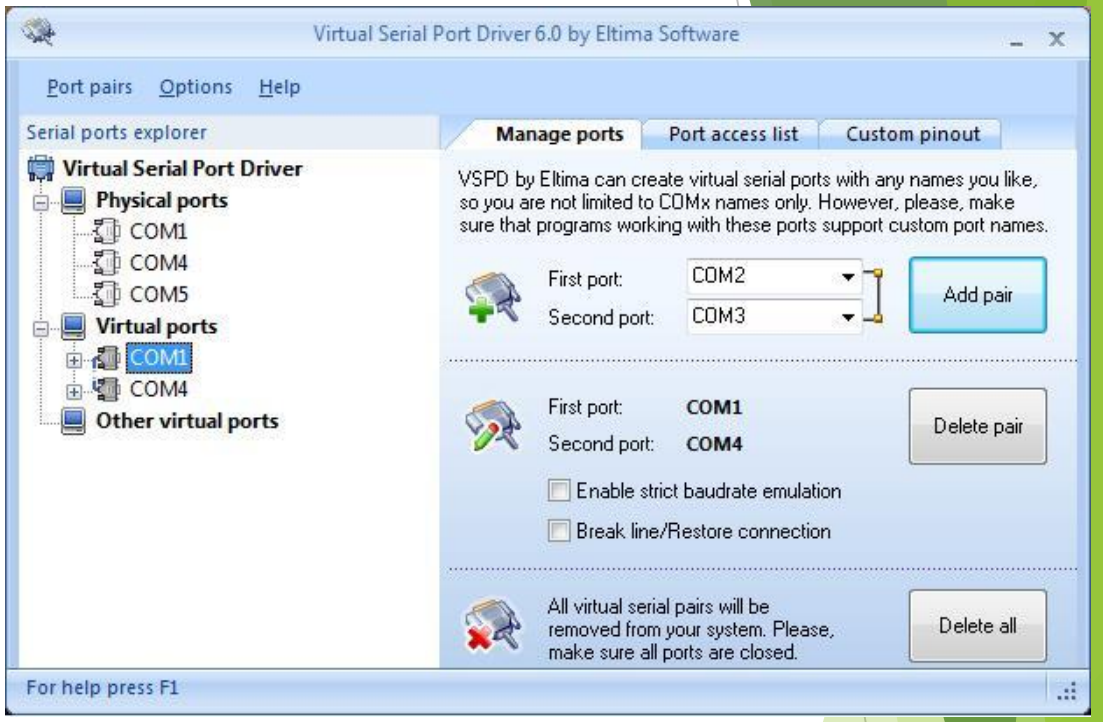
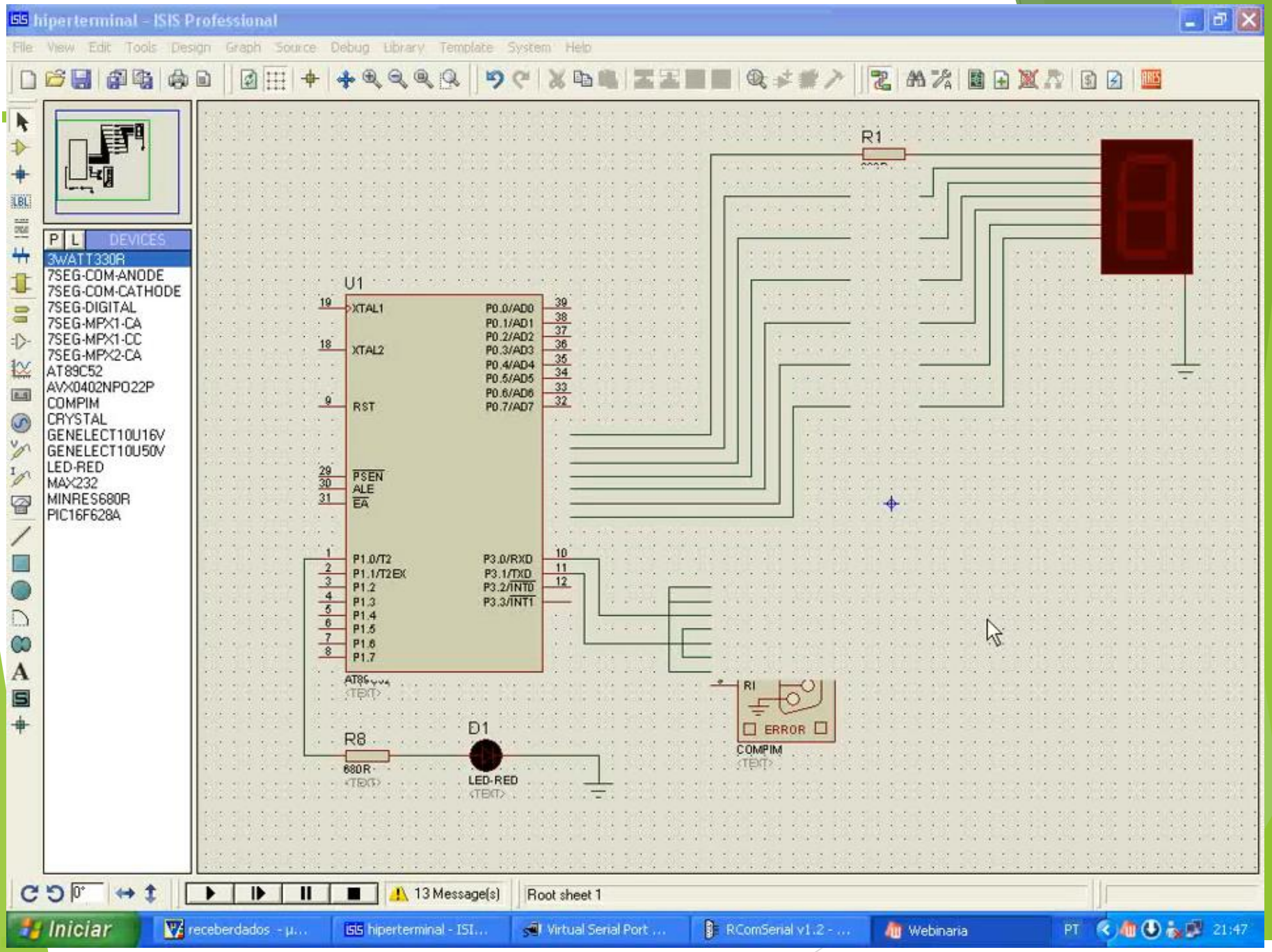
Comunicação Serial




