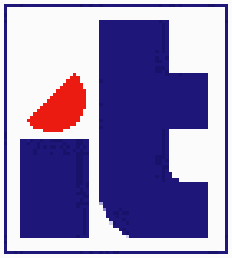
Desenvolvimento de uma controladora reconfigurável para um *loop* óptico

Relatório

Universidade de Aveiro



Instituto de Telecomunicações
Pólo de Aveiro

06/Maio/2004

Alunos:
Amadeu Santos N.º 21127
António Oliveira N.º 21366

Orientador:
Prof. Doutor Armando Nolasco Pinto
Prof. Doutor Rui Sousa Ribeiro

Índice

Índice de Tabelas

Índice de Figuras

Convenções

Lógica negativa

Os sinais de lógica negativa – activos a '0' – são precedidos por uma barra – "\". e.g. "\SCLR"

Sinais com duas funções

Os sinais com duas funções são representados na forma "X\Y". "X" está activo a '1' e "Y" está activo a '0'. Estes sinais são disjuntos.

Barramentos

Os barramentos são referidos pela primeira vez na forma "$X_{n-1}..X_0$" ou "X_1X_0" no caso de o barramento ser de 2 *bits*. Isto significa que o barramento tem *n bits*. Em referências posteriores a "$X_{n-1}..X_0$" será escrito apenas "*X*" em itálico.

Quantidades em base 2 ou 16

Para representar números binários de vários bits é usado o equivalente hexadecimal do número com a notação 0x"XXXX..." . e.g. 10100100_2 = 0x"A4". Em quantidades com muitos bits usamos um ponto "." Como separador de 2 *bytes* – e.g. 0x"1234.FFA4"

Acrónimos

PC	Computador Pessoal
TCP/IP	*Transfer Control Protocol*/*Internet Protocol*
PCB	*Printed Circuit Board*
HTTP	*Hyper Text Transfer Protocol*
UART	Universal Asynchronous Reciever/Transmitter
CX	Conector de Expansão
SGO	Sincronismo e Geração de Ondas
FPGA	Field Programable Gate Array
CTRL	Palavra de Controlo
FSM	Máquina de Estados Finitos
AmpOp	Amplificador Operacional

1 – Introdução

O *loop* óptico é um instrumento laboratorial que permite simular um percurso de até vários milhares de quilómetros de fibra óptica intercalados com amplificadores ópticos e que usa um número reduzido de componentes. O princípio geral de funcionamento consiste em injectar uma sequência de *bits* num circuito fechado de fibra óptica e amplificadores. O sinal circula durante o número desejado de voltas e um equipamento faz a medição da taxa de erros ou de outra grandeza que se queira testar.

1.1 – Funcionamento do *loop*

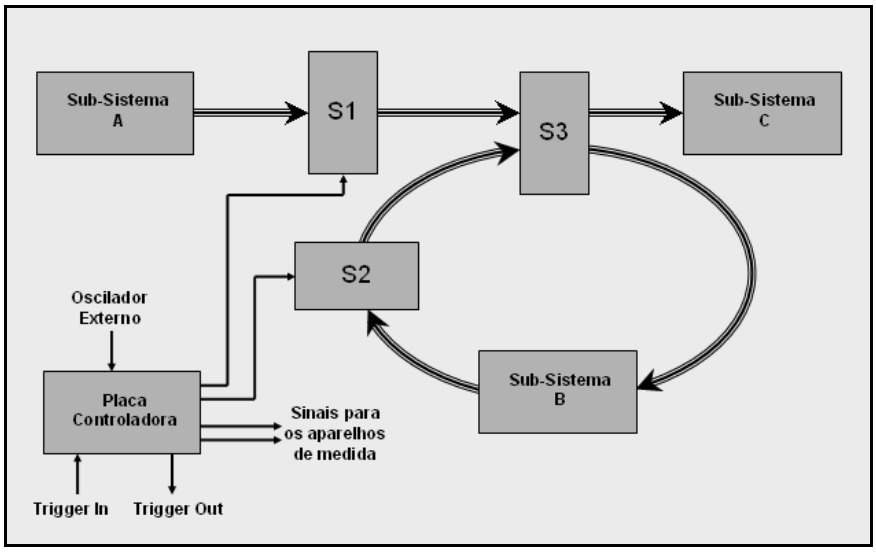


Figura 1 - Diagrama de Blocos de um *loop* óptico.

1.1.1 – Descrição dos componentes

- S1 e S2 são comutadores ópticos que permitem ou inibem a passagem do sinal;
- S3 é um acoplador óptico que retira parte do sinal para o sub-sistema C;
- O sub-sistema A é um emissor óptico. Produz a sequência de *bits*;
- O sub-sistema B é um conjunto de troços de fibra e amplificadores ópticos intercalados;
- O sub-sistema C é o equipamento de medida;

1.1.2 – Processo da simulação

1. Inicialmente S1 está fechado e S2 aberto. A envia o sinal, que passa para o interior do *loop*.
2. A seguir S1 abre e S2 fecha. O sinal circula dentro do *loop* um número pré-definido de voltas.
3. O sub-sistema C recolhe parte do sinal e mede as grandezas pretendidas.

NB: A comutação de S1 e S2 não é instantânea, o que gera interferência entre *bits* durante a transição. Esta interferência dura dezenas de nanossegundos e repete-se a cada circulação do sinal dentro do *loop*.
Para a medição dos erros é necessário sincronizar C de modo a não contar os erros gerados pela interferência dos comutadores.

Outros equipamentos de medida, que funcionam em modo *burst*, adquirem sincronismo de trama muito rapidamente, relativamente ao tempo de circulação, e simplificam o processo de contagem de erros.

A placa controladora gera os sinais para o sincronismo de todo o sistema.

2 – Objectivos

Com base na placa controladora já existente os nossos objectivos são:

i. Tornar a controladora totalmente reconfigurável, sem remoção de qualquer componente.

O circuito lógico interno da FPGA pode ser alterado por forma a corrigir eventuais *bugs*, melhorar a performance ou estabilidade, adicionar nova funcionalidade ou adaptar a controladora a uma situação específica.

ii. Aumentar o número de saídas para 8, totalmente configuráveis garantindo compatibilidade com uma gama mais vasta de níveis de sinalização.

Os níveis de tensão lógicos são definidos pelo utilizador para possibilitar a interacção da controladora com uma gama vasta de equipamento laboratorial.

iii. Tornar as entradas *Trigger In* e *Clock In* configuráveis em nível e sentido da transição.

O utilizador pode sincronizar a controladora com sinais externos com possibilidade de definir os níveis lógicos ou de transição para poder operar a controladora com o maior número possível de geradores de sinal.

iv. Transformar a controladora existente, que assenta sobre o barramento ISA, numa externa com interface RS232.

O formato externo é muito mais versátil do que a versão interna. As ligações da controladora externa são mais acessíveis do que numa instalada no interior de um computador e é possível transportar a controladora facilmente e sem risco de danos.

A porta série permite armazenar numa memória o endereço de rede da controladora, a versão do circuito e outros dados relevantes. Com os valores presentes na memória podemos operar a controladora remotamente recorrendo à porta de rede.

v. Dotar a controladora de um conector de expansão com acesso a todas as suas funções.

O conector de expansão permite o desenvolvimento *a posteriori* de módulos que aumentem a funcionalidade da controladora. Será criada documentação detalhada referente ao protocolo de comunicação que permita ao projectista o desenvolvimento dos módulos sem necessidade de conhecer os detalhes da implementação da controladora.

O formato físico do conector está por definir e dependerá da quantidade de sinais a disponibilizar, bem como da dimensão da caixa da controladora e do seu PCB.

NB: O conector de expansão permite adicionar funcionalidade à controladora numa fase posterior a este projecto. Uma sugestão é a implementação de uma ligação tipo *ethernet*.

É possível desenvolver uma unidade com capacidade de processamento e com *software* que crie uma sessão HTTP possibilitando ao utilizador aceder à configuração da controladora com apenas um *browser,* sem software adicional e independentemente do sistema operativo. A controladora criaria uma interface de utilizador com HTML ou outra linguagem.

Esta função confere à controladora bastante versatilidade, permitindo a sua utilização em ambientes muito mais diversos e sem dependência de PC's externos específicos, por exemplo em caso de avaria.

O desenvolvimento de tal função constitui uma proposta aliciante para um projecto a desenvolver nos anos seguintes.

3 – Especificação Geral

3.1 – Saídas

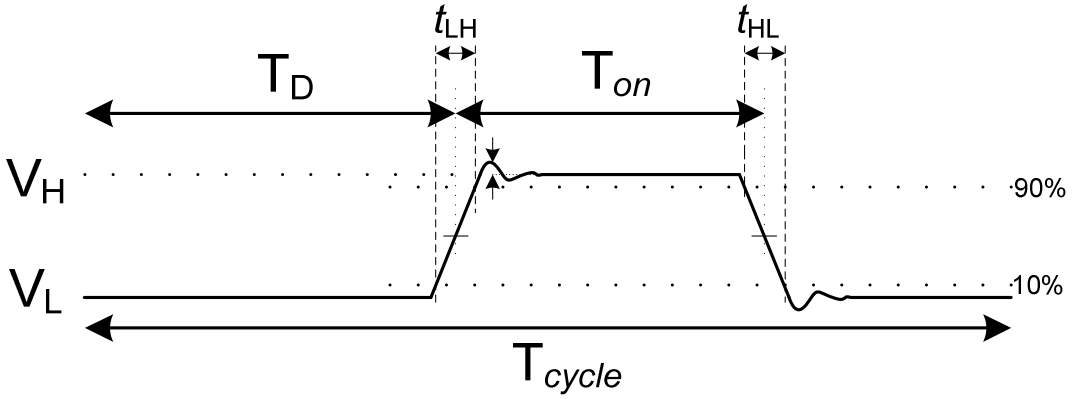


Figura 2 – Forma de onda da saída.

- A controladora terá oito saídas, todas com os parâmetros T_D, T_{on}, V_H e V_L configuráveis;
- Cada saída poderá ser colocada no modo *standby*, ficando com uma tensão constante de valor definido pelo utilizador;
- As tensões V_H e V_L variam entre -10V e +10V, com 255 passos. A onda poderá ser, ainda, de lógica negativa.
- Os tempos de transição da onda, t_{LH} e t_{HL}, não devem exceder 150ns, com *overshoot* inferior a 5%.
- A controladora terá dois modos de funcionamento:
  - *Master*: a forma de onda é periódica e repete-se a cada T_{cycle}, este comum a todas as saídas.
  - *Slave*: o varrimento da onda é iniciado por um sinal de sincronismo externo, *Trigger In*.

NB: Do ponto de vista do utilizador é visto o tempo de atraso, T_D, e de duração do estado alto, T_{on} como nos mostra a Figura 2. Para a implementação é bastante mais simples definirmos os instantes de transição T_1 e T_2, sendo T_i o instante de uma transição da onda de saída. Isto implica que no início de cada ciclo a onda seja colocada num estado bem definido. O software tem a função de fazer a conversão dos parâmetros, sendo apresentada a forma de onda e tempos associados ao utilizador como indica a Figura 2, ou, eventualmente, nas duas formas.

3.2 – Entradas

Trigger In é o sinal de sincronismo externo necessário para o funcionamento no modo *Slave*. *Clock In* é o sinal de relógio externo para usar em alternativa ao relógio embutido.

- Ambos os sinais são configuráveis no nível e no sentido da transição: ascendente ou descendente;
- Devem também tolerar sinais de entrada com níveis entre -10 e +10V. O nível de transição tem a mesma gama de valores;

A frequência máxima admissível para o relógio externo será determinada pela complexidade do circuito implementado e pela *FPGA* utilizada.

3.3 – Comunicação

A controladora comunicará com o PC via RS232, porta série, e, opcionalmente, *ethernet**;*** Ambas as ligações são bidireccionais, servem para configurar os parâmetros das entradas e das saídas, obter o estado do sistema, carregar uma versão nova do circuito e outras operações que venha a tornar-se necessárias;

Um conector de expansão permite comunicar com módulos adicionais que se possa desenvolver posteriormente.

As funções que a controladora deve disponibilizar ao utilizador são:

- Ler e escrever em registos de configuração de entradas e saídas;
- Ler e escrever na memória de massa;
- Escrever um ficheiro novo de configuração da FPGA;

A arquitectura deve ser tal que permita adicionar novas operações e instruções respectivas sem a necessidade de reformulação dos blocos existentes.

3.4 – Outras características

- Entradas e saídas com impedância característica de 50 Ω;
- Alimentação da controladora com 220VAC
- Regulação interna das tensões de alimentação;
- Isolamento dos pinos da *FPGA* acessíveis ao exterior, ou seja, nenhum pino da *FPGA* poderá ser directamente acessível do exterior;
- Uma memória de massa não volátil para armazenar a versão do circuito, endereços e configuração da controladora e outros dados necessários;
- Circuito embutido para escrita na memória que armazena a configuração da *FPGA*;
- Sinalizadores luminosos para informar do estado da controladora sem o auxílio do PC;

4 – Arquitectura Interna

A arquitectura de baixo nível e, sobretudo, a implementação são fortemente dependentes dos circuitos externos à FPGA usados em cada função. Por este motivo começaremos por especificar a controladora com os blocos individuais numa perspectiva funcional. Posteriormente serão descritos os detalhes específicos à implementação, com todos os sinais e operações elementares envolvidos, de acordo com os circuitos a utilizar e dados fornecidos pelos fabricantes relativamente à sua utilização, e que permita directamente o desenvolvimento de todos os circuitos correspondentes.

4.1 – Visão Geral

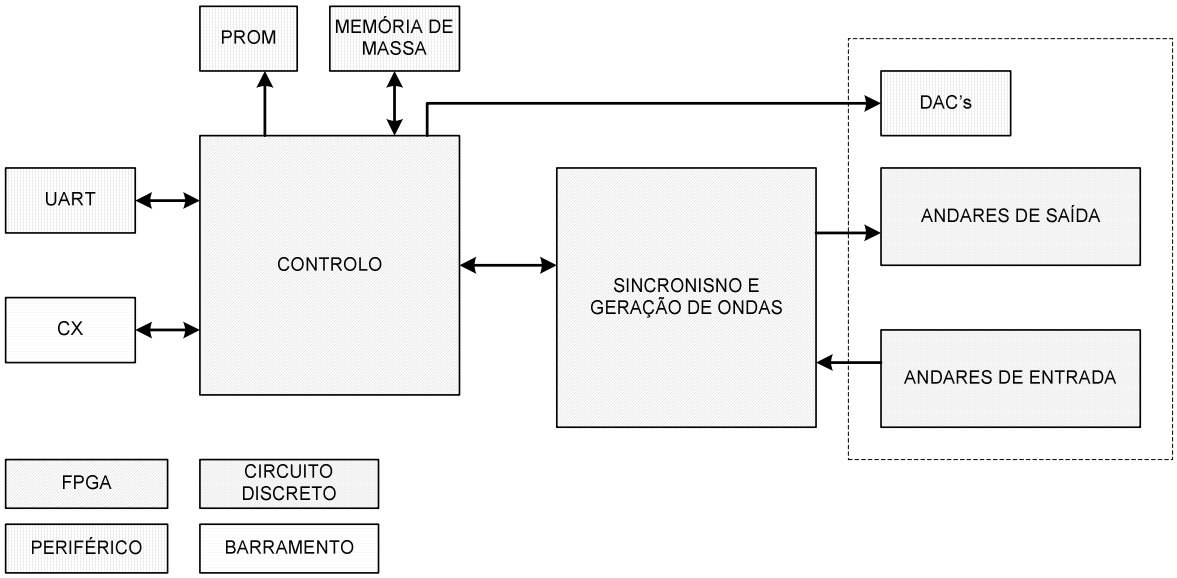


Figura 3 – Arquitectura de alto nível

SINCRONISMO E GERAÇÃO DE ONDAS – SGO

Bloco encarregue da geração dos sinais lógicos das ondas de saída. Neste bloco é feita a selecção entre modo *Master* ou *Slave* e *Clock* interno ou externo.

ANDARES DE SAÍDA

Andares que convertem o nível lógico de saída na tensão V_H ou V_L correspondente.

ANDARES DE ENTRADA:

Convertem os sinais de entrada (*Clock* e *Trigger* externo) em sinais digitais para processamento na FPGA.

DAC's:

As DAC's geram as tensões necessárias ao funcionamento dos andares de entrada/saída.

CONTROLO:

O bloco de controlo gere a comunicação da controladora com o exterior, executando as instruções recebidas, e configura o bloco SGO e as DAC's adequadamente.

