
KNIGHT RIDER
Sequencial “Supermáquina”
(TEMPORIZAÇÃO SIMPLES)
UNIVERSAL
© 2004 Universal Studios. Alle Rechte vorbehalten.

Roteiro

- Sequencial Supermáquina
- Materiais
- Pinos do PIC
- Fluxograma
- Temporização
- Circuito no protoboard
- Melhoramentos
- Sugestão de Atividades

Sequencial

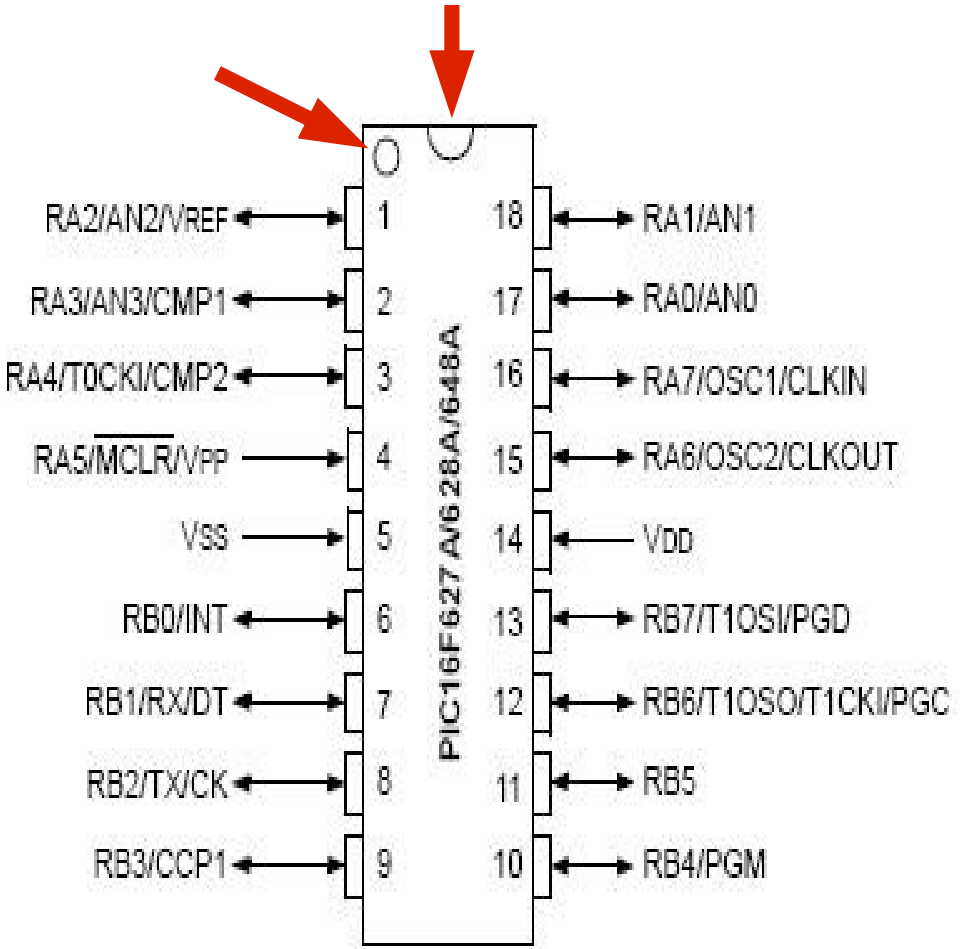
Um sequencial é basicamente um conjunto de ***leds*** que acedem e apagam em uma determinada ordem provocando algum tipo de efeito visual.

O sequencial supermáquina é assim chamado graças a série Supermáquina dos anos 80, onde o carro do protagonista possuía na sua parte frontal um sequencial onde as luzes acendiam de forma a provocar um efeito de “vai-e-vem”.

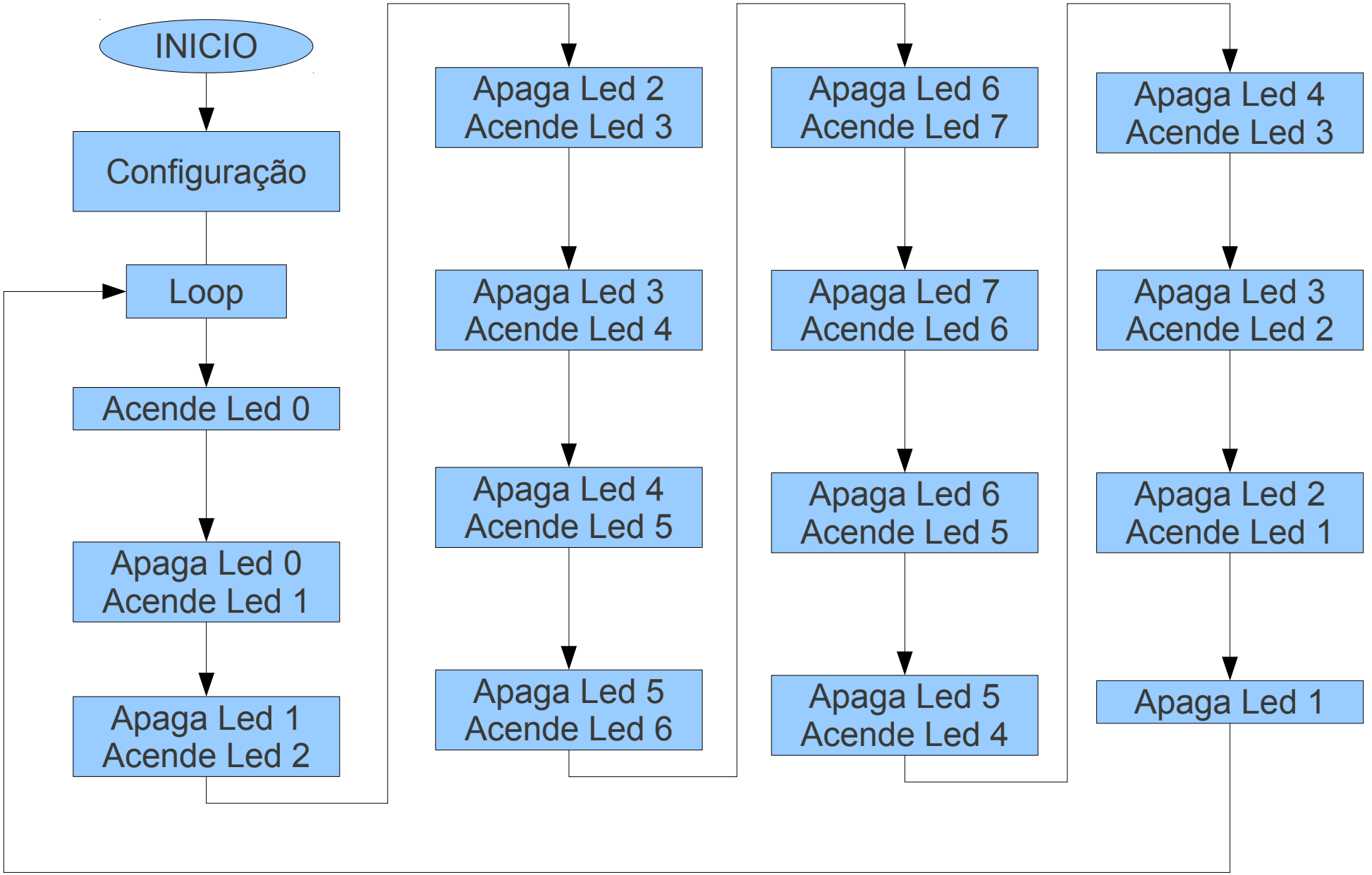
Materiais

- Gravador
- PIC 16F628A
- 8 Leds
- 8 Resistores de 1K
- 2 Suportes para Pilhas AA
- 4 Pilhas AA
- Protoboard e Fios