hiperterminal - ISIS Professional
File View Edit Tools Design Graph Source Debug Library Template System Help
P L DEVICES
3WATT330R
7SEG-COM-ANODE
7SEG-COM-CATHODE
7SEG-DIGITAL
7SEG-MPX1-CA
7SEG-MPX1-CC
7SEG-MPX2-CA
AT89C52
AVX0402NP022P
COMPIM
CRYSTAL
GENELECT10U16V
GENELECT10U50V
LED-RED
MAX232
MINRES680R
PIC16F628A
R1
U1
XTAL1
XTAL2
RST
PSEN
ALE
EA
P1.0/T2
P1.1/T2EX
P1.2
P1.3
P1.4
P1.5
P1.6
P1.7
P0.0/AD0
P0.1/AD1
P0.2/AD2
P0.3/AD3
P0.4/AD4
P0.5/AD5
P0.6/AD6
P0.7/AD7
P3.0/RXD
P3.1/TXD
P3.2/INT0
P3.3/INT1
RI
ERROR
COMPIM
R8
680R
D1
LED-RED
13 Message(s)
Root sheet 1
Iniciar
receberdados - µ...
hiperterminal - ISI...
Virtual Serial Port ...
RComSerial v1.2 - ...
Webinaria
PT
21:47

Fim de um programa

```
1       ORG     0
2
3       MOV     00h, #10h ;(Direto / Imediato)
4       MOV     00h, 10h  ;(Direto / Direto)
5       MOV     R0, #10h  ;(Rn / Imediato)
6       MOV     R1, #10h  ;(Rn / Imediato)
7       MOV     A, #10h   ;(Imediato)
8
9       END
```

- Fim físico (diretiva END) não significa a parada do programa!
- Indica ao compilador que não há mais código após aquele ponto.
- O processador continuará executando instruções desconhecidas presentes após o fim físico do programa.
- É de responsabilidade do programador criar um fim lógico para o programa.

Fim lógico de um programa

- Sempre feito com um loop infinito
- Pode ser um loop que não faça nada ou que realize o mesmo procedimento várias vezes
- Necessário para impedir a execução de código desconhecido

Fim lógico de um programa

Programa fica parado sempre na mesma instrução

```
 1        ORG     0
 2
 3        MOV     00h, #10h ;(Direto / Imediato)
 4        MOV     00h, 10h  ;(Direto / Direto)
 5        MOV     R0, #10h  ;(Rn / Imediato)
 6        MOV     R1, #10h  ;(Rn / Imediato)
 7        MOV     A, #10h   ;(Imediato)
 8
 9        SJMP    $         ;Fim lógico do programa
10
11        END
```

Programa não sairá desta posição.

Uma interrupção pode tirar o programa desta instrução, mas voltará após esta ser finalizada

Fim lógico de um programa

Programa em loop contínuo:

```
1  ;Programa para debounce de uma chave
2  ;Necessário para remover ruídos da chave ao pressioná-la que pode
3  ;causar mais de uma contagem
4
5           ORG     0               ;Início do programa (PC=0)
6
7           CLR     A               ;Inicializa o contador em zero
8  LOOP:    JB      P1.0, $         ;Aguarda por nível lógico 0 (Chave aberta)
9           INC     A               ;Incrementa o contador
10          MOV     P2, A           ;Coloca o valor do contador em P2
11          LCALL   ATRASO          ;Chama rotina de atraso
12          JNB     P1.0, $         ;Aguarda por nível lógico 1(Chave fechada)
13          SJMP    LOOP            ;Retorna para nova contagem de pulso
14
15 ATRASO:  MOV     R0, #0FFh       ;Valor de atraso para debounce (aproximadamente 500us)
16          DJNZ    R0, $           ;Decrementa até zero
17          RET                     ;Retorna da rotina
18
19          END                     ;Fim do programa
```

Processador nunca executará bytes desconhecidos pois sempre volta para uma parte conhecida do programa

Comunicação Serial

A comunicação serial consiste em enviar ou receber pacotes de informação bit a bit.

No caso do 8051 o canal de comunicação serial é do tipo ”*full duplex*”, o que significa que ele pode, ao mesmo tempo, receber e transmitir dados.

Uma grande questão da transmissão serial é
“como informar o receptor do início e do final do pacote de informação”, ou seja, qual o primeiro bit da informação e qual é o último.

Assim, existem dois tipos de comunicação:

síncrona e assíncrona.

Na comunicação serial síncrona, são utilizados dois canais:

um para transmitir/receber os dados e outro para transmitir/receber o sinal de sincronismo.

No caso do 8051 a transmissão e também a recepção síncrona de dados são feitas através do pino RxD (pino P3.0).
O pino TxD (pino P3.1) é usado para o sinal de sincronismo.

Na comunicação assíncrona não há a sinal de sincronismo e, portanto, alguns cuidados especiais devem ser tomados:

As taxas de recepção e de transmissão devem ser iguais.

Um bit de início e outro de fim de transmissão devem ser enviados, além dos dados.

O bit de início de transmissão é zero, isto porque o canal normalmente fica em repouso no nível lógico alto.

Assim, a primeira passagem para zero, após a habilitação da transmissão, é interpretada como o sinal de início.

O sinal de parada é de nível lógico alto, após ser recebida a quantidade de bits previstos.