NB: O sinal de relógio usado para a parte de controlo é **sempre** o relógio interno da controladora, independentemente do funcionamento em modo *Master* ou *Slave*

Os blocos identificados como periféricos na figura 3 são circuitos integrados externos à FPGA.

4.2 – Blocos Elementares

4.2.1 – Sincronismo e Geração de Ondas

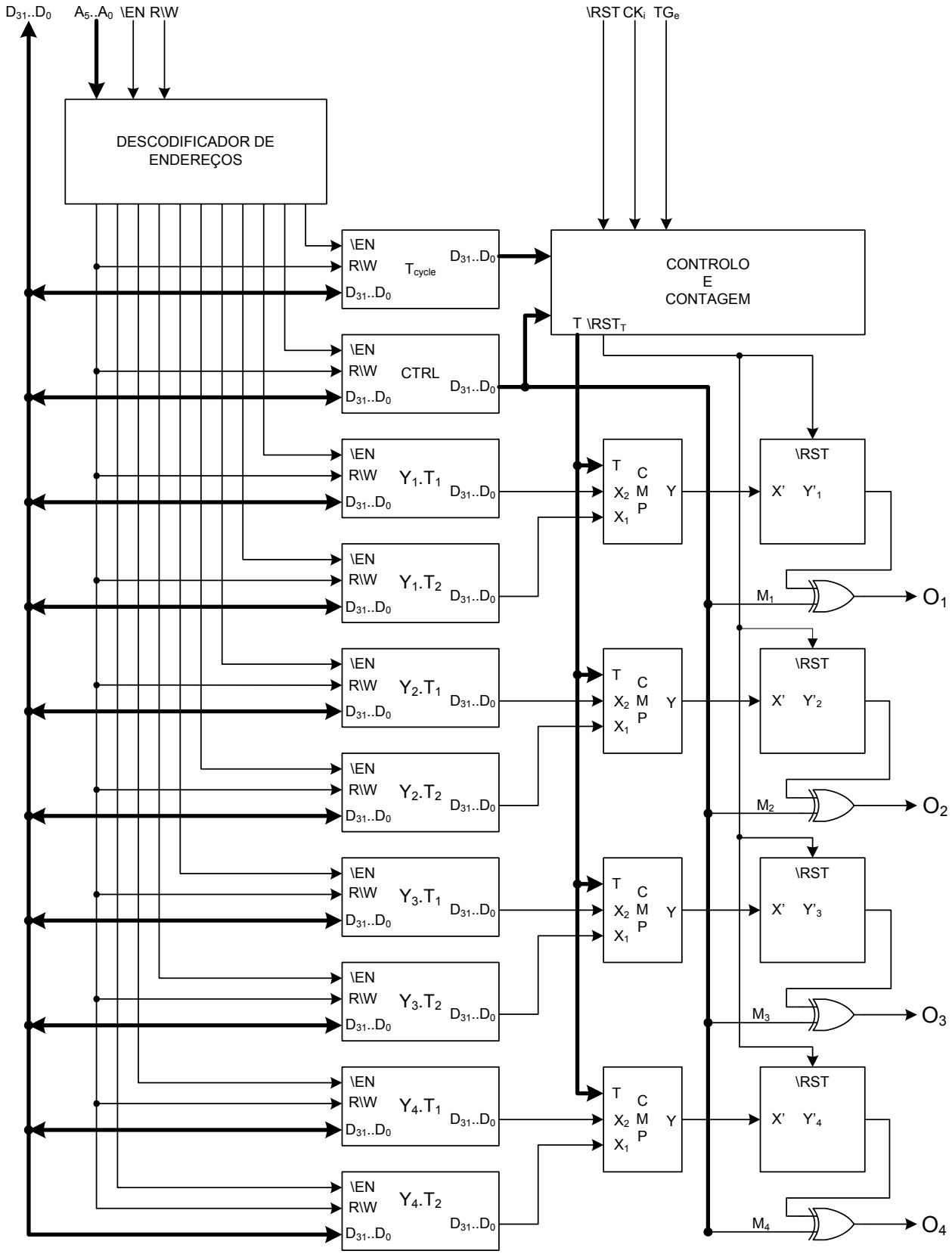


Figura 4 – Sincronismo e Geração de Ondas

NB: Na figura estão apenas representadas quatro saídas. A controladora tem oito saídas, sendo uma extensão da figura anterior.

i. Controlo e Contagem

Este componente do sistema sincroniza os blocos de geração de onda, faz a selecção do sinal de relógio interno ou externo e modo *Master* ou *Slave*, de acordo com a palavra CTRL. Faz a contagem dos ciclos de relógio de 0 até T_{cycle} sendo o valor máximo admissível para T_{cycle} de 0xFFFFFFFE. O valor da contagem é distribuído pelos comparadores.

Os sinais deste bloco são:

TG_e – *Trigger* Externo

Este sinal indica se o sinal na entrada de *Trigger* está acima ou abaixo do nível de transição do *Trigger* externo. Com a controladora no modo Slave uma transição (cuja direcção depende de CTRL) inicia um novo varrimento das formas de onda. No funcionamento em Modo *Master* esta entrada é ignorada.

CK_i – Relógio Interno

Sinal proveniente do oscilador embutido do sistema.

\RST_T

Saída de *reset* activa a '0' sempre que o valor de T é 0.

ii. Descodificador

Faz a tradução das linhas de endereço para até 64 linhas individuais de *Enable* de cada registo.

Este bloco tem os seguintes sinais:

R\W – Read\Write

Sinal de selecção de leitura/escrita. Quando a '1' os registos ficam em modo de leitura. Se estiver a '0' os registos ficam em modo de escrita.

\EN – *Enable*

Enable activa o *array* de registos para leitura ou escrita. Com R\W a '1' e \EN a '0' o conteúdo do registo endereçado é colocado no barramento de dados. Se R\W é '0' o endereço e o conteúdo do barramento dos dados são amostrados na transição descendente de \EN e os dados ficam escritos no registo endereçado.

Quando \EN se encontra a '1' o barramento de dados fica em alta impedância.

$D_{31}..D_0$ – Barramento de dados

Barramento bidireccional. Neste barramento são colocados os dados envolvidos nas operações de leitura/escrita.

NB: O barramento de dados não faz parte do bloco descodificador mas a sua descrição pode ser feita no âmbito deste bloco sem perda de generalidade.

$A_4..A_0$ – Endereço dos registos

Barramento de entrada. São entradas que, conforme o seu valor, seleccionam um registo para leitura ou escrita. A tabela 1 mostra o endereço de cada registo e a sua descrição.

Endereço	Registo	Descrição
0	O1.1	Instante de transição da saída 1
1	O1.2	Instante de transição da saída 1
2	O2.1	Instante de transição da saída 2
3	O2.2	Instante de transição da saída 2
4	O3.1	Instante de transição da saída 3
5	O3.2	Instante de transição da saída 3
6	O4.1	Instante de transição da saída 4
7	O4.2	Instante de transição da saída 4
8	O5.1	Instante de transição da saída 5
9	O5.2	Instante de transição da saída 5
10	O6.1	Instante de transição da saída 6
11	O6.2	Instante de transição da saída 6
12	O7.1	Instante de transição da saída 7
13	O7.2	Instante de transição da saída 7
14	O8.1	Instante de transição da saída 8
15	O8.2	Instante de transição da saída 8
16 até 29	RPUF	Reservado para uso futuro
30	T_{cycle}	Período das ondas
31	CTRL	Palavra de controlo

Tabela 1 – Endereçamento dos registos de SGO

iii. Comparador

Este bloco compara o valor dos dois registos de entrada com o valor da contagem, T. A saída é '1' sempre que T iguala o conteúdo de qualquer um dos dois registos.

Cada onda tem associados dois registos, Oi.1 e Oi.2, que representam dois instantes de transição. Não há quaisquer imposições quanto ao seu valor, isto é, podem ser iguais, ou qualquer um menor do que o outro. O modo *standby* é definido escrevendo em Oi.1 e Oi.2 o valor 0xFFFFFFFF.

iv. Y'

Este bloco complementa o a saída a cada transição do sinal de entrada, proveniente do comparador. Possui uma entrada de *reset* assíncrona que força '0' na saída, quando activada.

O resultado final de cada saída da FPGA é dado por $Y_i = Y_i' \oplus M_i$, em que M_i é o bit de máscara para a onda i. Quando Mi='0' a saída da FPGA é um sinal ascendente. Devido ao andar de saída ser inversor, a onda de saída da controladora é o complemento de Yi.

NB: Por *sinal de saída da FPGA* ou *sinal de saída do gerador* entende-se o sinal digital que sai da FPGA e entra no andar de saída da controladora. O sinal a que o utilizador de controladora tem acesso é referido por *onda de saída da controladora*.

v. Registos de Controlo

T_{cycle}

Valor máximo da contagem, T, ao fim do qual a contagem é reiniciada, em modo *Master*. Não tem significado em modo *Slave.*

CTRL – *Control Word*

Este registo é lido pelo bloco do contador para determinar o modo de funcionamento e contém a máscara que define a polaridade das saídas.

A alteração isolada de um único parâmetro de funcionamento tem de ser feita segundo o princípio de Ler-Modificar-Escrever.

A seguir apresenta-se o formato da palavra de controlo.

31	30	29	28	27	26	25	24	23 até 2	1	0
M_1	**M_2**	**M_3**	**M_4**	**M_5**	**M_6**	**M_7**	**M_8**	**Não Utilizado**	**T_a\T_d**	**M\S**

Tabela 2 – Formato da palavra de controlo

M\S – *MasterSlave***

Selecciona modo *Master* quando a '1' e *Slave* quando a '0'. No modo *Slave* o *trigger* externo é responsável pelo início do varrimento de um ciclo.

T_a\T_d – *Trigger* ascendente \ descendente

Quando a '1' a controladora reage à passagem do *trigger* pelo nível de transição no sentido ascendente. Quando a '0' a controladora reage à passagem descendente pelo nível de transição. Caso M\S seja '1' este *bit* é ignorado.

M_i

Este bit define a polaridade da onda i de saída da controladora. Se M_i estiver a '1' a sequência na onda de saída i da controladora é 010. Se M_i estiver a '0' a sequência na saída i é 101.

4.2.2 Controlo

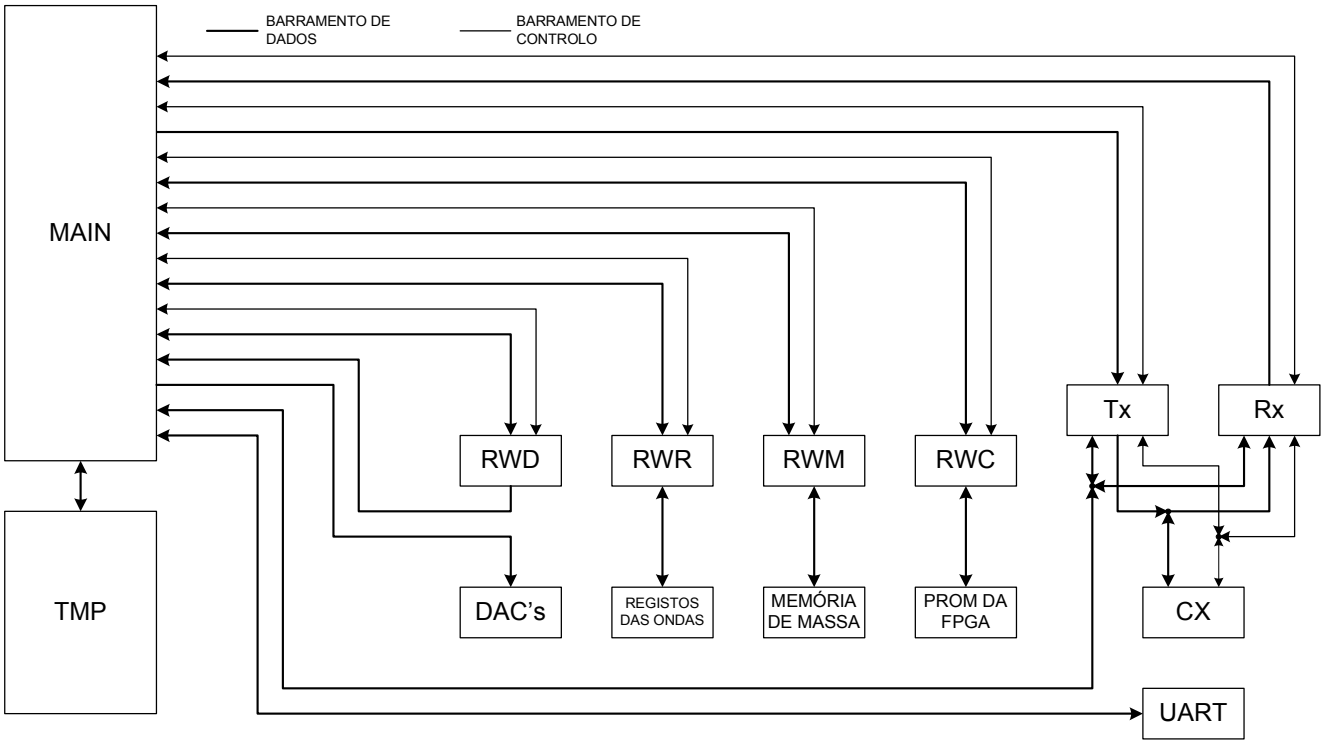


Figura 5 – Arquitectura interna de controlo (baixo nível)

O bloco de controlo é constituído por FSM's, cada uma delas dedicada a uma função elementar específica a um determinado periférico. Deste modo foi necessário definir FSM's para:

- Receber uma mensagem do PC ou de CX. (Rx)
- Enviar uma mensagem para o PC ou para CX. (Tx)
- Ler e escrever no *array* dos registos de SGO. (RWR)
- Ler e escrever os valores das DAC's dos andares de entrada e saída. (RWD)
- Ler e escrever na memória de massa. (RWM)
- Escrever novos dados na memória de configuração da FPGA. (RWC)
- Sincronizar todas as FSM's anteriores. (MAIN)
- Gerir o arranque da controladora (STARTUP), configurando todos os periféricos.

Para passagem de parâmetros entre as várias FSM's existe uma memória, TMP, (8 bytes) onde são armazenadas a mensagem recebida e a mensagem a transmitir.

i. **Princípio de funcionamento (exemplo para uma instrução)**

Após ligar a alimentação MAIN configura a UART e as DACS. O ciclo de funcionamento começa quando é iniciada a FSM Rx e a controladora fica à espera de uma instrução válida, quer de CX, quer da UART. Rx coloca na memória TMP a mensagem recebida e sinaliza a MAIN que acabou. De acordo com a instrução recebida MAIN inicia a FSM correspondente. A FSM iniciada executa a instrução, coloca em TMP uma mensagem de resposta que pode ser de *acknowledge*, ACK, erro, ERR, ou os dados pedidos na instrução e regressa ao estado de espera. MAIN retoma e inicia Tx para enviar a mensagem deixada em TMP. A partir deste ponto o funcionamento e cíclico.

5 – Descrição funcional das FSM's

Nesta secção será descrita a sequência de operações de cada FSM sem atender aos sinais envolvidos nessas operações. No Anexo 1. encontram-se os diagramas de estados detalhados de cada FSM.

5.1 – Rx

Comunica directamente com a UART, CX e com a memória TMP. A cada início de Rx é decidido de onde vai ficar à espera da próxima instrução e guarda numa *flag*. Esta *flag* é lida por Tx para saber a quem enviar cada resposta. Se a origem for CX, Rx sinaliza este para que possa escrever em TMP a mensagem com a devida formatação; se a mensagem vier da UART vamos ler 8 bytes desta. Após a recepção do primeiro byte da UART é iniciado um temporizador para evitar o bloqueio da controladora à espera de um byte que se tenha, eventualmente, perdido.

