[17. Espaço de I/O (entrada/saída) 17-2](#_Toc257452570)

[17.1 Portos paralelos de entrada e saída 17-2](#_Toc257452571)

[17.1.1 Porto paralelo de saída 17-2](#_Toc257452572)

[17.1.2 Porto paralelo de entrada 17-3](#_Toc257452573)

[17.1.3 Exercício 17-3](#_Toc257452574)

[17.2 GPIO (*General Propose Input Output*) 17-4](#_Toc257452575)

[17.2.1 Exercício 17-6](#_Toc257452576)

# Espaço de I/O (entrada/saída)

O objectivo principal de um CPU é providenciar interlocução com o exterior, no sentido de receber, processar e expedir informação de acordo com algoritmos que lhe são determinados por programas carregados em memória. Esta interlocução é estabelecida através de dispositivos periféricos tais como teclados, monitor, discos, etc., onde o CPU escreve e lê informação. A escrita e a leitura nos dispositivos periféricos é realizada através de portos de entrada/saída (Input/Output), e que são acedidos pelo CPU de uma forma semelhante à que acede aos dispositivos de memória. Na verdade o P16\_V1 não distingue o espaço de memória do espaço de I/O, ou seja, o I/O é tratado como um dispositivo de memória. Esta funcionalidade tem como vantagem o facto de não ser necessário disponibilizar instruções específicas para a entrada e saída de dados. A desvantagem é tornar a descodificação de endereços mais complexa devido à heterogeneidade existente entre as dimensões dos dispositivos de memória e os de I/O.

## Portos paralelos de entrada e saída

Denomina-se porto paralelo a um porto de entrada ou de saída, que disponibiliza vários bits em simultâneo. Estes portos poderão ser utilizados para controlo individual de actuadores, observação instantânea de sensores ou troca de informação entre sistemas. Na troca de informação entre sistemas é necessário recorrer a um protocolo de validação e controlo de fluxo, ou seja, um conjunto de regras estabelecidas entre os interlocutores que assegure a correcta transferência de informação entre ambos.

### Porto paralelo de saída

Um porto paralelo de saída é constituído por um registo inserido no espaço de endereços do CPU no qual é possível escrever uma palavra. A palavra escrita no registo fica disponível em pinos através de um andar de saída, que lhe confere uma impedância capaz de interligar o CPU com os circuitos lógicos digitais mais comuns (Ex. TTL) bem como outro tipo de electrónica. A Figura 17‑1 apresenta a estrutura característica de um porto de saída de 8 bits. O registo poderá ter característica latch ou edge-triggered, desde que a memorização se dê num momento em que os dados no bus se encontrem estáveis.



Figura 17‑1 – Porto de Saída de 8 bits

### Porto paralelo de entrada

Um porto paralelo de entrada, é constituído por um buffer TRISTAT que, quando endereçado pelo CPU, coloca em baixa impedância no bus de dados do CPU a informação presente nos pinos de entrada do porto. A Figura 17‑2 apresenta a estrutura base de um porto de entrada paralelo de 16 bits.



Figura 17‑2 – Porto de Entrada de 16 bits

### Exercício

Utilizando o processador P16, realizar um sistema para controlo de uma lâmpada temporizada L, accionada por um botão B. Ao premirmos o botão B o sistema tem o seguinte comportamento: se a lâmpada L se encontrar apagada, acende-se de imediato e permanece acesa durante 10 segundos, se já se encontrar acesa apaga-se de imediato.

Admita que: o botão B está ligado ao bit de peso 2 de um porto de entrada de 8 bits, localizado no endereço 0x8000, a lâmpada L está ligado ao bit de menor peso de um porto de saída de 8 bits, localizado no endereço 0x8000 e o CPU tem um clock de 1KHz (T=1ms).