Pinos do PIC

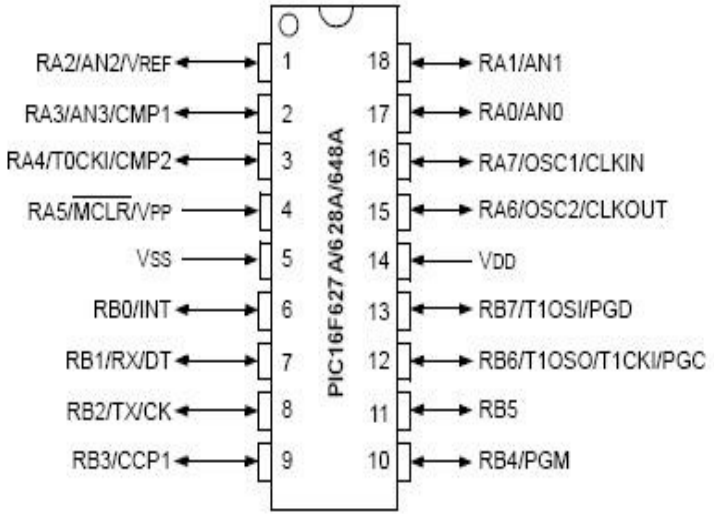

RA2/AN2/VREF
1
RA3/AN3/CMP1
2
RA4/T0CKI/CMP2
3
RA5/MCLR/VPP
4
VSS
5
RB0/INT
6
RB1/RX/DT
7
RB2/TX/CK
8
RB3/CCP1
9
18
RA1/AN1
17
RA0/AN0
16
RA7/OSC1/CLKIN
15
RA6/OSC2/CLKOUT
14
VDD
13
RB7/T1OSI/PGD
12
RB6/T1OSO/T1CKI/PGC
11
RB5
10
RB4/PGM
PIC16F627 A/628A/648A

Fluxograma


INICIO
Configuração
Loop
Acende Led 0
Apaga Led 0
Acende Led 1
Apaga Led 1
Acende Led 2
Apaga Led 2
Acende Led 3
Apaga Led 3
Acende Led 4
Apaga Led 4
Acende Led 5
Apaga Led 5
Acende Led 6
Apaga Led 6
Acende Led 7
Apaga Led 7
Acende Led 6
Apaga Led 6
Acende Led 5
Apaga Led 5
Acende Led 4
Apaga Led 4
Acende Led 3
Apaga Led 3
Acende Led 2
Apaga Led 2
Acende Led 1
Apaga Led 1

Fazendo Saída com o PIC

- O controlador PIC tem 2 Portas para Entrada e Saída uma conhecida por Port A e a outra Port B. Usaremos Port B neste exemplo.
- O Port B do PIC tem o endereço 06H, e tem os seu bits “ligados” ao pinos externos do PIC, de acordo com a tabela a seguir:



BIT PORTB	Pino
0	6
1	7
2	8
3	9
4	10
5	11
6	12
7	13

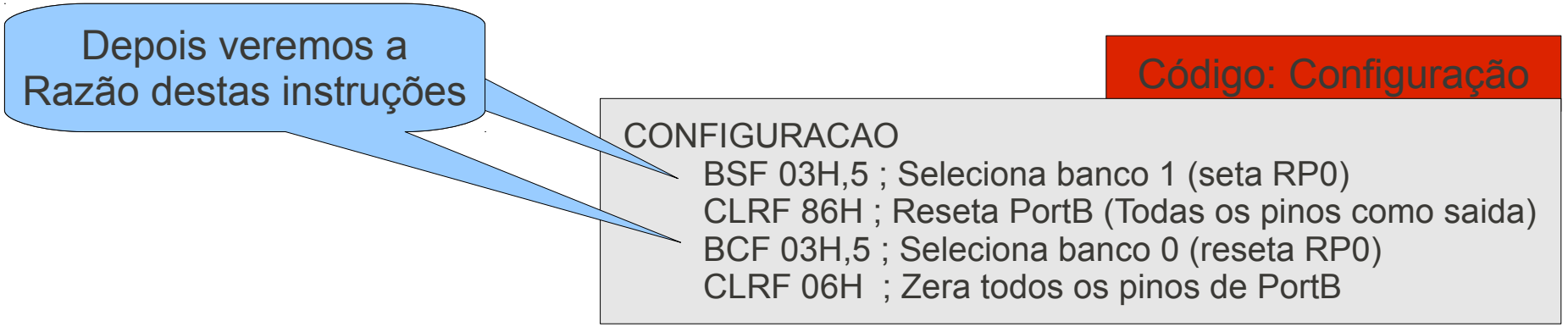
Configurando as Saídas

- As configurações de E/S para cada Port são feitas usando os registradores TRISA e TRISB onde TRISA configura o PortA e TRISB o PortB, os endereços são 85H e 86H respectivamente.
- Para configurar algum pino de Port B como saída devemos desligar o bit correspondente em TRISB. Ou seja se queremos que o pino 6 de PortB seja saída (output) configuramos o Bit 6 de TRISB como 0. Para configurar como entrada configuramos o bit 6 como 1;
- **É fácil lembrar desta regra 0 para 0utput e 1 para 1nput**
- Desta forma para configurar todos os pinos de port B como saída devemos zerar todos os bits de TRISB;

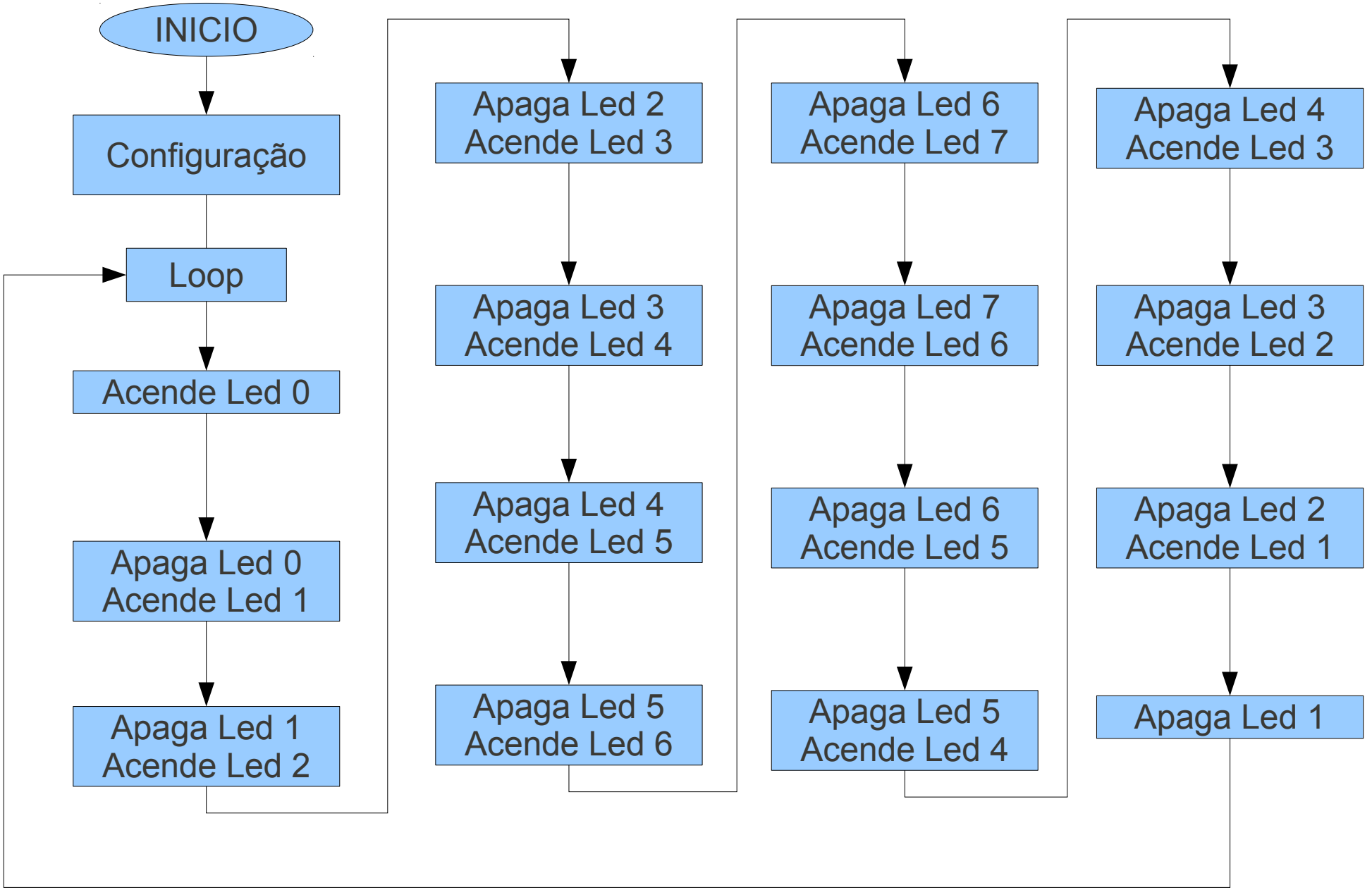
Instruções BCF e BSF

- As instruções BCF (Bit Clear File) e BSF (Bit Set File) são usadas para desligar e ligar bits específicos de um registrador.
- Ex
  - BCF 06H,1 ; desliga o bit 1 de 6H
  - BSF 06H,1 ; Liga o bit 1 de 6H
- A sintaxe destes comandos é
  - BCF Registrador, Bit
  - BSF Registrador, Bit