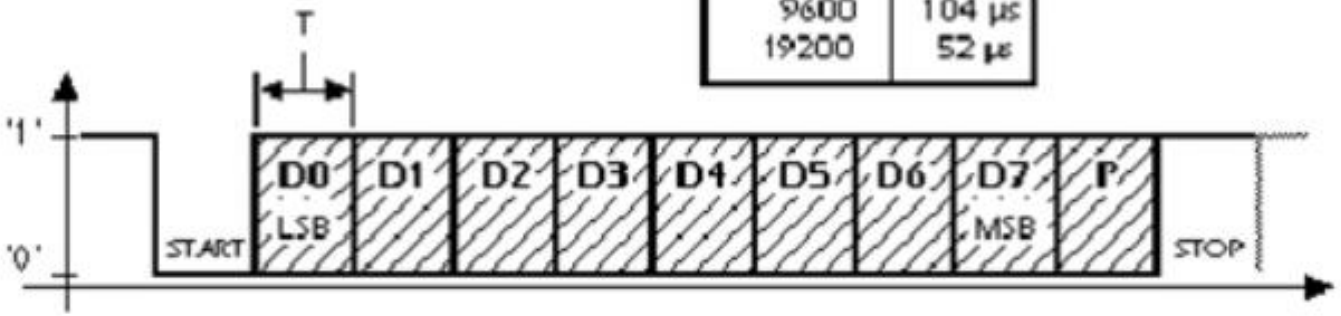
Além do bit de início (Start bit) e do bit de fim (Stop bit), também pode existir um terceiro bit extra, que é o bit de paridade, usado para verificar a consistência dos dados.

Formato da Comunicação Serial Assíncrona

FORMATO TÍPICO

1 Start Bit
8 Bits de Dados
1 Bit de Paridade
1 Stop Bit

Baud Rate	T
110	9.09 ms
300	3.33 ms
1200	833 µs
2400	417 µs
4800	208 µs
9600	104 µs
19200	52 µs



MENSAGEM

exatamente
8 bits de dados
aceitos

Start Bit
início de
transmissão

Bit de Paridade
checa a precisão
da transmissão

Stop Bit
fim de transmissão
e tempo para
receptor reiniciar

Registrador SCON – Configuração da Comunicação Serial

(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*	variável
1	0	2	UART de 9 *bits*	$f_{osc.}/64$ ou $f_{osc.}/32$
1	1	3	UART de 9 *bits*	variável

Símbolo	*Nome e Significado*
SM2	Habilita a característica de comunicação de multiprocessadores no modo 2 e 3. Nesses modos, se 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, deverá 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 limpado por *software*.
RB8	No modo 2 e 3, é o nono *bit* de dado que foi recebido. 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 8° *bit* no modo 0 ou no início do *stop bit* em outros modos, em qualquer transmissão serial. Deverá ser limpado por *software*.
RI	É o *flag* de interrupção de recepção. Setado por *hardware* no final do tempo do 8° *bit* no modo 0 ou na metade do tempo do *stop bit* em outros modos, em qualquer recepção serial. Deverá ser limpado por *software*.

Serial – Modo 0 (síncrono – registrador de deslocamento)

Taxa de comunicação (*baud rate*): igual à frequência de *clock* dividida por 12.

$$BaudRate = \frac{f_{clock}}{12} bits/s$$

Recepção: Fica habilitada fazendo REN = 1 e RI = 0.
Ao final da recepção o bit RI é setado por hardware, e deve ser resetado por software (CLR RI) antes da recepção seguinte.
O conteúdo recebido é transferido para o registrador SBUF.

Transmissão: É iniciada automaticamente quando o conteúdo de um registrador é transferido para o registrador SBUF.
Ao final da transmissão o bit TI é setado por hardware, e deve ser resetado por software (CLR TI) antes da transmissão seguinte.

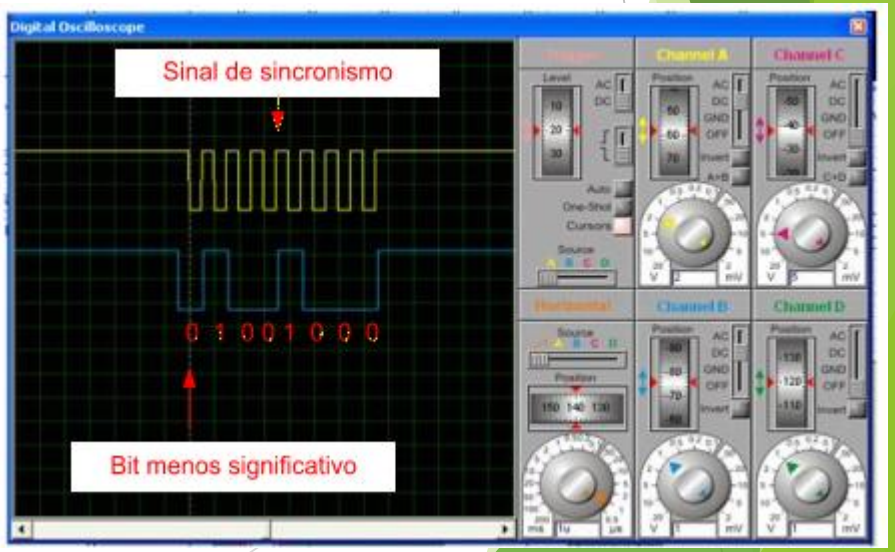
Obs.: Há um registrador SBUF para transmissão e outro para recepção

Rótulo	Instruções
	MOV SCON,#00H
	MOV A,#00H
V1:	MOV SBUF,A
	JNB TI,$
	CLR TI
	INC A
	SJMP V1
	END

Instrução que dá início à transmissão

Aguarda a transmissão de todos os bits.
Quando o processo acaba, o micro faz TI = 1.

Sinais de RxD e TxD durante uma transmissão síncrona



Serial – Modo 1 (assíncrono de 8 bits)

Taxa de comunicação (baud rate):

Bit 7 do registrador especial PCON

$$BaudRate = \frac{2^{SMOD}}{32} * \frac{f_{clock}}{12 * (256 - TH1)} bits/s$$

Recarga do Temporizador 1 no modo 2

O pacote de comunicação inclui 8 bits de dados,
1 bit de start e um bit de stop.

