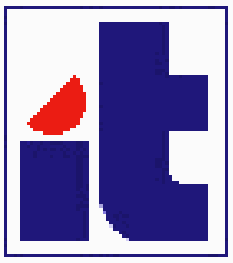
RELATÓRIO DO PROJECTO FINAL DO CURSO EM ENGENHARIA ELECTRÓNICA E TELECOMUNICAÇÕES

"Implementação e Optimização de uma Placa Controladora de um *Loop Óptico*"



Realizado por:

Silvio Jorge Abano Cordeiro N.º Mec. 15193

Orientado por:

Prof. Doutor Armando Nolasco Pinto

Prof. Doutor Rui Sousa Ribeiro

Setembro de 2002

LISTA DE ACRÓNIMOS

ISA – Industry Standard Architecture

PC – Personal Computer

TTL – Transistor-Transistor Logic

FPGA – Field Programmable Gate Array

DAC – Digital to Analogue Converter

PROM – Programmable Read Only Memory

DLL – Dynamic Link Library

PCB – Print Circuit Board

SMB – Sub-Miniature B

ÍNDICE

1. INTRODUÇÃO

Um sistema de transmissão óptico é, usualmente, constituído por um conjunto de sub-sistemas ópticos interligados. Por exemplo, um troço de fibra com comprimento de 100 km e um amplificador óptico podem ser vistos como um sub-sistema, que é repetido 10 vezes num sistema de transmissão ponto-a-ponto de 1000 km.

Para efeitos de estudo e análise laboratorial a transmissão de um sinal óptico no sistema anteriormente referido, pode ser emulada pela transmissão repetida no sub-sistema constituído pelo troço de fibra e amplificador, permitindo assim reduzir a quantidade de componentes necessários ao ensaio laboratorial.

Porém para que tal seja possível é necessário dispormos de um sistema capaz de controlar a propagação e monitoria do sinal óptico de uma forma repetida no sub-sistema em estudo.

Tal sistema é usualmente designado por *loop óptico*. O elemento central do *loop óptico* é a placa controladora, que como o nome indica controla o funcionamento dos vários módulos do *loop óptico* e gera sinais de disparo, síncronos com eventos relevantes, de modo a permitir a monitoria do sinal óptico.

2. O *LOOP ÓPTICO*

O *loop óptico* pode ser decomposto em quatro módulos. Três módulos de interface S1, S2 e S3 e a placa controladora.

- S1 e S2 são comutadores ópticos que permitem ou inibem a passagem do sinal óptico entre, respectivamente, o sub-sistema A e o módulo S3, e o sub-sistema B e o módulo S3, ver figura 1;

- S3 é um acoplador óptico que conjuntamente com os comutadores S1 e S2 permite fazer chegar ao sub-sistema B e C o sinal proveniente do sub-sistema B (fase repetitiva) ou do sub-sistema A (inicialização e reinicialização do processo), ver figura 1;

- a placa controladora vai gerar um conjunto de sinais síncronos de modo a controlar o estado dos comutadores S1 e S2, e de modo a permitir a monitoria do sinal óptico em instantes relevantes. A placa controladora pode funcionar em dois modos distintos, no modo mestre em que o sinal de reinicialização do processo é gerado internamente ou no modo escravo em que o sinal de reinicialização do processo é gerado externamente (*Trigger In*). A placa controladora operara com base num oscilador interno (33 MHz) ou com base num oscilador externo com níveis de tensão *TTL* (nível baixo → 0 V; nível alto → 5 V) .

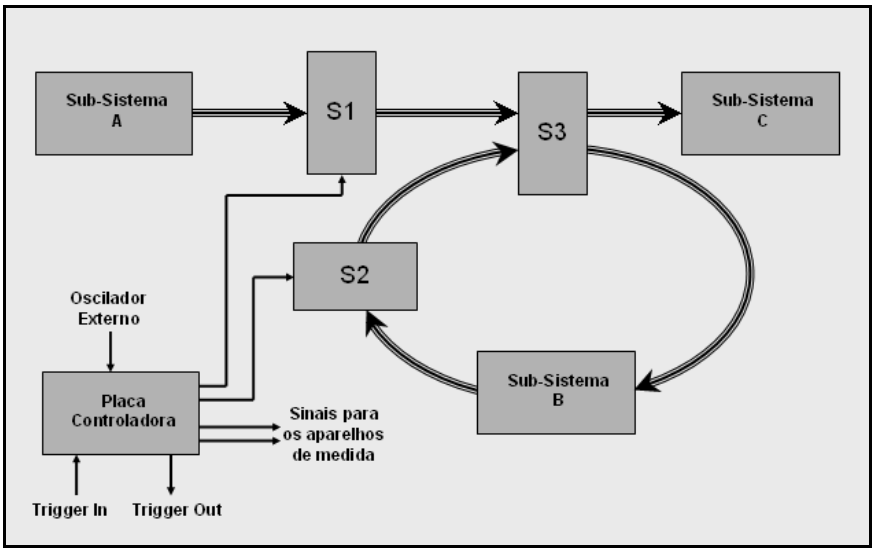


Figura 1 – *Loop Óptico*

Na fase de inicialização os dois comutadores ópticos S1 (fechado) e S2 (aberto) permitem que o sinal óptico passe do sub-sistema A para o sub-sistema B e C. Quando o sinal, que se pretende propagar, tiver atravessado o comutador S1 este é aberto, inibindo assim a passagem de mais sinal, e o comutador S2 é fechado, passando o sistema para a fase repetitiva. Durante a fase repetitiva o sinal vai repetidamente atravessar o sub-sistema B, sendo, em cada passagem pelo módulo S3, uma parte do sinal retirada para o sub-sistema C. Após o sinal ter atravessado o sub-sistema B o número de vezes pretendido, o processo é reinicializado abrindo-se o comutador S2 e fechando-se S1.

3. PLACA CONTROLADORA DO *LOOP ÓPTICO*

A placa controladora tem como função gerar os sinais de controlo para os vários módulos do *loop* e para os equipamento de medida. A placa controladora admite dois sinais de entrada; um relógio externo e um sinal de disparo externo. Estes sinais de entrada são opcionais, no caso de não estarem presente são gerados internamente.

3.1. Especificação

Definiu-se que a placa controladora iria ser incluída num computador do tipo *PC*, como um periférico com ligação ao barramento *ISA*, sendo controlada por intermédio de software.

A placa controladora disponibiliza dois tipos de sinais eléctricos programáveis: sinais de controlo com níveis de tensão programáveis e sinais de sincronismo com níveis de tensão *TTL* .

Os sinais de controlo são definidos no tempo em múltiplos do sinal de relógio e em amplitude entre -10 V e 10 V. O sinal de relógio é obtido dividindo o período de um sinal proveniente de um oscilador, local ou externo, por um número inteiro programável. A opção pelo oscilador local ou pelo oscilador externo é feita pelo utilizador via software.

Os sinais de controlo são definidos por cinco parâmetros (*TCYCLE, TDELAY, TON, VON e VOFF*), ver Figura 2. O *TCYCLE* é um parâmetro que assume o mesmo valor para todos os sinais de controlo e define o período dos sinais de controlo, este parâmetro pode assumir valores entre 0 e 65535 (2 bytes) períodos do sinal de relógio. Os outros quatro parâmetros podem assumir diferentes valores para os diferentes sinais de controlo. O *TDELAY* define o tempo até à transição para o nível de tensão alto, pode assumir valores entre 0 e *TON* períodos do sinal de relógio, o *TON* define o tempo de permanência no nível de tensão alto, pode assumir valores entre *TDELAY* e *TCYCLE* períodos do sinal de relógio, *VON* define o valor de tensão alto e o *VOFF* o valor de tensão baixo, ambos os valores, *VON* e *VOFF*, podem variar entre 0 e 255 (1 byte), correspondendo o 0 ao valor

de tensão de -10 V e o 255 ao valor de +10 V, variando linearmente o valor da tensão com o valor de *VON* e *VOFF*.

O principal sinal de sincronismo fornecido pela placa é o *TRIGGER OUT*, este sinal é activado no final de cada período dos sinais de controlo (*TCYCLE*), ver Figura 2. Estão também disponíveis sinais de disparo associados aos vários sinais de controlo, sendo estes activados quando o valor de tensão transita do nível alto para o nível baixo, ver Figura 2.

A Figura 2 mostra o funcionamento da placa no modo mestre. Neste modo de funcionamento, quando termina um período dos sinais de controlo (*TCYCLE*) um novo período é inicializado. No entanto é possível operar a placa no modo de escravo, neste modo um novo período só é inicializado quando o sinal de *TRIGGER IN* transitar para o nível de tensão alto, ver Figura 3. A selecção do modo de funcionamento da placa (escravo ou mestre) é efectuada pelo utilizador via software.

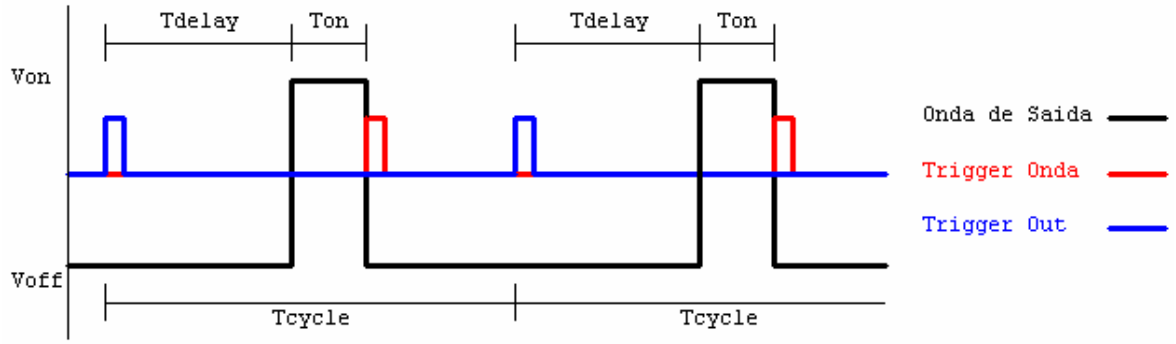


Figura 2 – Placa controladora no modo mestre

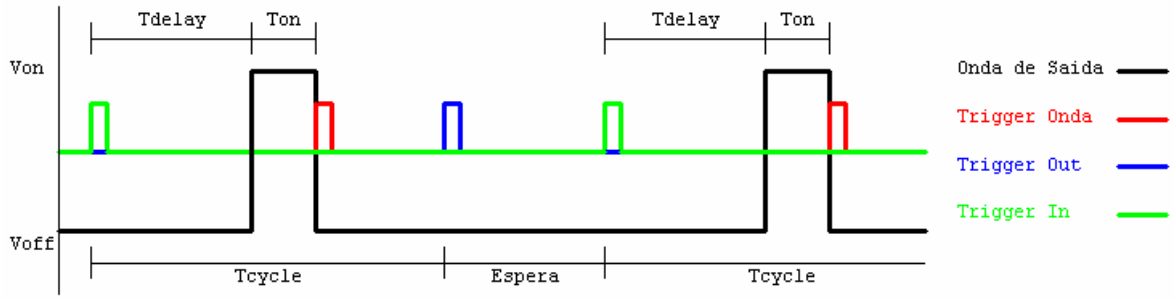


Figura 3 – Placa controladora no modo escravo

3.2. Arquitectura

A arquitectura idealizada para a placa controladora, de acordo com a especificação, é mostrada na Figura 4.