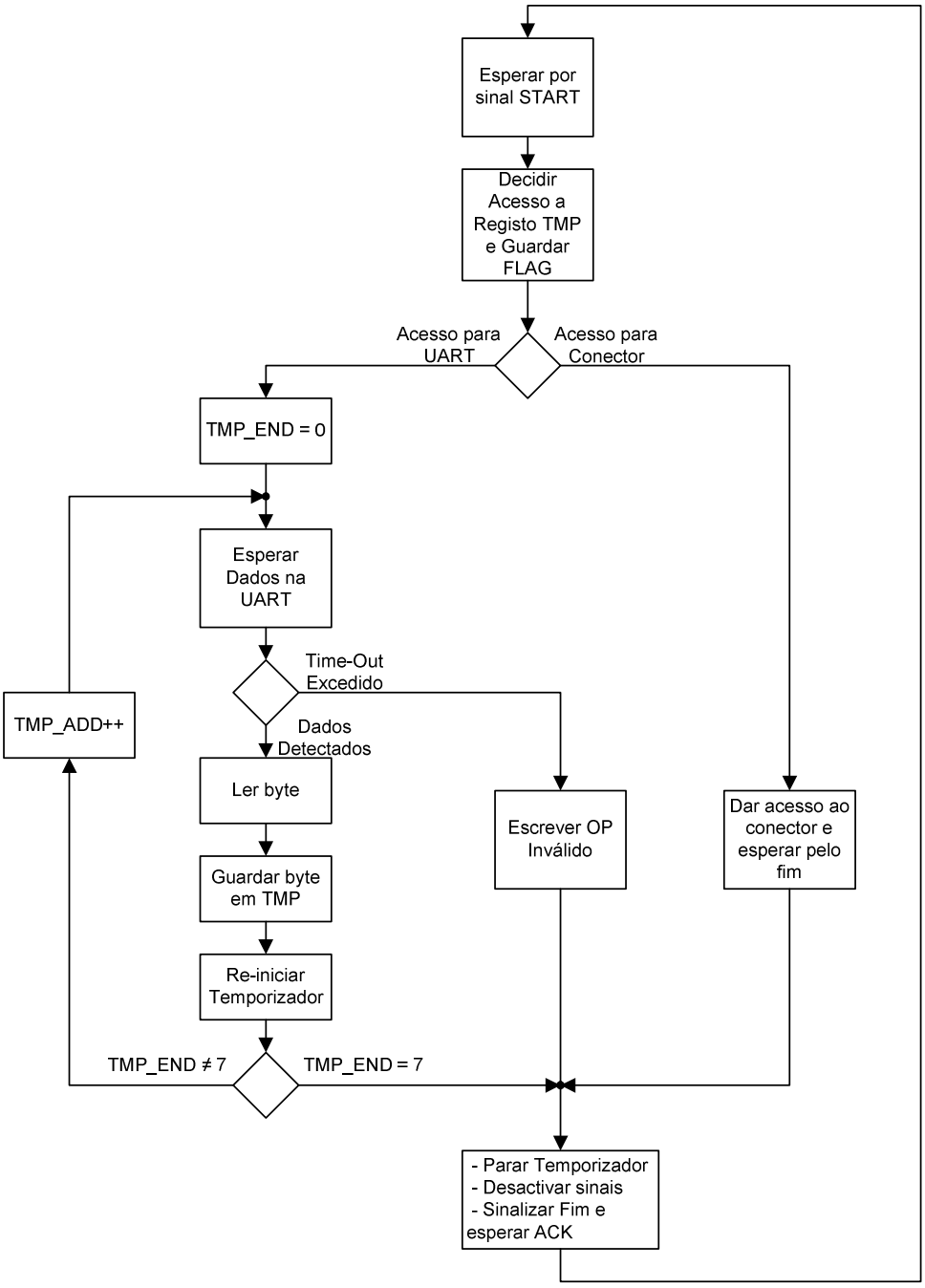


Figura 6 – Diagrama de estados abstracto de Rx

5.2 – Tx

Comunica directamente com a UART, CX e com a memória TMP. Usa a *flag* de Rx para decidir a quem enviar a mensagem.

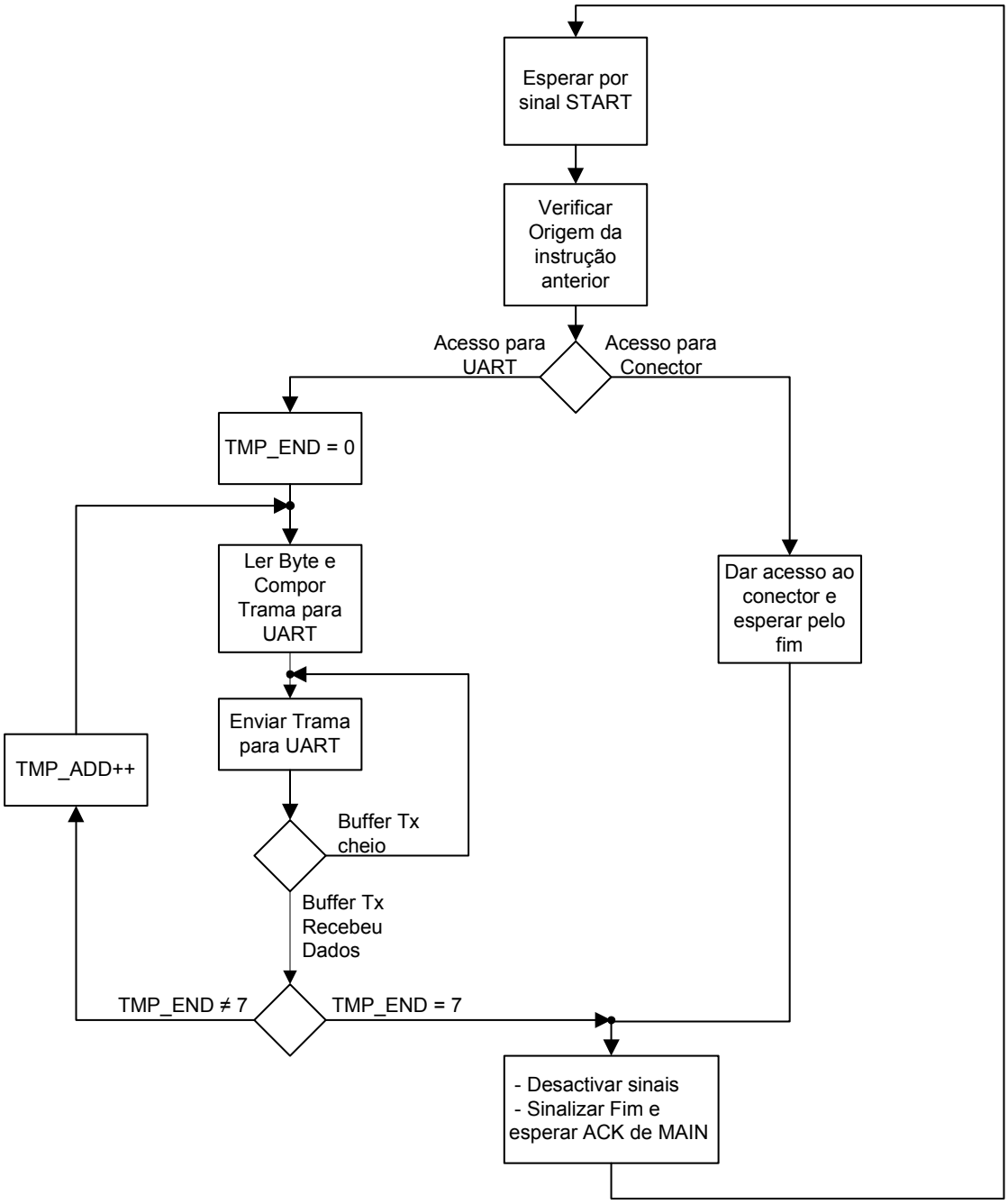


Figura 7 – Diagrama de estados abstracto de Tx

5.3 – RWR (*Read / Write Register*)

Comunica com TMP para ler a instrução e compor a mensagem de resposta, e com SGO para escrever ou ler nos registos pretendidos.

Instruções:

- **WTR –** Escrita num registo. Lê os dados e os endereços de TMP e escreve adequadamente nos registos de SGO;
- **RTR –** Leitura de um registo. Lê o endereço e coloca em TMP uma mensagem com os dados pedidos.

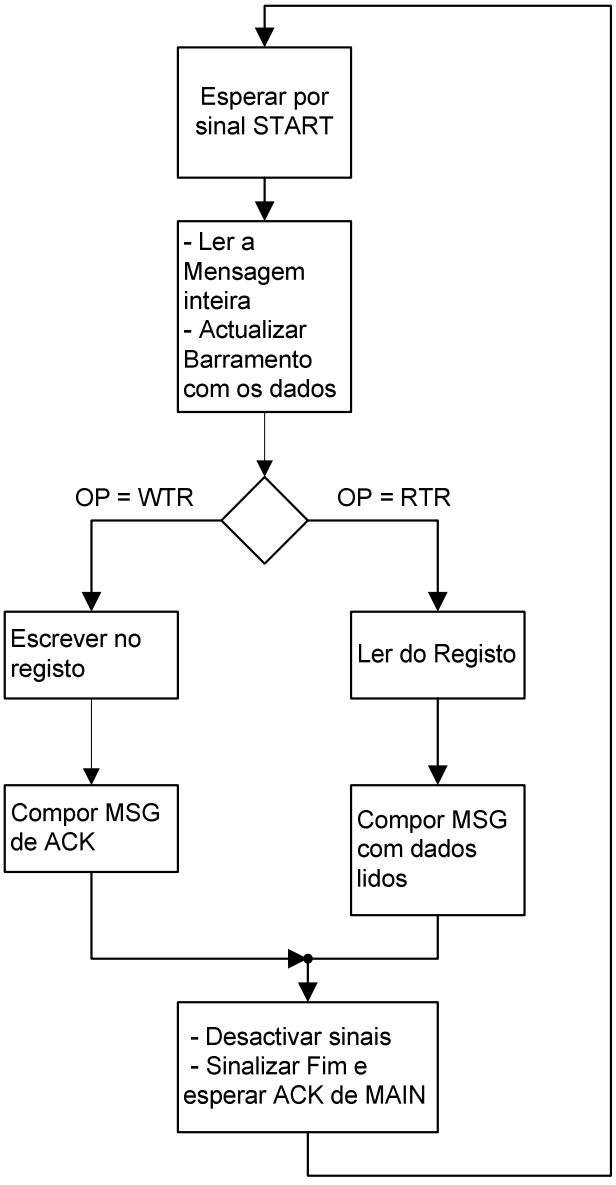


Figura 8 – Diagrama de estados abstracto de RWR

5.4 – RWM (*Read / Write Memory*)

Comunica com TMP para ler a instrução e compor a mensagem de resposta, e com a memória de massa para escrever ou ler nos endereços pretendidos.

Instruções:

- **WBM –** Escrita de um byte na memória de massa. São lidos os dados e o endereço para escrever na memória.
- **RBM –** Leitura de um byte na memória de massa. Lê o endereço e coloca em TMP uma mensagem com os dados pedidos.

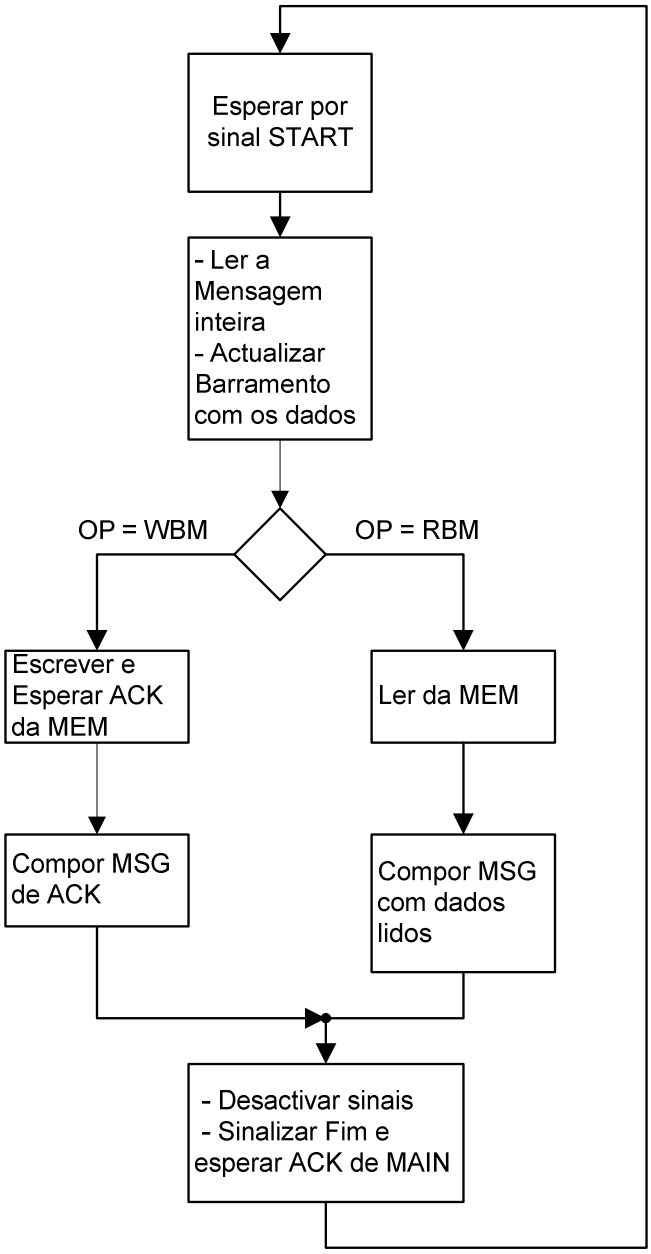


Figura 9 – Diagrama de estados abstracto de RWM

5.5 – RWD (*Read / Write DAC*)

Comunica com TMP e com as DAC's. Como não é possível ler directamente os valores das DAC's é usada uma memória onde são mantidas cópias dos valores digitais de cada DAC. Na instrução é fornecida a trama a enviar para a DAC e o respectivo endereço, que faz parte do código da instrução.

Instruções:

- **WDD** – Escrever dados numa DAC. É lida a trama e o endereço da DAC de TMP. A DAC é actualizada, bem como um registo com a cópia do seu valor.
- **RDD** – Leitura dados de uma DAC. O endereço do registo cópia é lido de TMP e é colocado em TMP o valor desse registo.

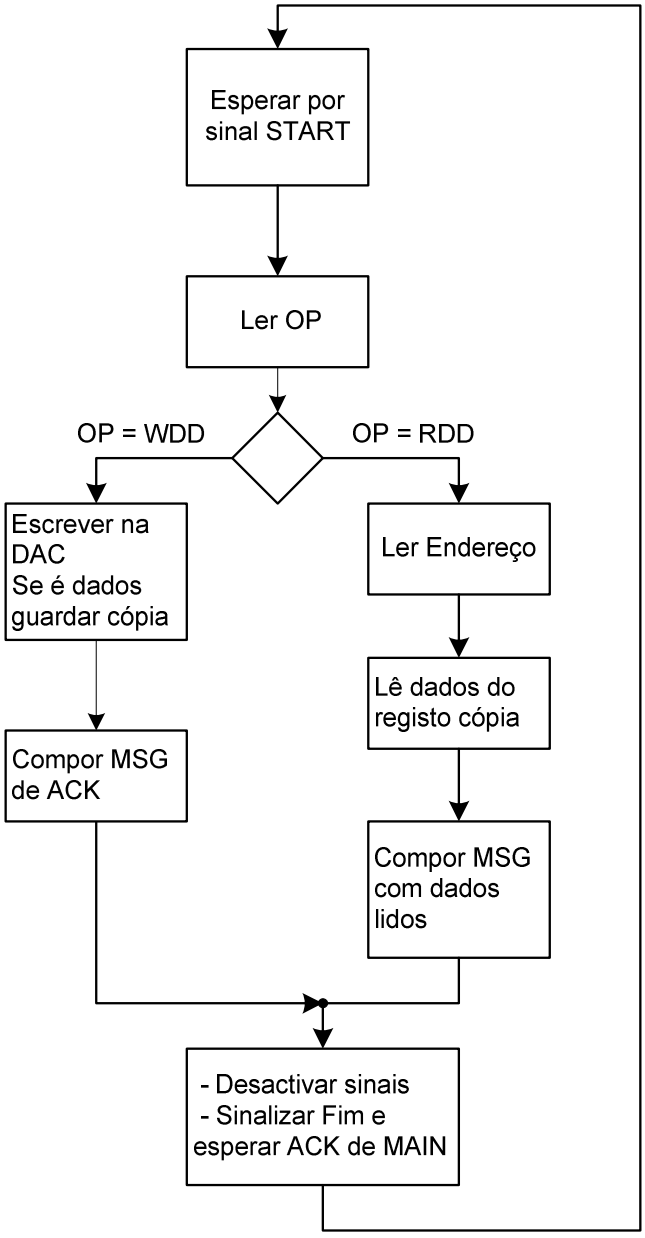


Figura 10 – Diagrama de estados abstracto de RWD

5.6 – RWC (*Read / Write Configuration*)

Comunica com TMP e a memória de configuração.

