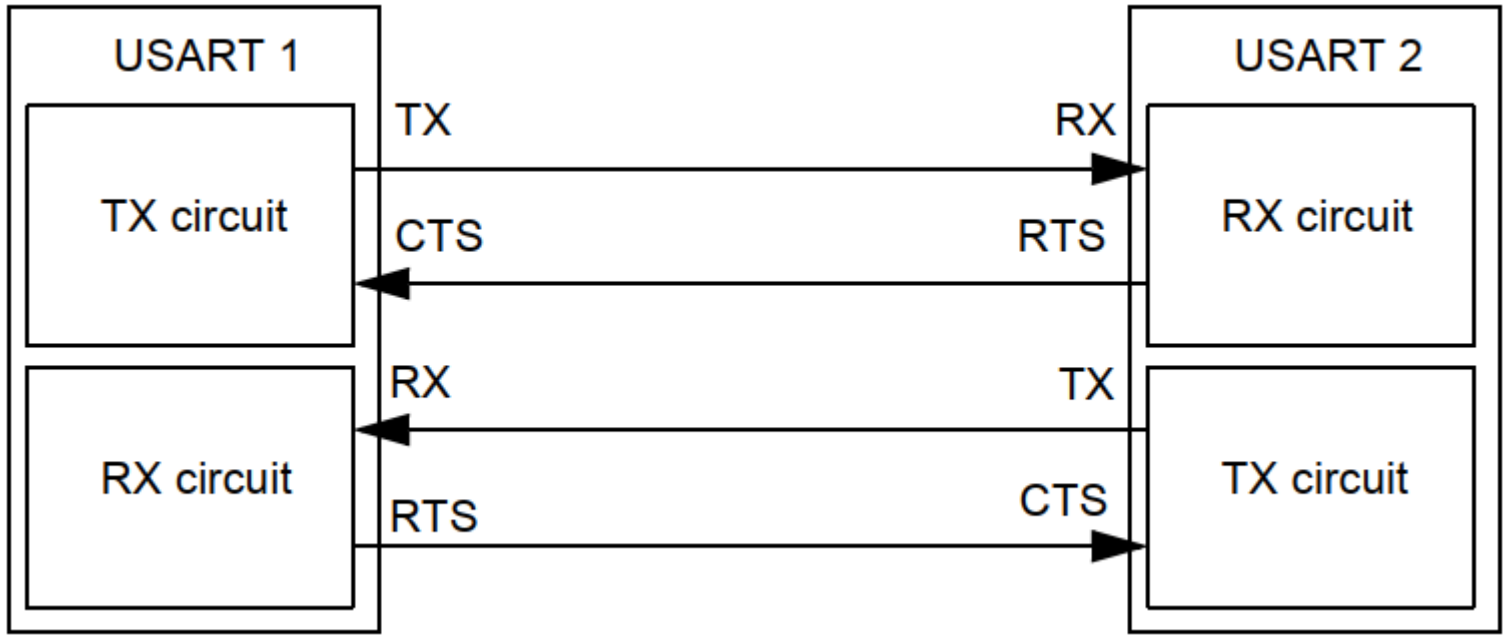
- وجود سیگنال Clock علاوه بر سیگنال‌های دیتا به منظور همگام سازی فرستنده و گیرنده.
- با توجه به وجود سیگنال Clock، گیرنده همواره می‌داند در چه لحظه‌ای اطلاعات را بخواند. از این رو دیگر نیازی به تنظیم پارامتر Baud rate پیش از شروع رابطه برای طرفین نمی‌باشد...
- پروتکل USART، نیازمند مدار پیچیده‌تر، مصرف توان بیشتر و هزینه‌بر تری می‌باشد. بنابراین در برخی سیستم‌ها فقط پروتکل UART وجود دارد.
- سرعت پروتکل USART بیشتر است. چرا؟!



