UFPEL

Microcontroladores

Aula 9 – Comunicação serial

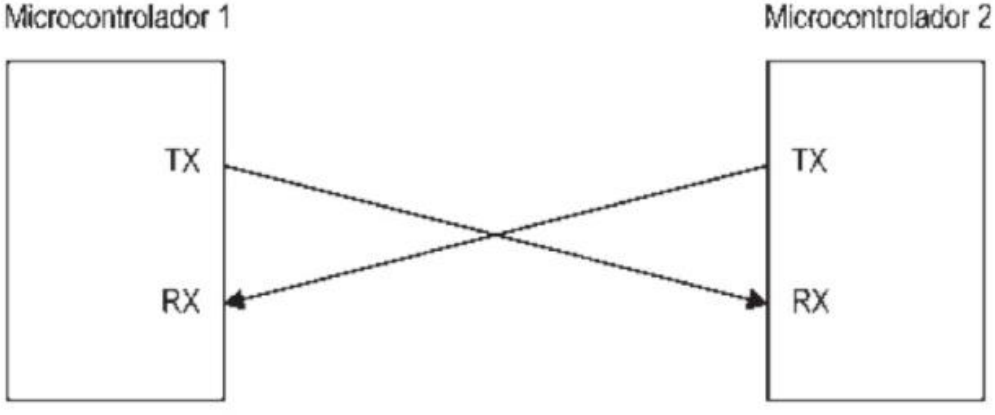
Prof. Dr. Alan Carlos Junior Rossetto

Introdução

- O periférico de comunicação serial é um componente de extrema importância em computadores, pois possibilita a comunicação bidirecional entre dispositivos utilizando apenas dois fios;
- Permite também, por meio de um modem (modulador/demodulador), transferir e receber dados via um sistema de comunicação tradicional (e.g., telefonia), sendo vital nas aplicações de comunicação cotidianas;
- Recebe o nome de serial o tipo de comunicação na qual a informação (geralmente em *bytes*) é particionada (geralmente em *bits*), transmitida em uma sequência pré-estabelecida por uma única via, recebida e recomposta na sua forma original;
- Esta se opõe conceitualmente à transmissão em paralelo, onde toda a informação é transmitida de uma única vez, mas ao custo de uma conexão com mais vias;
- Alguns exemplos de comunicação serial são os protocolos *ethernet*, I2C, SPI, UART, CAN, USB, entre outros.

Introdução

- Além da informação propriamente dita (codificada em ASCII), também são transmitidos pelo canal serial alguns *bits* para o controle da transmissão / recepção. Estes *bits* sinalizam ao receptor / transmissor, por exemplo, o ponto de início e o ponto de término da transmissão / recepção da informação;
- Um requisito fundamental é que a frequência utilizada durante a comunicação seja a mesma tanto para o transmissor quanto para o receptor. Esta frequência é chamada de *baudrate*;
- No contexto do µC 8051, a comunicação serial em questão é a *universal asynchronous receiver / transmitter* ou UART, podendo também ser chamada de *serial communication interface* (SCI);



Introdução

- No contexto do µC 8051, a comunicação serial em questão é a *universal asynchronous receiver / transmitter* ou UART, podendo também ser chamada de *serial communication interface* (SCI);
- Ela é do tipo *full duplex*, i.e., o µC 8051 pode transmitir informação por uma via (TX) enquanto recebe informação por outra (RX), e fazer isso de maneira síncrona ou assíncrona;
- Além da informação propriamente dita (codificada em ASCII), também são transmitidos pelo canal serial alguns bits para o controle da transmissão / recepção:
  - *Bit* inicial (*start bit*): sincroniza o transmissor e receptor indicando que a transmissão do dado vai começar através de uma transição 1 → 0;
  - *Bit* de paridade: indica a paridade do *byte* transmitido / recebido, sendo 0 para número par de ‘1’s e 1 para número ímpar de ‘1’s;
  - *Bit* de parada (*stop bit*): indica que a transmissão foi concluída.