Instruções:

- **SATP –** Início da programação. É apagada a memória e iniciada uma operação de escrita a partir do endereço 0.
- **WP –** Escrita na PROM. Escreve os novos dados de configuração.
- **FP –** Fim de programação. É enviada uma sequência de finalização da escrita de dados na memória.

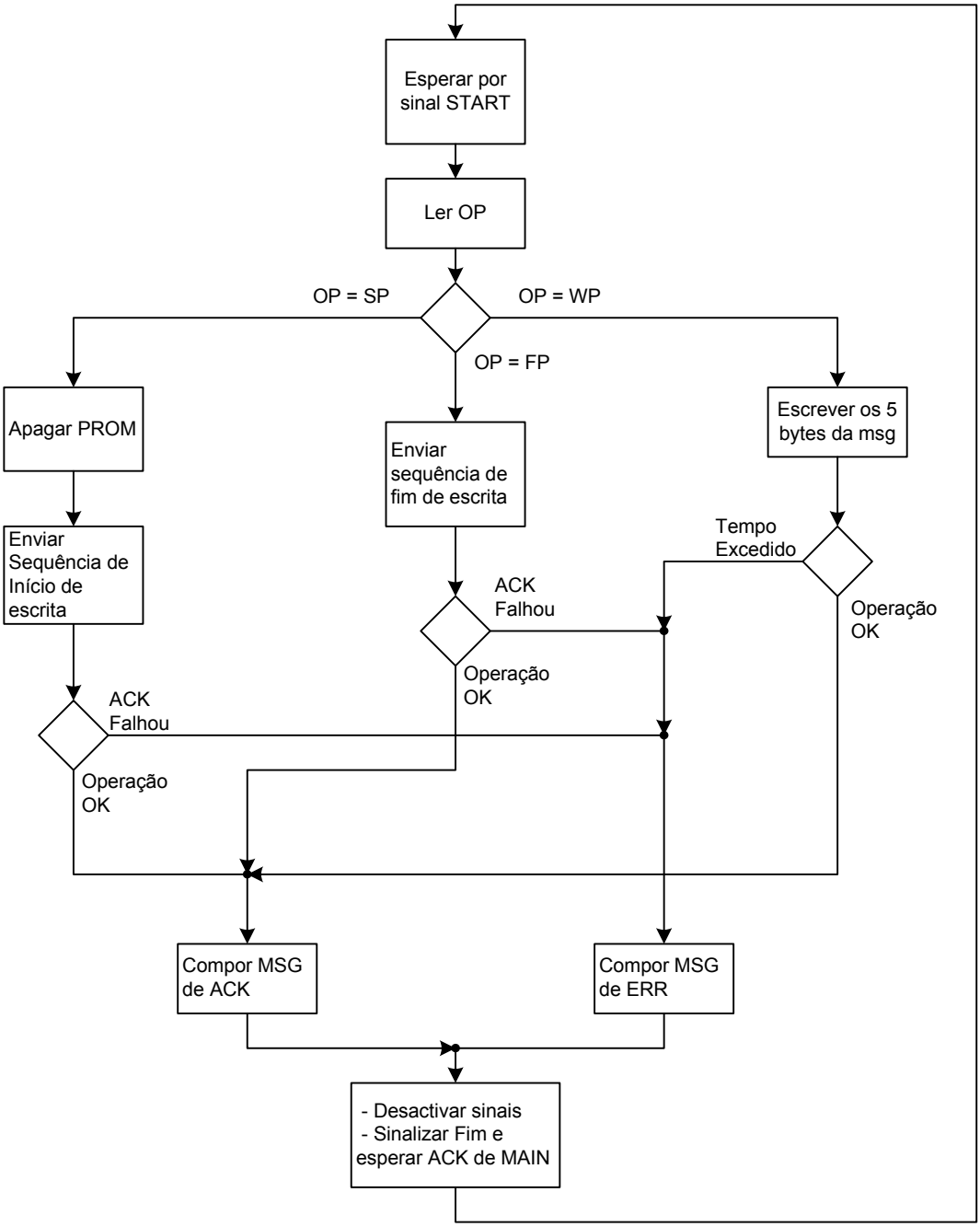


Figura 11– Diagrama de estados abstracto de RWC

5.7 – MAIN

Coordena entre si as FSM's anteriores, atribuindo o acesso a TMP consoante necessário e iniciando a FSM correspondente a cada instrução. Verifica a correcção das mensagens e gera mensagens de erro caso sejam inválidas.

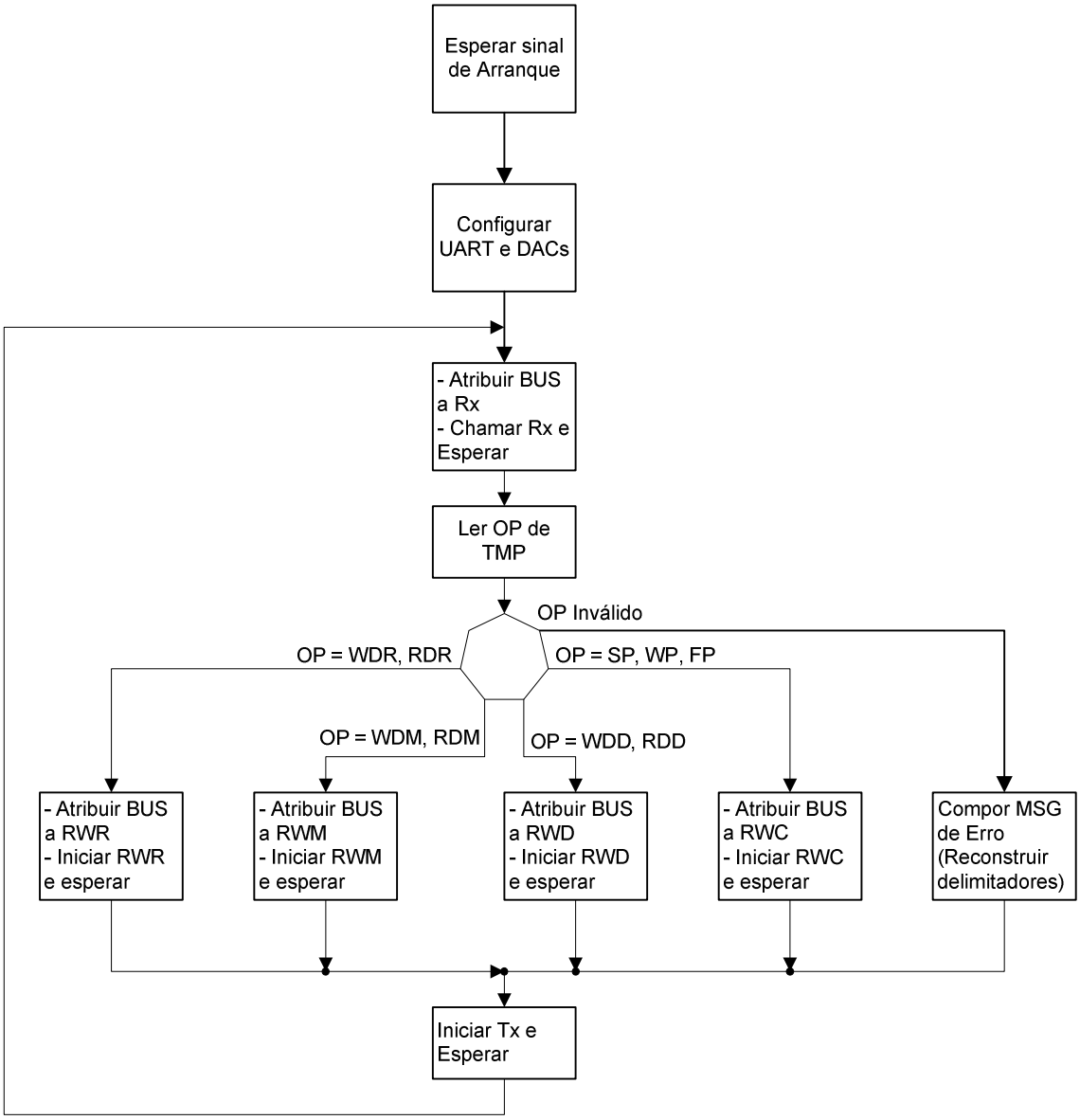


Figura 12 – Diagrama de estados abstracto de MA

No Anexo 1. estão os diagramas de estados detalhados das unidades que compõem a controladora. Os diagramas de estados representam as operações individuais das máquinas de estados numa sintaxe semelhante à da linguagem VHDL. A sequência de operações é determinada pela informação retirada dos *datasheets* dos componentes utilizados.

6 – Formato das mensagens

A controladora escuta os portos para receber uma instrução, executa a instrução e envia pelo porto de origem uma mensagem que confirma a execução e, quando aplicável, devolve dados ao chamador.

As mensagens envolvidas na comunicação em qualquer um dos sentidos têm todas 8 bytes. A formatação das mensagens é a mesma quer no caso de comunicação com a UART ou com o conector de expansão. A recepção e transmissão com a UART é feita usando uma FSM que acede à UART e lê ou escreve em TMP. Por outro lado, para o conector de expansão é disponibilizado na ficha o barramento que acede directamente à memória TMP para leitura e escrita, tendo o controlador acima de CX a tarefa de escrever ou ler de TMP, consoante necessário.

Todas as mensagens começam com o *byte* 0x00 e terminam com 0xFF. O *byte* 2 de qualquer mensagem, OP, define a natureza da mensagem. Os restantes *bytes* são interpretados no contexto da instrução OP.

A figura seguinte mostra o formato de uma mensagem genérica, incluindo a ordem em que os *bytes* são enviados.

1	2	3	4	5	6	7	8
0x00	**OP**	**X**	**X**	**X**	**X**	**X**	**0xFF**

Figura 13 – Formato genérico de uma mensagem.

Nas instruções dadas à controladora OP é dividido em 2 *nibbles*. O mais significativo identifica a FSM que MAIN deve iniciar. O menos significativo indica, dentro da FSM iniciada, qual a instrução a executar. A tabela 4 contém todos os valores de OP usados nas instruções dadas à controladora.

6.1 – Mensagens de resposta genéricas

Quando não é necessário a controladora devolver quaisquer dados, para além da confirmação de mensagem recebida é enviada a mensagem representada na figura 14.

1	2	3	4	5	6	7	8
0x00	**OP**	**X**	**X**	**X**	**X**	**X**	**0xFF**

Figura 14 – Mensagem de resposta genérica

Os valores possíveis de OP e significado correspondente são:

OP	Código	Descrição
ACK	0X01	Instrução executada com sucesso
ERR	0X02	Erro ao executar instrução ou instrução inválida

Tabela 3 – Códigos OP nas mensagens de resposta.

6.2 – Instruções e mensagens de resposta

Neste ponto são descritas as instruções recebidas pela controladora e respectiva resposta.

WTR – *Write Timing Register*

Escrita da palavra de 32 bits no registo de SGO.

1	2	3	4	5	6	7	8
0x00	**OP**	**ADD**	**D_{HH}**	**D_{HL}**	**D_{LH}**	**D_{LL}**	**0xFF**

Figura 15 – Instrução *Write to Timing Register*

Byte ADD:

0	**0**	**ADD_5**	**ADD_4**	**ADD_3**	**ADD_2**	**ADD_1**	**ADD_0**

Figura 16 – Byte de endereço em WTR

No byte ADD os bits de ADD_5 a ADD_0 contém o endereço do registo a escrever; os bits não usados são colocados a '0'. O *byte* D_{HH} é a parte mais significativa a escrever e D_{LL} é a parte menos significativa.

Após terminar a instrução de escrita no registo a controladora responde com ACK.

RTR – *Read from Timing Register*

Leitura da palavra de 32 bits armazenada no registo de SGO.

1	2	3	4	5	6	7	8
0x00	**OP**	**ADD**	**X**	**X**	**X**	**X**	**0xFF**

Figura 17 – Instrução *Read from Timing Register*

O byte ADD tem a estrutura descrita para a instrução WTR.

A controladora responde com a seguinte mensagem. OP tem o valor ACK.

1	2	3	4	5	6	7	8
0x00	**OP**	**D_{HH}**	**D_{HL}**	**D_{LH}**	**D_{LL}**	**X**	**0xFF**

Figura 18 – Resposta a RTR

WBM – *Write Byte to Memory*

Escrita de um Byte na memória de massa.

1	2	3	4	5	6	7	8
0x00	**OP**	**A_H**	**A_L**	**DATA**	**X**	**X**	**0xFF**

Figura 19 – Instrução *Write Byte to Memory*

Estrutura do endereço:

	A_H								**A_L**							
MSB								LSB	MSB							LSB
0	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8		A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0

Figura 20 – Endereço na instrução WBM

Os bits de endereço não utilizados devem ser colocados a '0'; DATA é o byte a escrever.

A controladora responde após a escrita com uma mensagem de ACK.

RBM – *Read Byte from Memory*

Leitura de um byte da memória de massa.

1	2	3	4	5	6	7	8
0x00	**OP**	**A_H**	**A_L**	**X**	**X**	**X**	**0xFF**

Figura 21 – Instrução Read Byte from Memory

O bytes campo de endereço é formado de modo semelhante ao da instrução WBM.

Após a leitura do byte pretendido, é devolvido no campo DATA da mensagem da figura seguinte:

1	2	3	4	5	6	7	8
0x00	**ACK**	**DATA**	**X**	**X**	**X**	**X**	**0xFF**

Figura 22 – Resposta a RBM

WDD – *Write Data to DAC*

Escrita de dados numa DAC.

1	2	3	4	5	6	7	8
0x00	**OP**	**W_H**	**W_L**	**X**	**X**	**X**	**0xFF**

Figura 23 – Instrução WDD

Os bytes W_H e W_L compõem a palavra de 16 bits, tal qual como deve ser escrita na DAC correspondente. No *nibble* menos significativo do byte OP é indicada qual a DAC onde será escrita a palavra de 16 bits. A controladora responde com ACK.

Os valores de tensão armazenados nas DAC's dos sinais de saída são o nível baixo, V_L, e a amplitude da onda, V_{HL}. Na tabela seguinte encontram-se os endereços da DAC correspondentes à tensão de cada saída.

Endereço	Função
0	$Y1.V_L$
1	$Y1.V_{HL}$
2	$Y2.V_L$
3	$Y2.V_{HL}$
4	$Y3.V_L$
5	$Y3.V_{HL}$
6	$Y4.V_L$
7	$Y4.V_{HL}$
8	$Y5.V_L$
9	$Y5.V_{HL}$
10	$Y6.V_L$
11	$Y6.V_{HL}$
12	$Y7.V_L$
13	$Y7.V_{HL}$
14	$Y8.V_L$
15	$Y8.V_{HL}$
16	Vtr

Tabela 4 – Endereços das DAC's

Construção da instrução WDD:

O campo OP da instrução WDD é calculado por:
OP = 0x20 + *add*/8, sendo *add* obtido pela na tabela 4.

O byte W_H deve construir-se da seguinte maneira:
W_{H7} = '0'
$W_{H6}..W_{H4}$ = add % 8
$W_{H3}..W_{H0} = X_7..X_4$

O byte W_L deve construir-se da seguinte maneira:
$W_{L7}..W_{L4} = X_3..X_0$
$W_{L3}..W_{L0}$ = "0000"

NB: Os operadores "/" e "%" significam divisão inteira e resto da divisão inteira , respectivamente. A palavra $X_7..X_0$ é o valor do byte que se quer escrever na DAC com endereço add.

RDD – *Read Data from DAC*

Leitura do valor digital de uma DAC.

1	2	3	4	5	6	7	8
0x00	**OP**	**ADD**	**X**	**X**	**X**	**X**	**0xFF**

Figura 24 – Instrução Read Data from DAC

Endereço:

0	**0**	**0**	**A_4**	**A_3**	**A_2**	**A_1**	**A_0**

Figura 25 – Endereço na instrução RDD

Os bits de endereço não utilizados são colocados a '0'. O endereço é obtido pela tabela 4.
A controladora responde à instrução com o byte lido.

1	2	3	4	5	6	7	8
0x00	**ACK**	**DATA**	**X**	**X**	**X**	**X**	**0xFF**

Figura 26 – Resposta à instrução RDD

SP – *Start Programming*

Iniciar a sequência de programação da memória de configuração.

1	2	3	4	5	6	7	8
0x00	**OP**	**X**	**X**	**X**	**X**	**X**	**0xFF**

Figura 27 – Instrução SATP

A memória é apagada e inicializada para escrita de dados novos.

WP – *Write To PROM*

Escrita de dados novos na memória de configuração (5 bytes de cada vez).