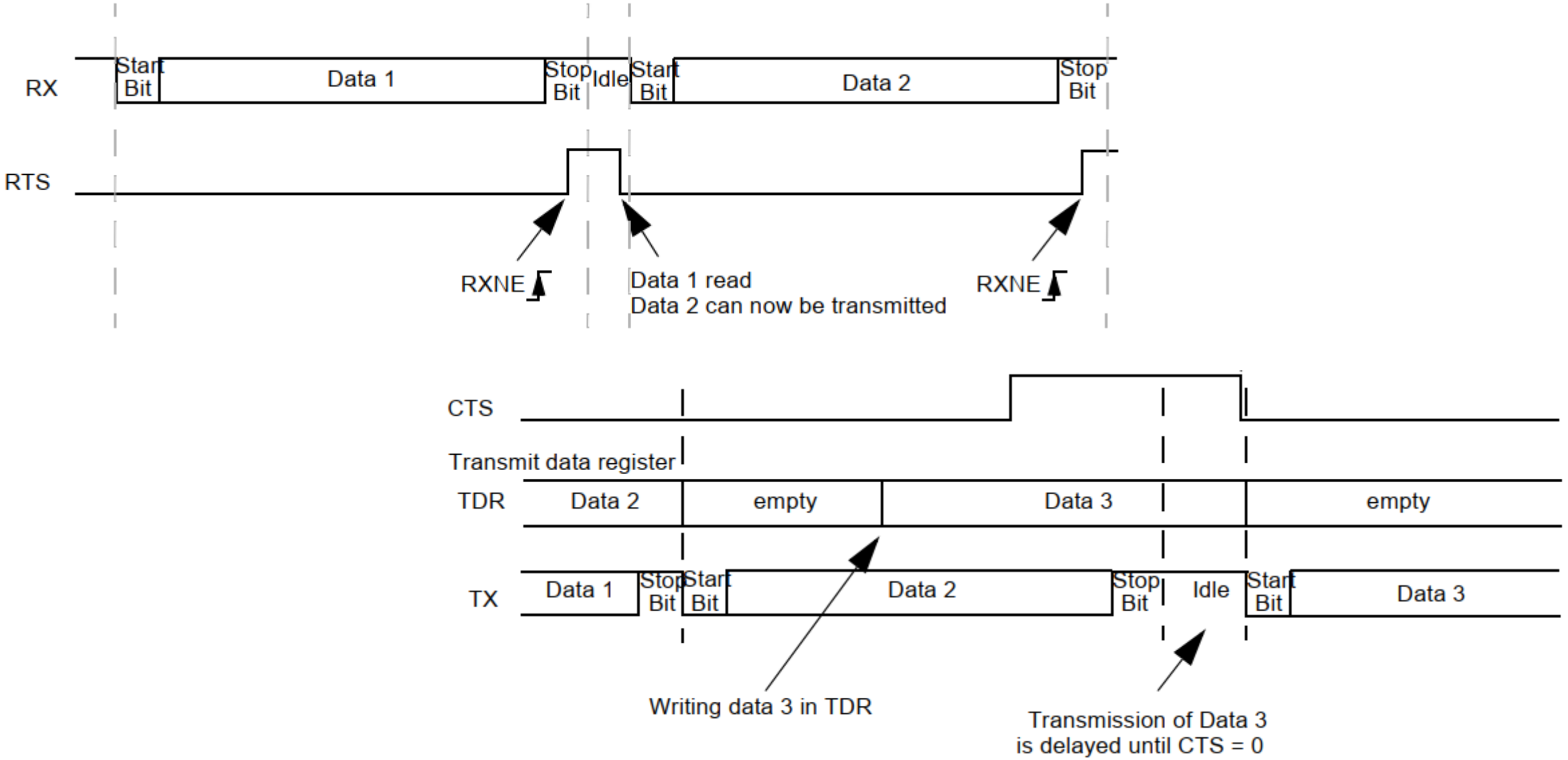


کنترل جریان داده سخت‌افزاری

- راهکار کنترل جریان داده نرم‌افزاری با استفاده از تنظیم پارامتر Baud rate صورت می‌گرفت. اما به منظور ایجاد کنترل سخت‌افزاری، بایستی مداری (شامل پایه‌هایی جدید) به این پروتکل اضافه شود که به شرح زیر می‌باشند :
- CTS (Clear To Send) : در صورتی که مقدار آن 1 باشد، پس از ارسال دیتای کنونی، عملیات ارسال اطلاعات متوقف می‌شود.
- RTS (Request To Send) : در صورت که مقدار آن 0 باشد، به این معناست که گیرنده آمادگی دریافت دیتای بعدی را دارد.