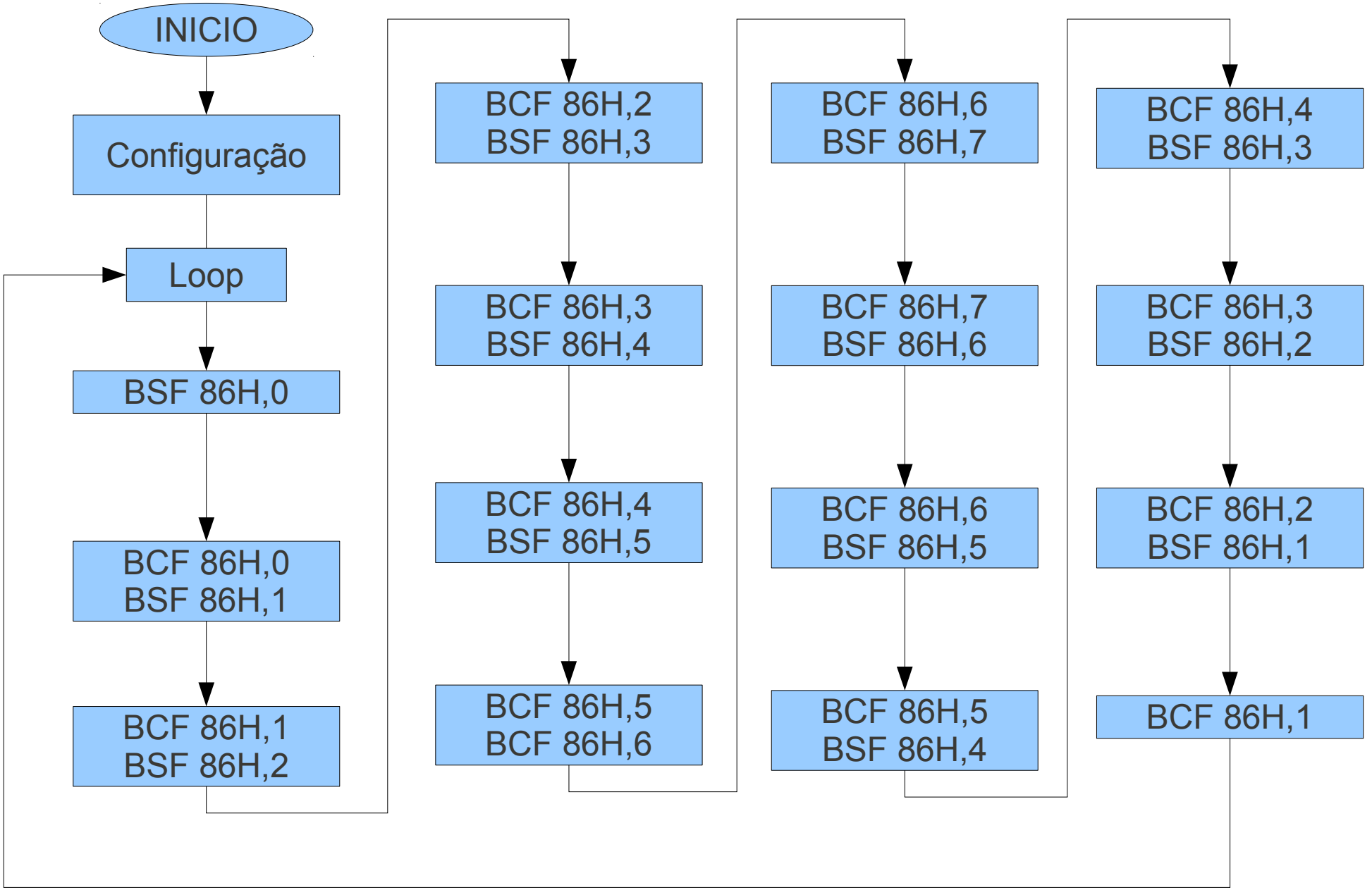
Configuração



Fluxograma


INICIO
Configuração
Loop
Acende Led 0
Apaga Led 0
Acende Led 1
Apaga Led 1
Acende Led 2
Apaga Led 2
Acende Led 3
Apaga Led 3
Acende Led 4
Apaga Led 4
Acende Led 5
Apaga Led 5
Acende Led 6
Apaga Led 6
Acende Led 7
Apaga Led 7
Acende Led 6
Apaga Led 6
Acende Led 5
Apaga Led 5
Acende Led 4
Apaga Led 4
Acende Led 3
Apaga Led 3
Acende Led 2
Apaga Led 2
Acende Led 1
Apaga Led 1

Fluxograma


INICIO
Configuração
Loop
BSF 86H,0
BCF 86H,0
BSF 86H,1
BCF 86H,1
BSF 86H,2
BCF 86H,2
BSF 86H,3
BCF 86H,3
BSF 86H,4
BCF 86H,4
BSF 86H,5
BCF 86H,5
BCF 86H,6
BCF 86H,6
BSF 86H,7
BCF 86H,7
BSF 86H,6
BCF 86H,6
BSF 86H,5
BCF 86H,5
BSF 86H,4
BCF 86H,4
BSF 86H,3
BCF 86H,3
BSF 86H,2
BCF 86H,2
BSF 86H,1
BCF 86H,1

Código

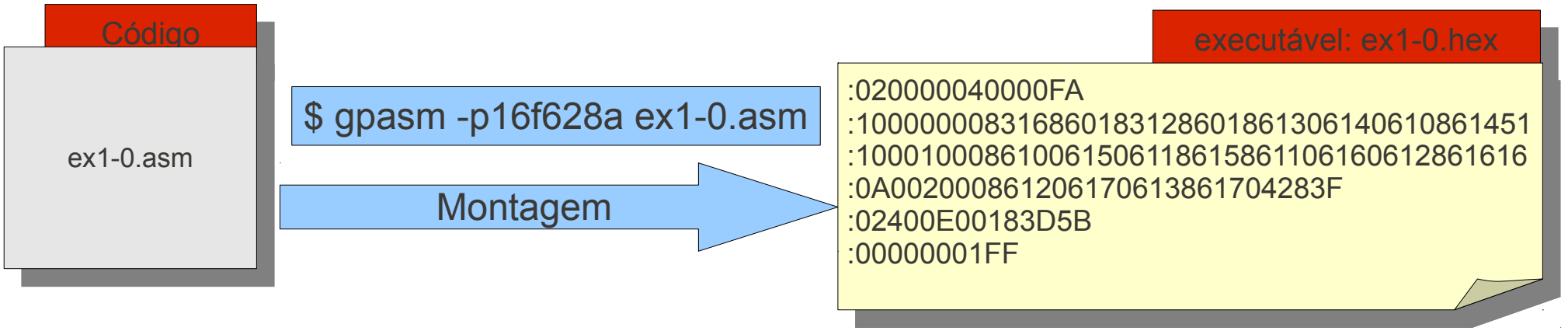
Código: ex1-0.asm

```
__CONFIG  0x3D18
ORG 0x00
CONFIGURACAO
        BSF 03H,5 ; Seleciona banco 1 (seta RP0)
        CLRF 86H ; Reseta PortB (Todas os pinos como saida)
        BCF 03H,5 ; Seleciona banco 0 (reseta RP0)
        CLRF 06H  ; Zera todos os pinos de PortB

LOOP

        BSF 06H,0 ; Liga RB0 (bit 0 de PortB)

        BCF 06H,0 ; Desliga RB0
        BSF 06H,1 ; Liga RB1 (bit 1 de PortB)

        BCF 06H,1 ; Desliga RB1
        BSF 06H,2 ; Liga RB2 (bit 2 de PortB)

        BCF 06H,2 ; Desliga RB2
        BSF 06H,3 ; Liga RB3 (bit 3 de PortB)

        BCF 06H,3 ; Desliga RB3
        BSF 06H,4 ; Liga RB4 (bit 4 de PortB)

        BCF 06H,4 ; Desliga RB4
        BSF 06H,5 ; Liga RB5 (bit 5 de PortB)
```