1	2	3	4	5	6	7	8
0x00	**OP**	**D_1**	**D_2**	**D_3**	**D_4**	**D_5**	**0xFF**

Figura 28 – Instrução *Write to PROM*

Os bytes de D_1 até D_5 contém os dados a escrever na memória de configuração. D1 á o primeiro byte a ser escrito e D5 o último.

FP – *Finish Programming*

Finalização da escrita de dados novos na memória de configuração.

1	2	3	4	5	6	7	8
0x00	**OP**	**X**	**X**	**X**	**X**	**X**	**0xFF**

Figura 29 – Instrução *Finish Programming*

A controladora responde às três ultimas mensagens com ACK ou ERR, dependendo da execução normal ou ocorrência de um erro durante a configuração. Em nenhum dos casos são devolvidos dados.

6.2.1 - Códigos OP das instruções recebidas pela controladora

Instrução	OP	Descrição
WTR	0x80	Escrever num registo de SGO
RTR	0x81	Ler de um registo de SGO
WBM	0x40	Escrever um byte na memória de massa
RBM	0x41	Ler um byte da memória de massa
WDD	0x20	Escrever mensagem na DAC 1 (integrado)
	0x21	Escrever mensagem na DAC 2 (integrado)
	0x22	Escrever mensagem na DAC 3 (integrado)
RDD	0x23	Ler uma copia do conteúdo da DAC
SP	0x10	Inicializar e memória de configuração para escrita
WP	0x11	Escrever 5 bytes de dados de configuração na mem.
FP	0x12	Finalizar a programação da memória de configuração

Tabela 5 – Códigos OP

7 – Andares de entrada e saída

7.1 – Andares de Saída

O andar de saída recebe um sinal digital da FPGA e converte-o na tensão correspondente. É alimentado a ±15V e deve ser capaz de debitar uma corrente de 200mA numa carga de 50Ω.
Em 3.1. encontra-se uma descrição mais detalhada das características do sinal de saída.

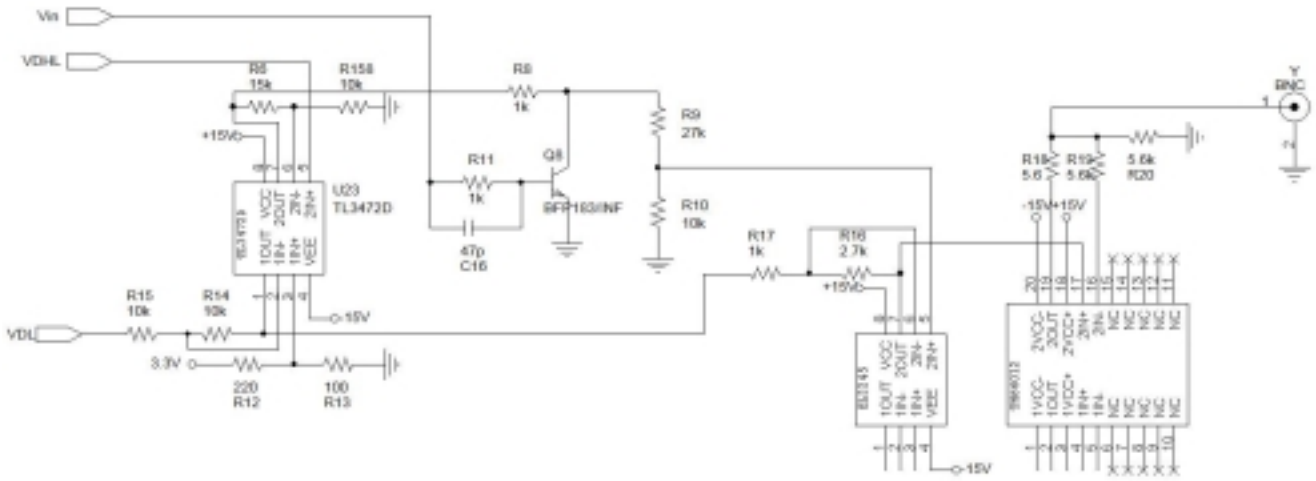


Figura 30 – Esquemático do andar de saída.

Princípio de funcionamento

O circuito base para o andar de saída é um AmpOp (EL2245) montado em configuração não-inversora. As tensões VDL e VDHL provêm de uma DAC e Vin é o sinal lógico da FPGA. Para ter na saída o nível baixo de tensão Vin tem de der '1' para a saída do inversor (BFP183) ser aproximadamente 0V. Este andar comporta-se como um inversor.

A tensão de saída é dada pela seguinte equação:

$V_{out} = 5VDHL - 10.8 + 5.4VDL$, com $0 \leq VDL, VDHL \leq 4$

VDL controla o nível baixo da onda. Com VDL igual a 0V e VDHL igual a 0V temos uma onda com $V_L = -10V$. Se VDL é 4V temos $V_L = +10V$.

VDHL controla a amplitude da onda. Com VDHL igual a 0V temos uma onda com amplitude 0V. Se VDHL for 4V a amplitude é a máxima, ou seja, 20V, o que permite ter uma onda com ±10V, se desejado.

7.2 – Andar de Entrada

O andar de entrada converte o sinal de entrada analógico numa entrada digital para a FPGA. Para a sua implementação é necessário um comparador e uma DAC que gere a tensão de referência.

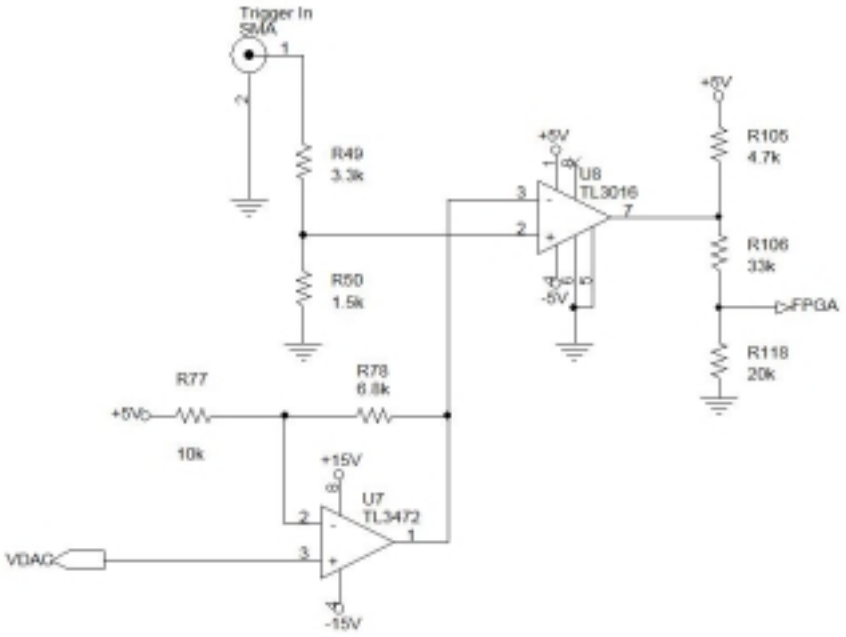


Figura 31 - Esquemático do andar de entrada

VDAC é gerado numa DAC e controla o nível de referência do comparador.

A entrada de Trigger passa por um divisor de tensão para ajustar o sinal à gama suportada pelo comparador.

A tensão de referência é dada por $V_{ref} = 1.68VDAC - 3.4$

Quando a entrada de *trigger* tem -10V temos -3.1V na entrada "+" do comparador. Para ter -3.1V na tensão de referência VDAC tem de ser 0.17V. Quando a entrada de trigger tem 10V temos 3.1V na entrada "+" do comparador. Para ter 3.1V na tensão de referência precisamos de 3.86V na DAC.

Como as DACS usadas são de oito bits e têm uma gama de saída de aproximadamente 0-4V temos aproximadamente 255 incrementos, tanto nos parâmetros da saída, como na entrada de trigger.

8. Conector de expansão:

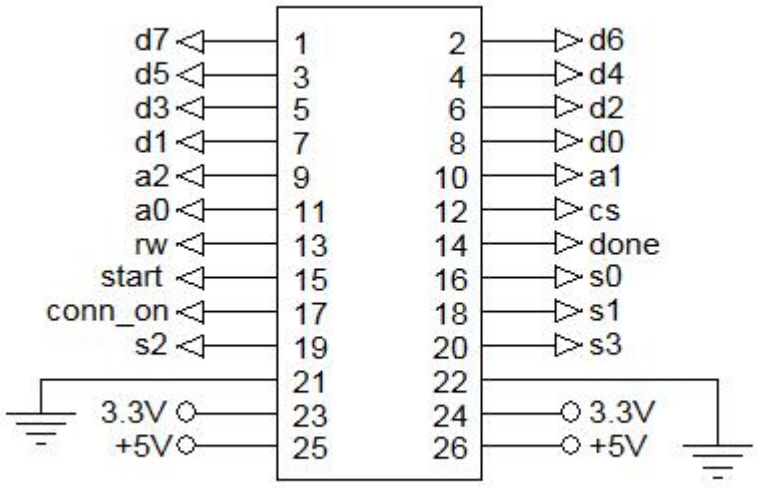


Figura 32 – Conector de expansão.

O conector é um porto que permite enviar e receber instruções com o formato descrito em 6. A diferença está no modo como as instruções são passadas, isto é, o conector tem um barramento paralelo, de oito bits, que acede directamente ao *buffer* de instruções para leitura e escrita.

Sinais:

$d_7..d_0$ – Dados

Barramento de dados bidireccional. Este barramento liga à memória que armazena as instruções.

$a_2..a_0$ – Endereço

Barramento de endereços.

rw – Leitura/Escrita

Sinal de controlo para selecção entre modo de leitura ou escrita. '1' em rw selecciona modo de leitura e '0' significa modo de escrita.

cs – *Chip Select*

Sinal de activação da memória de instrução. É activo a '0'. Enquanto estiver a '1' o barramento *d* está em alta impedância.

start – Sinal de coordenação

Esta linha a '1' indica ao dispositivo do conector que pode aceder ao buffer de instrução para ler ou escrever.

done – Sinal de coordenação

Esta linha deve estar a '0' enquanto o dispositivo do conector está inactivo ou em espera. Quando o dispositivo termina a execução deve activar o sinal *done* e só o colocar a '0' depois de *start* passar para '0'.

conn_on – Dispositivo ligado ao conector

Indicador de presença de um dispositivo no conector. Deve estar a '0' para assinalar a presença. Quando não há um dispositivo o sinal fica em aberto e um *pull-up* trá-lo para o nível '1'.

$s_3..s_0$: Quatro linhas reservadas para uso futuro.

Passagem de instruções:

Para o dispositivo do conector dar uma instrução à controladora deve primeiro assinalar presença colocando a linha *conn_on* a '0'. A seguir aguarda que a controladora active a linha *start*, após o que pode aceder ao buffer e escrever a instrução, segundo o diagrama temporal da figura 33. Após o fim da escrita activa a linha *done* e espera que *start* volte a '0' para desactivar a linha *done*.

Sempre que se escreve uma instrução é preciso ler a seguir a mensagem de retorno. Para tal deve aguardar que, a seguir à passagem da instrução, a linha *start* volte a activar, a seguir ao que pode ler do *buffer* a mensagem. Por fim activa o sinal *done* e só o desactiva depois de *start* voltar a '0'. O diagrama temporal da figura 34 mostra como deve ser feita a leitura do *buffer*.

Diagramas temporais do conector

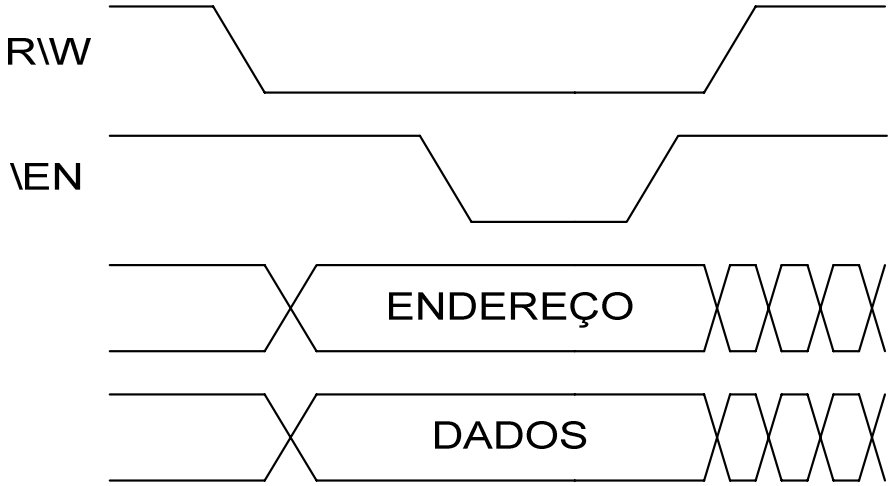


Figura 33 - Ciclo de escrita com sinais do conector

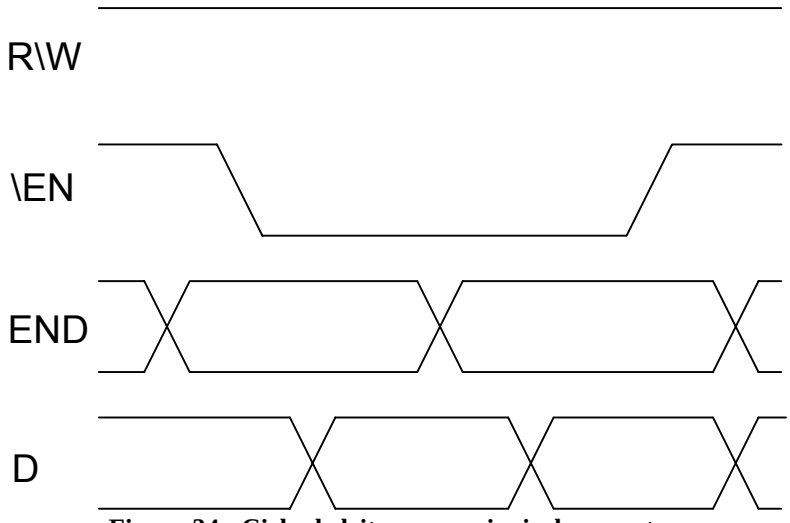


Figura 34 - Ciclo de leitura com sinais do conector.

9. Implementação:

Fizemos um circuito do andar de saída, para testar os tempos de transição do sinal. O andar cumpre a especificação do tempo de transição, isto é, uma transição de -10V para 10V demora cerca de 100ns. Este tempo mantém-se praticamente constante, independentemente da amplitude do degrau. Devido à saturação do transístor no andar de saída existe um atraso entre a transição de V_{in} de '1' para '0' e a transição ascendente da saída. Este tempo é de aproximadamente 200ns e pode ser medido facilmente. O software permite compensar este atraso ao definir a transição ascendente 200ns antes do instante definido pelo utilizador.

Para o andar de entrada também foi feita uma placa da teste. O comparador responde bem para sinais de frequência até aproximadamente 2MHz.

Decidimos dividir a controladora em duas partes, uma com os reguladores de tensão e a outra com a FPGA, andares de entrada e de saída.

Para o funcionamento são necessárias 6 tensões de alimentação:
±15V , ±5V , 3.3V e 1.8V.
As tensões de 15V são usadas nos andares de saída.
As tensões de 5V são usadas nos *rails* analógicos das DAC's e no comparador do andar de entrada.
Os 3.3V são necessários para o funcionamento dos blocos de I/O da FPGA e para alimentar toda a parte digital da controladora.
Por fim, a tensão de 1.8V é necessária para o funcionamento da lógica interna (núcleo) da FPGA.

Construímos a placa com os reguladores e encontrámos um problema com o regulador que gera as tensões da FPGA. Após algum tempo em carga a tensão começa a decair para valores que ultrapassam a tolerância dos circuitos. Por esse motivo tivemos de retirar o regulador da placa e adicionar 3 fios para ligar a uma fonte de alimentação externa, uma vez que não havia tempo para desenvolver uma solução alternativa. Com a fonte externa não tivemos problemas com a alimentação da controladora.

A segunda placa também foi montada, faltando apenas um componente que não foi possível obter junto dos representantes nem do fabricante. Trata-se da memória de configuração AT17F040-30JC, da ATMEL, necessária para armazenar a configuração da FPGA. Sem esta memória a FPGA tem de ser programada por JTAG, com um cabo ligado ao PC.