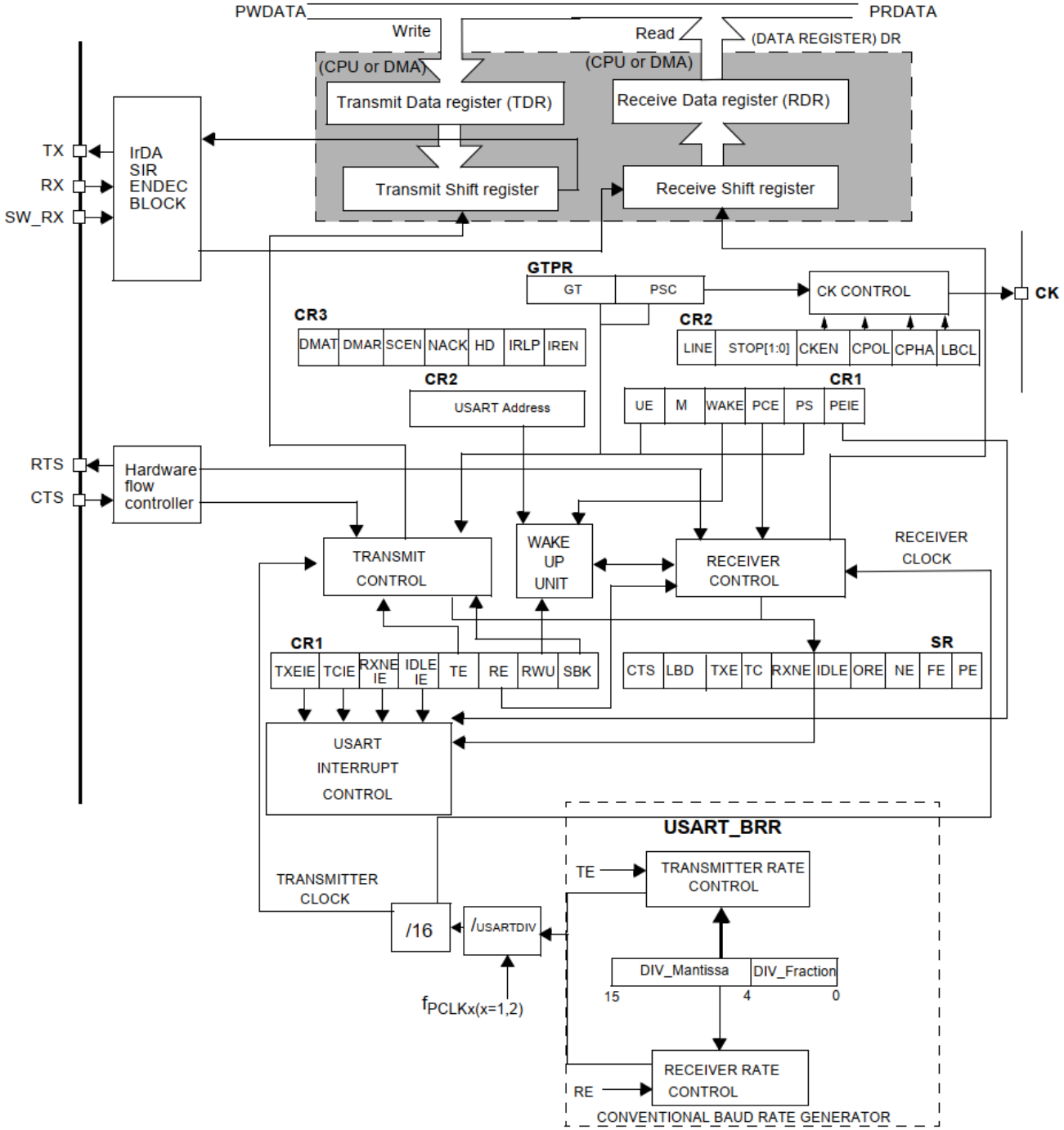


عملکرد پایه‌های CTS و RTS



USART در میکروکنترولر های STM32F1

- مدار تولید clock
- رجیسترهای TDR و RDR
- پایه‌های RTS و CTS



نحوه ارسال و دریافت کاراکترها

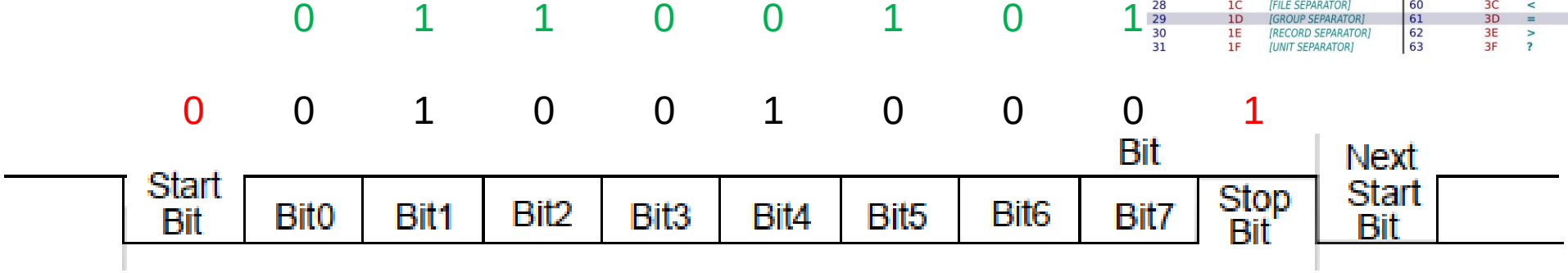
- **فرض کنید قصد داریم عبارت Hello World را به کمک پروتکل UART یا USART ارسال نماییم.