CONTINUAÇÃO...

```
        BCF 06H,5 ; Desliga RB5
        BSF 06H,6 ; Liga RB6 (bit 6 de PortB)

        BCF 06H,6 ; Desliga RB6
        BSF 06H,7 ; Liga RB7 (bit 7 de PortB)

        BCF 06H,7 ; Desliga RB7
        BSF 06H,6 ; Liga RB6 (bit 6 de PortB)

        BCF 06H,6 ; Desliga RB6
        BSF 06H,5 ; Liga RB5 (bit 5 de PortB)

        BCF 06H,5 ; Desliga RB5
        BSF 06H,4 ; Liga RB4 (bit 4 de PortB)

        BCF 06H,4 ; Desliga RB4
        BSF 06H,3 ; Liga RB3 (bit 3 de PortB)

        BCF 06H,3 ; Desliga RB3
        BSF 06H,2 ; Liga RB2 (bit 2 de PortB)

        BCF 06H,2 ; Desliga RB2
        BSF 06H,1 ; Liga RB1 (bit 1 de PortB)

        BCF 06H,1 ; Desliga RB1

        GOTO LOOP
    END
```

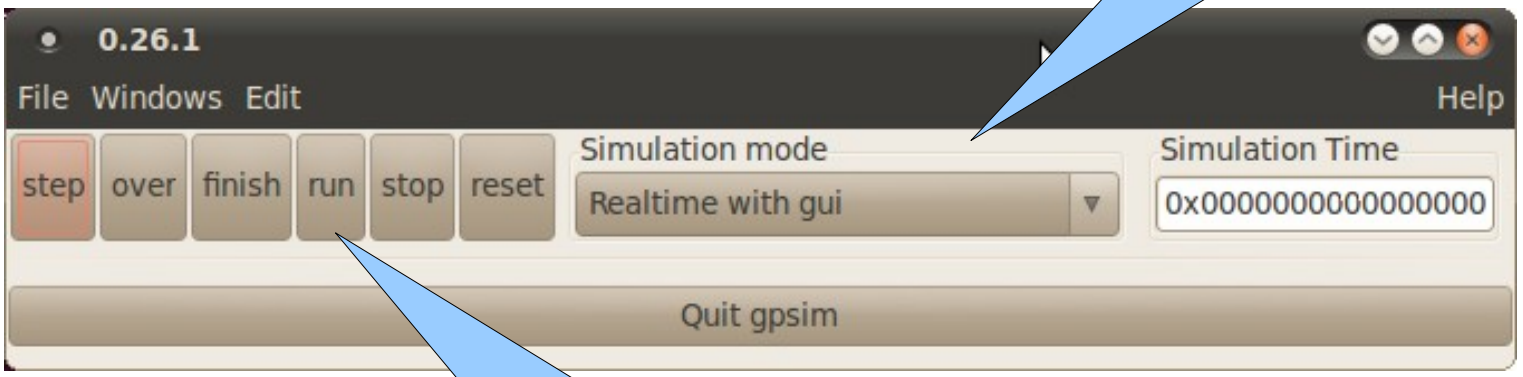
Processo de Montagem



Simulação

$ gpsim ex1-0.cod

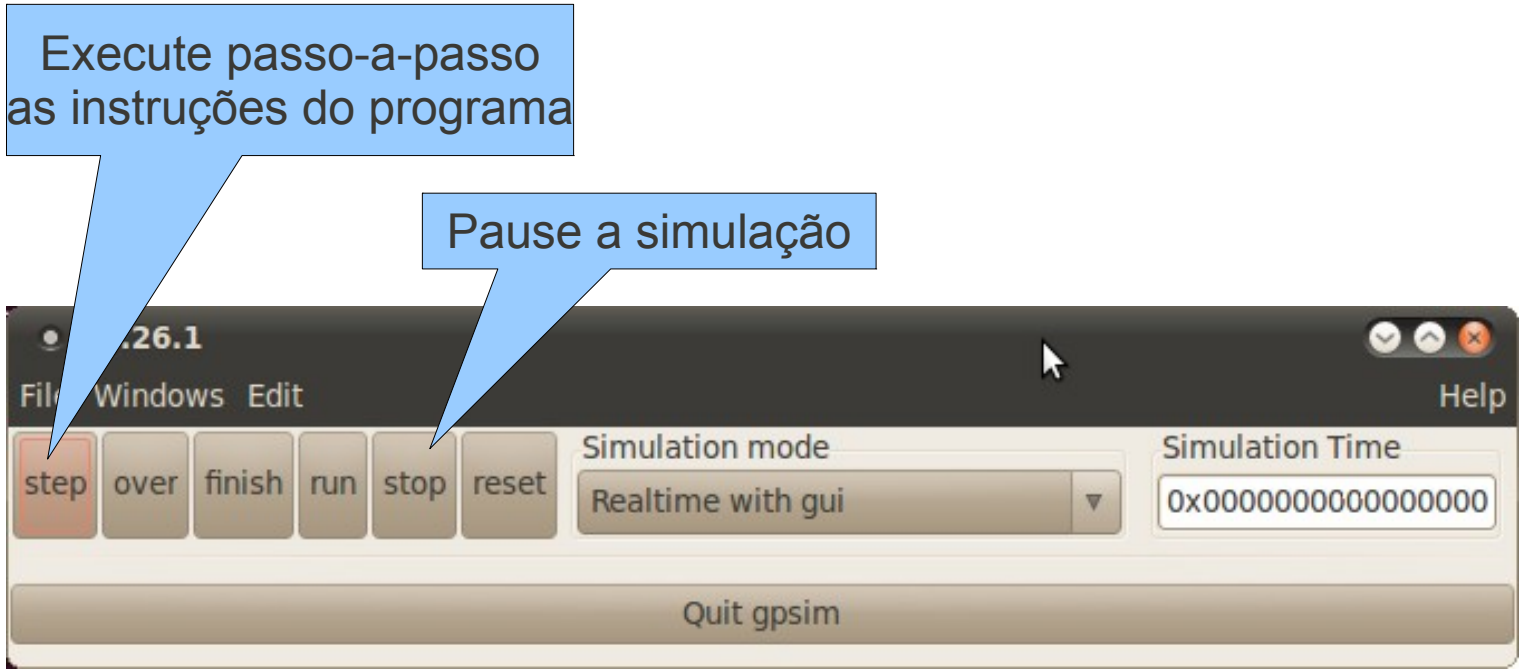
Escolha Realtime with GUI
(simulação em tempo real)



Inicie a simulação
Verifique a Breadboad

Observe que durante a simulação as saídas do controlador não estão alternando entre acesso e apagado com esperado.

Simulação



No passo-a-passo o comportamento é o esperado. **Qual será o problema?**
Resposta: Próximo slide.