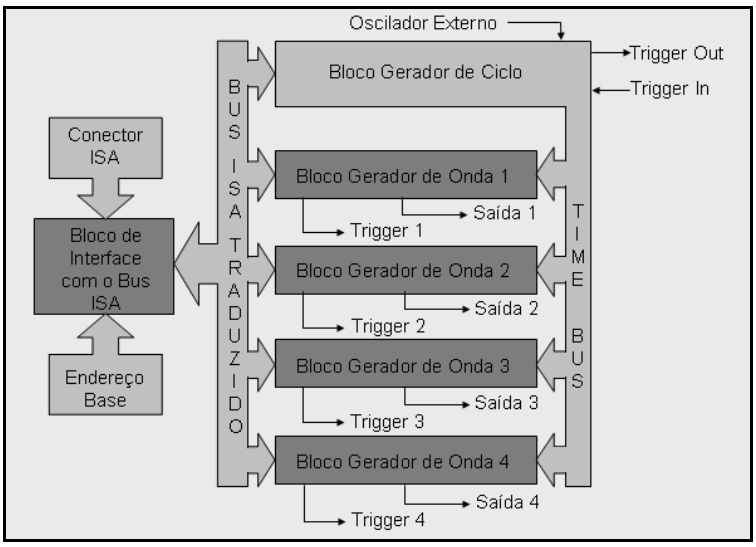


Figura 4 – Arquitectura da placa controladora

A comunicação com o computador é feita através do barramento *ISA*, disponibilizando este 16 bits de endereçamento, bits A0 a A15 figura 5. Os 8 bits mais significativos, A8 ao A15, do endereço são usados para seleccionar a placa controladora, de entre o conjunto dos periféricos do PC. Para isso é especificado um endereço base, por hardware, que é comparado com o endereço proveniente do bus ISA, e com base nesta comparação é gerado o sinal *BSel*. O bit A0 e A7 não são usados, devendo assumir sempre o valor 0. Os restantes 6 bits, são usados para seleccionar os blocos da placa controladora, A4 a A6, e dentro de cada bloco os respectivos registos, A1 a A3, ver figura 4. Dos restantes sinais do bus ISA fazem parte 16 bits de dados e um conjunto de sinais de controlo, designados na figura 5 como Sinais ISA.

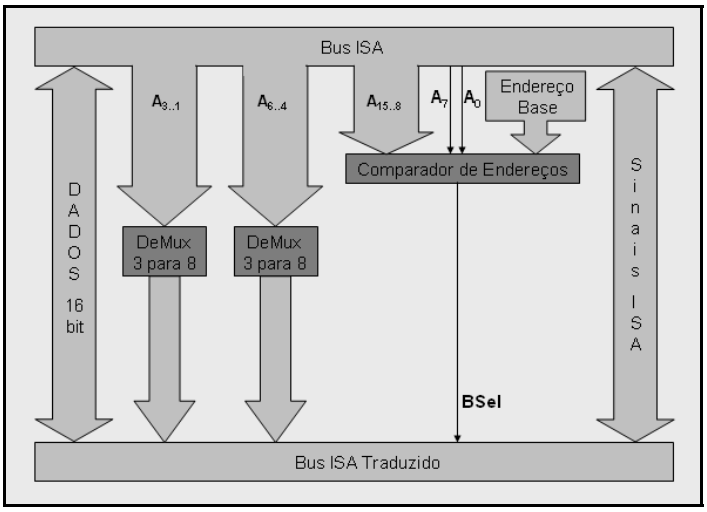


Figura 5 – Arquitectura do bloco de interface com o barramento *ISA*

De modo a garantir que todas os sinais de controlo são síncronos, ou seja, têm o mesmo período e a diferença de fase entre eles é constante, entendendo-se por diferença de fase a diferença entre os instante de transição do nível baixo para o nível alto, ver figura 2 ou 3, há a necessidade de usar um sinal de relógio comum a todos os blocos geradores de onda. Este sinal é gerado no bloco gerador de ciclo e distribuído pelos blocos geradores de onda, na forma de um sinal digital de 16 bits, através do TIME BUS, ver figura 4. Este sinal de relógio é obtido tendo por base o sinal proveniente do oscilador, local ou externo. Este sinal é dividido por um valor especificado pelo utilizador e é introduzido num contador de 16 bits. O contador é colocado a zero quando a sua saída iguala o valor do *TCYCLE*, no modo mestre, a contagem inicia-se imediatamente, no modo escravo a contagem é iniciada pelo sinal *TRIGGER IN*. Um sinal de saída, *TRIGGER OUT*, é activado quando o valor do contador iguala o *TCYCLE*. O registo de controlo, é um registo de 16 bits, em que os 8 bits mais significativos (R8 ao R15) é o valor pelo qual o sinal do oscilador será dividido para obtermos o sinal de relógio, os restantes servem para especificar o oscilador de referência (R0, local ou externo), para activar ou inibir o sinal de relógio (R2), para especificar o modo de funcionamento (R3, mestre ou escravo) e para activar um sinal de disparo por software (R4). Este sinal de disparo por software funciona quando a opção de *TRIGGER IN* está seleccionada. A transição do bit R4, de "0" para "1", é em tudo análoga à transição do nível baixo para o nível alto do sinal de *TRIGGER IN.* Os bits R1 e R5 a R7 não são usados.

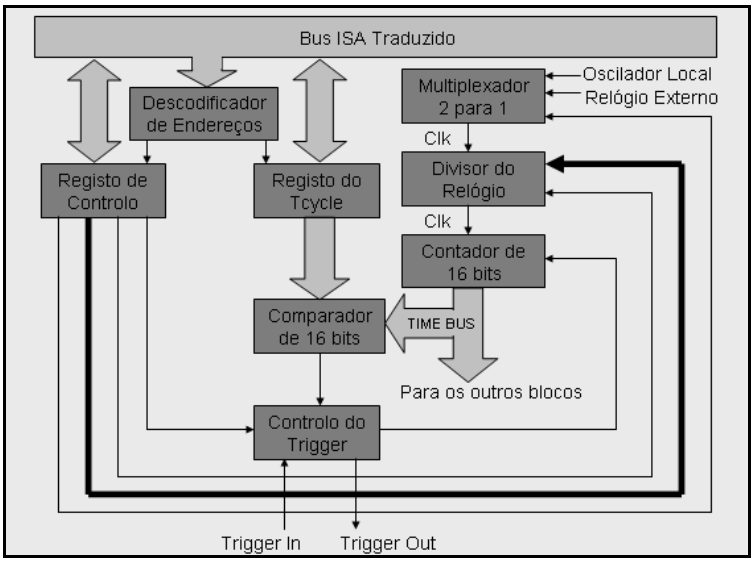


Figura 6 – Arquitectura do bloco gerador de ciclo

A geração das ondas de saída é feita com base nos valores armazenados nos registos *Von*, *Voff*, *Ton* e *Tdelay*. Quando o valor proveniente do *TIME BUS* for menor que o valor de *Tdelay* à saída do bloco temos o valor de *Voff*, quando for maior que *Tdelay* e menor ou igual a *Ton* temos à saída *Von*, para valores superiores de *Ton* temos à saída *Voff*. É ainda possível colocar a saída a zero, independentemente dos valores dos outros registos e do valor do relógio, activando a opção de *Stand*-By.

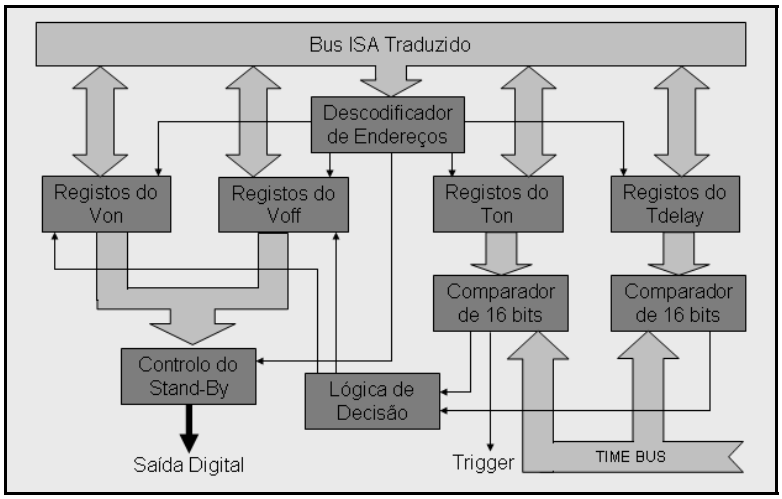


Figura 7 – Arquitectura do bloco gerador de onda

3.3. Implementação

Para implementação da placa controladora optou-se pela utilização de uma *FPGA*, em detrimento da implementação em hardware discreto, o que apresenta um conjunto de

vantagens. Destas são de referir, o menor tamanho e complexidade do hardware, reduzindo-se a uma só placa em vez dum conjunto de placas que implicariam a necessidade de construir uma caixa capaz de englobar todo o sistema, a maior flexibilidade na implementação, pois o desenvolvimento das funcionalidades é todo feito em software em vez de se construir e alterar hardware, a maior imunidade a ruídos externos na interligação dos vários blocos funcionais e por fim o menor consumo de corrente, que possibilita que a placa seja alimentada com as alimentações disponíveis no barramento *ISA* sem a necessidade de se recorrer a uma fonte de alimentação externa ao computador.

No anexo 7.1 é apresentado o esquema utilizado na construção do hardware, do qual se descreve seguidamente os pontos mais relevantes.

Sendo a comunicação com o computador feita através barramento *ISA*, utilizou-se uma configuração em que o periférico, ou seja a placa controladora, é controlada pelo barramento e em que os endereços de 16 bits são mapeados no espaço de endereçamento de entrada/saída do PC.

Para a configuração do endereço base, foram implementadas ligações em ponte (*jumpers*) que permitem ligar os pinos da *FPGA* para este fim dedicados à massa ("0" lógico) ou à alimentação ("1" lógico), definindo-se assim o endereço base. Por defeito o valor deste endereço é 00000001. Este valor deve ser alterado sempre que existam incompatibilidades entre este periférico e os restantes periféricos do PC.

Para a alimentação do sistema desenvolveu-se um bloco de reguladores de tensão, isto na eventualidade das linhas de alimentação do barramento não apresentarem as características desejadas, facto que não se veio a verificar tornando este bloco desnecessário ao bom funcionamento da placa controladora, ou seja a alimentação é retirada directamente do barramento *ISA*.

Visto que para o funcionamento da placa é essencial um sinal de relógio, implementou-se um oscilador local de alto factor de qualidade e precisão, utilizando-se um oscilador a cristal de 33 MHz, assim como uma entrada externa TTL. As características limites do oscilador externo não foram determinadas.

Para a construção das ondas de saída foi desenvolvido um bloco de conversores de digital para analógico (*DAC*) com saídas em corrente e respectivos conversores de corrente para tensão, já que se pretende que os sinais de saída sejam em tensão. O

ajuste dos níveis de tensão e do *overshoot* de cada onda de saída é feito por duas resistências variáveis e por dois condensadores variáveis, devendo-se fazer este ajuste sempre que a placa for alimentada por fontes diferentes, visto as ondas de saída estarem dependentes dos valores das tensões de alimentação.

Cada onda tem associada um sinal de disparo, *TRIGGER*, estes sinais estão disponíveis apenas internamente, nos pinos junto aos pinos para configuração do endereço base. Estes sinais foram apenas usados na fase de teste não estando por isso protegidos por circuitos de protecção. Tal situação poderá ser alterado futuramente seguindo o mesmo tipo de implementação utilizada nos outros sinais.

De referir que todos os pinos da *FPGA* com acesso do exterior foram protegidos por circuitos de protecção, sendo a ligação para o exterior do tipo coaxial *SMB* com impedância característica de 50 Ω.