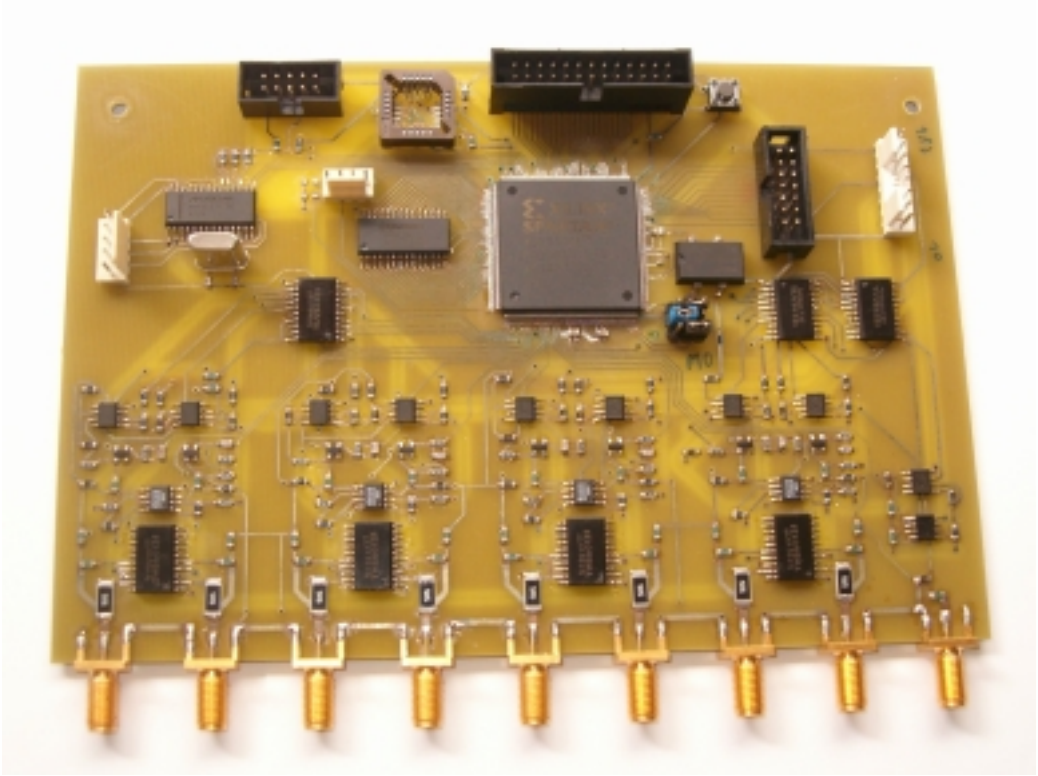


Figura 35 – Placa com a FPGA e andares de entrada/saída

Após a montagem da segunda placa falta programar a FPGA com o código VHDL que criámos. A comunicação com a FPGA é feita usando o porto JTAG da FPGA e o paralelo III da *Xilinx*, com o software iMPACT. É possível aceder à FPGA com o JTAG mas como não foi possível fazer o *debug* do código VHDL atempadamente não é possível concluir a controladora.

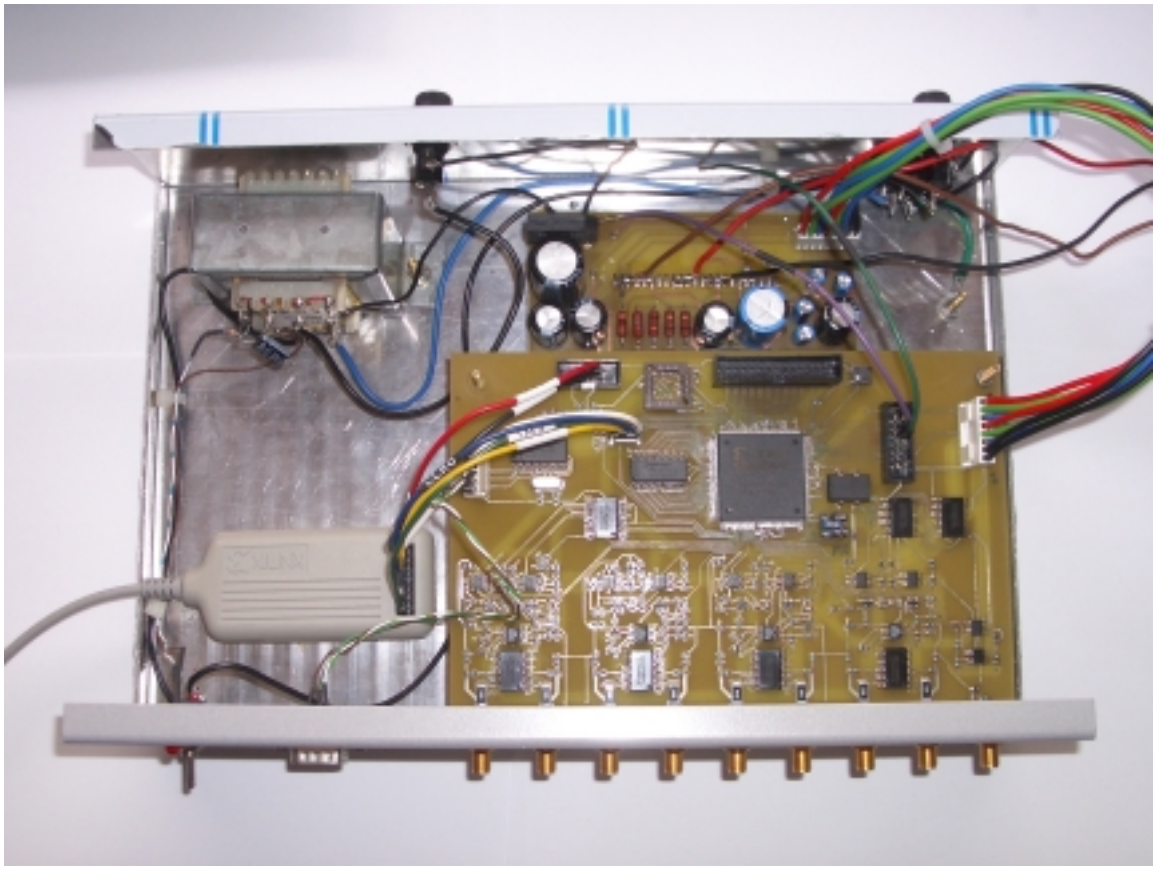


Figura 36 – Controladora montada na caixa

10. Conclusões:

Por falta de tempo não foi possível terminar o código VHDL para programar a FPGA, sendo esta a última etapa da construção da controladora.

Conseguimos cumprir com o objectivo de melhorar a performance das saídas, relativamente à da antecessora desta controladora.

Devido ao aumento do número de saídas para oito, deixou de ser possível usar o andar de saída como estava implementado na controladora antiga. A arquitectura nova tem DAC's de oito canais que são carregadas por um barramento série. Isto permitiu reduzir a complexidade do circuito impresso, que de outra forma não se poderia fazer em apenas duas camadas.

Devido ao facto da controladora passar a ter uma forma externa foi necessário redefinir toda arquitectura, representando uma ruptura com a arquitectura da controladora antiga. Enquanto que o barramento ISA passava em paralelo o endereço e os dados simultaneamente, a comunicação série implica a criação de um protocolo de mensagens e detecção de erros. Adicionalmente são necessários blocos para interpretar e executar as mensagens.

Ao aumentar a dimensão dos registo internos de contagem de tempo para 32 bits e usar um oscilador de 50MHz, o tempo de ciclo máximo é de aproximadamente 85 segundos. Por este motivo concluímos que uma entrada de relógio externo deixa de ser necessária porque a arquitectura permite um tempo de ciclo bastante maior, sem recorrer sequer a um divisor de relógio. A controladora antiga só permite um tempo de ciclo máximo de 500ms.

Apesar de não ter ficado concluída a controladora, alguns dos objectivos foram cumpridos, e conseguimos construir o hardware, que representa uma plataforma bastante flexível para desenvolvimento no futuro.

Anexo 1 – Diagramas de estado detalhados das FSM

Os diagramas de estados apresentados são baseados nos dados dos fabricantes sobre o funcionamento do circuito.

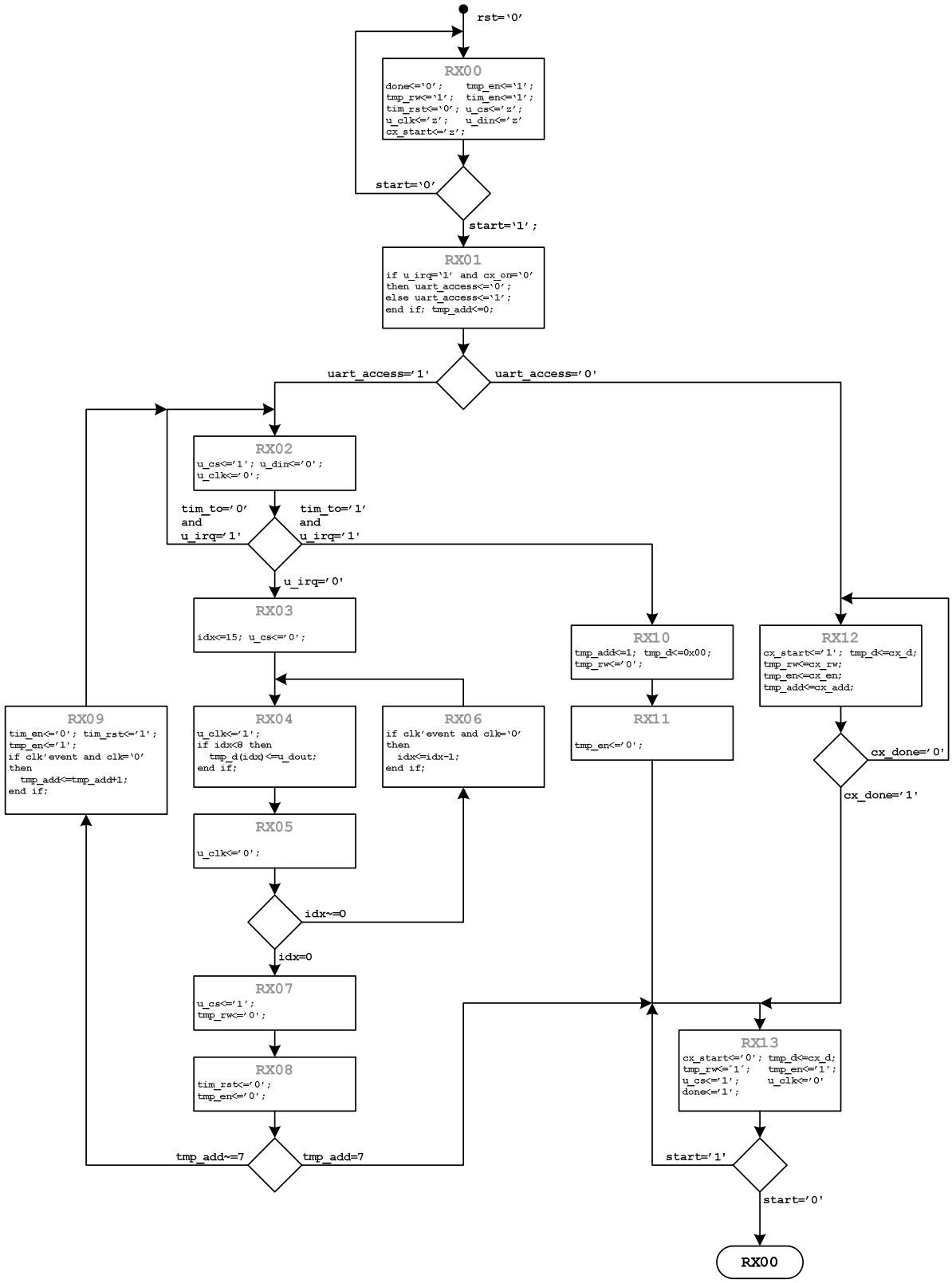


Figura 37 – Diagrama de Estados de RX

Rx está ligada à UART MAX3111E e lê os bytes recebidos pela UART.

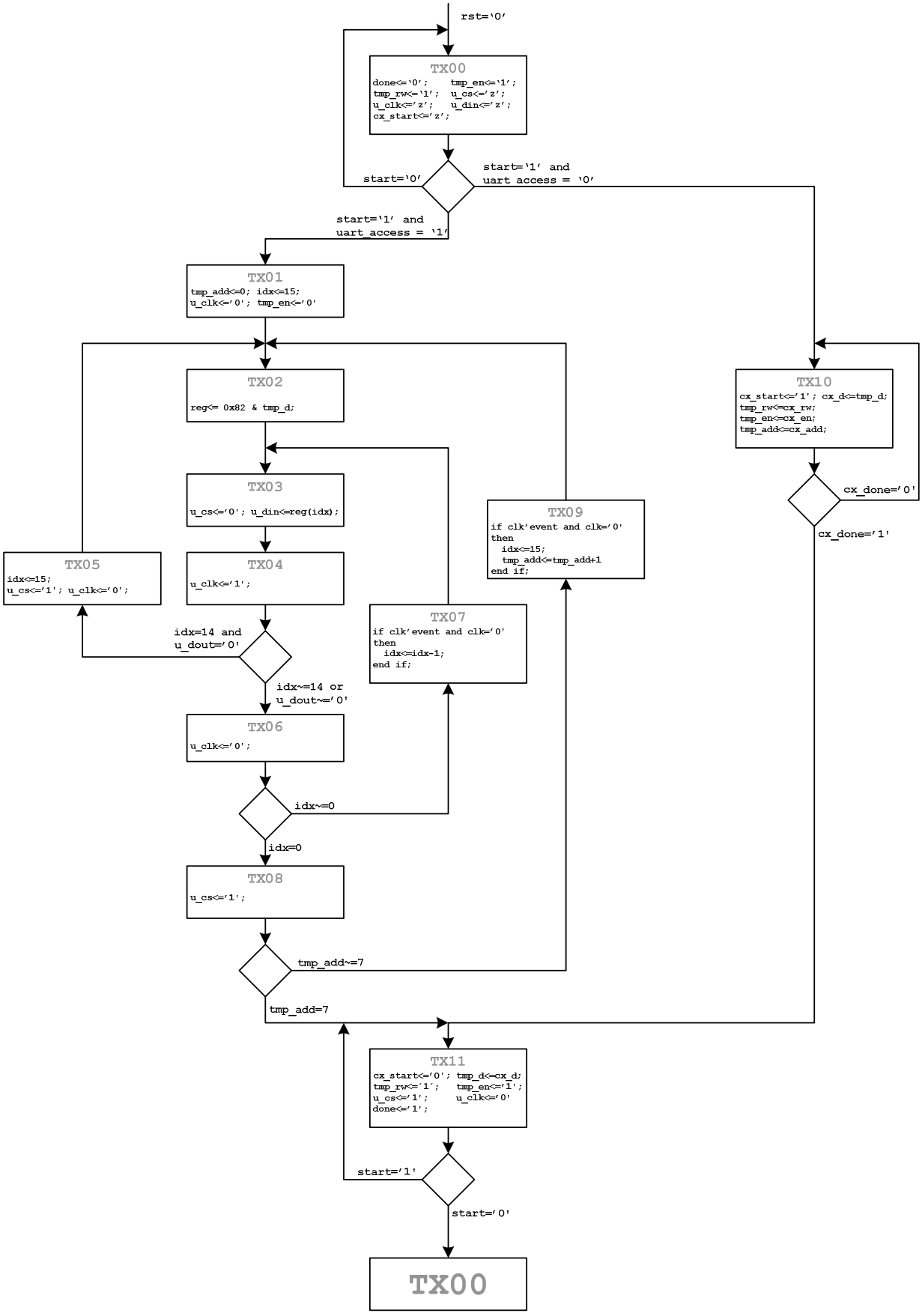


Figura 38 – Diagrama de estados de TX.