**

H = 72decimal = 48hex = 0x48 = 0100 1000

e = 101decimal = 65hex = 0x65 = 0110 0101

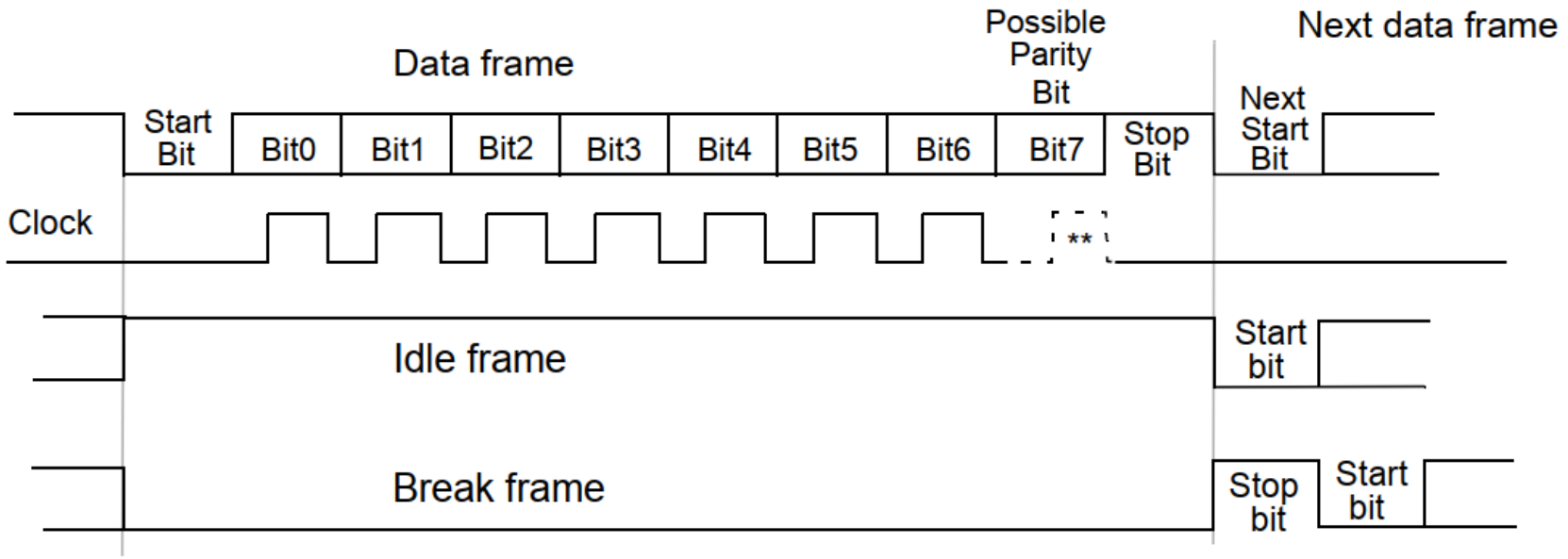
ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[END OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]