O circuito implementado, e armazenado na *PROM*, que é carregado na *FPGA* aquando da inicialização do sistema é apresentado no anexo 7.2 e os blocos funcionais são descritos nas secções seguintes.

3.3.1. Bloco de interface com o barramento ISA

Este bloco, apresentado na Figura 8, tem como entradas internas os sinais do barramento ISA e tem como função inibir ou activar a comunicação entre o PC e a placa controladora, assim como traduzir os endereços do barramento *ISA*.

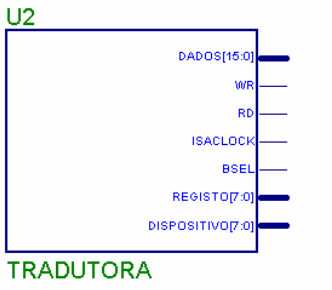


Figura 8 – Bloco de interface com o barramento *ISA*

Para mais fácil compreensão dos protocolos de comunicação com o barramento *ISA* é, de seguida, feita uma breve descrição dos sinais utilizados, encontrando-se informação mais detalhada em http://www.techfest.com/hardware/bus/isa.htm e http://sunsite.tut.fi/hwb/co_ISA_Tech.html.

Como já referido utilizou-se uma configuração em que o barramento controla o periférico, com os endereços mapeados no espaço de endereçamento de entrada/saída do PC. Assim recorreu-se aos seguintes sinais do barramento *ISA*:

–*SA15* a *SA0*: linhas de endereço;

–*SD15* a *SA0*: linhas de dados;

–*I/O READ#*: indica ao periférico seleccionado uma operação de leitura;

–*I/O WRITE#*: indica ao periférico seleccionado uma operação de escrita;

–*CLK*: relógio do barramento *ISA*;

–*I/O CH RDY*: indica ao barramento que o periférico está preparado para comunicação;

–*I/O CS16#*: transferência de dados de 16 bits.

(Nota: o símbolo # indica que o sinal é em lógica negativa)

Numa comunicação de escrita, Figura 9, o barramento começa por activar as linhas de endereço com o endereço do dispositivo a comunicar. Este detecta o seu endereço e activa as linhas *I/O CH RDY* e *I/O CS16#* indicando que está pronto a comunicar com transferência de dados de 16 bits. Seguidamente, o barramento identifica que pretende efectuar uma operação de escrita e activa as linhas de dados com a informação a escrever.

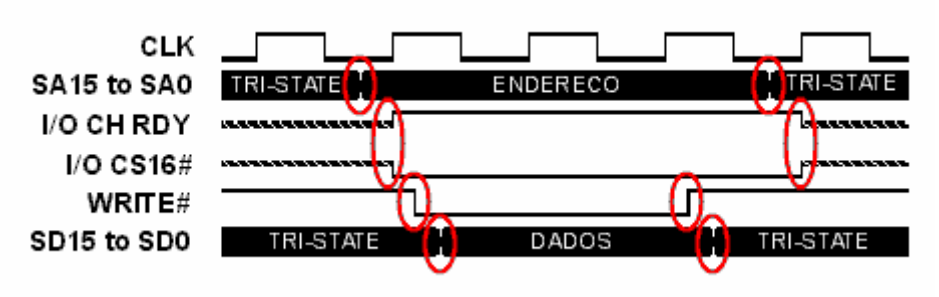


Figura 9 – Ciclo de escrita do barramento *ISA*

Passados dois ciclos do sinal de relógio, aproximadamente, do barramento *ISA* o barramento assume que os dados foram transferidos correctamente. O barramento desactiva as linhas de comunicação com o periférico (I/O WRITE# e DADOS), que por

sua vez detecta que a comunicação cessou e desactiva as linhas *I/O CH RDY* e *I/O CS16#*, dando-se por completo o ciclo de escrita do barramento *ISA*.

Numa comunicação de leitura, Figura 10, o barramento começa activar as linhas de endereço com o endereço do dispositivo a comunicar. Este detecta o seu endereço e activa as linhas *I/O CH RDY* e *I/O CS16#* indicando que está pronto a comunicar com transferência de dados de 16 bits. Seguidamente, o barramento identifica que pretende efectuar uma operação de leitura. Aguarda aproximadamente dois ciclos de relógio e lê a informação disponibilizada pelo periférico, nas linhas de dados.

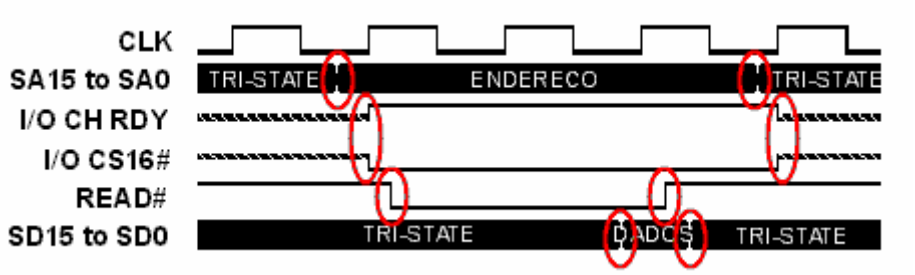


Figura 10 – Ciclo de leitura do barramento *ISA*

Depois do barramento receber a informação a si destinada, desactiva a linha *I/O READ #*. O periférico, por sua vez, desactiva as linhas de dados e espera que o barramento desactive as linhas de endereço. Dado isto, o periférico desactiva as linhas *I/O CH RDY* e *I/O CS16#*, ficando completo o ciclo de leitura do barramento *ISA*.

De referir que, em comunicações com os endereços mapeados no espaço de endereçamento de entrada/saída com 16 bits de dados, só são aceites endereços pares, o que obriga a que a linha *SA0* esteja no nível lógico baixo para que a comunicação seja válida.

Feita esta breve introdução ao barramento *ISA*, descreve-se agora este bloco. Como sinais de interface com os restantes blocos temos:

–Barramento *DADOS*: é do tipo entrada/saída de 16 bits e é utilizado para a transferência de informação entre o barramento *ISA* e os restantes blocos implementados na *FPGA*.

–Barramento *DISPOSITIVO*: é do tipo saída de 8 bits, utilizando-se para seleccionar o bloco a aceder. Sendo que cada bit

selecciona um bloco, pode servir um máximo de 8 blocos.

–Barramento *REGISTO*: é do tipo saída de 8 bits e é utilizado para seleccionar o registo a aceder dentro do bloco referenciado. Sendo que cada bit selecciona um registo, pode referenciar um máximo de 8 registos.

–Sinal *READ*: é do tipo saída de lógica negativa e indica uma operação de leitura.

–Sinal *WRITE*: é do tipo saída de lógica negativa e indica uma operação de escrita.

–Sinal *BSEL*: é do tipo saída e indica aos blocos internos que a placa controladora está seleccionada para comunicação.

–Sinal *ISACLOCK*: é do tipo saída e, sendo um sinal de relógio síncrono com os sinais do barramento *ISA*, poderá ser utilizado para sincronismo.

Internamente, anexo 7.2, temos o comparador *ENDBSEL* que compara o endereço base configurado na placa com endereço presente no barramento *ISA*. Quando os valores são iguais é gerado o sinal *BSEL* que activa a comunicação com a placa controladora. Assim, são activados os sinais *I/O CH RDY* e *I/O CS16#* do barramento *ISA*, é definido o sentido da comunicação mediante os sinais *IOR#* e *IOW#* e descodificados os endereços que definem as saídas, descodificador *ENDDISP*, e que definem os registos, descodificador *ENDREG*.

A correspondência entre o endereço e a saída e o registo a aceder é a seguinte:

A_6	A_5	A_4	LINHA ACTIVA
0	0	0	DISPOSITIVO0
0	0	1	DISPOSITIVO1
0	1	0	DISPOSITIVO2
0	1	1	DISPOSITIVO3
1	0	0	DISPOSITIVO4
1	0	1	DISPOSITIVO5
1	1	0	DISPOSITIVO6
1	1	1	DISPOSITIVO7

A_3	A_2	A_1	LINHA ACTIVA
0	0	0	REGISTO0
0	0	1	REGISTO1
0	1	0	REGISTO2
0	1	1	REGISTO3
1	0	0	REGISTO4
1	0	1	REGISTO5
1	1	0	REGISTO6
1	1	1	REGISTO7

Figura 11 – Descodificação dos endereços

3.3.2. Bloco gerador de ciclo

Este bloco, apresentado na Figura 12, tem como função fazer a contagem dos tempos, sincronizando as varias saídas, assim como introduzir mecanismos de sincronismo com os outros componentes do *loop óptico*.

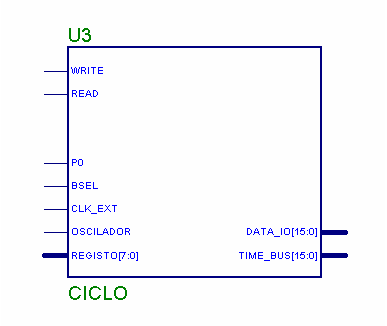


Figura 12 – Bloco gerador de ciclo

Como sinais de interface com os restantes blocos temos:

–Barramento *TIME_BUS*: é do tipo saída de 16 bits e é utilizado como relógio digital para criar um referencial temporal para os restantes blocos.

–Barramento *DATA_IO*: é do tipo entrada/saída de 16 bits e é utilizado para a transferência de informação.

–Barramento *REGISTO*: é do tipo entrada de 8 bits e é utilizado para seleccionar o registo a aceder.

–Sinal *READ*: é do tipo entrada de lógica negativa e indica uma operação de leitura.

–Sinal *WRITE*: é do tipo entrada de lógica negativa e indica uma operação de escrita.

–Sinal *BSEL*: é do tipo entrada e indica que a placa controladora está seleccionada para comunicação.

–Sinal *P0*: é do tipo entrada e indica que este bloco está seleccionado para comunicação.

–Sinal *CLK_EXT*: é do tipo entrada e é para ligação do oscilador externo.

–Sinal *OSCILADOR*: é do tipo entrada e é para ligação do oscilador local.

Internamente, anexo 7.2, temos dois registo: o registo que armazena o *TCYCLE* e o registo que armazena o *TCONTROL*, cujos bits têm a seguinte função:

–Bit 0: Define o oscilador de referência (0→Oscilador local, 1→Oscilador externo).

–Bit 2: Activa ou inibe o relógio de referência (0 → Inibição, 1 → Activação).

–Bit 3: Activa ou inibe o modo escravo (0 → Inibição, 1 → Activação).

–Bit 4: Disparo por software (Transição 0 → 1).

–Bits 15 a 8: Número de divisões do oscilador de referência.

–Bits restantes: Não utilizados, estando disponíveis para implementações futuras.

Existe, também, um conjunto de portas *E* que fazem a selecção do registo a comunicar da forma seguinte:

–REGISTO0: Selecciona o registo *TCONTROL*.

–REGISTO1: Selecciona o registo *TCYCLE*.

–REGISTO2 a REGISTO7: Não utilizados.

O relógio de contagem é fornecido na saída do divisor *CONT_CLK*, que divide a frequência do oscilador de referência, escolhido no multiplexador 2:1, pelo valor do byte mais significativo do registo *TCONTROL*. Esta implementação para divisão da frequência do oscilador de referência permite divisões de 2 a 256. Atendendo que, para um valor de divisão 0 a saída é sempre 1 lógico, e para os restantes valores até 255, como a contagem é iniciada em 0 até ao valor de divisão presente no registo *TCONTROL*, a saída tem a frequência do oscilador de referência dividida pelo valor escrito no registo mais uma unidade.