Resposta: Temporização

Temporização

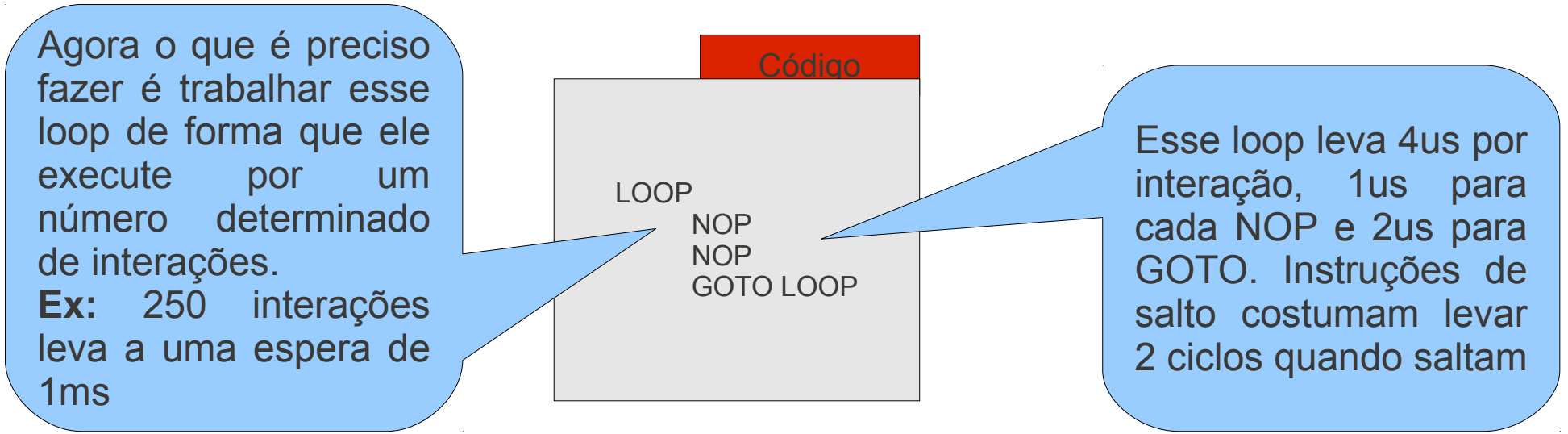
- Entende-se por temporização a capacidade de programar um evento para acontecer após a passagem de um tempo específico.
- No PIC não existem instruções que permitam fazer o controlador esperar um determinado intervalo de tempo, desta forma é responsabilidade do programador prover esse tipo de funcionalidade.
- Existem 2 formas de temporização com o PIC, a primeira e mais simples é a “queima” de ciclos. A segunda é usar os timers internos do PIC para essa finalidade. Aqui veremos apenas a primeira.

Temporização

- A palavra de configuração que estamos usando aqui configura o oscilador interno do PIC que é de 4Mhz.
- O PIC executa instruções a ¼ da velocidade do oscilador o que nos leva a uma taxa de 1 milhão de instruções por segundo, ou seja uma instrução demora 1 microssegundo para ser executada.
- Com base nisto se quisermos um tempo de espera 1s temos que fazer o PIC executar 1 milhão de instruções. Mas como fazer isso?

Temporização

- Para prover a temporização usaremos loops com a instrução: NOP (No OPeration) que serve apenas para “Queimar” um ciclo
- Observe o código abaixo de loop infinito:



Temporização

Essas instruções inicializam o endereço de memória com o valor 250.

Código

```
        MOVLW d'250'
        MOVWF 20H
DL_1
        NOP
        DECFSZ 20H,1
        GOTO DL_1
```

DECFSZ
Essa Instrução decrementa o conteúdo da posição 20h até este chegar a zero. Quando o valor armazenado em 20H chegar a zero, a instrução DECFSZ efetuará o salto da próxima instrução (GOTO DL_1) e sairá do loop. Resultado: 1ms de espera (250x4us).

Em termos da percepção humana, 1ms é muito pouco. Para o olho humano perceber, com certa persistencia precisamos de um intervalo de pelo menos 1/50s ou seja 20ms

Temporização

DELAY1MS
Esse conjunto de instruções foi transformado em uma **FUNÇÃO**. Isso permite que em qualquer parte do código o programador execute esse conjunto de instruções apenas executando **CALL DELAY1MS**. Isso tem algumas vantagens a principal delas é não precisar replicar esse código em todas as partes do programa

Código

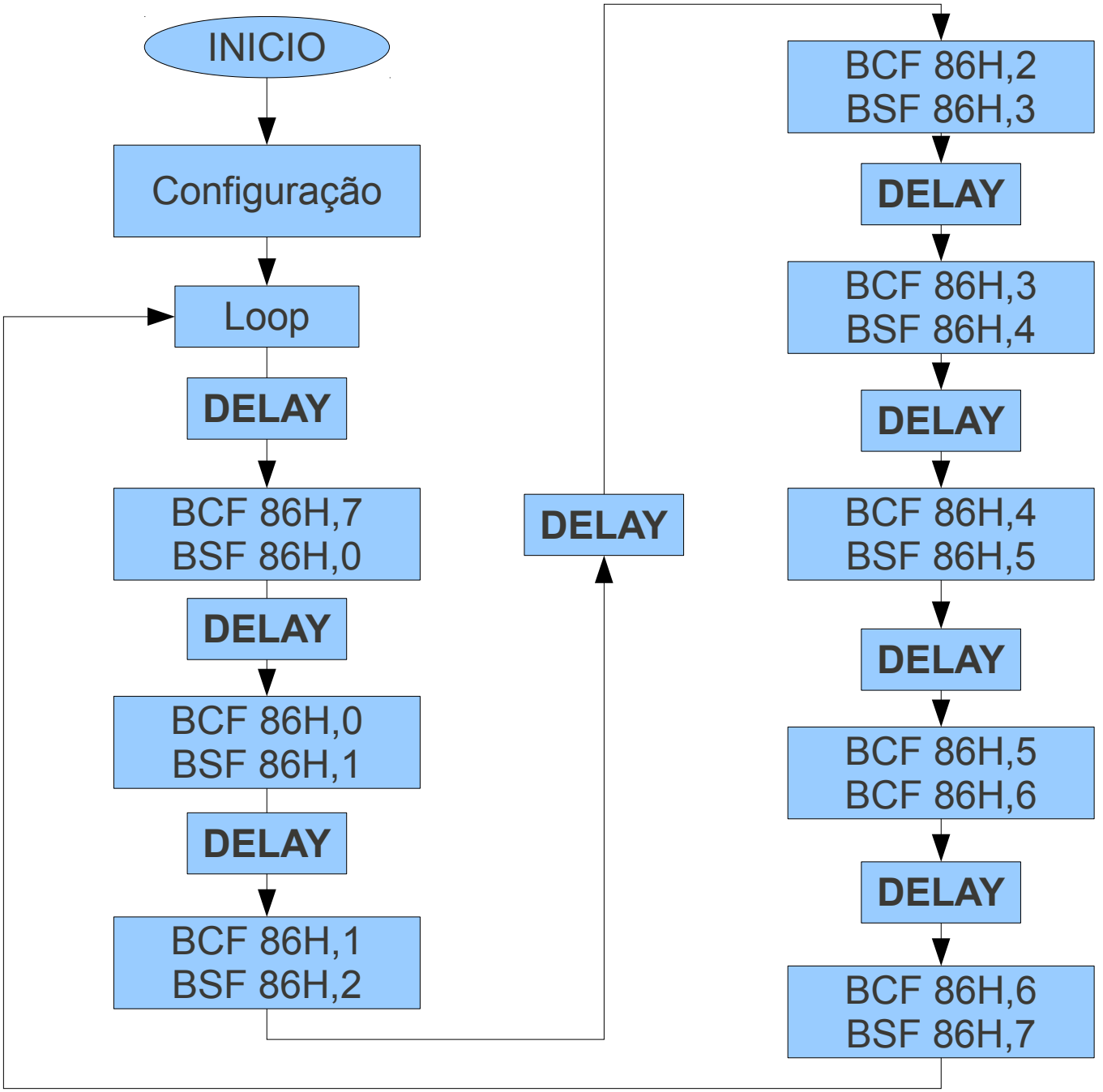
```
DELAY1MS
        MOVLW d'250'
        MOVWF 20H
DL_1
        NOP
        DECFSZ 20H,1
        GOTO DL_1
        RETURN
```

DELAY
Esse laço executa 250 vezes, a cada vez que executa chama a função DELAY1MS 4 vezes. Ou seja uma espera aproximada de 1s. Convém transformar esse código em função.