Recepção: Fica habilitada fazendo REN = 1 e RI = 0.
Tem início quando há uma transição de nível alto para baixo no pino P3.0 (RxD)
Ao final da recepção o bit RI é setado por hardware, e deve ser resetado por software (CLR RI) antes da recepção seguinte.
O conteúdo recebido é transferido para o registrador SBUF.

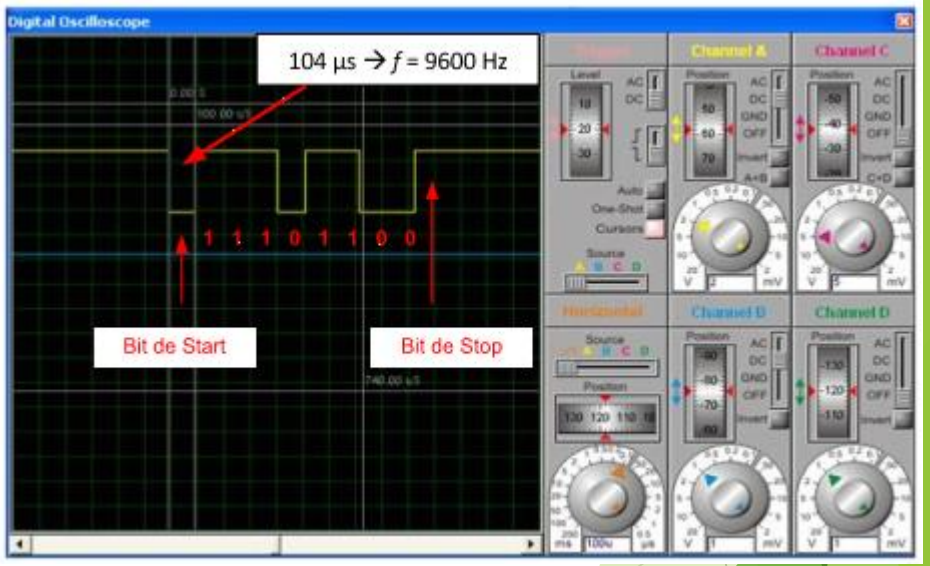
Transmissão: É iniciada automaticamente quando o conteúdo de um registrador é transferido para o registrador SBUF.
Ao final da transmissão o bit TI é setado por hardware, e deve ser resetado por software (CLR TI) antes da transmissão seguinte.

Rótulo	Instruções
	MOV SCON,#40H
	MOV TMOD,#20H
	MOV TH1,#0FDH
	MOV TL1,#0FDH
	SETB TR1
	MOV A,#00H
V1:	MOV SBUF,A
	JNB TI,$
	CLR TI
	INC A
	SJMP V1
	END

Configura serial modo 1 (assíncrono modo 1)

Timer 1 no modo 2. *Baud rate* de 9600 bps, para cristal de 11,0592 MHz.

Sinal de TxD (P3.1) durante uma transmissão assíncrona no modo 1



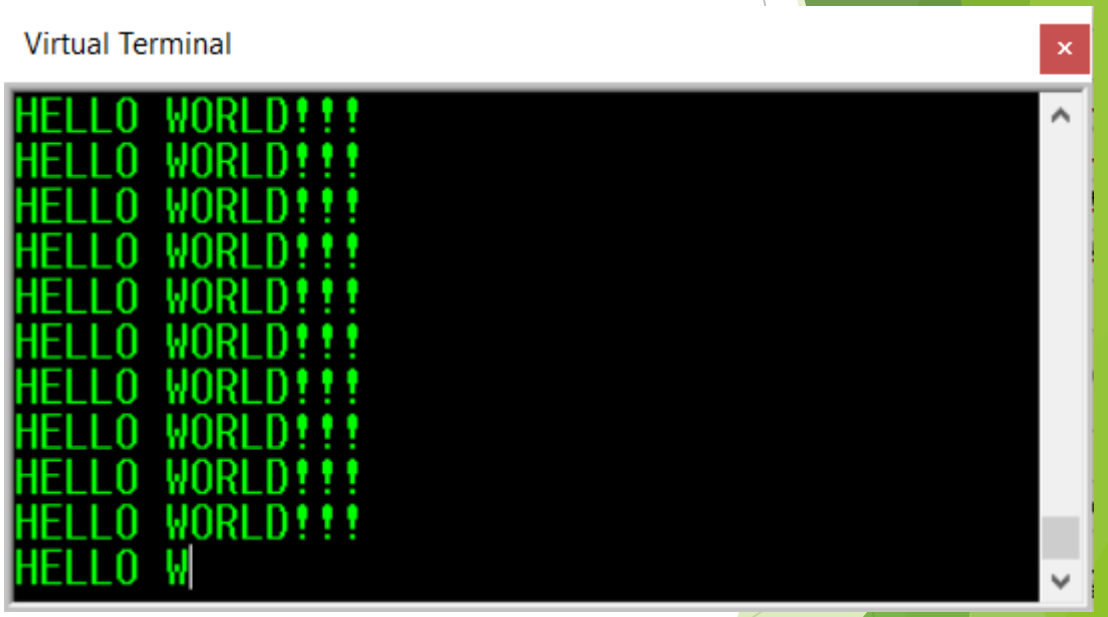
Instrução que dá início à transmissão

Aguarda a transmissão de todos os bits. Quando o processo acaba, o micro faz TI = 1.