Para gerar o relógio digital *TIME BUS* são utilizados um contador de 16 bits, *CONT_TBUS*, e um comparador, *COMP_TBUS*, comparando o valor do registo *TCYCLE* com o valor do *TIME BUS*, que aquando da igualdade reinicia o contador.

3.3.3. Bloco gerador de onda

Este bloco, apresentado na Figura 13, tem como função criar uma onda mediante a configuração feita por software, isto é, as tensões do nível alto e do nível baixo e os tempos de atraso e do nível alto. A saída deste bloco é digital mas posteriormente é convertida num sinal analógico recorrendo a um conversor de digital para analógico.

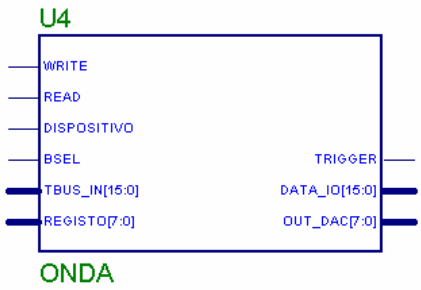


Figura 13 – Bloco gerador de onda

Como sinais de interface com os restantes blocos temos:

–Barramento *TIME_BUS*: é do tipo entrada de 16 bits e é utilizado referencial temporal para a criação da saída.

–Barramento *DATA_IO*: é do tipo entrada/saída de 16 bits e é utilizado para a transferência de informação.

–Barramento *REGISTO*: é do tipo entrada de 8 bits e é utilizado para seleccionar o registo a aceder.

–Barramento *OUT_DAC*: é do tipo saída de 8 bits e representa o valor binário da tensão de saída.

–Sinal *READ*: é do tipo entrada de lógica negativa e indica uma operação de leitura.

–Sinal *WRITE*: é do tipo entrada de lógica negativa e indica uma operação de escrita.

–Sinal *BSEL*: é do tipo entrada e indica que a placa controladora está seleccionada para comunicação.

–Sinal *DISPOSITIVO*: é do tipo entrada e indica que este bloco está seleccionado para comunicação.

–Sinal *TRIGGER*: é do tipo saída e é síncrono com a transição descendente da onda configurada.

Internamente, anexo 7.2, temos quatro registos para armazenar os valores *TDELAY*, *TON*, *VOFF* e *VON*.

Existe, também, um conjunto de portas *E* que fazem a selecção do registo a comunicar da forma seguinte:

–REGISTO0: Selecciona o registo *TON*.

–REGISTO1: Selecciona o registo *TDELAY*.

–REGISTO2: Selecciona o registo *VON*.

–REGISTO3: Selecciona o registo *VOFF*.

–REGISTO7: Controla a função *STAND-BY*.

–REGISTO4 a REGISTO6: Não utilizados.

Para selecção do valor de tensão a colocar na saída recorreu-se a dois comparadores de magnitude, *COMP_ONDA* e *COMP_ONDA_TON*, e uma porta *OU EXCLUSIVO* que geram um sinal de controlo segundo a seguinte tabela:

Time Bus >= TDelay	Time Bus >= TOn	Saída
0	0	VOfff
1	0	VOn
1	1	VOff

Figura 14 – Lógica de decisão do valor de saída

Para implementar a função *STAND-BY*, utilizou-se um multiplexador de 2 para 1 que coloca na saída, mediante a configuração feita por software, o valor vindo dos registos *VOFF* e *VON* ou uma palavra pré-definida em hardware correspondente à tensão 0 V.

4. SOFTWARE DE COMUNICAÇÃO

Sendo a placa controladora um periférico de um computador, houve a necessidade de criar software de comunicação para permitir a configuração dos diversos parâmetros. Assim, tendo como ferramenta o programa *Microsoft® Visual Studio 6.0*™, desenvolveu-se uma aplicação e uma livraria com primitivas de comunicação para os sistemas operativos *Microsoft® Windows 95*™ e *Microsoft® Windows 98*™.

4.1. Livraria com rotinas de baixo nível

Criou-se uma *Dynamic Link Library (DLL)* na linguagem *C++* de forma a introduzir um conjunto de rotinas de acesso ao espaço de endereçagem de entrada/saída, livraria esta que não é mais que uma forma de traduzir as instruções de alto nível em instruções *assembly* de baixo nível do tipo *IN* e *OUT*.

```
IOCOM_API void DeviceWrite (unsigned char MsAddr, unsigned char lsAddr, unsigned char MsVal, unsigned char lsVal)
{
        unsigned short Address;
        unsigned short Value;

        Address=0;
        Value=0;

        Address=MsAddr;

        Address=Address<<8;
        Address=Address+lsAddr;

        Value=MsVal;
        Value=Value<<8;
        Value=Value+lsVal;
        IOWrite(Address,Value);
}
```

Figura 15 – Rotina *DeviceWrite*

As rotinas *DeviceWrite* e *IOWrite* implementam instruções de escrita. A rotina *DeviceWrite*, que tem como parâmetros de entrada 2 bytes referentes a 16 bits de endereços e 2 bytes referentes a 16 bits de dados, converte as 4 palavras de 8 bits em 2 palavras de 16 bits correspondentes aos endereços e aos dados. Seguidamente, é invocada a rotina *IOWrite*, que tem como parâmetros de entrada as 2 palavras de 16 bits

referidas anteriormente, com a finalidade de enviar os dados para a placa controladora. Isto é conseguido através da instrução *assembly OUT*.

```
IOCOM_API void IOWrite(unsigned short Addr,unsigned short Val)
{
  //done in assembly code
        _asm mov        dx,Addr;
        _asm mov        ax,Val;
        _asm out dx,ax;
}
```

Figura 16 – Rotina *IOWrite*

```
IOCOM_API unsigned short DeviceRead (unsigned char MsAddr, unsigned char lsAddr)
{
        unsigned short Address;
        unsigned short Value;

        Address=0;
        Value=0;

        Address=MsAddr;
        Address=Address<<8;
        Address=Address+lsAddr;

        Value=IORead(Address);
        Return Value;
}
```

Figura 17 – Rotina *DeviceRead*

As rotinas *DeviceRead* e *IORead* implementam instruções de leitura. A rotina *DeviceRead*, que tem como parâmetros de entrada 2 bytes referentes a 16 bits de endereços, converte as 2 palavras de 8 bits numa palavra de 16 bits. Seguidamente, é invocada a rotina *IORead*, que tem como parâmetro de entrada a palavras de 16 bits referida anteriormente, com a finalidade de ler os dados da a placa controladora. Isto é conseguido através da instrução *assembly IN*.

```
IOCOM_API unsigned short IORead(unsigned short Addr)
{
        unsigned short val;
  //done in assembly code
        _asm mov        dx,Addr;
        _asm in         ax,dx;
        _asm mov        val,ax;
        return val;
}
```

Figura 18 – Rotina *IORead*

4.2. Aplicação de interface com o utilizador

Para interface das rotinas de baixo nível com o utilizador foi desenvolvida uma aplicação, aplicação esta que se considera de teste pois não é amigável ao utilizador requerendo alguma compreensão do funcionamento do hardware implementado.

O interface gráfico, Figura 19, apresenta uma secção dedicada à comunicação com o bloco gerador de ciclo e outra dedicada à comunicação com os diversos blocos geradores de ciclo.

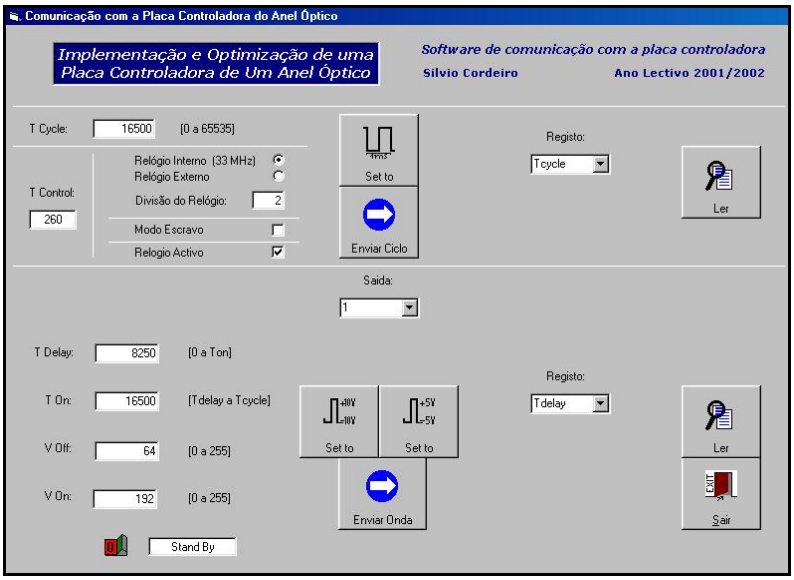


Figura 19 – Interface gráfico da aplicação

Quando a aplicação é iniciada, o bloco gerador de ciclo é configurado, por defeito, com um tempo de ciclo de 1ms, utilizando o oscilador local dividido por um factor de dois. Nos blocos geradores de onda é mantida a configuração existente.

Posteriormente, na secção de comunicação com o bloco gerador de ciclo, Figura 20, é possível definir no campo *TCycle* o tempo de ciclo das ondas de saída e configurar as diversas funcionalidades como, o relógio de referência desejado, se o oscilador local ou o relógio externo e o factor de divisão, activar ou inibir o funcionamento em modo escravo e activar ou inibir o funcionamento do relógio. O envio das novas configurações é feita pressionando o botão *"Enviar Ciclo"*, o que provoca a actualização do campo *TControl* com o valor que reflecte as funcionalidades configuradas. É, também, possível fazer a leitura dos registos, escolhendo o registo que se pretende ler, seguido do pressionar do botão *"Ler"*. Existe um botão *"Set to"* que restitui os valores por defeito a este bloco.

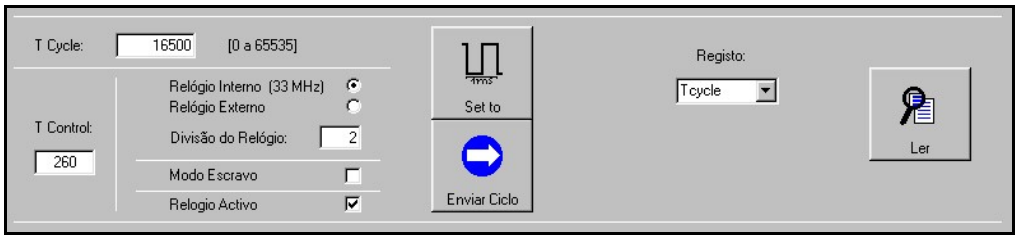


Figura 20 – Secção de comunicação com o bloco gerador de ciclo

Na secção de comunicação com o bloco gerador de onda, Figura 21, é possível fazer a configuração dos campos *TDelay*, *TOn*, *VOff* e *VOn* da saída seleccionada, sendo o envio das configurações é feita pressionando o botão *"Enviar Ciclo"*. O estado de *StandBy* é controlado por um botão do tipo *Liga/Desliga*. É, também, possível fazer a leitura dos registos, escolhendo o registo que se pretende ler, seguido do pressionar do botão *"Ler"*. Existem dois botões *"Set to"* com configurações pré-definidas de ondas a variar entre [+10 V , -10 V] e [+5 V , -5 V].