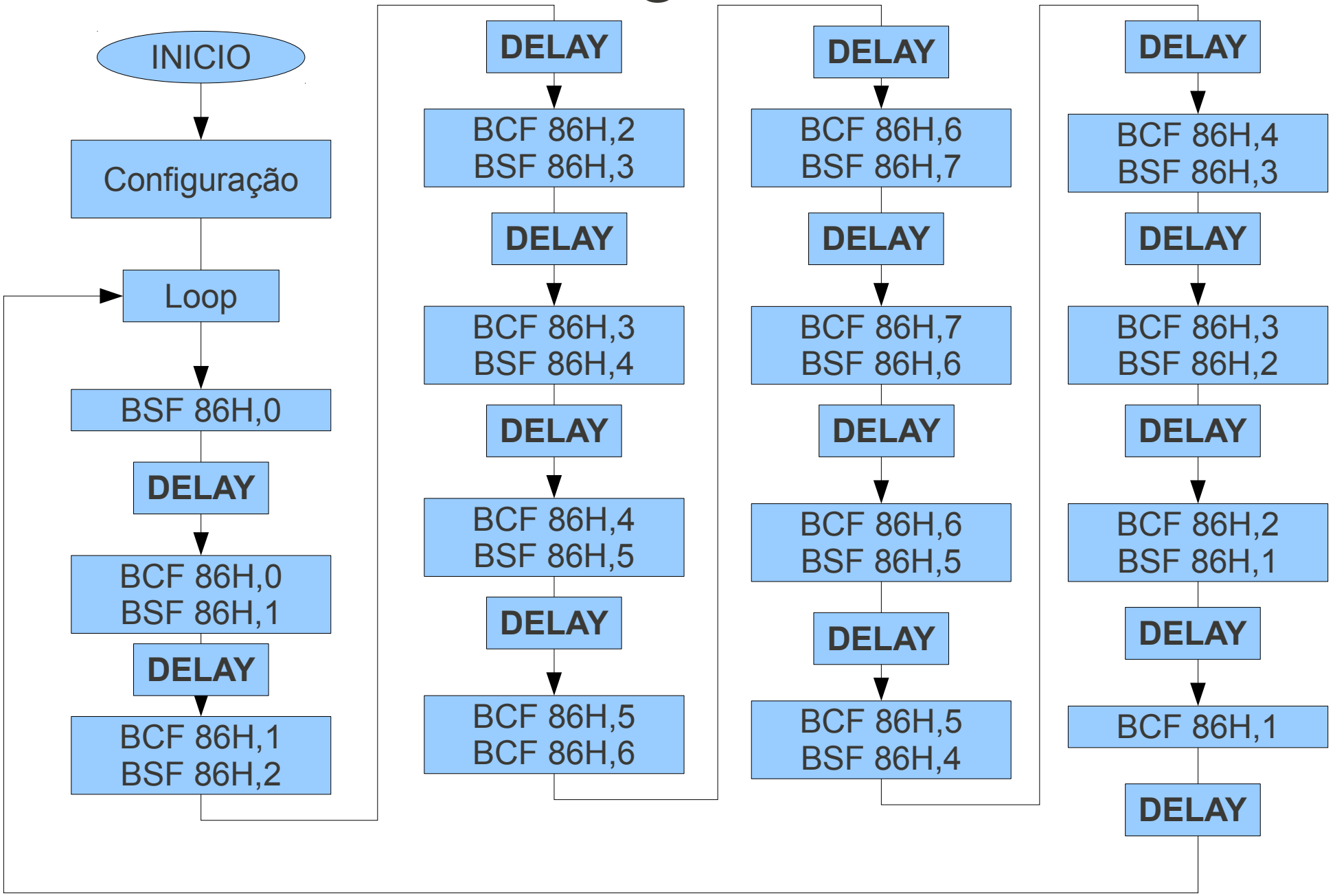
Código

```
        MOVLW d'250'
        MOVWF 21H
DL_2
        CALL DELAY1MS
        CALL DELAY1MS
        CALL DELAY1MS
        CALL DELAY1MS
        DECFSZ 21H,1
        GOTO DL_2
```

Fluxograma (com temporização)



Fluxograma


INICIO
Configuração
Loop
BSF 86H,0
DELAY
BCF 86H,0
BSF 86H,1
DELAY
BCF 86H,1
BSF 86H,2
DELAY
BCF 86H,2
BSF 86H,3
DELAY
BCF 86H,3
BSF 86H,4
DELAY
BCF 86H,4
BSF 86H,5
DELAY
BCF 86H,5
BCF 86H,6
DELAY
BCF 86H,6
BSF 86H,7
DELAY
BCF 86H,7
BSF 86H,6
DELAY
BCF 86H,6
BSF 86H,5
DELAY
BCF 86H,5
BSF 86H,4
DELAY
BCF 86H,4
BSF 86H,3
DELAY
BCF 86H,3
BSF 86H,2
DELAY
BCF 86H,2
BSF 86H,1
DELAY
BCF 86H,1
DELAY

Código com temporização

Código: ex1-1.asm

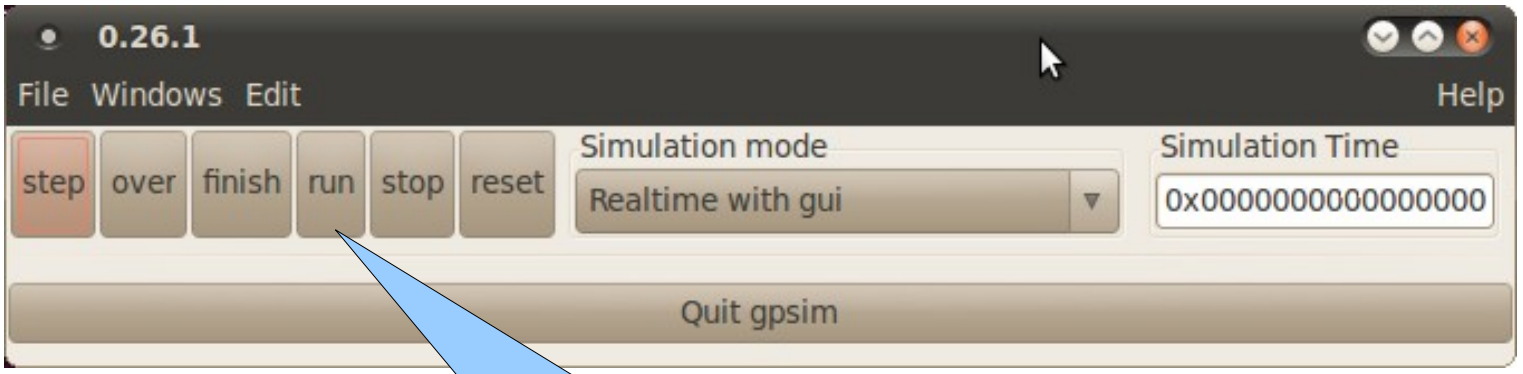
```
__CONFIG  0x3D18
ORG 0x00
CONFIGURACAO
        BSF 03H,5 ; Seleciona banco 1 (seta RP0)
        CLRF 86H ; Reseta PortB (Todas os pinos como saida)
        BCF 03H,5 ; Seleciona banco 0 (reseta RP0)
        CLRF 06H  ; Zera todos os pinos de PortB
        GOTO LOOP
DELAY1MS
        MOVLW d'250'
        MOVWF 20H
DL_1
        NOP
        DECFSZ 20H,1
        GOTO DL_1
        RETURN

DELAY1S
        MOVLW d'250'
        MOVWF 21H
DL_2
        CALL DELAY1MS
        CALL DELAY1MS
        CALL DELAY1MS
        CALL DELAY1MS
        DECFSZ 21H,1
        GOTO DL_2
        RETURN

LOOP

        BSF 06H,0 ; Liga RB0 (bit 0 de PortB)

        CALL DELAY1S
        BCF 06H,0 ; Desliga RB0
        BSF 06H,1 ; Liga RB1 (bit 1 de PortB)

        CALL DELAY1S
        BCF 06H,1 ; Desliga RB1
        BSF 06H,2 ; Liga RB2 (bit 2 de PortB)
```

CONTINUAÇÃO...

```
        CALL DELAY1S
        BCF 06H,2 ; Desliga RB2
        BSF 06H,3 ; Liga RB3 (bit 3 de PortB)

        CALL DELAY1S
        BCF 06H,3 ; Desliga RB3
        BSF 06H,4 ; Liga RB4 (bit 4 de PortB)

        CALL DELAY1S
        BCF 06H,4 ; Desliga RB4
        BSF 06H,5 ; Liga RB5 (bit 5 de PortB)

        CALL DELAY1S
        BCF 06H,5 ; Desliga RB5
        BSF 06H,6 ; Liga RB6 (bit 6 de PortB)

        CALL DELAY1S
        BCF 06H,6 ; Desliga RB6
        BSF 06H,7 ; Liga RB7 (bit 7 de PortB)

        CALL DELAY1S
        BCF 06H,7 ; Desliga RB7
        BSF 06H,6 ; Liga RB6 (bit 6 de PortB)

        CALL DELAY1S
        BCF 06H,6 ; Desliga RB6
        BSF 06H,5 ; Liga RB5 (bit 5 de PortB)

        CALL DELAY1S
        BCF 06H,5 ; Desliga RB5
        BSF 06H,4 ; Liga RB4 (bit 4 de PortB)

        CALL DELAY1S
        BCF 06H,4 ; Desliga RB4
        BSF 06H,3 ; Liga RB3 (bit 3 de PortB)

        CALL DELAY1S
        BCF 06H,3 ; Desliga RB3
        BSF 06H,2 ; Liga RB2 (bit 2 de PortB)

        CALL DELAY1S
        BCF 06H,2 ; Desliga RB2
        BSF 06H,1 ; Liga RB1 (bit 1 de PortB)

        CALL DELAY1S
        BCF 06H,1 ; Desliga RB1

        CALL DELAY1S
        GOTO LOOP
END
```