TX também comunica com a UART e escreve bytes na UART para enviar

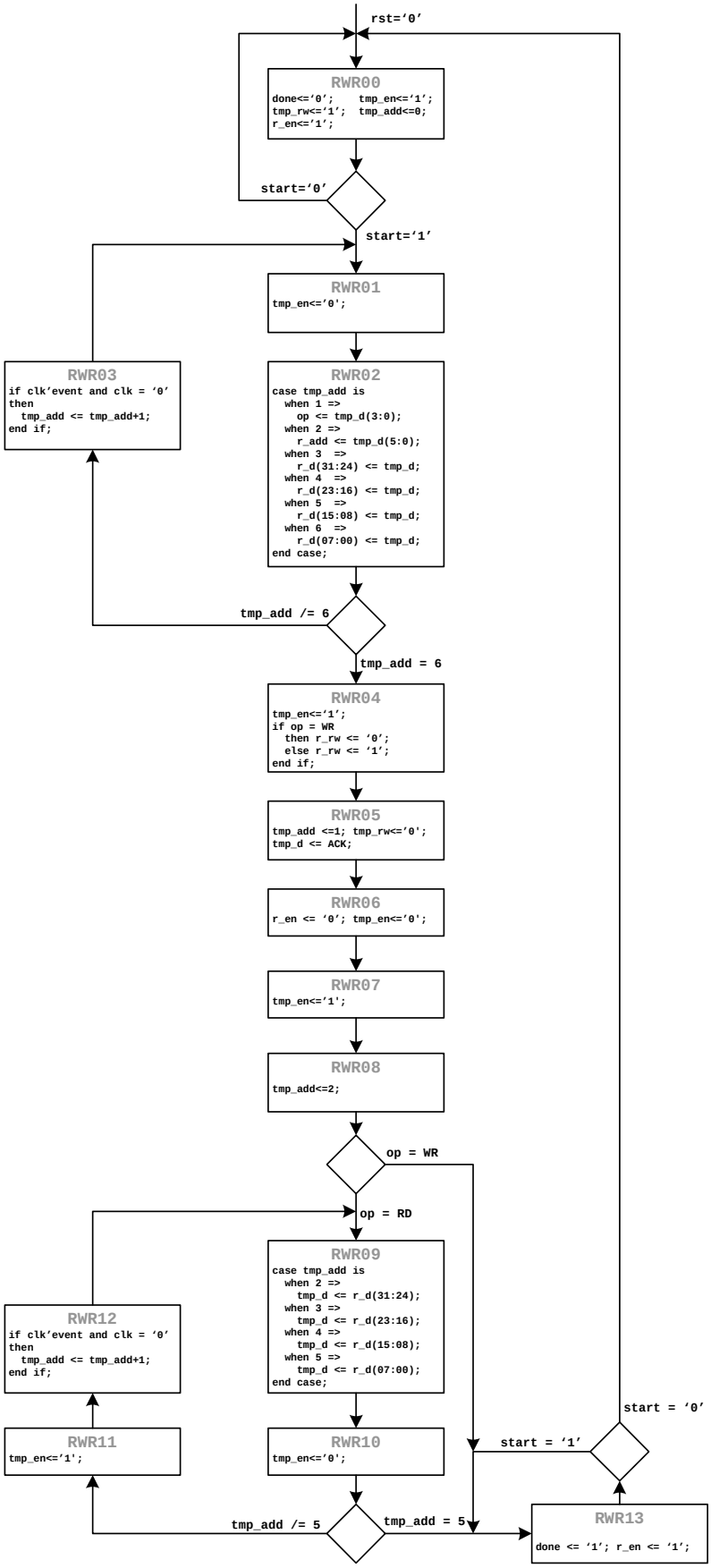


Figura 39 – Diagrama de estados de RWR

RWR Escreve e lê nos registos de controlo e tempos de transição.

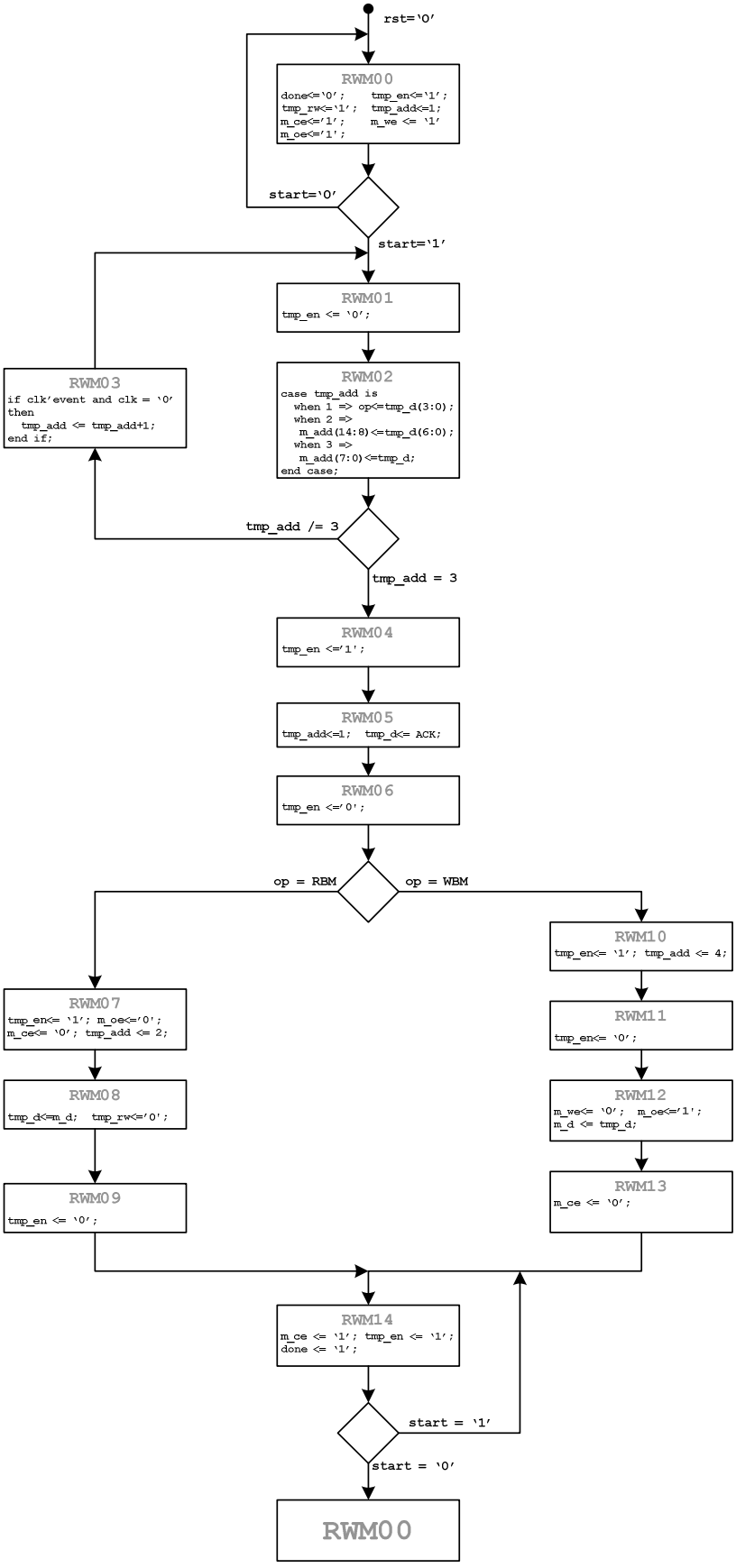


Figura 40 – Diagrama de estados de RWM

RWM controla o barramento da memória de massa para leitura e escrita.

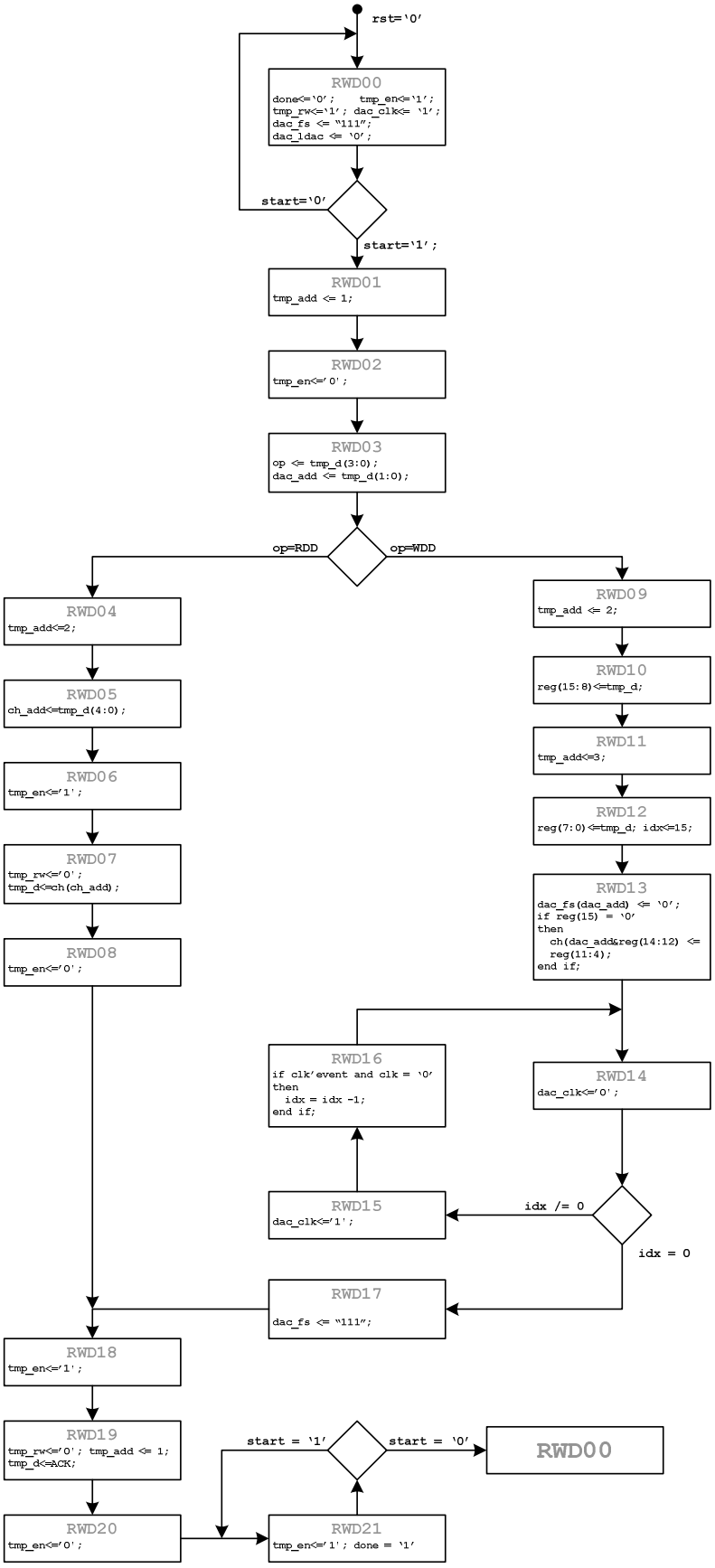


Figura 41 – diagrama de estados de RWD

RWD escreve nas DAC's o valor de tensão desejado e mantém um array com uma cópia desse valor para leitura.

A FSM RWC realiza os passos necessários para programar a memória de configuração. Como as operações são repetitivas decidimos criar a FSM AUX para realizar as operações elementares de escrita de um byte, leitura de um byte, envio de uma condição de início e envio de uma condição de paragem.