.EQU B\_POSITION,3

.EQU DELAY\_5S,1000/50

.code

main:

mov r0,#0

movt r0,#0x80

mov r1,#0

str r1,[r0] ;apaga a lâmpada L

;detectar transição ascendente em B

main\_1:

ldr r1,[r0]

lsrs r1,r1,#B\_POSITION

bcs main\_1

main\_2: ldr r1,[r0]

lsrs r1,r1,#B\_POSITION

bcc main\_2

;detectou transição ascendente em B

mov r1,#1

str r1,[r0] ;acende a lâmpada L

mov r7,#DELAY\_5S

;aguarda nova transição ascendente em B ou fim de tempo

main\_3: bl delay

bcs main

ldr r1,[r0]

lsrs r1,r1,#B\_POSITION

bcs main\_3

main\_4: bl delay

bcs main

ldr r1,[r0]

lsrs r1,r1,#B\_POSITION

bcc main\_4

;detectou transição ascendente em B

b main

delay: sub r7,r7,#1

mov pc,rl

## GPIO (General Propose Input Output)

Em sistemas baseados em microprocessadores é usual dispormos de dispositivos de entrada/saída em paralelo com uma arquitectura programável. A arquitectura ser programável/configurável, permite uma fácil adaptação às especificidades do sistema que pretendamos construir. A título de exemplo consideremos um GPIO (General Propose Input Output) como é mostrado na Figura 17‑3 e que denominaremos por GPIO\_V1. O GPIO\_V1 está mais adaptado a aplicações de controlo, uma vez que não põe disponíveis sinais de protocolo.



Figura 17‑3 – diagrama do GPIO\_V1

O GPIO\_V1 apresenta a seguinte especificação:

* Cada pino pode ser configurado como entrada ou saída através da escrita no registo CTR. Esta funcionalidade permite uma boa adaptação em termos de número de pinos necessários para entrada e saída. Cada um dos 8 bits do registo CTR determina se o bit do porto é de entrada ou de saída.
* O porto de saída disponibiliza três endereços: um para escrita de uma palavra (bytewide), outro para colocar pinos ao valor lógico “1” (set), e um terceiro para colocar pinos ao valor lógico “0” (clear). Esta funcionalidade diminui a complexidade do programa e o tempo necessário para colocar a um ou a zero um ou mais pinos, sem interferir com os restantes.

Na Tabela 17‑1 são apresentados os endereços e o significado de cada bit do registo de CTR e das várias acções sobre os bits do registo de saída (WIDE, SET e CLR).



Tabela 17‑1 – Registos aplicativos do GPIO\_V1

Na Figura 17‑4 é apresentada o diagrama de blocos de uma possível arquitectura para o GPIO\_V1. Como se pode observar, quando realizamos a leitura do porto também se lêem os valores presentes nos bits configurados como saídas. A implementação assíncrona mostrada na Figura 17‑4 só é possível se o CPU garantir a estabilidade do bus de dados antes da activação do sinal nWE, de outro modo é possível realizar acções set ou reset em células não seleccionadas.



Figura 17‑4 – Diagrama de blocos do GPIO\_V1 implementação assíncrona

Como o PDS16 não garante a estabilidade do bus de dados antes da activação do sinal WE é necessário que implementação seja síncrona. Na Figura 17‑5 é apresentada o diagrama de blocos duma possível implementação utilizando por exemplo uma PAL que disponibiliza saídas registadas com flip-flops tipo D edge-trigger.



Figura 17‑5 – Implementação síncrona do GPIO\_V1

### Exercício

Sobre uma fila de 4 LEDs, deslocar para a direita ou para a esquerda um LED aceso ao ritmo das transições ascendentes do sinal CLK. O sentido de deslocação é estabelecido pelo sinal DIR que, quando a zero, indica deslocamento para a direita. Existe ainda um LED denominado MAX que, quando aceso, indica que a fila já se deslocou consecutivamente na presente direcção mais que quatro vezes. O MAX apaga quando existe mudança de direcção. Utilizaremos para a fila de LEDs os bits de menor peso do GPIO (P0 a P3), o LED MAX em P4, o sinal CLK em P5 e o sinal DIR em P6.