Simulação

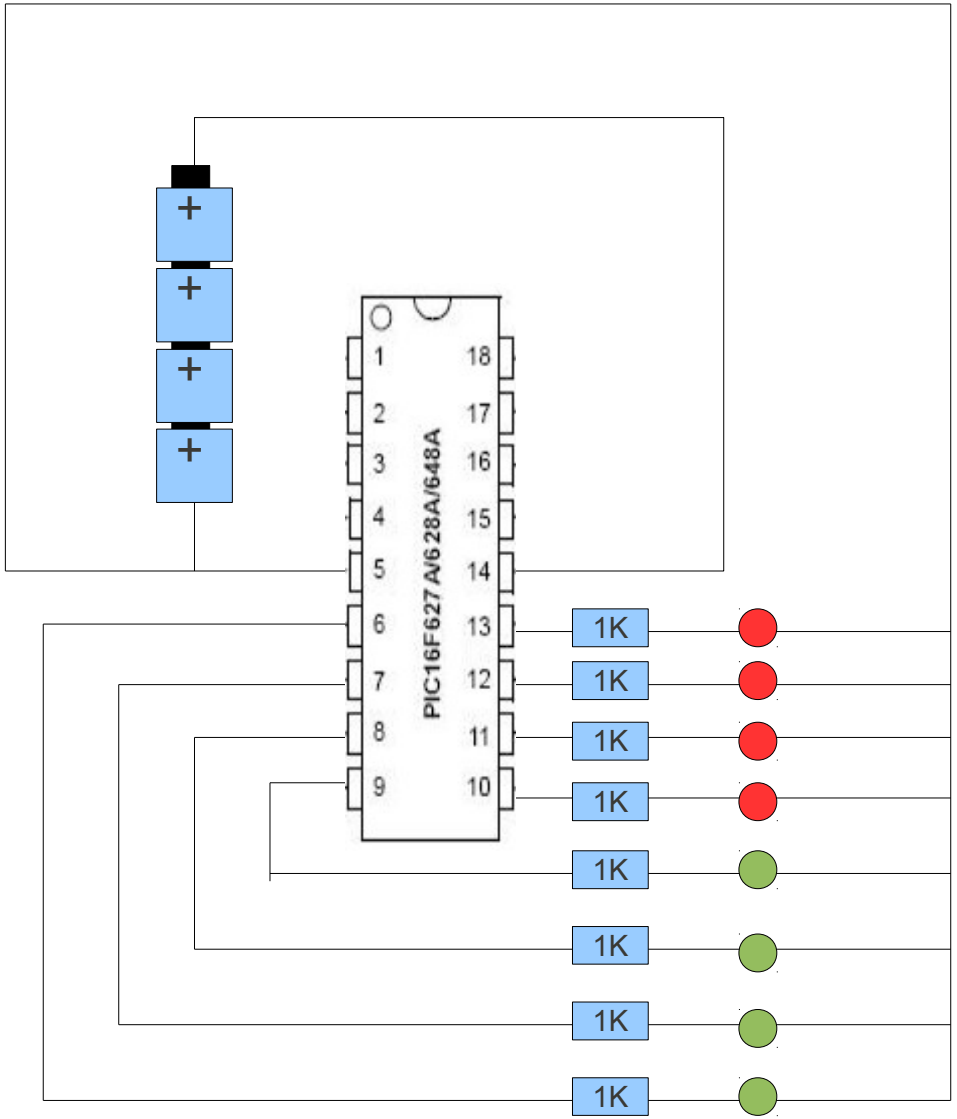
$ gpsim ex1-0.cod



Inicie a simulação
Verifique a Breadboad

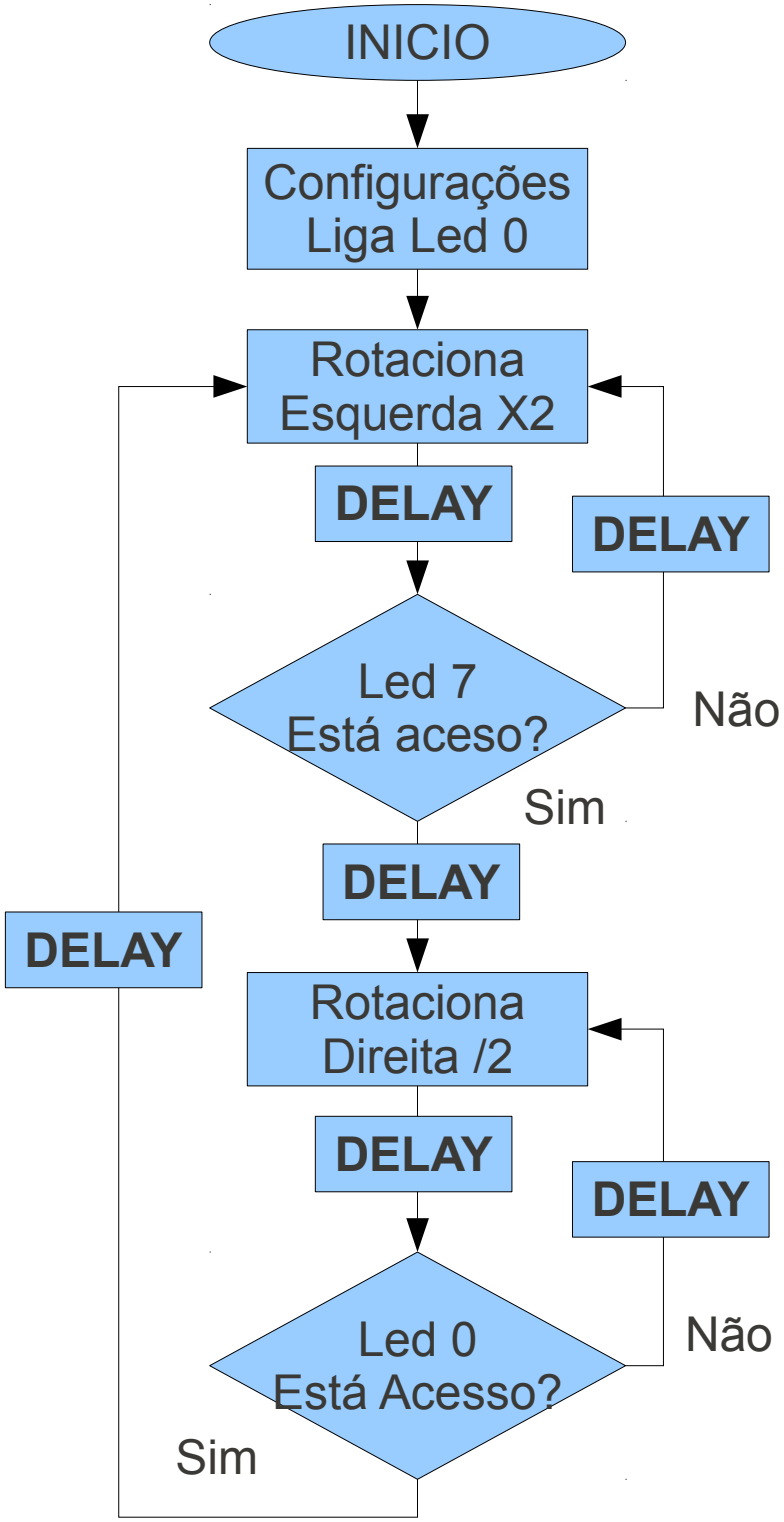
Observe que o funcionamento agora está correto. Parabéns você conseguiu!