SFRs envolvidos

- A operação do canal serial do µC 8051 faz uso dos seguintes registradores de função especial:
  - **SBUF** ou *serial buffer*: *buffer* de dados recebidos (**SBUF** *in*) ou a serem transmitidos (**SBUF** *out*);
  - **SCON** ou *serial control*: registrador de controle do modo de operação do canal serial;
  - **PCON** ou *power control*: registrador de gerenciamento energético, mas que é utilizado para modificar a taxa de transmissão de dados através da flag **SMOD** (*serial mode*);

Fluxo de comunicação

- O fluxo de comunicação serial no µC 8051 se dá da seguinte forma:
  - Para o envio:
    - Habilitação da *flag* **`REN`**;
    - Limpeza da *flag* **`TI`**;
    - Escrita no **`SBUF`**;
    - Transmissão do dado;
  - Para a recepção:
    - Habilitação da *flag* **`REN`**;
    - Limpeza da *flag* **`RI`**;
    - Recebimento do dado no **`SBUF`**.

Modos de operação

- O modo de operação do canal serial do µC 8051 é definido pelo SFR **SCON**. Nele está contido também o 9º bit da informação (**RB8**), o qual indica a paridade do *byte* recebido, além de outras *flags* que indicam a situação de operação desta interface.

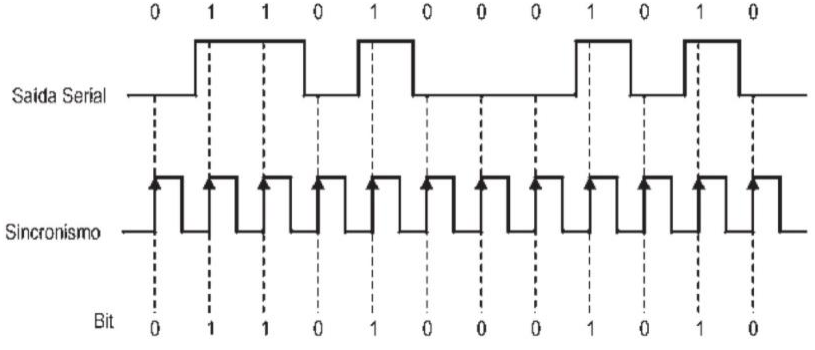
bit	7	6	5	4	3	2	1	0
(SCON) =	SM0	SM1	SM2	REN	TB8	RB8	TI	RI

SM0	SM1	Modo	Descrição	*Baud rate*
0	0	0	Registrador de deslocamento	$f_{osc.}/12$
0	1	1	UART de 8 *bits*	programável pelo *timer*/contador 1
1	0	2	UART de 9 *bits*	$f_{osc.}/64$ ou $f_{osc.}/32$
1	1	3	UART de 9 *bits*	programável pelo *timer*/contador 1

Símbolo	Nome e significado
SM2	Habilita a característica de comunicação entre múltiplos microcontroladores no modo 2 e 3, e caso SM2=1, RI não será ativado se o nono *bit* de dado recebido for igual a 0. No modo 1, se SM2=1, RI não será ativado se um *stop bit* válido não for recebido. No modo 0, deve ser 0.
REN	*Bit* habitador da recepção serial. Setado/limpado por *software* para habilitar ou desabilitar a recepção serial.
TB8	É o nono *bit* de dado que será transmitido no modo 2 e 3. Setado ou limpo por *software*. Geralmente ele é usado para transmitir a paridade do *byte*.
RB8	No modo 2 e 3, é o nono *bit*, geralmente usado quando se transmite a paridade do *byte*. No modo 1, se SM2=0, RB8 é o *stop bit* que foi recebido. No modo 0, RB8 não é usado.
TI	É o *flag* de interrupção de transmissão. Setado por *hardware* no final do tempo do 8º *bit* transmitido no modo 0 ou no início do *stop bit* em outros modos, em qualquer tipo de transmissão serial. Deve ser limpo por *software*. Ele sinaliza que um dado serial foi transmitido pelo $SBUF_{out}$.
RI	É o *flag* de interrupção de recepção. Setado por *hardware* no final do tempo do 8º bit no modo 0 ou na metade do tempo do *stop bit* em outros modos, em qualquer tipo de recepção serial. Deve ser limpo por *software*. Ele sinaliza que um dado serial acabou de ser montado no $SBUF_{in}$.

