O módulo *Keyboard Reader* é constituído por três blocos principais: *i*) o descodificador de teclado (*Key Decode*); *ii*) o bloco de armazenamento (designado por *Ring Buffer*); e *iii*) o bloco de entrega ao consumidor (designado por *Output Buffer*), de acordo com o diagrama representado na Figura 1. Neste caso, o módulo *Control*, implementado em software, é a entidade consumidora. Este módulo tem como seus componentes, Key Decode, Ring Buffer e OutputBuffer, todos internamente ligados e que transmitem os valores Q3:0 e Dval ao UsbPort, como apresentado na figura1

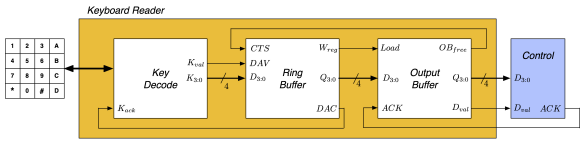


Figura 1: Diagrama de blocos do módulo *Keyboard Reader*

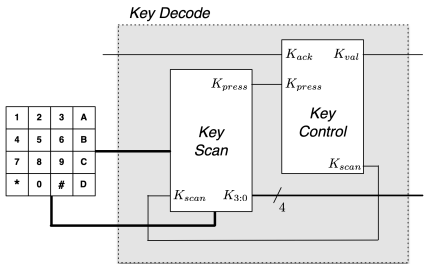
1 Keyboard Reader

1.1 Key Decode

O bloco *Key Decode* implementa um descodificador de um teclado matricial 4x4 por *hardware*, sendo constituído por três sub-blocos: *i*) um teclado matricial de 4x4; *ii*) o bloco *Key Scanner*, responsável pelo varrimento do teclado; e *iii*) o bloco *Key Control*, que realiza o controlo do varrimento e o controlo de fluxo, conforme o diagrama de blocos representado na Figura 2a.

O controlo de fluxo de saída do bloco *Key Decode*, (para o módulo *Control*), define que o sinal K_{val} é ativado quando é detetada a pressão de uma tecla, sendo também disponibilizado o código dessa tecla no barramento $K_{0:3}$. Apenas é iniciado um novo ciclo de varrimento ao teclado quando o sinal K_{ack} for ativado e a tecla premida for libertada. O diagrama temporal do controlo de fluxo está representado na Figura 2b.

O bloco *Key Scanner* foi implementado de acordo com o diagrama de blocos representado na Figura 3. Para este bloco, foi escolhida a 3ª versão, devido ao menor número de clocks (4) necessários para fazer o varrimento das 16 teclas presentes no teclado.



(a) Diagrama de blocos do módulo *Key Decode*

(b) Diagrama temporal

Figura 2: Bloco *Key Decode*

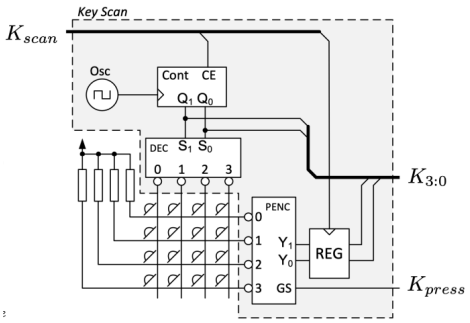


Figura 3: Diagrama de blocos do módulo *Key Scanner*

O bloco *Key Control* foi implementado pela máquina de estados representada em *ASM-chart* na Figura 4, de acordo com o funcionamento esperado do KeyDecode, acima mencionado.

A descrição hardware dos blocos *Key Reader, Key Decode, Key Scanner e Key Control* em *VHDL* encontra-se nos anexos VHDL.