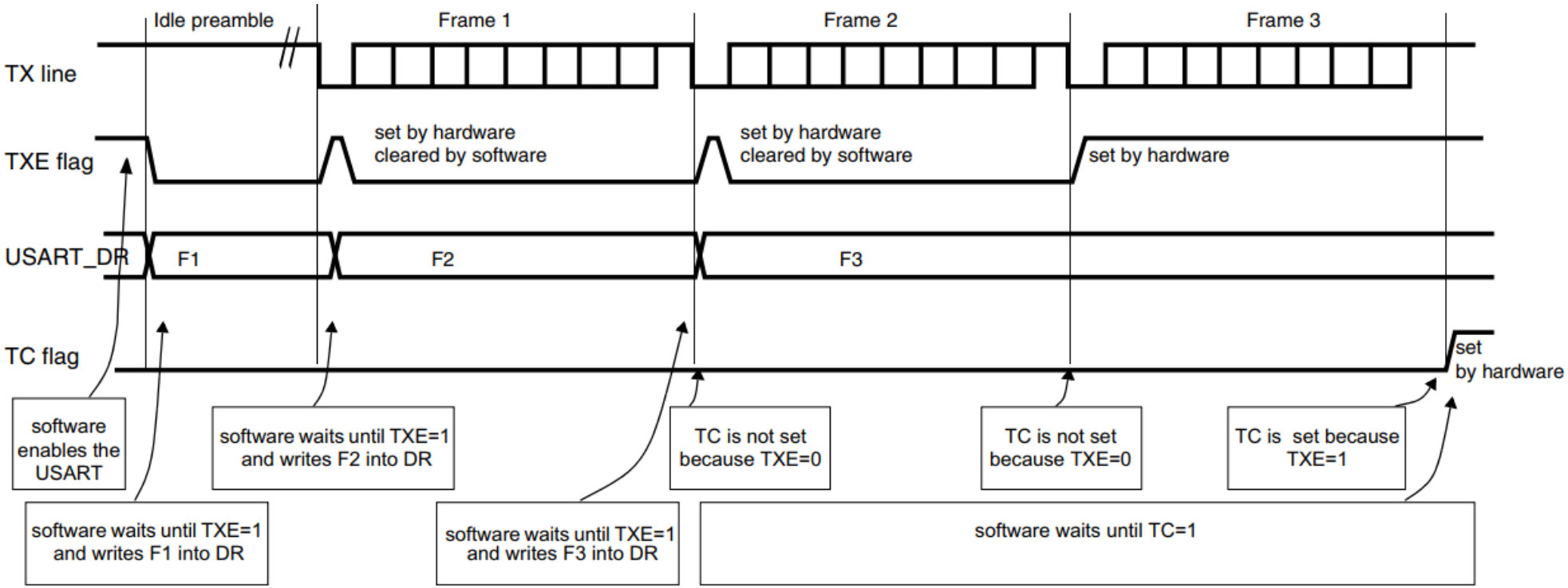
Idle Frame & Break Frame

- Idle Frame : فرمی از اطلاعات که شامل زمان ارسال 1 کاراکتر با فرض تمامی بیت‌های 1 به همراه بیت Start (0) فریم بعد باشد.
- Break Frame : فرمی از اطلاعات که شامل زمان 0 بودن بیش از یک کاراکتر باشد!
- کاربرد Idle Frame!؟
- کاربرد Break Frame!؟



TXE Flag & TC Flag

- در پروتکل UART پیش از ارسال اولین بسته دیتا، یک Idle ارسال می‌گردد.
- TXE Flag!?
- TC Flag?!