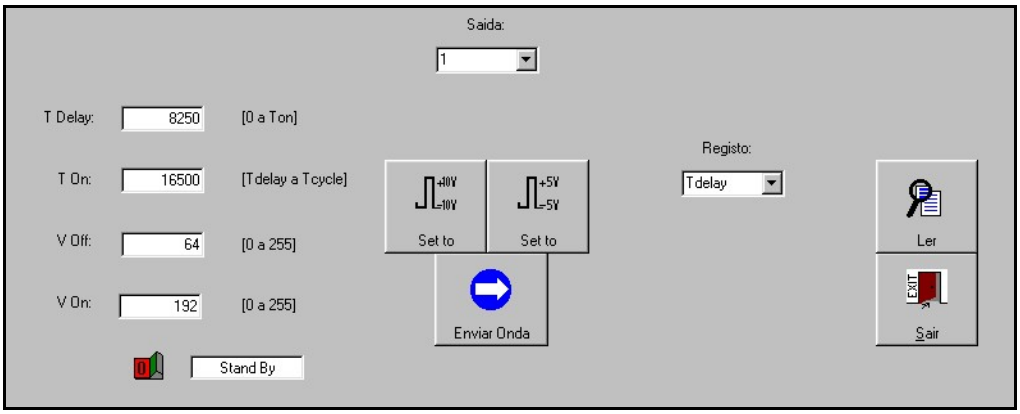


Figura 21 - Secção de comunicação com o bloco gerador de onda

Para mais fácil compreensão do funcionamento dos botões e campos existentes, foram adicionados comentários que são visíveis quando o rato permanece sobre o botão ou campo em questão.

4.3. Instalação

A instalação do software nos sistemas operativos *Microsoft® Windows 95*™ e *Microsoft® Windows 98*™ compreende três passos distintos: instalação da livraria de rotinas, instalação da aplicação e a reserva de recursos no espaço de endereçamento de entrada/saída.

A instalação da *DLL* implica colocar esta no espaço partilhado pelas diversas aplicações, criado pelo sistema operativo, para a colocação deste tipo de ficheiros. Assim, a forma mais imediata de instalar é colocar uma copia do ficheiro *IOCOM.DLL* no directório *C:\WINDOWS\SYSTEM*.

Visto que a aplicação é constituída unicamente pelo ficheiro *ONODE.EXE*, não é necessária instalação, basta correr a aplicação na unidade e directório onde se encontra.

Sendo a comunicação com a placa controladora feita utilizando o espaço de endereçamento de entrada/saída, é necessário reservar os endereços utilizados por esta, de forma a que mais nenhum dispositivo use estes endereços. Então, é necessário seguir uma série de passos que se descrevem a seguir.

Fazendo *Start* → *Settings* → *Control Panel* → *System*, começamos por aceder as *System Properties*, Figura 22.

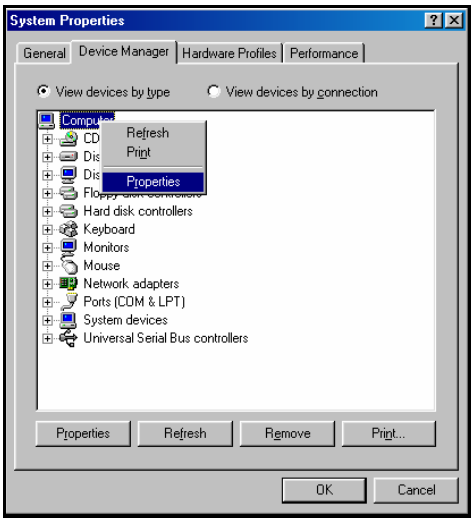


Figura 22 – *System Properties*

No separador *Device Manager*, pressionando o botão direito do rato sobre *Computer* acede-se a uma janela na qual se terá de seleccionar *Properties*. Encontra-se agora aberta a janela *Computer Properties*.

Acedendo ao separador *Reserve Resources*, e fazendo a selecção de *Input/Output (I/O)* aparecem as reservas já efectuadas, Figura 23.

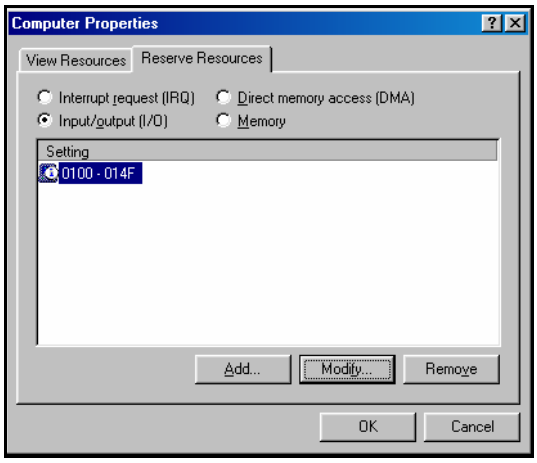


Figura 23 – *Computer Properties*

Pressionando *Add...* tem-se acesso à janela *Edit Resource Setting*, Figura 24, onde se fará a reserva de endereços, sendo *Start Value:* o endereço base configurado na placa controladora e *End Value:* a soma do endereço base com os endereços utilizados pelos diversos blocos geradores de onda e ciclo.

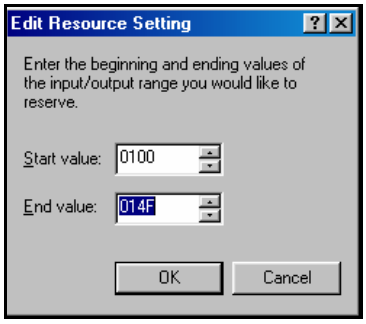


Figura 24 – *Edit Resource Setting*

No caso presente, encontrava-se disponível uma gama de endereços entre 0x0100 e 0x016F. Assim, definiu-se como endereço base, *Start Value:*, o endereço 0x0100 e, sendo que cada bloco compreende 16 endereços e estando em utilização um bloco gerador de ciclo e 4 geradores de onda, definiu-se como *End Value:* 0x014F.

5. TESTE DO SISTEMA

Implementada a placa controladora, Figura 25, montou-se o sistema da Figura 26 e desenvolveram-se um conjunto de testes com o intuito de caracterizar as ondas de saída.

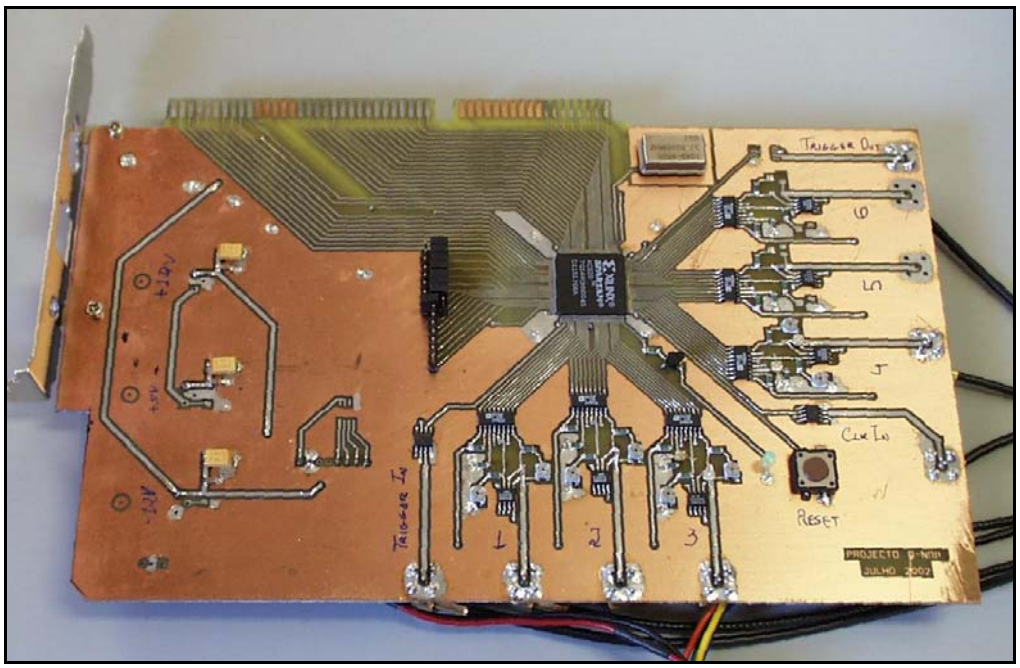


Figura 25 – Placa controladora

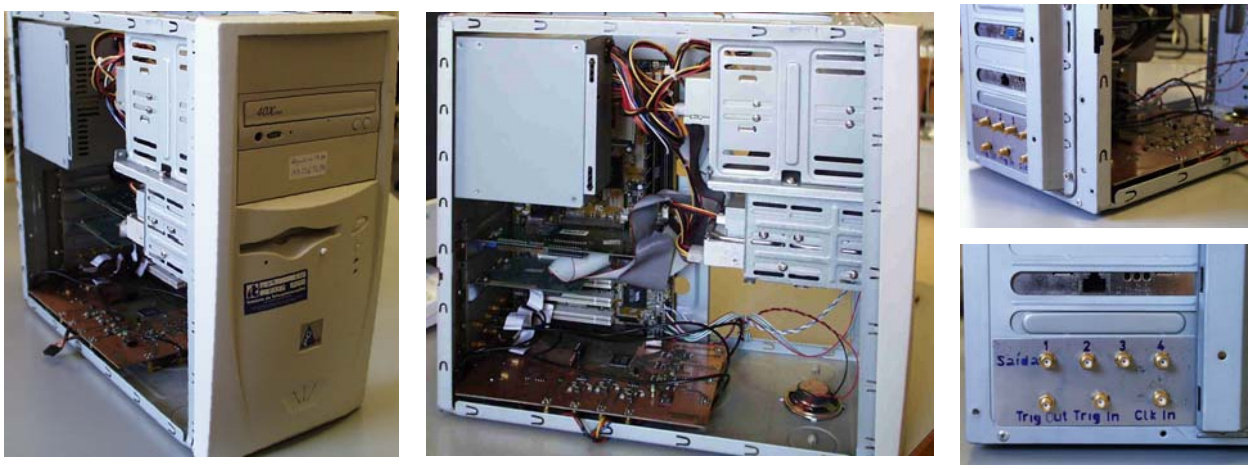


Figura 26 – Sistema que aloja a placa controladora

Os sinais de saída para controlo dos comutadores ópticos e outros poderão ser configurados pelo utilizador tendo como limitações os seguintes parâmetros:

Parâmetro	Valor	Unidade
Excursão em tensão	-10 a +10	V
Tempo máximo da transição	300	ns
Duração mínima do pulso	424	ns
Passo em tensão	350	mV
Número de passos em tensão	255	passos
Passo temporal mínimo	60	ns
Passo temporal máximo	7,76	µs
Número de passos temporais	65536	passos
Período máximo	508	ms

Figura 27 – Parâmetros característicos das saídas de controlo

Os valores anteriores foram obtidos tendo como referência temporal o oscilador interno, *VON* = +10V e *VOFF* = -10V. Utilizando outra referência temporal com frequência ν os valores de referência para os parâmetros são:

– Passo temporal mínimo: 2 / ν (s).

– Passo temporal máximo: 256 / ν (s).

– Período máximo: (2^24 –1) / ν (s).

Como exemplo apresenta-se um sinal, Figura 28, configurado com *TCYCLE* = 1 ms, *TDELAY* = 500 µs, *TON* = 500 µs, *VOFF* = -5V e *VON* = +5 V.