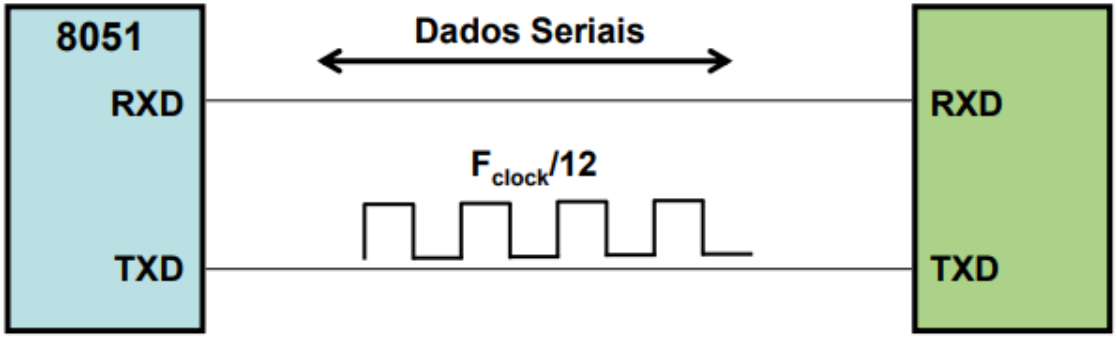
Modos de operação

- No **Modo 0**, os dados serializados entram e saem pelo pino RX (**`P3.0`**), sendo o primeiro *bit* a ser transmitido ou recebido é o menos significativo (LSB);
- O pino TX (**`P3.1`**) é usado para a transmissão do sinal de *clock* de transmissão (i.e., *baudrate*). Nesse modo, o *baudrate* é fixo em 1/12 da frequência do oscilador do µC;
- Neste modo, a comunicação é dita **síncrona**;



- Para que ocorra a recepção, é preciso ter **`REN`** = 1 e **`RI`** = 0;
- A transmissão é iniciada por qualquer instrução que usa **`SBUF`** como um registrador de destino e finalizada quando **`TI`** = 1.

Modo 0



Início de transmissão: SBUF ← Dados (SW)

TB8 ← 1 (HW)

Fim de transmissão: TI ← 1 (HW)

Início de recepção: REN ← 1 (SW)

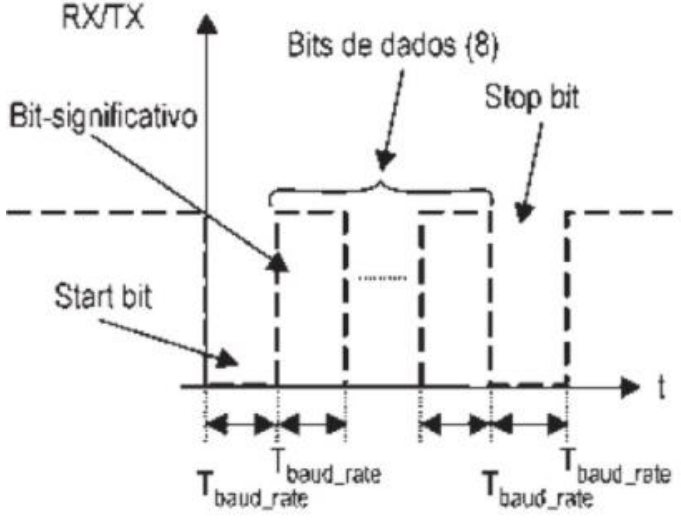
RI ← 0 (SW)

Fim de recepção: RI ← 1 (HW)

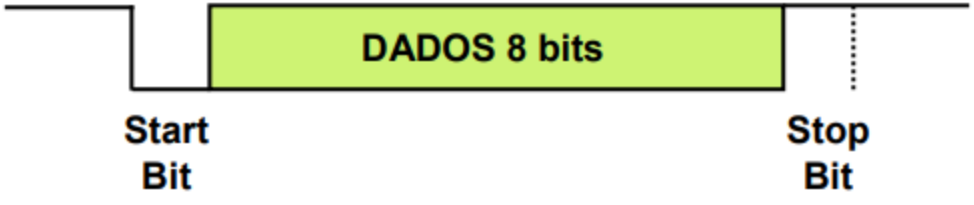
TB8, TI, REN e RI são bits do SCON

Modos de operação

- No **Modo 1**, a comunicação é dita **assíncrona**, e o *baudrate* é programável, como será visto em seguida. Ademais, a informação transmitida pelo TX (`P3.1`) e recebida pelo RX (`P3.0`) é composta de 10 *bits*, a saber:
  - Um *bit* de sincronização (*start bit*) em 0 lógico, uma vez que o sinal em TX inicialmente está em 1 lógico;
  - Oito *bits* de dados, sendo o primeiro o LSB e o último o MSB;
  - Um *bit* de sincronização (*stop bit*) em 0 lógico sinalizando o fim da transmissão pelo TX ou a recepção pelo RX.