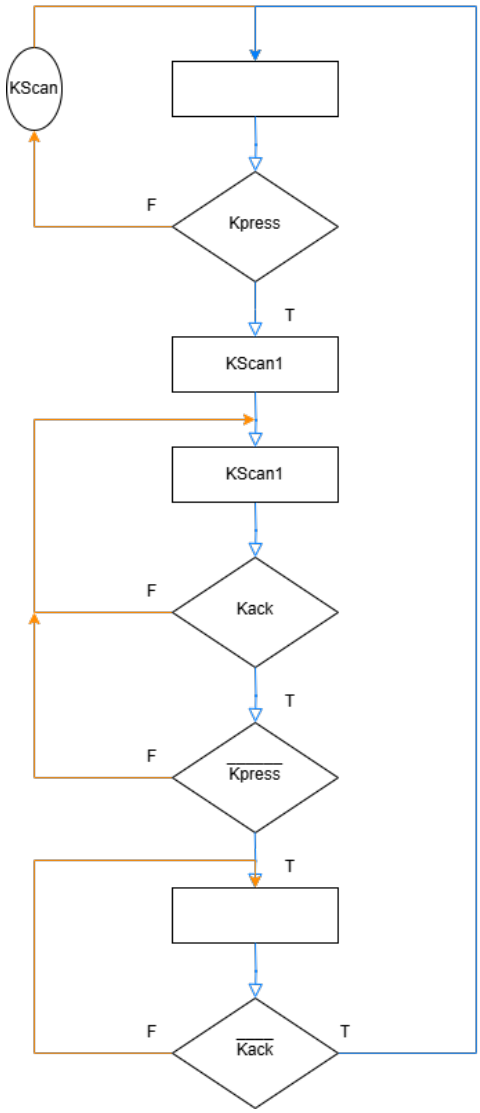


Figura 4: Máquina de estados do *Key Control*.
As linhas a laranja e azul representam o valor *false* e *true* respetivamente.

1.2 Ring Buffer

O bloco *Ring Buffer* é responsável pelo armazenamento temporário dos códigos das teclas premidas, implementando um método de acesso FIFO (*First In, First Out*). A estrutura tem capacidade para armazenar até 16 palavras de 4 bits, garantindo que os dados são entregues à entidade consumidora na mesma ordem em que foram recebidos.

A operação do *Ring Buffer* inicia-se quando o bloco produtor (*KeyDecode*) ativa o sinal **DAV** (*Data Available*), indicando a existência de um novo código a armazenar. Quando o *Ring Buffer* estiver pronto para receber dados, este grava o valor presente no barramento $D_{3:0}$ e ativa o sinal **DAC** (*Data Accepted*) como resposta. O sinal **DAV** deve permanecer ativo até o **DAC** ser ativado. Por sua vez, o *Ring Buffer* apenas desativa o sinal **DAC** após **DAV** ser desativado, assegurando a transferência segura dos dados.

A implementação do *Ring Buffer* assenta numa memória RAM, com dois ponteiros de acesso geridos pelo bloco *MAC (Memory Address Control)*: o **putIndex** para escrita e o **getIndex** para leitura. Estes índices são controlados por sinais de incremento (**incPut** e **incGet**), e o bloco MAC também é responsável por determinar os estados de **cheio (Full)** e **vazio (Empty)** da estrutura.

A entrega dos dados ao consumidor só é realizada quando este sinaliza disponibilidade através do sinal **CTS** (*Clear To Send*). O funcionamento global é ilustrado no diagrama da Figura 5 e a estrutura interna do Ring Buffer é representada pela figura 6.

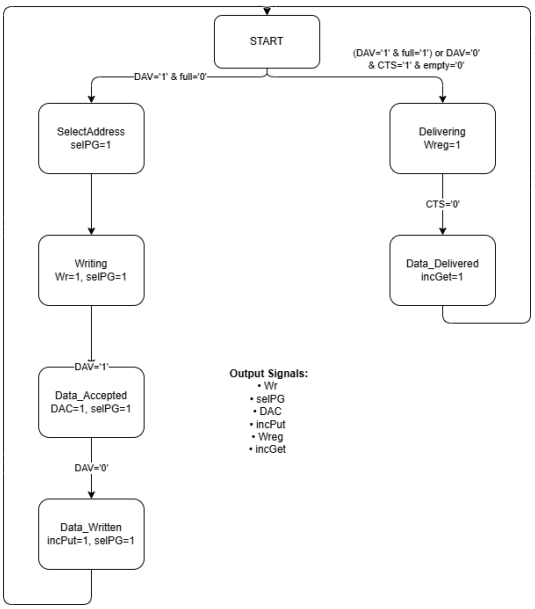


Figura 5: Diagrama da State machine do controlador do *Ring Buffer*.