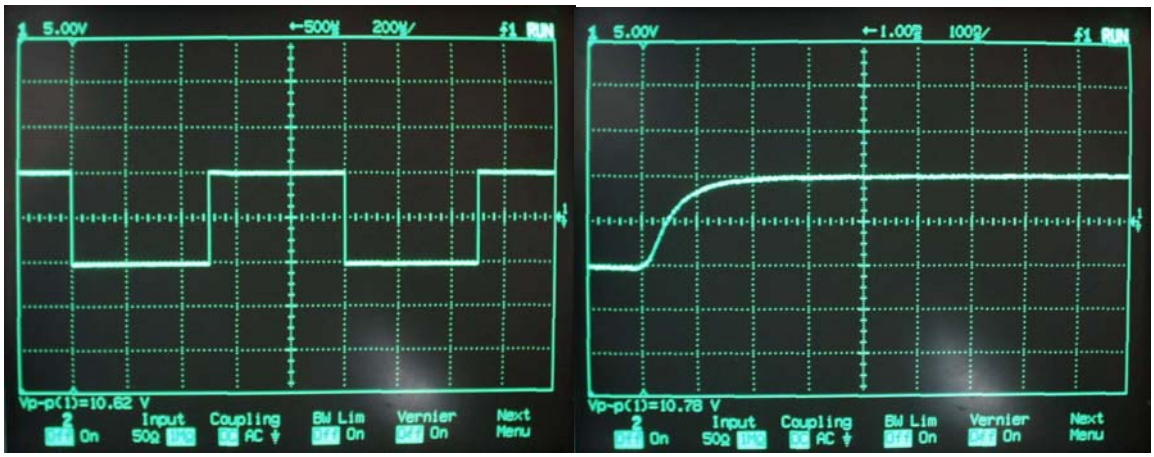


Figura 28 – Onda de saída

As características das saídas de sincronismo não dependem das configurações feitas, exceptuando o período que é o mesmo das ondas de controlo, têm como parâmetros:

Parâmetro	Valor	Unidade
Excursão em tensão	0 a +5	V
Tempo máximo da transição	5	ns
Duração do pulso	120	ns
Período máximo	508	ms

Figura 29 - Parâmetros característicos das saídas de sincronismo

Como exemplo apresenta-se na Figura 30 um sinal de saída e o respectivo sinal de sincronismo.

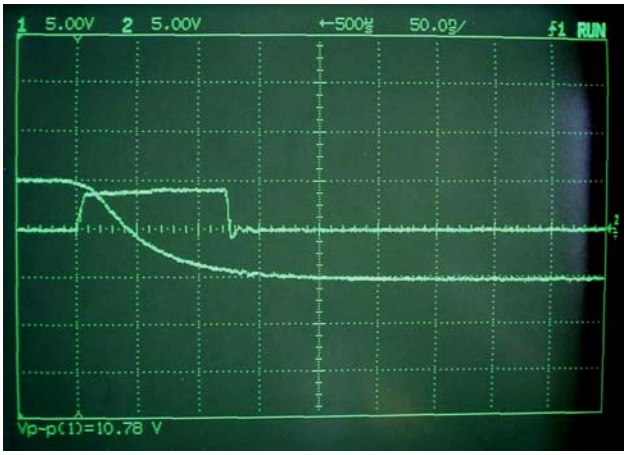


Figura 30 – Onda e sinal de sincronismo

6. CONCLUSÕES

Os resultados obtidos mostram que a implementação da placa controladora baseada numa *FPGA* foi um passo acertado, apresentando um conjunto de vantagens relativamente à implementação em hardware discreto feita anteriormente. Assim, neste momento, existe todo um hardware plenamente funcional para a aplicação para que foi desenvolvido, o *loop óptico*.

Apesar de se ter desenvolvido software para a comunicação com a placa controladora, este está muito associado ao funcionamento do hardware, pois tinha em vista a fase de testes. O próximo passo será, então, a construção de uma aplicação que permita uma utilização mais amigável ao utilizador.

Agradecimentos:

Agradeço a orientação dos professores Armando Nolasco Pinto e Rui Sousa Ribeiro. Deixo, também, um agradecimento ao Instituto de Telecomunicações – Pólo de Aveiro, por disponibilizar instalações e equipamento e ao projecto *ONODE* pelo fornecimento de componentes essenciais à realização do presente projecto.

Uma especial referência ao apoio técnico do IT, Paulo Gonçalves, Miguel Lacerda e Pedro Silva, assim como aos colegas de laboratório, Tiago Maia e Helena Silva, que proporcionaram um bom ambiente de trabalho e mostraram-se sempre disponíveis para auxiliar.

7. ANEXOS

7.1. Orcad

Nesta secção é apresentado o trabalho realizado no programa *Orcad® 9.1* que inclui o esquema da placa projectada e o desenho da placa de circuito impresso (*PCB*).

7.2. Xilinx

Nesta secção é apresentado o trabalho realizado no programa *Xilinx Foundation® F3.1i* que inclui o esquema principal da configuração assim como os esquemas dos blocos construídos para implementação na *FPGA*.

7.3. Visual Studio

Nesta secção é apresentado o trabalho realizado no programa *Microsoft® Visual Studio 6.0*™, que inclui o código da livraria de rotinas e o código da aplicação de interface.