.equ gpio\_base\_addr,0xFF

.equ GPIO\_CTR\_ADDR,0

.equ GPIO\_WIDE\_ADDR,1

.equ GPIO\_SET\_ADDR,2

.equ GPIO\_CLR\_ADDR,3

.equ MAX\_POS,5

.equ CLK\_POS,6

.equ DIR\_POS,7

.equ DIR\_MASK,0x40

.equ MAX\_MASK,0x20

.section start

jmp main

.section main

;void initGPIO(char \* ptr); inicia o GPIO com os bits de 0 a 4 output e 5 a 7 input

initGPIO:

mov r1,#0xff

str r1,[r0,#GPIO\_WIDE\_ADDR]

mov r1,#00011111b ;programar P0-4 output, P5-7 input

stb r1,[r0,#GPIO\_CTR\_ADDR]

mov pc,lr

; void getClkAscTrans(char \* ptr); detecta transição ascendete do CLK

getClkAscTrans:

ldr r1,[r0,#GPIO\_WIDE\_ADDR] ;input P0-7

lsrs r1,r1,#CLK\_POS ;cy=CLK

bcs getClkAscTrans ; while(CLK)

clkT\_1:

ldr r1,[r0,#GPIO\_WIDE\_ADDR] ;input P0-7

lsrs r1,r1,#CLK\_POS ;cy=CLK

bcc clkT\_1 ; while(!CLK)

mov pc,lr

; void setBitMax(char \* ptr); set do bit MAX

setBitMax:

mov r1,#MAX\_MASK

str r1,[r0,#GPIO\_SET\_ADDR]

mov pc,lr

; void clrBitMax(char \* ptr); clear do bit MAX

clrBitMax:

mov r1,#MAX\_MASK

str r1,[r0,#GPIO\_CLR\_ADDR]

mov pc,lr

;void roda(char port\_in,char \* ptr);roda esquerda direita (esquerda se DIR=0)

roda:

lsrs r1,r1,#DIR\_POS

bcs esquerda

ldr r2,roda\_image ;roda para adireita

rrl r2,r2,#1

st r2,roda\_image

roda\_1:

shl r2,r2,#12,0 ; roda\_image & 0x0f

shr r2,r2,#12,0

ldi r3,#MAX\_MASK ; port & MAX\_MASK

anl r1,r1,r3

orl r2,r2,r1

st r2,[r0,#GPIO\_WIDE\_ADDR] ; port= (roda\_image & 0x0f) | (port & MAX\_MASK)

ret

esquerda:

ld r2,roda\_image

rrl r2,r2,#15 ; equivale a rodar para a esquerda uma vez

st r2,roda\_image

jmp roda\_1

main:

mov r0,#gpio\_base\_addr

movt r0,#gpio\_base\_addr >> 8

bl initGPIO

mov r1,#00010001b

lsls r2,r1,#8

orl r7,r1,r2 ;r7=0001000100010001b

mov r1,#1

str r1,[r0,#GPIO\_WIDE\_ADDR] ;LED da direita aceso

mov r6,#0 ;iniciar contador de deslocamentos

ldr r5,[r0,#GPIO\_WIDE\_ADDR] ;old direction

main\_1:

bl getClkAscTrans

ldr r1,[r0,#GPIO\_WIDE\_ADDR]

eors r8,r5,r1

mov r5,r1 ;actualiza old direction

lsrs r2,r2,#DIR\_POS

bcs diferente

mov r9,#0

bl clrBitMax

main\_2:

ld r1,old\_dir

jmpl roda

jmp main\_1

diferente:

ld r1,cont\_max

inc r1

st r1,cont\_max

ldi r2,#4

subs r1,r2,r1

jb main\_2

jmpl setBitMax

jmp main\_2

.section direct\_data

.org 8

old\_dir:

.space 1

cont\_max:

.space 1

roda\_image:

.space 1

.end