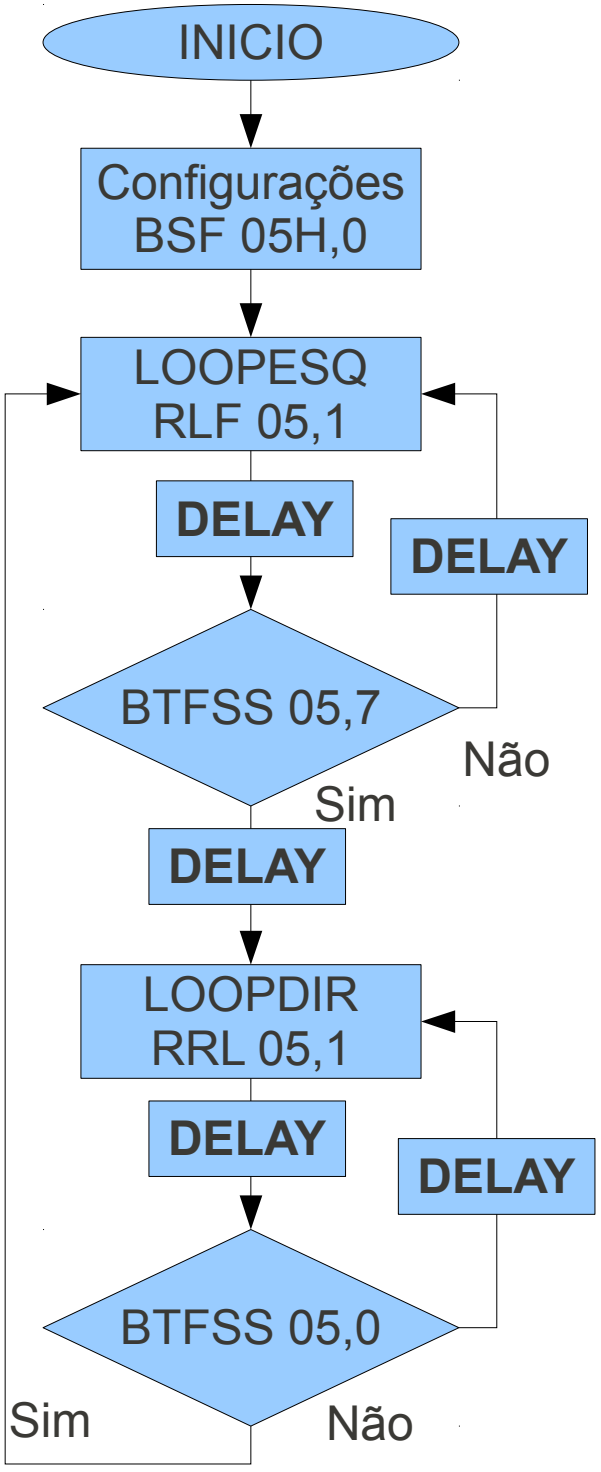
O Circuito


+
+
+
+
PIC16F627A/628A/648A
1
2
3
4
5
6
7
8
9
18
17
16
15
14
13
12
11
10
1K
1K
1K
1K
1K
1K
1K

Melhoramentos deste código

- No programa optamos por ligar e desligar os bits da porta individualmente através das instruções **BCF** e **BSF**.
- Podemos minimizar bastante o código tratando a porta como um byte completo, ao invés de tratar cada bit. Existe duas instruções que proporcionam o mesmo efeito do programa anterior que é basicamente o rotacionamento do byte para direita ou esquerda, essas instruções são **RRF** e **RLF** respectivamente.


INICIO
Configurações
Liga Led 0
Rotaciona
Esquerda X2
DELAY
DELAY
Led 7
Está aceso?
Não
Sim
DELAY
DELAY
Rotaciona
Direita /2
DELAY
DELAY
Led 0
Está Acesso?
Não
Sim


INICIO
Configurações
BSF 05H,0
LOOPESQ
RLF 05,1
DELAY
DELAY
BTFSS 05,7
Não
Sim
DELAY
LOOPDIR
RRL 05,1
DELAY
DELAY
BTFSS 05,0
Sim
Não

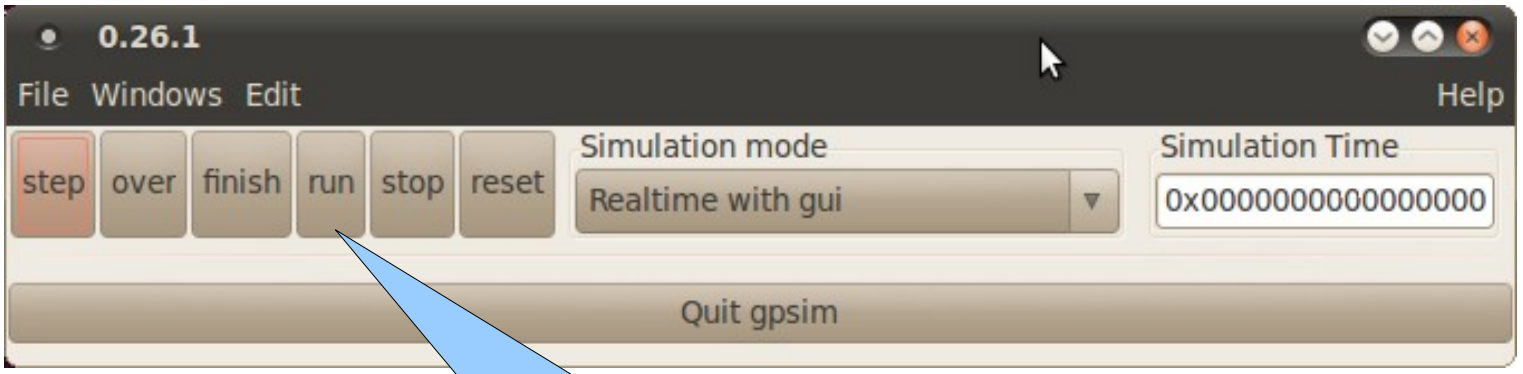
Código com Temporização Melhorado

Código: ex1-2.asm

```
__CONFIG  0x3D18
ORG 0x00
        GOTO CONFIGURACAO
DELAY1MS                ; Funcao de Espera 1ms
        MOVLW d'250'    ; $(20H) = 250
        MOVWF 20H       ;
DL_1
        NOP             ; Queima 1 Ciclo
        DECFSZ 20H,1    ; Queima 1 Ciclo se <> 0
        GOTO DL_1       ; Queima 2 Ciclos
                        ; Total 4 ciclos X 250 = 1000 ciclos gastos
                        ; 1000 X 0,000001s = 0,001s = 1ms
        RETURN                  ; Retorna

DELAY1S                         ; Funcao espera 1s
        MOVLW d'50'     ; $(21h) = 250
        MOVWF 21H       ;
DL_2
        CALL DELAY1MS
        CALL DELAY1MS
        CALL DELAY1MS
        CALL DELAY1MS
        DECFSZ 21H,1
        GOTO DL_2
        RETURN          ; Retorna

CONFIGURACAO
        BSF 03H,5 ; Seleciona banco 1 (seta RP0)
        CLRF 86H ; Reseta PortB (Todas os pinos como saida)
        BCF 03H,5 ; Seleciona banco 0 (reseta RP0)
        CLRF 06H  ; Zera todos os pinos de PortB
        BSF 06H,0 ; Liga RB0 (bit 0 de PortB)
LOOPESQ
        CALL DELAY1S
        RLF 06H,1  ; Rotaciona para esquerda
        BTFSS 06H,7 ; Sai do Loop se led 7 acesso
        GOTO LOOPESQ
LOOPDIR
        CALL DELAY1S
        RRF 06H,1 ; Rotaciona para direita
        BTFSS 06H,0 ; Sai do Loop se led 7 acesso
        GOTO LOOPDIR
        GOTO LOOPESQ
END
```

Simulação

$ gpsim ex1-0.cod



Inicie a simulação
Verifique a Breadboad

Observe que o funcionamento TAMBEM está correto. Só que agora com muito menos linhas de código e consequentemente menos espaço na memória do programa.

Sugestão de Atividades

- 1) diminua o tempo de forma a parecer mais continuo o deslocamento do led acesso;
- 2)Altere o código de forma que no ligamento do circuito todos os leds pisquem 5 vezes como forma de teste;
- 3) Altere o código para que existam 2 leds sempre acessos um se deslocando para direita e outro para esquerda;
- 4) Altere o código de forma que os dois leds acessos voltem quando se encontrarem.