Transmissão Serial no Modo 1

```
ORG 00H
LJMP INICIO

ORG 30H
INICIO:     MOV SP,#2FH
            MOV SCON,#40H
            MOV TMOD,#20H
            MOV TH1,#0FDH
            MOV TL1,#0FDH
            SETB TR1
            MOV DPTR,#MSG1

V1:         MOV R7,#00
V2:         MOV A,R7
            MOVC A,@A+DPTR
            CJNE A,#0FFH,ENVIA
            SJMP V1

ENVIA:      MOV SBUF,A
            JNB TI,$
            CLR TI
            INC R7
            SJMP V2

MSG1: DB 'HELLO WORLD!!!', 0DH, 0FFH
MSG2: DB 'MICRO 2017 ', 0Dh, 0FFH

end
```

Baud-rate de 9600 bps, no modo 1
Cristal oscilador: 11,0592 MHz



Código ASCII para mudança de linha

; Transmissão de mensagem via serial. Cristal: 11.0592 MHz. Taxa de transmissão: 4800 bps

```
ORG 00H
LJMP INICIO

ORG 30H
INICIO:   MOV SP,#2FH          ; Posição inicial da Pilha: 2FH
          MOV SCON,#40H        ; Serial no modo 1: assíncrona de 8 bits
          MOV TMOD,#20H        ; Timer 1 no modo 2 (recarga automática)
          MOV TH1,#0FAH        ; Valor da recarga: FAH → baud rate: 4800 bps
          SETB TR1             ; Dispara Timer 1

          MOV DPTR,#MSG        ; DPTR assume o valor do endereço inicial da tabela MSG
V2:       MOV R7,#00H          ; Offset para leitura da MSG assume valor inicial 00H

V3:       MOV A,R7             ; Acumulador recebe o valor atual do Offset
          MOVC A,@A+DPTR       ; Acumulador recebe o conteúdo da posição A+DPTR da MSG
          CJNE A,#0FFH,V1      ; Verifica se A = FFH (fim da MSG). Se não for, desvia para V1
          SJMP V2              ; Retorna para V2 após cada fim de MSG

V1:       MOV SBUF,A           ; Transfere de A para SBUF o valor  ser transmitido via serial
          JNB TI,$             ; Aguarda final da transmissão do conteúdo de SBUF
          CLR TI               ; Limpa a flag TI, de transmissão serial
          INC R7               ; Incrementa o valor do Offset
          SJMP V3              ; Retorna para V3, para ler o próximo caractere de MSG

MSG:      DB ' COTUCA ', 0DH, 0FFH
MSG1:     DB 4DH, 49H, 43H, 52H, 4FH, 20H, 32H, 30H, 31H, 37H, 0DH, 0FFH
END
```

As mensagens MSG e MSG1 são equivalentes

Recepção Serial no Modo 1

```
LAMP        EQU P2.7

            ORG 00H
            LJMP INICIO

            ORG 30H
INICIO:     MOV SP,#2FH
            MOV SCON,#40H
            MOV TMOD,#20H
            MOV TH1,#0FDH
            MOV TL1,#0FDH
            SETB TR1
            CLR RI
            SETB REN
            MOV A,#01H

V0:         JNB RI,$
            MOV R0,SBUF
            CLR RI

            CJNE R0,#'D',V1
            SJMP LED_DIREITA

V1:         CJNE R0,#'E',V3
            SJMP LED_ESQUERDA

V3:         CJNE R0,#'L',V4
            CPL LAMP
            SJMP V0

V4:         CJNE R0,#'M',V0

MOTOR:      CLR P2.3
            CPL P2.2
            SJMP V0

LED_DIREITA:
V5:         MOV P1,A
            RR A
            LCALL ATRASO
            JNB RI,V5
            SJMP V0

LED_ESQUERDA:
            MOV R3,#64
V6:         MOV P1,A
            RL A
            LCALL ATRASO

            DJNZ R3,V6
            SJMP V0

ATRASO:     MOV R1,#100
V2:         MOV R2,#200
            DJNZ R2,$
            DJNZ R1,V2
            RET

END
```

Se R0 = 4DH (ASCII de M), aciona o motor CC (liga/desliga)

Aguarda receber dados via serial e transfere os dados para R0

Se R0 = 43H (ASCII de D), rotaciona Leds para a direita. Enquanto RI = 0, continua rotacionando para a esquerda

Se R0 = 44H (ASCII de E), rotaciona Leds para a esquerda. São 64 passos para a esquerda, que corresponde a 8 giros completos para a esquerda.

Se R0 = 4CH (ASCII de L), aciona a lâmpada (apaga/liga)

Recepção Serial no Modo 1 Com interrupção

```
LAMP        EQU P2.7

ORG 00H
LJMP INICIO

ORG 23H
            MOV R0,SBUF
            CLR RI
            RETI

ORG 30H
INICIO:     MOV SP,#2FH
            MOV IE,#90H
            MOV SCON,#40H
            MOV TMOD,#20H
            MOV TH1,#0FDH
            MOV TL1,#0FDH
            SETB TR1
            CLR RI
            SETB REN
            MOV A,#01H

V0:         CJNE R0,#'D',V1
            SJMP LED_DIREITA
V1:         CJNE R0,#'E',V3
            SJMP LED_ESQUERDA
V3:         CJNE R0,#'L',V4
            SETB LAMP
            SJMP V0
V4:         CJNE R0,#'F',V5
            CLR LAMP
            SJMP V0
```

Se R0 = 4DH (ASCII de M), aciona o motor CC (liga/desliga)

R0 recebe os dados recebidos via serial, através de interrupção

Se R0 = 43H (ASCII de D), rotaciona *Leds* para a direita. Continua rotação, até R0 receber outro caractere.

Se R0 = 44H (ASCII de E), rotaciona *Leds* para a esquerda. Enquanto R0 = 44H, continua rotacionando para a esquerda

Se R0 = 4CH (ASCII de L), liga a lâmpada.
Se R0 = 45H (ASCII de F), desliga a lâmpada.

```
V5:         CJNE R0,#'M',V6

MOTOR:      CLR P2.3
            SETB P2.2
            SJMP V0

V6:         CJNE R0,#'N',V0
            CLR P2.3
            CLR P2.2
            SJMP V0

LED_DIREITA:
            MOV P1,A
            RR A
            LCALL ATRASO
            SJMP V0

LED_ESQUERDA:
            MOV P1,A
            RL A
            LCALL ATRASO

            SJMP V0

ATRASO:     MOV R1,#100
V2:         MOV R2,#200
            DJNZ R2,$
            DJNZ R1,V2
            RET

END
```

Serial – Modo 2 (assíncrono de 9 bits)

Taxa de comunicação (*baud rate*):

Bit 7 do registrador especial PCON

$$BaudRate = 2^{SMOD} * \frac{f_{clock}}{64} bits/s$$

O pacote de comunicação inclui 8 bits de dados, 1 bit extra (RB8 ou TB8), 1 bit de start e um bit de stop. O bit extra pode ser a paridade.