– Aplicação

```
VERSION 5.00
Begin VB.Form frmEnviar
   Caption        =   "Comunicação com a Placa Controladora do Anel Óptico"
   ClientHeight   =   8595
   ClientLeft     =   60
   ClientTop      =   345
   ClientWidth    =   11880
   LinkTopic      =   "Form1"
   ScaleHeight    =   8595
   ScaleWidth     =   11880
   StartUpPosition =   2  'CenterScreen
   WindowState    =   2  'Maximized
   Begin VB.CommandButton cmd1ms
      Caption        =   "Set to"
      Height         =   1100
      Left           =   5040
      Picture        =   "Com_FPGA.frx":0000
      Style          =   1  'Graphical
      TabIndex       =   13
      ToolTipText    =   "Configuração predefinida (Tempo de ciclo de 1ms)."
      Top            =   1320
      Width          =   1200
   End
   Begin VB.OptionButton Rel_Ext
      Alignment      =   1  'Right Justify
      Caption        =   "Relógio Externo"
      Height         =   195
      Left           =   1920
      TabIndex       =   40
      ToolTipText    =   "Relogio ligado externamente com frequência pretendida."
      Top            =   2160
      Width          =   2295
   End
   Begin VB.OptionButton Rel_Int
      Alignment      =   1  'Right Justify
      Caption        =   "Relógio Interno  (33 MHz)"
      Height         =   195
      Left           =   1920
      TabIndex       =   39
      ToolTipText    =   "Oscilador a cristal de 33 MHz."
      Top            =   1920
      Value          =   -1  'True
      Width          =   2295
   End
   Begin VB.TextBox Div_Clk
      Alignment      =   1  'Right Justify
      Height         =   285
      Left           =   3720
      TabIndex       =   32
      Text           =   "2"
      ToolTipText    =   "Valor pelo qual o relogio de referência é dividido."
      Top            =   2480
      Width          =   495
   End
   Begin VB.TextBox Stand_Run
      Alignment      =   2  'Center
      Height         =   285
      Left           =   2160
      TabIndex       =   31
      Text           =   "Stand By"
      ToolTipText    =   "Indica o estado da saida (Stand By ou Run)."
      Top            =   7680
```

```
    Width          =   1335
  End
  Begin VB.CommandButton set10volt
    Caption        =   "Set to"
    Height         =   1100
    Left           =   4440
    Picture        =   "Com_FPGA.frx":030A
    Style          =   1  'Graphical
    TabIndex       =   14
    ToolTipText    =   "Configuração predefinida (+10volt/-10volt)"
    Top            =   5400
    Width          =   1200
  End
  Begin VB.CheckBox Trigger_In
    Alignment      =   1  'Right Justify
    Caption        =   "Modo Escravo"
    Height         =   255
    Left           =   1920
    TabIndex       =   30
    ToolTipText    =   "Activa a entrada de disparo."
    Top            =   2940
    Width          =   2295
  End
  Begin VB.CheckBox Stand_By
    Alignment      =   1  'Right Justify
    Caption        =   "Relogio Activo"
    Height         =   195
    Left           =   1920
    TabIndex       =   29
    ToolTipText    =   "Coloca o relogio em funcionamento."
    Top            =   3300
    Value          =   1  'Checked
    Width          =   2295
  End
  Begin VB.ComboBox SelRegCiclo
    Height         =   315
    ItemData       =   "Com_FPGA.frx":0614
    Left           =   7920
    List           =   "Com_FPGA.frx":061E
    TabIndex       =   27
    Text           =   "Tcycle"
    ToolTipText    =   "Selecção do registo a ler"
    Top            =   1920
    Width          =   1215
  End
  Begin VB.CommandButton LerCiclo
    Caption        =   "Ler"
    Height         =   1100
    Left           =   10200
    Picture        =   "Com_FPGA.frx":0634
    Style          =   1  'Graphical
    TabIndex       =   26
    ToolTipText    =   "Leitura do registo seleccionado"
    Top            =   1800
    Width          =   1200
  End
  Begin VB.TextBox Ref
    Alignment      =   2  'Center
    BackColor      =   &H8000000D&
    BeginProperty Font
      Name           =   "Verdana"
      Size           =   11.25
      Charset        =   0
      Weight         =   700
      Underline      =   0   'False
      Italic         =   -1  'True
      Strikethrough  =   0   'False
    EndProperty
    ForeColor      =   &H80000009&
    Height         =   660
    Left           =   720
    MultiLine      =   -1  'True
    TabIndex       =   24
    Text           =   "Com_FPGA.frx":093E
    ToolTipText    =   "Nome do projecto"
    Top            =   240
```

```
    Width          =   4935
  End
  Begin VB.ComboBox SelDisp
    Height         =   315
    ItemData       =   "Com_FPGA.frx":098B
    Left           =   5040
    List           =   "Com_FPGA.frx":099B
    TabIndex       =   22
    Text           =   "1"
    ToolTipText    =   "Selecção da saída a configurar"
    Top            =   4080
    Width          =   1215
  End
  Begin VB.CommandButton EnviarCiclo
    Caption        =   "Enviar Ciclo"
    Height         =   1095
    Left           =   5040
    Picture        =   "Com_FPGA.frx":09AB
    Style          =   1  'Graphical
    TabIndex       =   21
    ToolTipText    =   "Envio dos valores para dos registos"
    Top            =   2400
    Width          =   1200
  End
  Begin VB.TextBox txtTControl
    Alignment      =   2  'Center
    BackColor      =   &H00FFFFFF&
    Height         =   300
    Left           =   360
    Locked         =   -1  'True
    TabIndex       =   20
    Text           =   "260"
    ToolTipText        =    "Valor do registo de controlo das funcionalidades (divisão do relógio, selecção do relógio,
configuração do trigger, etc.)"
    Top            =   2760
    Width          =   720
  End
  Begin VB.CommandButton set5volt
    Caption        =   "Set to"
    Height         =   1095
    Left           =   5640
    Picture        =   "Com_FPGA.frx":0CB5
    Style          =   1  'Graphical
    TabIndex       =   17
    ToolTipText    =   "Configuração predefinida (+5volt/-5volt)"
    Top            =   5400
    Width          =   1200
  End
  Begin VB.ComboBox SelRegOnda
    Height         =   315
    ItemData       =   "Com_FPGA.frx":0FBF
    Left           =   7920
    List           =   "Com_FPGA.frx":0FCF
    TabIndex       =   16
    Text           =   "Tdelay"
    ToolTipText    =   "Selecção do registo a ler"
    Top            =   5520
    Width          =   1215
  End
  Begin VB.CommandButton LerOnda
    Caption        =   "Ler"
    Height         =   1100
    Left           =   10200
    Picture        =   "Com_FPGA.frx":0FEB
    Style          =   1  'Graphical
    TabIndex       =   15
    ToolTipText    =   "Leitura do registo seleccionado"
    Top            =   5400
    Width          =   1200
  End
  Begin VB.CommandButton EnviarOnda
    Caption        =   "Enviar Onda"
    Height         =   1100
    Left           =   5020
    Picture        =   "Com_FPGA.frx":12F5
    Style          =   1  'Graphical
```

```
    TabIndex       =   12
    ToolTipText    =   "Envio dos valores para dos registos"
    Top            =   6480
    Width          =   1335
  End
  Begin VB.CommandButton cmdSair
    Caption        =   "&Sair"
    Height         =   1095
    Left           =   10200
    Picture        =   "Com_FPGA.frx":15FF
    Style          =   1  'Graphical
    TabIndex       =   11
    ToolTipText    =   "Clique para sair (Alt-s)"
    Top            =   6480
    Width          =   1200
  End
  Begin VB.TextBox txtTDelay
    Alignment      =   1  'Right Justify
    Height         =   300
    Left           =   1350
    TabIndex       =   10
    Text           =   "0"
    ToolTipText    =   "Instante em que o sinal comuta para ON"
    Top            =   4770
    Width          =   975
  End
  Begin VB.TextBox txtTOn
    Alignment      =   1  'Right Justify
    Height         =   300
    Left           =   1350
    TabIndex       =   9
    Text           =   "0"
    ToolTipText    =   "Instante em que o sinal comuta para OFF"
    Top            =   5490
    Width          =   975
  End
  Begin VB.TextBox txtVOff
    Alignment      =   1  'Right Justify
    Height         =   300
    Left           =   1350
    TabIndex       =   8
    Text           =   "0"
    ToolTipText    =   "Valor de tensão OFF"
    Top            =   6240
    Width          =   975
  End
  Begin VB.TextBox txtVOn
    Alignment      =   1  'Right Justify
    Height         =   300
    Left           =   1320
    TabIndex       =   7
    Text           =   "0"
    ToolTipText    =   "Valor de tensão ON"
    Top            =   6960
    Width          =   975
  End
  Begin VB.TextBox txtTCycle
    Alignment      =   1  'Right Justify
    BackColor      =   &H00FFFFFF&
    Height         =   300
    Left           =   1320
    TabIndex       =   1
    Text           =   "16500"
    ToolTipText    =   "Tempo de ciclo (periodo)"
    Top            =   1400
    Width          =   945
  End
  Begin VB.Line Line6
    DrawMode       =   4  'Mask Not Pen
    X1             =   1560
    X2             =   4440
    Y1             =   2880
    Y2             =   2880
  End
  Begin VB.Line Line5
    DrawMode       =   4  'Mask Not Pen
```

```
    X1            =   1560
    X2            =   4440
    Y1            =   3240
    Y2            =   3240
  End
  Begin VB.Line Line4
    DrawMode      =   4  'Mask Not Pen
    X1            =   1320
    X2            =   1320
    Y1            =   1920
    Y2            =   3480
  End
  Begin VB.Line Line2
    DrawMode      =   4  'Mask Not Pen
    X1            =   120
    X2            =   4560
    Y1            =   1800
    Y2            =   1800
  End
  Begin VB.Label Label5
    BackColor     =   &H8000000A&
    Caption       =   "[0 a 65535]"
    Height        =   255
    Left          =   2640
    TabIndex      =   38
    Top           =   1440
    Width         =   855
  End
  Begin VB.Label Label4
    Caption       =   "[0 a 255]"
    Height        =   255
    Left          =   2760
    TabIndex      =   37
    Top           =   6990
    Width         =   855
  End
  Begin VB.Label Label3
    Caption       =   "[0 a 255]"
    Height        =   255
    Left          =   2760
    TabIndex      =   36
    Top           =   6280
    Width         =   855
  End
  Begin VB.Label Label2
    Caption       =   "[Tdelay a Tcycle]"
    Height        =   255
    Left          =   2760
    TabIndex      =   35
    Top           =   5520
    Width         =   1335
  End
  Begin VB.Label Label1
    Caption       =   "[0 a Ton]"
    Height        =   255
    Left          =   2760
    TabIndex      =   34
    Top           =   4800
    Width         =   855
  End
  Begin VB.Label lblDivRel1
    AutoSize      =   -1  'True
    Caption       =   "Divisão do Relógio:"
    Height        =   195
    Left          =   1965
    TabIndex      =   33
    ToolTipText   =   "Controlo das funcionalidades (divisão do relógio, selecção do relógio, configuração do trigger, etc.)"
    Top           =   2520
    Width         =   1380
  End
  Begin VB.Label lblRegCiclo
    AutoSize      =   -1  'True
    Caption       =   "Registo:"
    Height        =   195
    Left          =   8160
    TabIndex      =   28
```

```
    ToolTipText     =   "Selecção do registo a ler"
    Top             =   1560
    Width           =   585
  End
  Begin VB.Label Autor
    Alignment       =   2  'Center
    BackColor       =   &H8000000B&
    Caption         =   "Silvio Cordeiro                    Ano Lectivo 2001/2002"
    BeginProperty Font
      Name          =   "Verdana"
      Size          =   8.25
      Charset       =   0
      Weight        =   700
      Underline     =   0   'False
      Italic        =   0   'False
      Strikethrough =   0   'False
    EndProperty
    ForeColor       =   &H8000000D&
    Height          =   255
    Left            =   6240
    TabIndex        =   25
    ToolTipText     =   "Autor do software"
    Top             =   600
    Width           =   5295
  End
  Begin VB.Line Line3
    DrawMode        =   4  'Mask Not Pen
    X1              =   120
    X2              =   11760
    Y1              =   3600
    Y2              =   3600
  End
  Begin VB.Label Titulo
    Alignment       =   2  'Center
    BackColor       =   &H8000000B&
    Caption         =   "Software de comunicação com a placa controladora"
    BeginProperty Font
      Name          =   "Verdana"
      Size          =   9
      Charset       =   0
      Weight        =   700
      Underline     =   0   'False
      Italic        =   -1  'True
      Strikethrough =   0   'False
    EndProperty
    ForeColor       =   &H8000000D&
    Height          =   615
    Left            =   6240
    TabIndex        =   23
    ToolTipText     =   "Descrição do software"
    Top             =   240
    Width           =   5295
  End
  Begin VB.Label lblTcontrol
    AutoSize        =   -1  'True
    Caption         =   "T Control:"
    Height          =   195
    Left            =   360
    TabIndex        =   19
    ToolTipText     =   "Controlo das funcionalidades (divisão do relógio, selecção do relógio, configuração do trigger, etc.)"
    Top             =   2400
    Width           =   690
  End
  Begin VB.Label lblRegistoOnda
    AutoSize        =   -1  'True
    Caption         =   "Registo:"
    Height          =   195
    Left            =   8160
    TabIndex        =   18
    ToolTipText     =   "Selecção do registo a ler"
    Top             =   5160
    Width           =   585
  End
  Begin VB.Image imgBotaoON
    Height          =   480
    Left            =   1320
```

```
    Picture        =   "Com_FPGA.frx":1909
    ToolTipText    =   "Clique para colocar a saida em Stand-By."
    Top            =   7560
    Visible        =   0   'False
    Width          =   480
  End
  Begin VB.Image imgBotaoOFF
    Height         =   480
    Left           =   1440
    Picture        =   "Com_FPGA.frx":1C13
    ToolTipText    =   "Clique para sair de Stand-By."
    Top            =   7560
    Width          =   480
  End
  Begin VB.Label lblTDelay
    AutoSize       =   -1  'True
    Caption        =   "T Delay:"
    Height         =   195
    Left           =   435
    TabIndex       =   6
    ToolTipText    =   "Instante em que o sinal comuta para ON"
    Top            =   4800
    Width          =   600
  End
  Begin VB.Label lblTon
    AutoSize       =   -1  'True
    Caption        =   "T On:"
    Height         =   195
    Left           =   600
    TabIndex       =   5
    ToolTipText    =   "Instante em que o sinal comuta para OFF"
    Top            =   5520
    Width          =   405
  End
  Begin VB.Label lblVoff
    AutoSize       =   -1  'True
    Caption        =   "V Off:"
    Height         =   195
    Left           =   600
    TabIndex       =   4
    ToolTipText    =   "Valor de tensão OFF"
    Top            =   6240
    Width          =   405
  End
  Begin VB.Label lblVon
    AutoSize       =   -1  'True
    Caption        =   "V On:"
    Height         =   195
    Left           =   600
    TabIndex       =   3
    ToolTipText    =   "Valor de tensão ON"
    Top            =   6960
    Width          =   405
  End
  Begin VB.Label lblDispositivo
    AutoSize       =   -1  'True
    Caption        =   "Saida:"
    Height         =   195
    Left           =   5400
    TabIndex       =   2
    ToolTipText    =   "Selecção da saída a configurar"
    Top            =   3720
    Width          =   450
  End
  Begin VB.Line Line1
    BorderStyle    =   6  'Inside Solid
    DrawMode       =   4  'Mask Not Pen
    X1             =   120
    X2             =   11760
    Y1             =   1200
    Y2             =   1200
  End
  Begin VB.Label lblTCycle
    AutoSize       =   -1  'True
    Caption        =   "T Cycle:"
    Height         =   195
```

```
    Left          =   360
    TabIndex      =   0
    ToolTipText   =   "Tempo de ciclo (periodo)"
    Top           =   1440
    Width         =   585
  End
End
Attribute VB_Name = "frmEnviar"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
'----- Definicao de variaveis -----

Dim TCycle As Long
Dim ValDisp(1 To 6, 1 To 5) As Long

'----- Inicio -----

Private Sub Form_Load()

Ma = 1

'-------------------------------------------------'
'  Cálculo do endereço La do dispositivo          '
'                                                 '
'  DeviceADDR = DeviceNUM * 16 + DeviceREG * 2    '
'-------------------------------------------------'

'--------------------------------------------------------------------------------
'-------------------------     ENVIO DO VALOR DE TCONTROL    ---------------------
'--------------------------------------------------------------------------------

Mv = 1
Lv = 20 'Divisao por 2, Trg_In Off, Clk_En, Osc
La = 0
Call DeviceIOWrite(Ma, La, CByte(Mv), CByte(Lv))

Mv = 1
Lv = 4 'Divisao por 2, Trg_In Off, Clk_En, Osc
La = 0
Call DeviceIOWrite(Ma, La, CByte(Mv), CByte(Lv))


txtTControl.Text = 260

'--------------------------------------------------------------------------------
'-------------------------     ENVIO DO VALOR DE TCYCLE    ----------------------
'--------------------------------------------------------------------------------

Mv = 64
Lv = 116 '16500
La = 2
Call DeviceIOWrite(Ma, La, CByte(Mv), CByte(Lv))

TCycle = 16500
txtTCycle.Text = 16500

'----- Aspectos visuais -----
txtVOn.Text = "0"
txtVOff.Text = "0"
txtTOn.Text = "0"
txtTDelay.Text = "0"

Beep

End Sub

'----- Botao Enviar Onda -----

Private Sub EnviarOnda_Click()

'--------------------------------------------------------------------------------
'----------------     VERIFICAÇÃO DA EXISTÊNCIA DE DADOS       ------------------
'--------------------------------------------------------------------------------
```

```
If txtVOn.Text = "" Or txtVOff.Text = "" Or txtTOn.Text = "" Or txtTDelay.Text = "" Then
   MsgBox "Tem que introduzir um valor numérico em cada caixa. VOn, VOff, TOn e TDelay.", vbOKOnly, "ERRO"
   Exit Sub
End If

If SelDisp.Text = "" Then
   MsgBox "Tem que seleccionar uma saida para configurar.", vbOKOnly, "ERRO"
   Exit Sub
End If

disp = CInt(SelDisp.Text)

Ma = 1

'--------------------------------------------------------------------------------
'-------------------------     ENVIO DO VALOR DE TON     -------------------------
'--------------------------------------------------------------------------------

Aux1 = CLng(txtTOn.Text)
Aux2 = CLng(txtTDelay.Text)
Aux3 = CLng(txtVOn.Text)
Aux4 = CLng(txtVOff.Text)

If Aux1 > TCycle Or Aux2 > Aux1 Then
   MsgBox "Tem que introduzir valores de TDelay e TOn conforme as especificações.", vbOKOnly, "ERRO"
   Exit Sub
End If

ValDisp(disp, 1) = Aux1
a = ValDisp(disp, 1) \ 256
b = ValDisp(disp, 1) Mod 256

La = 16 * disp + 0
Call DeviceIOWrite(Ma, La, CByte(a), CByte(b))

'--------------------------------------------------------------------------------
'-------------------------     ENVIO DO VALOR DE TDELAY     ----------------------
'--------------------------------------------------------------------------------

ValDisp(disp, 2) = Aux2
c = ValDisp(disp, 2) \ 256
d = ValDisp(disp, 2) Mod 256

La = 16 * disp + 2
Call DeviceIOWrite(Ma, La, CByte(c), CByte(d))

'--------------------------------------------------------------------------------
'-------------------------     ENVIO DO VALOR DE VON     -------------------------
'--------------------------------------------------------------------------------

ValDisp(disp, 3) = Aux3
e = ValDisp(disp, 3) \ 256
f = ValDisp(disp, 3) Mod 256

La = 16 * disp + 4
Call DeviceIOWrite(Ma, La, CByte(e), CByte(f))

'--------------------------------------------------------------------------------
'-------------------------     ENVIO DO VALOR DE VOFF    -------------------------
'--------------------------------------------------------------------------------

ValDisp(disp, 4) = Aux4
g = ValDisp(disp, 4) \ 256
h = ValDisp(disp, 4) Mod 256

La = 16 * disp + 6
Call DeviceIOWrite(Ma, La, CByte(g), CByte(h))

Beep

End Sub

'----- Botao Enviar Ciclo -----

Private Sub EnviarCiclo_Click()
```

```
If txtTCycle.Text = "" Then
   MsgBox "Tem que introduzir um valor numérico na caixa Tcycle.", vbOKOnly, "ERRO"
   Exit Sub
End If

If Div_Clk.Text = "" Then
   MsgBox "Tem que introduzir um valor para a divisão do relógio.", vbOKOnly, "ERRO"
   Exit Sub
End If

Ma = 1

'--------------------------------------------------------------------------------
'-------------------------     ENVIO DO VALOR DE TCONTROL     --------------------------------
'--------------------------------------------------------------------------------

aa = Rel_Ext.Value + 4 * Stand_By.Value + 8 * Trigger_In.Value + 256 * (CLng(Div_Clk.Text) - 1)
bb = aa \ 256
cc = aa Mod 256

La = 0

If Trigger_In.Value = 0 Then
   Call DeviceIOWrite(Ma, La, CByte(bb), CByte(cc + 16))
End If

Call DeviceIOWrite(Ma, La, CByte(bb), CByte(cc))

txtTControl.Text = aa

'--------------------------------------------------------------------------------
'-------------------------     ENVIO DO VALOR DE TCYCLE     ------------------------------
'--------------------------------------------------------------------------------

Aux1 = CLng(txtTCycle.Text)

If Aux1 > 65535 Then
   MsgBox "Tem que introduzir valores de TCycle conforme a especificação.", vbOKOnly, "ERRO"
   Exit Sub
End If

TCycle = Aux1
dd = TCycle \ 256
ee = TCycle Mod 256

La = 2
Call DeviceIOWrite(Ma, La, CByte(dd), CByte(ee))

Beep

End Sub

'----- Botao Ler Ciclo -----

Private Sub LerCiclo_Click()

Ma = 1

If SelRegCiclo.Text = "" Then
   MsgBox "Tem que seleccionar um registo para ler.", vbOKOnly, "ERRO"
   Exit Sub
Else
   If SelRegCiclo.Text = "Tcontrol" Then
      La = 0
   End If
   If SelRegCiclo.Text = "Tcycle" Then
      La = 2
   End If
End If

data = DeviceIORead(Ma, La)
MsgBox (data And 65535), vbOKOnly, "VALOR LIDO"

Beep

End Sub
```

```
'----- Botao Ler Onda -----

Private Sub LerOnda_Click()

If SelDisp.Text = "" Then
   MsgBox "Tem que seleccionar uma saida para ler.", vbOKOnly, "ERRO"
   Exit Sub
End If

disp = 16 * CInt(SelDisp.Text)

Ma = 1

If SelRegOnda.Text = "" Then
   MsgBox "Tem que seleccionar um registo para ler.", vbOKOnly, "ERRO"
   Exit Sub
Else
   If SelRegOnda.Text = "Ton" Then
      La = disp + 0
   End If
   If SelRegOnda.Text = "Tdelay" Then
      La = disp + 2
   End If
   If SelRegOnda.Text = "Von" Then
      La = disp + 4
   End If
   If SelRegOnda.Text = "Voff" Then
      La = disp + 6
   End If
End If

data = DeviceIORead(Ma, La)
MsgBox (data And 65535), vbOKOnly, "VALOR LIDO"

Beep

End Sub

'----- Seleccao da divisao do relogio -----

Private Sub Div_Clk_Change()

If Div_Clk.Text = "" Then
   Exit Sub
End If

Val_Div = CLng(Div_Clk.Text)

If Val_Div < 2 Or Val_Div > 256 Then
   MsgBox "Tem que introduzir um valor do intervalo [2,256].", vbOKOnly, "ERRO"
   Exit Sub
End If

End Sub

Private Sub Rel_Ext_Click()

Rel_Ext.Value = True
Rel_Int.Value = False

End Sub

Private Sub Rel_Int_Click()

Rel_Int.Value = True
Rel_Ext.Value = False

End Sub

'----- Seleccao da saida -----

Private Sub SelDisp_Click()

disp = CInt(SelDisp.Text)
```

```
txtTOn.Text = ValDisp(disp, 1)

txtTDelay.Text = ValDisp(disp, 2)

txtVOn.Text = ValDisp(disp, 3)
txtVOff.Text = ValDisp(disp, 4)

If ValDisp(disp, 5) = 0 Then
   imgBotaoON.Visible = False
   imgBotaoOFF.Visible = True
   Stand_Run.Text = "Stand By"
Else
   imgBotaoON.Visible = True
   imgBotaoOFF.Visible = False
   Stand_Run.Text = "Run"
End If

End Sub

'----- Botao Set To 1 ms -----

Private Sub cmd1ms_Click()

Rel_Ext.Value = False
Rel_Int.Value = True
Stand_By.Value = 1
Trigger_In.Value = 0
txtTCycle.Text = "16500"
Div_Clk.Text = "2"

End Sub


'----- Botao Set To +10v/-10v -----

Private Sub set10volt_Click()

txtVOn.Text = "255"
txtVOff.Text = "0"
txtTOn.Text = "16500"
txtTDelay.Text = "8250"

End Sub

'---------- Botao Set To +5v/-5v ----------

Private Sub set5volt_Click()

txtVOn.Text = "192"
txtVOff.Text = "64"
txtTOn.Text = "16500"
txtTDelay.Text = "8250"

End Sub


'----- Botao Stand-By Off -----

Private Sub imgBotaoOFF_Click()

If SelDisp.Text = "Dispositivo" Then
   MsgBox "Tem que seleccionar um dispositivo.", vbOKOnly, "ERRO"
   Exit Sub
End If

disp = CInt(SelDisp.Text)
ValDisp(disp, 5) = 1

'----------------------------------------
'      Troca dos botões
'----------------------------------------

imgBotaoOFF.Visible = False
imgBotaoON.Visible = True
Stand_Run.Text = "Run"
```

```
'----------------------------------------
'      Envio da palavra de controlo
'----------------------------------------

Mv = 0
Lv = 0
Ma = 1
La = 16 * disp + 14
Call DeviceIOWrite(Ma, La, CByte(Mv), CByte(Lv))

Beep

End Sub

'----- Botao Stand-By On -----

Private Sub imgBotaoON_Click()

If SelDisp.Text = "" Then
   MsgBox "Tem que seleccionar uma saida.", vbOKOnly, "ERRO"
   Exit Sub
End If

disp = CInt(SelDisp.Text)
ValDisp(disp, 5) = 0
'----------------------------------------
'      Troca dos botões
'----------------------------------------

imgBotaoON.Visible = False
imgBotaoOFF.Visible = True
Stand_Run.Text = "Stand By"

'----------------------------------------
'      Envio da palavra de controlo
'----------------------------------------

Mv = 0
Lv = 0
Ma = 1
La = 16 * disp + 14
Call DeviceIOWrite(Ma, La, CByte(Mv), CByte(Lv))

Beep

End Sub

'----- Botao Sair -----

Private Sub cmdSair_Click()

Unload Me
End

End Sub
```