Modo 1



Início de transmissão: SBUF ← Dados (SW)

TB8 ← 1 (HW)

Fim de transmissão: TI ← 1 (HW)

Início de recepção: REN ← 1 (SW)

RXD ↓ (HW)

Fim de Recepção: RB8 ← 1 (Stop Bit)

RI ← 1 (HW)

TB8, RB8, TI, REN e RI são bits do SCON

Modos de operação

- No **Modo 2**, a comunicação também é **assíncrona**, porém o *baudrate* é programável somente para 1/32 ou 1/64 da frequência de *clock*.
- A informação transmitida pelo TX (**`P3.1`**) e recebida pelo RX (**`P3.0`**) é composta de 11 *bits*, a saber:
  - Um *bit* de sincronização (*start bit*) em 0 lógico;
  - Oito *bits* de dados, sendo o primeiro o LSB e o último o MSB;
  - Um *bit* de controle (tipicamente a paridade do byte);
  - Um *bit* de sincronização (*stop bit*) em 0 lógico.
- A recepção se inicia quando é detectada uma borda de descida em RX, e finalizada quando se detecta o *stop bit*. Neste momento é setada a *flag* RI;
- Na transmissão com 9 *bits* de informação, o 9º bit fica armazenado no **`RB8`** do registrador **`SCON`**.

Modo 2

Início de transmissão: SBUF ← Dados (SW)

TB8 ← Paridade, segundo stop bit (SW)

Nono bit ← TB8 (HW)

Fim de transmissão: TI ← 1 (HW)

Início de recepção: REN ← 1 (SW)

RXD ↓ (HW)

Fim de Recepção: RB8 ← Nono Bit

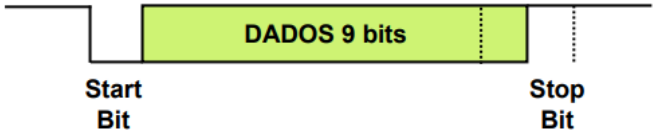
RI ← 1 (HW)

TB8, RB8, TI, REN e RI são bits do SCON

Modos de operação

- No **Modo 3**, a comunicação também é dita **assíncrona** e o *baudrate* programável, tal como no **Modo 1**. Contudo, a informação transmitida pelo TX e recebida pelo RX é composta de 11 *bits*, tal como no **Modo 2**;
- No contexto desta disciplina, trabalharemos com o **Modo 1** e o **Modo 3**.

Modo 3



Início de transmissão: SBUF ← Dados (SW)
TB8 ← Paridade, segundo stop bit (SW)
Nono bit ← TB8 (HW)

Fim de transmissão: TI ← 1 (HW)

Início de recepção: REN ← 1 (SW)
RXD ↓ (HW)

Fim de Recepção: RB8 ← Nono Bit
RI ← 1 (HW)

TB8, RB8, TI, REN e RI são bits do SCON

Cálculo do *baudrate*

- O *baudrate* da comunicação está associado à frequência de *clock* do µC. Nos **Modos 1** e **3**, este ainda depende da *flag* **SMOD** do registrados **PCON** e da taxa de estouro do timer / contador 1, o qual é utilizado justamente para definir o *baudrate.*

bit 7	*bit* 6	*bit* 5	*bit* 4	*bit* 3	*bit* 2	*bit* 1	*bit* 0
SMOD	-	-	-	GF1	GF0	PD	IDL

SMOD: *bit* que habilita a dupla taxa de transmissão/recepção (*baud rate*) do canal de comunicação serial. Quando igual a 1 lógico, o *baud rate* é dobrado, ou seja, a frequência de transmissão e recepção do canal de comunicação serial é dobrada, quando a programação do canal de comunicação serial estiver nos modos 1, 2, ou 3.