Recepção: Fica habilitada fazendo REN = 1 e RI = 0.
Tem início quando há uma transição de nível alto para baixo no pino P3.0 (RxD)
Ao final da recepção o bit RI é setado por hardware, e deve ser resetado por software (CLR RI) antes da recepção seguinte.
O nono bit chega através do bit RB8.

Transmissão: É iniciada automaticamente quando o conteúdo de um registrador é transferido para o registrador SBUF.
Ao final da transmissão o bit TI é setado por hardware, e deve ser resetado por software (CLR TI) antes da transmissão seguinte. O nono bit é transmitido através de TB8.

Serial – Modo 3 (assíncrono de 9 bits)

Taxa de comunicação (*baud rate*):

Bit 7 do registrador especial PCON

$$BaudRate = \frac{2^{SMOD}}{32} * \frac{f_{clock}}{12 * (256 - TH1)} bits/s$$

Recarga do Temporizador 1 no modo 2

O pacote de comunicação inclui 8 bits de dados, 1 bit extra, 1 bit de start e um bit de stop.

Recepção: Fica habilitada fazendo REN = 1 e RI = 0.
Tem início quando há uma transição de nível alto para baixo no pino P3.0 (RxD)
Ao final da recepção o bit RI é setado por hardware, e deve ser resetado por software (CLR RI) antes da recepção seguinte.
O conteúdo recebido é transferido para o registrador SBUF.

Transmissão: É iniciada automaticamente quando o conteúdo de um registrador é transferido para o registrador SBUF.
Ao final da transmissão o bit TI é setado por hardware, e deve ser resetado por software (CLR TI) antes da transmissão seguinte.

Taxas de Comunicação Serial mais Comuns

			Timer 1		
Baud Rate (bits/seg)	Freq. Osc. (MHz)	SMOD	C/Tbarra	Modo	Valor Recar.
Modo 0 Máx: 1MHz	12	X	X	X	X
Modo 2 Máx: 375K	12	1	X	X	X
Modo 1, 3: 62,5K	12	1	0	2	FFh
19,2K	11,059	1	0	2	FDh
9,6K	11,059	0	0	2	FDh
4,8K	11,059	0	0	2	FAh
2,4K	11,059	0	0	2	F4h
1.2K	11,059	0	0	2	E8h
137,5	11,059	0	0	2	1Dh
110	6	0	0	2	72h
110	12	0	0	1	FEEBh

Tabela ASCII

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Mensagens: Computador → Micro → LCD

```
RS   EQU  P3.5
RW   EQU  P3.6
EN   EQU  P3.7

DADOS  EQU P0

ORG 00H
LJMP INICIO

ORG 23H
CLR RI
MOV A,SBUF
MOV P1,A
LCALL TEXTO_WR
RETI

ORG 30H
INICIO:  MOV SP,#2FH
MOV SCON,#40H
MOV IE,#90H
MOV TMOD,#20H
MOV TL1,#0FDH
MOV TH1,#0FDH
MOV R7,#0FFH
SETB TR1
SETB REN
LCALL INICIA

SJMP $
```

```
INICIA:  MOV A,#38H
         LCALL INSTR_WR

         MOV A,#38H
         LCALL INSTR_WR

         MOV A,#0EH
         LCALL INSTR_WR

         MOV A,#06H
         LCALL INSTR_WR

         MOV A,#01H
         LCALL INSTR_WR
         RET

INSTR_WR: SETB EN
         CLR RW
         CLR RS
         MOV DADOS,A
         CLR EN
         LCALL ATRASO_LCD
         RET
```

```
TEXTO_WR:SETB EN
         CLR RW
         SETB RS
         MOV DADOS,A
         CLR EN
         LCALL ATRASO_LCD
         RET

ATRASO_LCD:
         MOV R4,#10
V6:      MOV R5,#80
         DJNZ R5,$
         DJNZ R4,V6
         RET

         END
```

A recebe valor via serial e envia para P1 e para o LCD

Sub-rotina para escrever instrução no LCD

Sub-rotina de inicialização do LCD

Sub-rotina para escrever dados no LCD

Serial_LCD_22_6.asm

Motor de Passo via serial

Rótulo	Mnemônico
	ORG 00H
	LJMP INICIO
	ORG 23H
	CLR RI
	MOV R0,SBUF
	RETI
	ORG 30H
INICIO:	MOV SP,#2FH
	MOV SCON,#40H
	MOV IE,#90H
	MOV TMOD,#20H
	MOV TL1,#0FDH
	MOV TH1,#0FDH
	MOV R0,#00H
	MOV A,#11H
	SETB TR1
	SETB REN

Rótulo	Mnemônico
V2:	CJNE R0,#44H,V1
	LJMP DIREITA
V1:	CJNE R0,#45H,V2
	LJMP ESQUERDA
DIREITA:	MOV P1,A
	RR A
	LCALL ATRASO
	SJMP V2
ESQUERDA:	MOV P1,A
	RL A
	LCALL ATRASO
	SJMP V2
ATRASO:	MOV R7,#200
V3:	MOV R6,#250
	DJNZ R6,$
	DJNZ R7,V3
	RET
	END