– Livraria DLL

```
// IOCom.cpp : Defines the entry point for the DLL application.
//

#include "StdAfx.h"
#include "IOCom.h"
#include "math.h"
#include "stdlib.h"

BOOL APIENTRY DllMain( HANDLE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved
                                                    )
{
   switch (ul_reason_for_call)
              {
```

```
                        case DLL_PROCESS_ATTACH:
                        case DLL_THREAD_ATTACH:
                        case DLL_THREAD_DETACH:
                        case DLL_PROCESS_DETACH:
                                break;
    }
    return TRUE;
}

IOCOM_API void IOWrite(unsigned short Addr,unsigned short Val)
{
   //done in assembly code
            _asm mov            dx,Addr;
            _asm mov            ax,Val;
            _asm out dx,ax;
}

IOCOM_API unsigned short IORead(unsigned short Addr)
{
            unsigned short val;
            //done in assembly code
            _asm mov            dx,Addr
            _asm in             ax,dx
            _asm mov            val,ax
            return val;
}

IOCOM_API void DeviceWrite(unsigned char MsAddr,unsigned char lsAddr,unsigned char MsVal,unsigned char lsVal)
{
            unsigned short Address;
            unsigned short Value;


            //MsAddr = 1;
            //lsAddr = 18;
            //MsVal = 64;
            //lsVal = 0;

            Address=0;
            Value=0;

            Address=MsAddr;

            Address=Address<<8;
            Address=Address+lsAddr;

            Value=MsVal;
            Value=Value<<8;
            Value=Value+lsVal;
            IOWrite(Address,Value);
}



IOCOM_API unsigned short DeviceRead(unsigned char MsAddr,unsigned char lsAddr)
{
            unsigned short Address;
            unsigned short Value;

            Address=0;
            Value=0;
            Address=MsAddr;
            Address=Address<<8;
            Address=Address+lsAddr;
            Value=IORead(Address);
            return Value;
}
```

7.4. Cd-Rom

Reuniu-se num disco compacto toda a informação pertinente relacionada com a realização deste projecto. A informação encontra-se organizada em directórios com a seguinte estrutura:

– *DOCUMENTOS*: Relatórios e outros.

– *DATASHEETS*: Documentos com características dos componentes utilizados.

– *IMAGENS*: Imagens e figuras relacionadas com o projecto.

– *XILINX*: Ficheiros de programação da *FPGA*.

– *VISUAL_STUDIO*: Ficheiros da aplicação e da livraria.

– *ORCAD*: Ficheiros dos esquemas da placa controladora.

ÍNDICE DE FIGURAS