GF1: *flag* de uso de propósito geral.

GF0: *flag* de uso de propósito geral.

PD: *bit* de baixa potência (*power down*). Fazendo esse *bit* igual a 1 lógico, ativa-se o modo de baixa potência.

IDL: *bit* de Modo *Idle*. Fazendo esse *bit* igual a 1 lógico, ativa-se o modo *Idle*.

- **Obs:** para a comunicação serial, somente o **SMOD** é utilizado. As demais opções dizem respeito ao gerenciamento energético do µC.

Cálculo do *baudrate*

- O *baudrate* da comunicação está associado à frequência de *clock* do µC. Nos **Modos 1** e **3**, este ainda depende da *flag* **`SMOD`** do registrados **`PCON`** e da taxa de estouro do timer / contador 1, o qual é utilizado justamente para definir o *baudrate*;
- Dessa forma, para estes modos, o *baudrate* é calculado pela seguinte expressão:

$$Taxa\ de\ baudrate = \frac{2^{SMOD}}{32} \cdot \frac{f_{CLK}}{12 \cdot (256 - TH1)}\ [bits/s]$$

- **Importante:** em razão da exclusividade, a utilização da comunicação serial compromete o uso do T/C 1 para outras temporizações dentro do programa.

Cálculo do *baudrate*

- Tabela de configuração do *baudrate* através da programação do timer/contador 1 para diferentes modos de operação e frequências de *clock*:

			Timer 1		
Baud rate **(*bits*/seg)**	**Frequência do oscilador (Cristal) (MHz)**	**SMOD**	**C/$\overline{T}$**	**Modo**	**Valor da recarga**
Modo 0 Máx: 1 MHz	12	X	X	X	X
Modo 2 Máx: 375 K	12	1	X	X	X
Modo 1, 3: 62,5 K	12	1	0	2	FF_h
19,2 K	11,059	1	0	2	FD_h
9,6 K	11,059	0	0	2	FD_h
4,8 K	11,059	0	0	2	FA_h
2,4 K	11,059	0	0	2	$F4_h$
1.2 K	11,059	0	0	2	$E8_h$
137,5	11,059	0	0	2	$1D_h$
110	6	0	0	2	72_h
110	12	0	0	1	$FEEB_h$

Exemplo 1

- Usando a placa de desenvolvimento **V0.8**, escreva um programa para enviar uma *string* via porta serial sem paridade utilizando um *baudrate* de 9600 bps e *clock* de 11,0592 MHz. Na inicialização do programa este deve ler o pino P1.0 para, se em nível lógico **alto**, a *string* permaneça sendo enviada continuamente.

Exemplo 2

- Usando a placa de desenvolvimento **V0.8**, escreva um programa que receba um **número** via serial por interrupção e escreva ele no display BCD / 7 segmentos ligado utilizando um *baudrate* de 1200 bps e *clock* de 6 MHz.

- Calculando o valor de recarga do T/C 1 para o *baudrate* estipulado com SMOD = 0:

$$Taxa\ de\ baudrate = \frac{2^{SMOD}}{32} \cdot \frac{f_{CLK}}{12 \cdot (256 - TH1)}\ [bits/s]$$

$$1200 = \frac{2^0}{32} \cdot \frac{6 \cdot 10^6}{12 \cdot (256 - TH1)}$$

$$256 - TH1 = \frac{6 \cdot 10^6}{32 \cdot 12 \cdot 1200}$$

$$256 - TH1 = 13{,}021$$

$$TH1 = 256 - 13{,}021 \approx \mathbf{243}\boldsymbol{d}\ ou\ \mathbf{0F3}\boldsymbol{h}$$

Referências

- NICOLOSI, D. E. C. *Microcontrolador 8051 detalhado*. 9ª ed., São Paulo: Érica, 2013.
- NICOLOSI, D. E. C.; BRONZERI, R. B. *Microcontrolador 8051 com linguagem C prático e didático: família AT89S8252 Atmel*. 2ª ed., São Paulo: Érica, 2009.
- GIMENEZ, S. P. *Microcontroladores 8051: teoria e prática*. São Paulo: Érica, 2010.

Tabela ASCII

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	[ENG 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]