Serial_Passo_22_6.asm

Serial_estado_22_6.asm

```
CHAVE   EQU  P3.3   ; CHAVE (P3.3) – MOTOR LIGA (P3.3 = 0) OU DESLIGA (P3.3 = 1)
STATUS  EQU 22H     ; registrador que guarda o estado das chaves
M0      EQU  P2.2   ; IN0 do driver para acionamento do motor CC
M1      EQU  P2.3   ; IN1 do driver para acionamento do motor CC
        ORG 00H
        LJMP INICIO

        ORG 30H
INICIO: MOV SP,#2FH             ; Pilha no endereço inicial 2Fh
        MOV TMOD,#20H           ; Temporizador 1 no modo 2 (recarga automática) para o gerar o baud rate
        MOV SCON,#40H           ; SCON = 0100 0000 – Serial no modo 1
        MOV TH1,#0FAH           ; Recarga para baud rate de 4800 bps
        MOV TL1,#0FAH           ; Valor inicial de contagem, desde a primeira contagem
        SETB TR1                ; Dispara temporizador 1
        MOV R7,#00H             ; Contador (offset) para leitura das mensagens a serem enviadas via serial
        MOV STATUS,#00H         ; Zera o registrador de STATUS para eliminar a possibilidade de “lixo”
        CLR M0
        CLR M1                  ; Motor CC parado

V1:     MOV A,P3                ; Leitura da porta P3 (onde está a chave que liga e desliga o motor)
        ANL A,#00001000B        ; Faz uma operação AND entre A e 08H, para isolar o pino P3.3
        XRL A,STATUS            ; Verifica se houve alteração no STATUS
        JZ V1                   ; Se A XOR STATUS for zero, não houve mudança, então volta para V1 (A=0 e Z=1)
                                ; Se houve mudança na posição da CHAVE, atualiza a situação e o STATUS
        JNB CHAVE,LIGA_M        ; Se CHAVE = 0, liga o motor
        MOV DPTR,#M_OFF         ; DPTR aponta para o início da mensagem de motor desligado
        LCALL SERIAL            ; Chama sub-rotina para mostrar a mensagem de motor desligado
        CLR M0
        CLR M1                  ; Motor desligado
        SETB STATUS.3           ; Atualiza registrador de STATUS com valor 1 na posição STATUS.3
        SJMP V1
```

Transfere conteúdo de P3 para A, isola o pino P3.3 e verifica se houve mudança de estado

Continuação do programa Serial_estado_22_6.asm

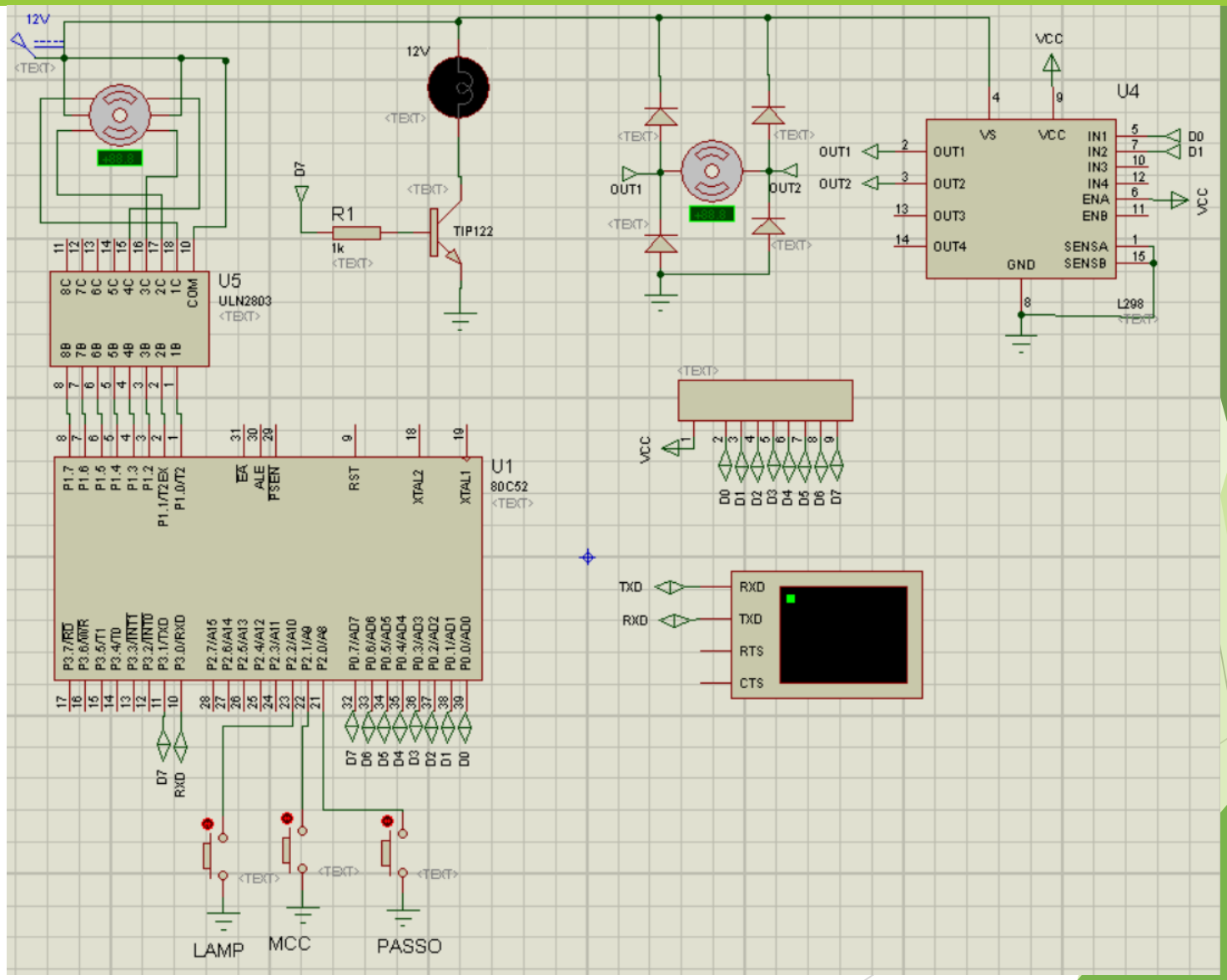
```
LIGA_M:   MOV DPTR,#M_ON        ; DPTR aponta para o início da mensagem de motor ligado
          LCALL SERIAL          ; Chama sub-rotina para mostrar a mensagem de motor ligado
          SETB M0
          CLR M1                ; Motor ligado
          CLR STATUS.3          ; Atualiza registrador de STATUS com valor 0 na posição STATUS.3
          SJMP V1

SERIAL:   MOV A,R7              ; Transfere para A o valor do offset
          MOVC A,@A+DPTR        ; A recebe o conteúdo da tabela
          CJNE A,#0FFH,ENVIA    ; Verifica se já chegou ao final da tabela
          MOV R7,#00H           ; Se a tabela já acabou, faz R7 = 0
          RET                   ; Retorna da sub-rotina SERIAL

ENVIA:    MOV SBUF,A            ; Envia conteúdo da tabela/mensagem para o computador
          JNB TI,$              ; Aguarda terminar a transmissão
          CLR TI                ; limpa flag de transmissão
          INC R7                ; incrementa R7 (offset para leitura da tabela / mensagem)
          SJMP SERIAL           ; retorna para SERIAL

M_ON:     DB 'MOTOR DE CORRENTE CONTINUA  LIGADO', 0DH, 0DH, 0FFH
M_OFF:    DB 'MOTOR DE CORRENTE CONTINUA  DESLIGADO', 0DH, 0DH, 0FFH

          END
```