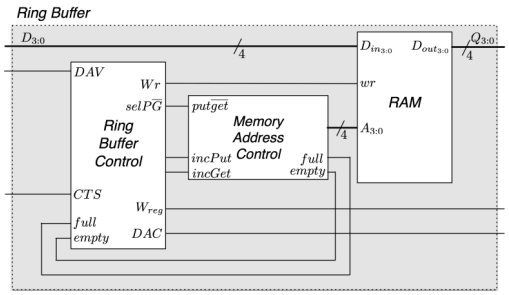


Figura 6: Diagrama de blocos do Ring Buffer

1.3 Output Buffer

O bloco *Output Buffer* estabelece a ligação final entre os dados armazenados e o módulo consumidor (*Control*). Este buffer atua como registo de saída, recebendo os dados provenientes do *Ring Buffer* e disponibilizando-os ao sistema, e está representado na figura 7.

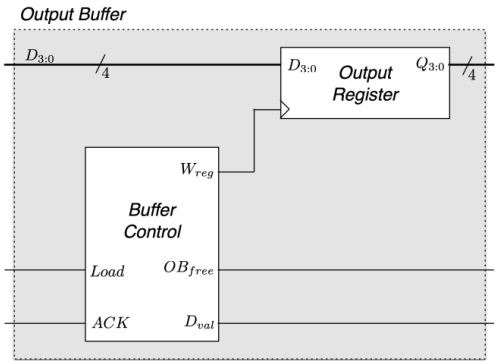


Figura 7: Diagrama de blocos do Output Buffer

Quando o *Output Buffer* está pronto para receber novos dados, ativa o sinal **OBfree**. Neste momento, o *Ring Buffer*, ao detetar este estado e tendo dados disponíveis, realiza a transferência de uma palavra ativando o sinal **Wreg**, após o que o *Output Buffer* armazena o valor e desativa **OBfree**.

O módulo consumidor (*Control*) monitoriza o sinal **Dval**, que indica a presença de dados válidos no *Output Buffer*. Quando este sinal se encontra ativo, o *Control* lê os dados e ativa o sinal **ACK** para confirmar a receção. Após esta confirmação, o *Output Buffer* invalida os dados (desativando **Dval**) e volta a sinalizar disponibilidade (reativando **OBfree**).

O ciclo completo garante a sincronização correta entre a receção e o consumo dos dados, e está representado graficamente na Figura 8.

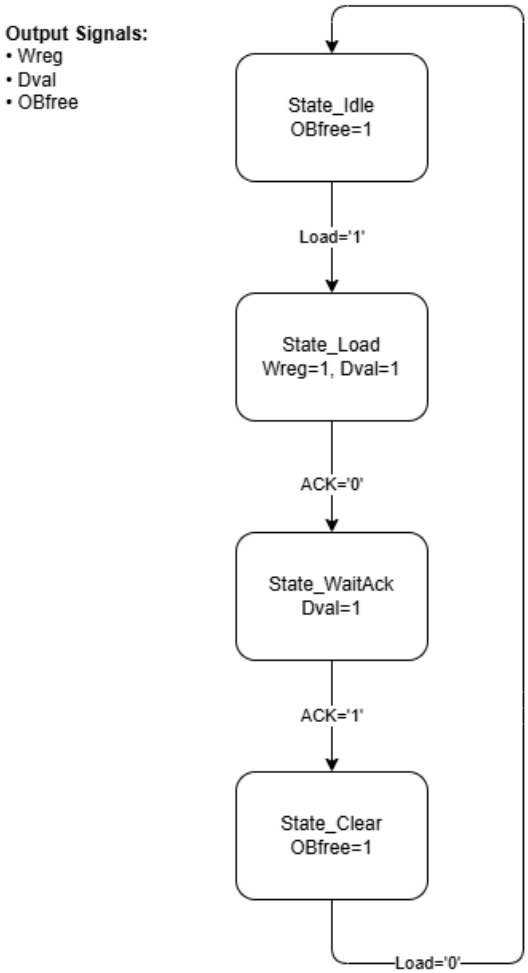


Figura 8: Diagrama de blocos do *Output Buffer*.