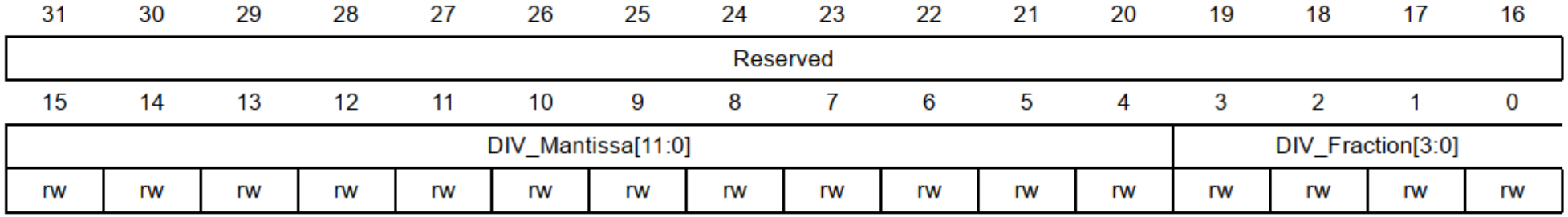


محاسبه Baud rate در پردازنده‌های STM32F1

- قابلیت تعریف Baud rate های اعشاری نیز در این سری از میکروکنترولرها وجود دارد. به منظور اجرای چنین عملیاتی، رجیستری به نام USART_BRR وجود دارد که 4 بیت نخست آن مربوط به قسمت اعشاری و 12 بیت بعد، مربوط به قسمت صحیح پارامتر USARTDIV می‌باشد.

$$\text{Tx/ Rx baud} = \frac{f_{CK}}{(16*\text{USARTDIV})}$$

- مثال : محاسبه مقدار رجیستر USART_BRR برای Baud rate = 9600 bps
- سوال : اگر DIV_Mantissa = 0d27 و DIV_Fraction = 0d12 باشد، سرعت انتقال اطلاعات چند است!؟
- سوال : با فرض UARTDIV = 0d25.62، مقدار رجیستر USART_BRR را بدست آورید؟
- سوال : با فرض USARTDIV = 50.99، مقدار رجیستر USART_BRR را بدست آورید؟!



توابع HAL کاربردی برای پروتکل UART

- تابع ارسال اطلاعات بصورت polling (سرکشی) HAL_UART_Transmit نام دارد.

آرایه‌ای از داده‌ها که قصد داریم ارسال نماییم
انتخاب شماره UART
مثال : &huart1
نام تابع
مقداری که تابع بر می‌گرداند

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

تعداد بایت‌هایی که قصد داریم ارسال نماییم

توابع HAL کاربردی برای پروتکل UART

- تابع دریافت اطلاعات بصورت polling (سرکشی) HAL_UART_Receive نام دارد.

مقداری که تابع بر می‌گرداند — نام تابع — انتخاب شماره UART (مثال : &huart1) — آرایه‌ای که قصد داریم داده‌های دریافتی را در آن ذخیره نماییم — تعداد بایت‌هایی که قرار است دریافت نماییم

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

توابع HAL کاربردی برای پروتکل UART

- تابع ارسال اطلاعات بصورت وقفه HAL_UART_Transmit_IT نام دارد.
- پس از ارسال موفق تمامی داده‌ها یا نیمی از آن‌ها، وقفه‌ای رخ می‌دهد. (در صورت وقوع خطا نیز، وقفه‌ای رخ می‌دهد.)

مقداری که تابع بر می‌گرداند — نام تابع — انتخاب شماره UART مثال : &huart1 — آرایه‌ای از داده‌ها که قصد داریم ارسال نماییم — تعداد بایت‌هایی که قصد داریم ارسال نماییم

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

```
void HAL_UART_TxCpltCallback (UART_HandleTypeDef *huart){
}
```

```
void HAL_UART_TxHalfCpltCallback (UART_HandleTypeDef *huart){
}
```

```
void HAL_UART_ErrorCallback (UART_HandleTypeDef * huart){
}
```

توابع HAL کاربردی برای پروتکل UART

- تابع دریافت اطلاعات بصورت وقفه HAL_UART_Receive_IT نام دارد.
- پس از دریافت موفق تمامی داده‌ها یا نیمی از آن‌ها، وقفه‌ای رخ می‌دهد. (در صورت وقوع خطا نیز، وقفه‌ای رخ می‌دهد.)

مقداری که تابع بر می‌گرداند — نام تابع — انتخاب شماره UART مثال : &huart1 — آرایه‌ای از داده‌ها که قصد داریم ارسال نماییم

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

تعداد بایت‌هایی که قصد داریم ارسال نماییم

```
void HAL_UART_RxCpltCallback (UART_HandleTypeDef *huart){
}
```

```
void HAL_UART_RxHalfCpltCallback (UART_HandleTypeDef *huart){
}
```

```
void HAL_UART_ErrorCallback (UART_HandleTypeDef * huart){
}
```