Serial_Leds_28_6.asm

Rotação de Leds via serial

```
Rótulo    Mnemônico
          ORG 00H
          LJMP INICIO
          
          ORG 23H
          CLR RI
          MOV R0,SBUF
          RETI
          
          ORG 30H
INICIO:   MOV SP,#2FH
          MOV SCON,#40H
          MOV IE,#90H
          MOV TMOD,#20H
          MOV TL1,#0FDH
          MOV TH1,#0FDH
          MOV R0,#00H
          MOV A,#01H
          SETB TR1
          SETB REN
          
Rótulo    Mnemônico
V2:       CJNE R0,#44H,V1
          MOV B,#44H
          SJMP DIREITA
          
V1:       CJNE R0,#45H,V4
          MOV B,#45H
          SJMP ESQUERDA
          
V4:       CJNE R0,#50H,V5
          SJMP V2
          
V5:       MOV R0,B
          SJMP V2
          
DIREITA:  MOV P1,A
          RR A
          LCALL ATRASO
          SJMP V2
          
ESQUERDA: MOV P1,A
          RL A
          LCALL ATRASO
          SJMP V2
          END
```

Acionamentos diversos com informação de status para o computador



```
          CH_STEP    EQU P2.0
          CH_MCC     EQU P2.1
          CH_LAMP    EQU P2.2

          M0     EQU  P0.0
          M1     EQU  P0.1
          LAMP   EQU  P0.7

          STATUS     EQU 22H

          ORG 00H
          LJMP INICIO

          ORG 30H
INICIO:   MOV SP,#2FH
          MOV TMOD,#20H
          MOV SCON,#40H
          MOV TH1,#0FAH
          MOV TL1,#0FAH
          SETB TR1
          MOV R7,#00H
          MOV P1,#00H

          MOV STATUS,#00H
          CLR M0
          CLR M1
          CLR LAMP

V1:       MOV A,P2
          ANL A,#00000111B
          XRL A,STATUS
          JZ V1
```

STATUS indica o estado atual dos dispositivos.

STATUS.0 = 0 → motor de passo ligado
STATUS.1 = 0 → motor CC ligado
STATUS.2 = 0 → lâmpada ligada

Compara o estado atual das chaves (leitura de P2) com o conteúdo de STATUS (**XRL A,STATUS**).

Enquanto eles forem iguais (Z=1), aguarda no loop.

Quando houver mudança da posição de qualquer das chaves, sai desse loop e atualiza tudo.

Envia a atualização via serial

```
;== Acionamento da Lâmpada ==========

V5:       JNB CH_LAMP,LIGA_Lamp
          MOV DPTR,#Lamp_OFF
          LCALL SERIAL
          CLR LAMP
          SETB STATUS.2
          SJMP V2

LIGA_Lamp: MOV DPTR,#Lamp_ON
          LCALL SERIAL
          SETB LAMP
          CLR STATUS.2              ;

;== Acionamento do Motor CC ==========

V2:       JNB CH_MCC,LIGA_M
          MOV DPTR,#M_OFF
          LCALL SERIAL
          CLR M0
          CLR M1
          SETB STATUS.1
          SJMP V3

LIGA_M:   MOV DPTR,#M_ON
          LCALL SERIAL
          SETB M0
          CLR M1
          CLR STATUS.1
```

```
;== Acionamento do Motor de Passo =======

V3:     JNB CH_STEP,LIGA_P
        MOV DPTR,#PASSO_OFF
        LCALL SERIAL
        MOV P1,#00H
        SETB STATUS.0
        SJMP V1

LIGA_P:MOV DPTR,#PASSO_ON
        LCALL SERIAL
        CLR STATUS.0
        MOV R6,#11H
V6:     MOV P1,R6
        LCALL ATRASO

        MOV A,P2
        ANL A,#00000111B
        XRL A,STATUS
        JNZ V5

        MOV A,R6
        RL A
        MOV R6,A
        SJMP V6
```

Para manter o motor de passo ligado, é necessário rotacionar os bits da porta P1.

Assim, é necessário manter-se nesse loop de V6, verificando se há mudança em alguma chave.

```
;== Envia Mensagem via Serial =======
SERIAL:     MOV A,R7
            MOVC A,@A+DPTR
            CJNE A,#0FFH,ENVIA
            MOV R7,#00H
            RET

ENVIA:      MOV SBUF,A
            JNB TI,$
            CLR TI
            INC R7
            SJMP SERIAL

ATRASO:     MOV R0,#100
V4:         MOV R1,#150
            DJNZ R1,$
            DJNZ R0,V4
            RET

Lamp_ON: DB 'Lampada Ligada', 0DH, 0FFH
Lamp_OFF: DB 'Lampada Desligada', 0DH, 0FFH

M_ON: DB 'Motor de Corrente Continua Ligado', 0DH, 0FFH
M_OFF: DB 'Motor de Corrente Continua Desligado', 0DH, 0FFH

PASSO_ON: DB 'Motor de Passo Ligado', 0DH, 0DH, 0FFH
PASSO_OFF: DB 'Motor de Passo Desligado', 0DH, 0DH, 0FFH

END
```