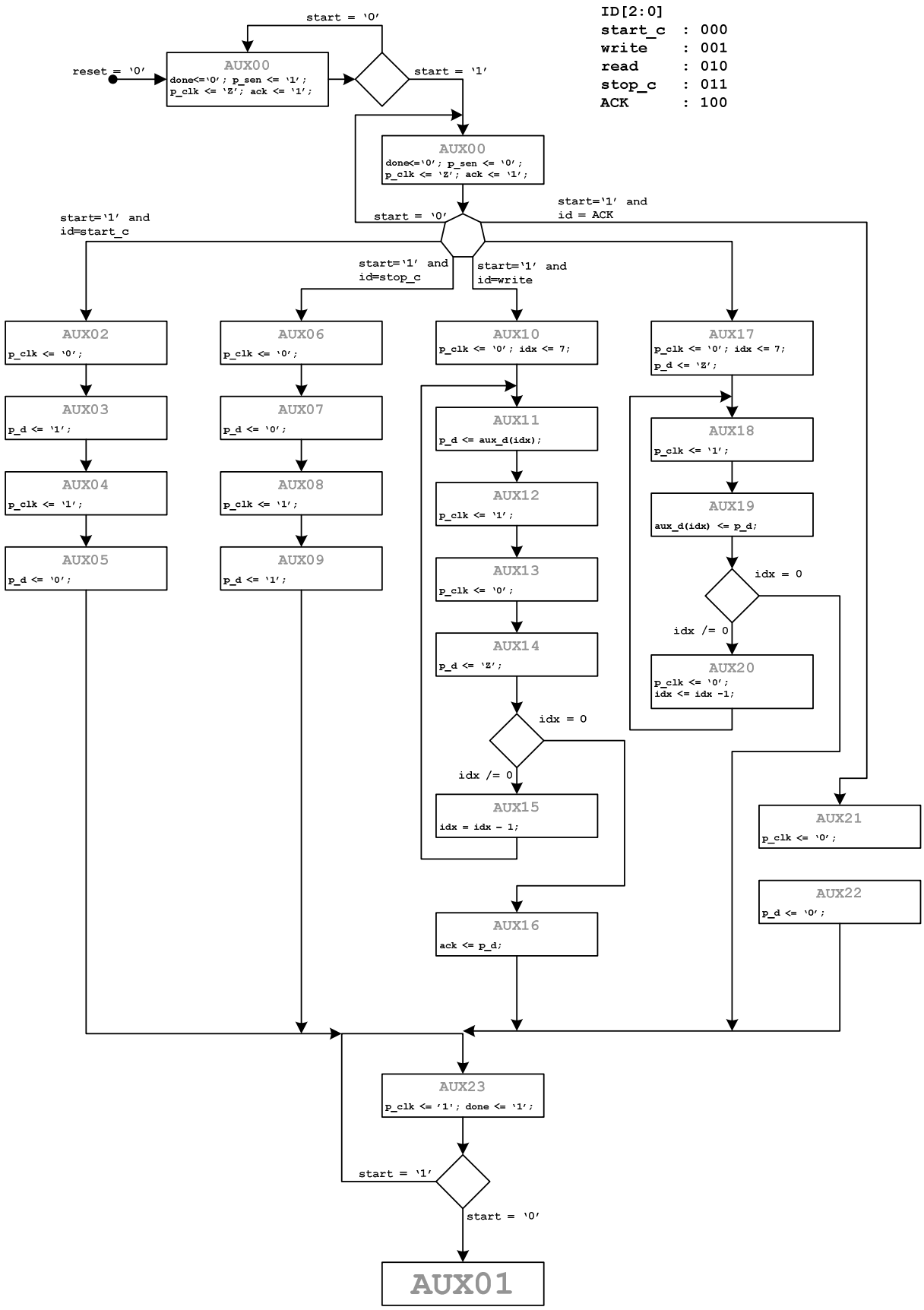


Figura 42 – Diagrama de estados de AUX

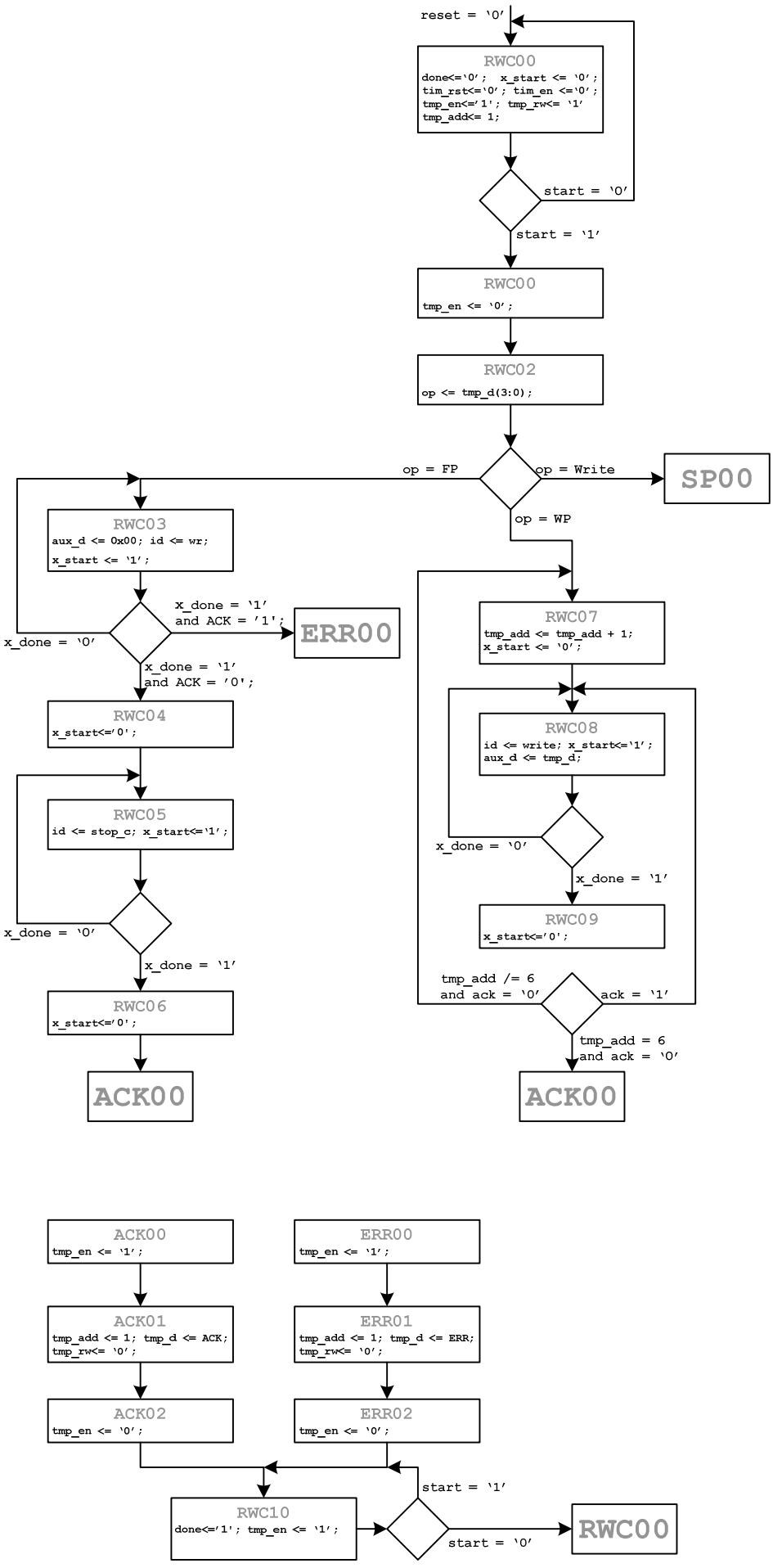


Figura 43 – Diagrama de estados de RWC, parte 1

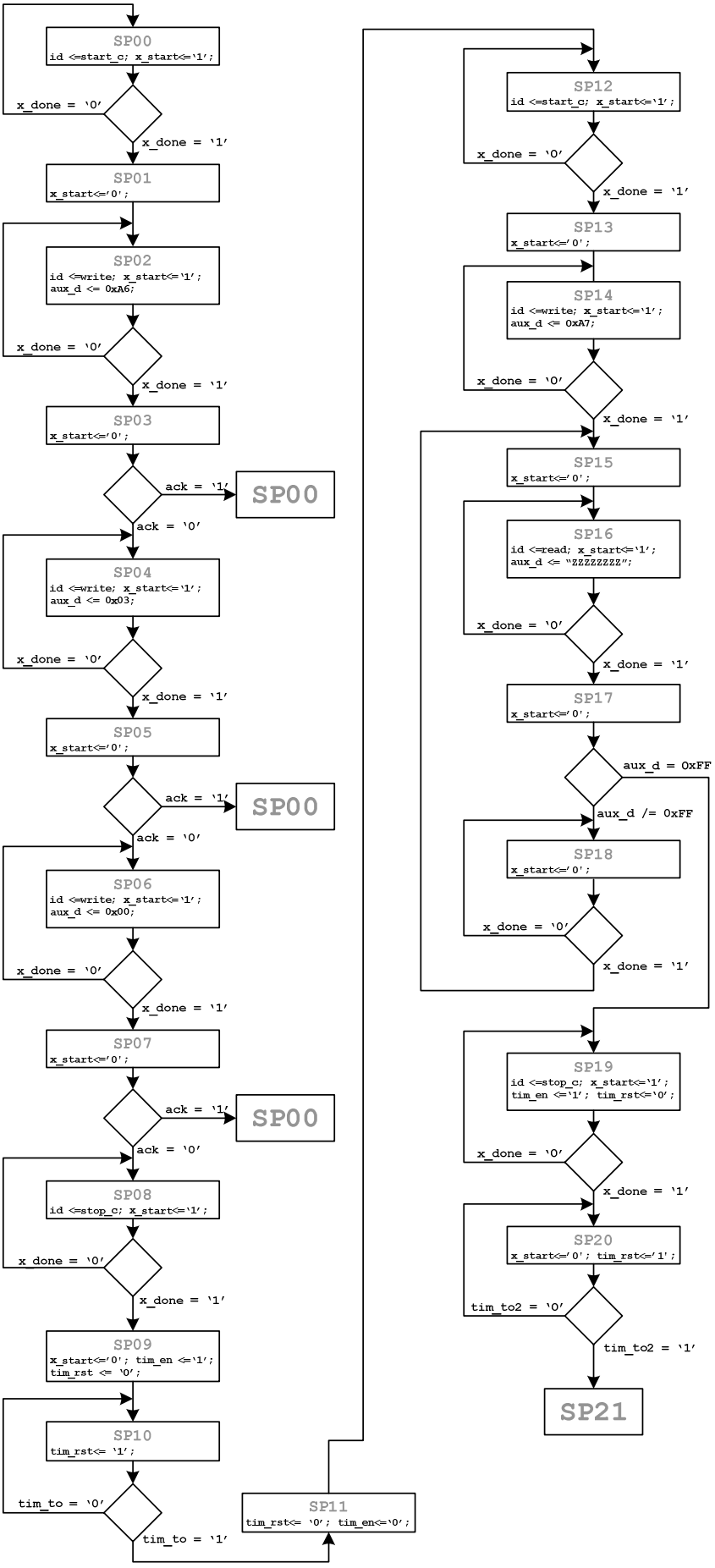


Figura 44 – Diagrama de estados de RWC, parte 2.

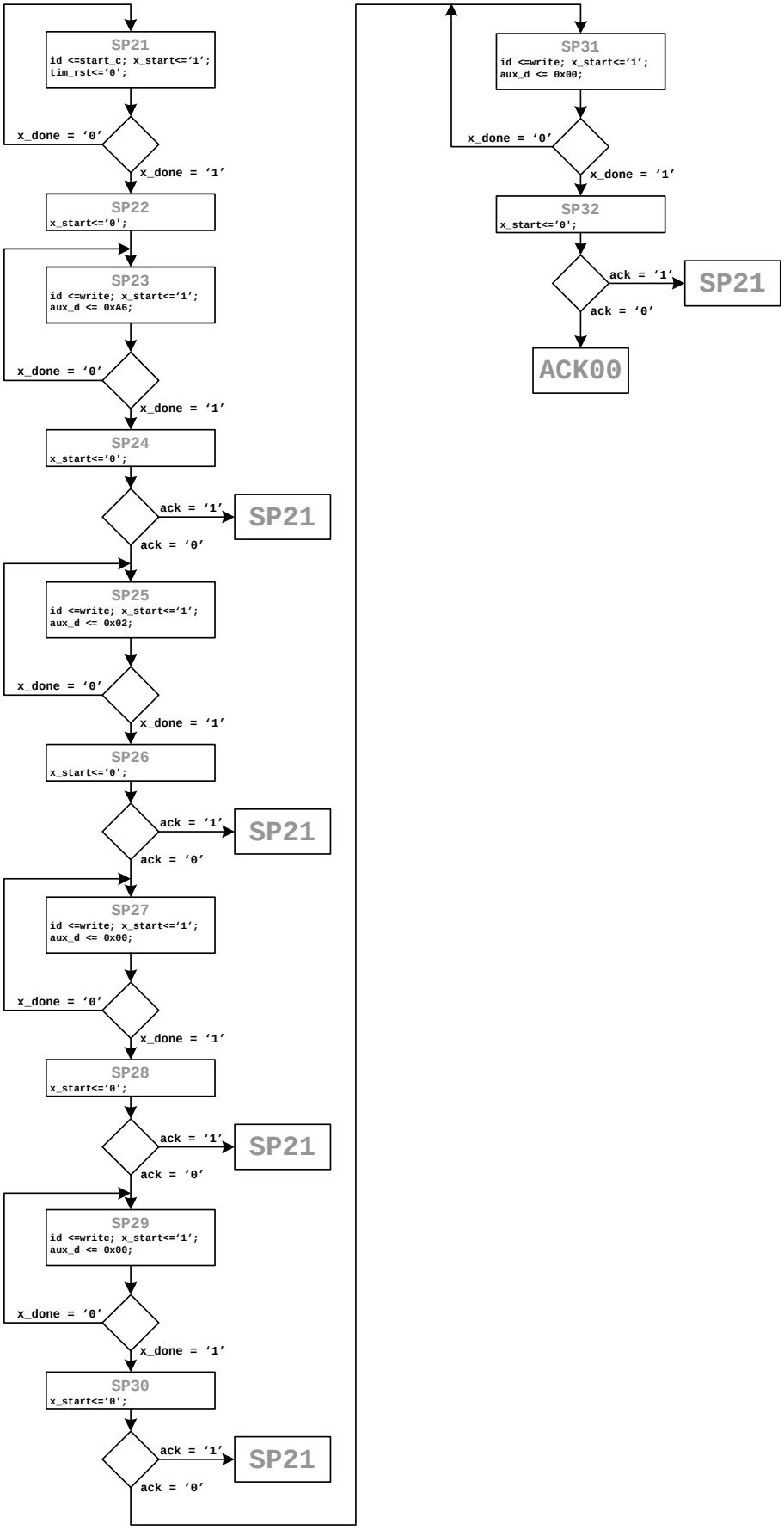


Figura 45 – Diagrama de estados de RWC, parte 3

RWC realiza a sequência das operações necessárias para programas a memória de configuração, passando o parâmetro a AUX, sendo AUX que realiza as operações elementares.

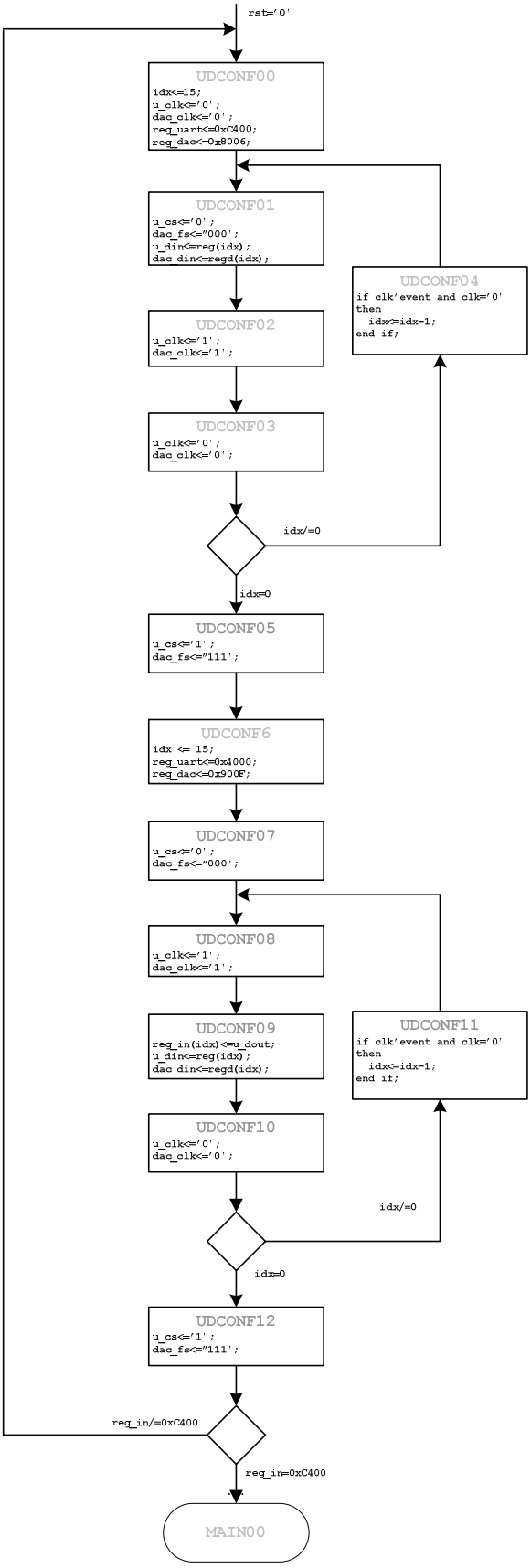


Figura 46 – Arranque de controladora

NOTA: Antes do arranque de todas as FSM's, é feita a configuração da UART e das DAC's, conforme especificado nos *datasheets* respectivos, através da FSM UDCONF. Quando a configuração estiver completa, MAIN é iniciada.

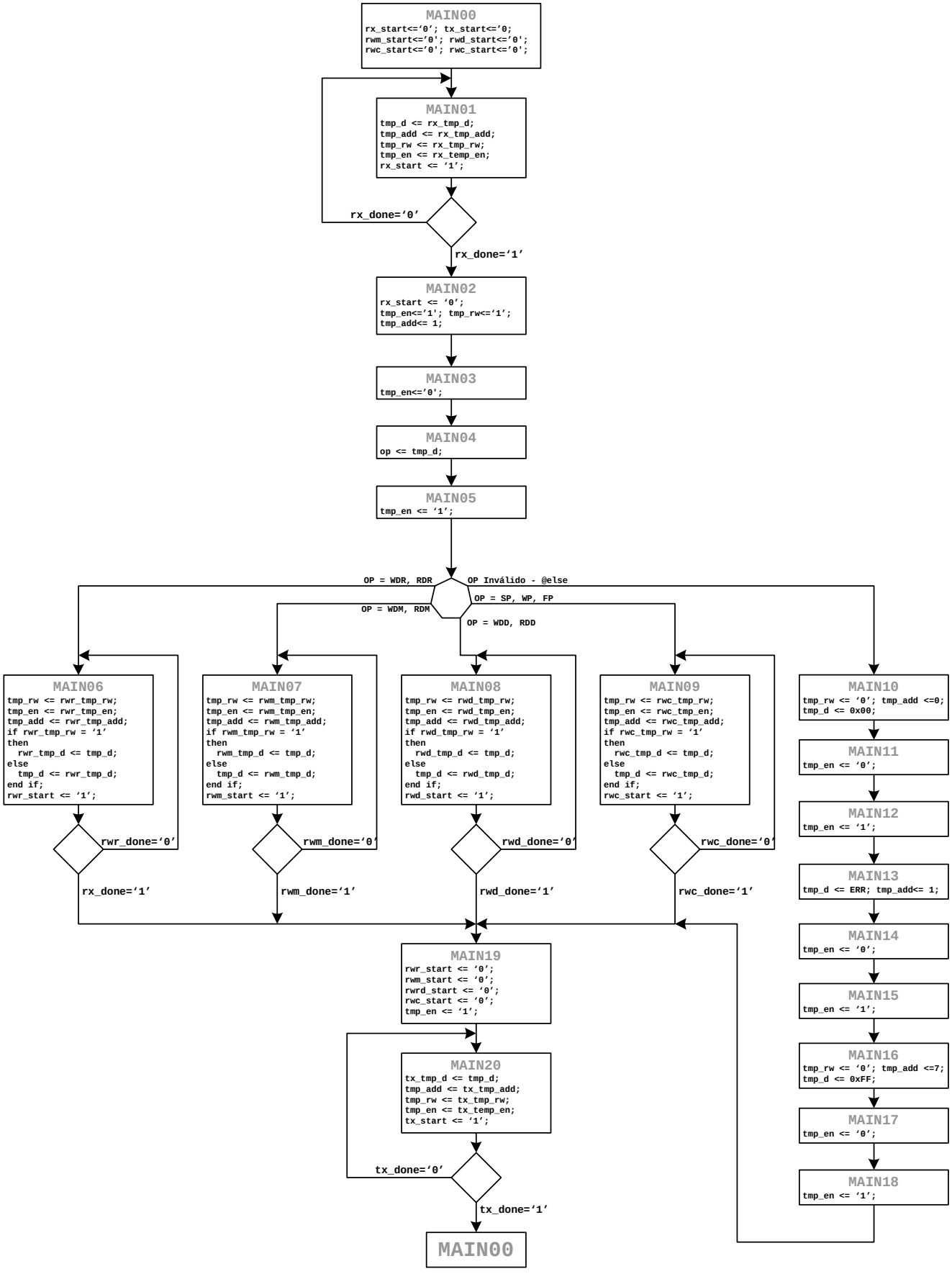


Figura 47 – Diagrama de estados de MAIN

MAIN liga a todas as outras FSM e controla-as dependendo da instrução recebida.

O esquema seguinte liga todas as FSM entre si, constituindo o bloco de nível mais alto na hierarquia do projecto das máquinas de estado. Da compilação deste esquema deve resultar o ficheiro de configuração de FPGA. Como não foi possível fazer o *debug* a tempo do código VHDL o esquema não se encontra isento de erros, pelo que não é possível ainda configurar a controladora.

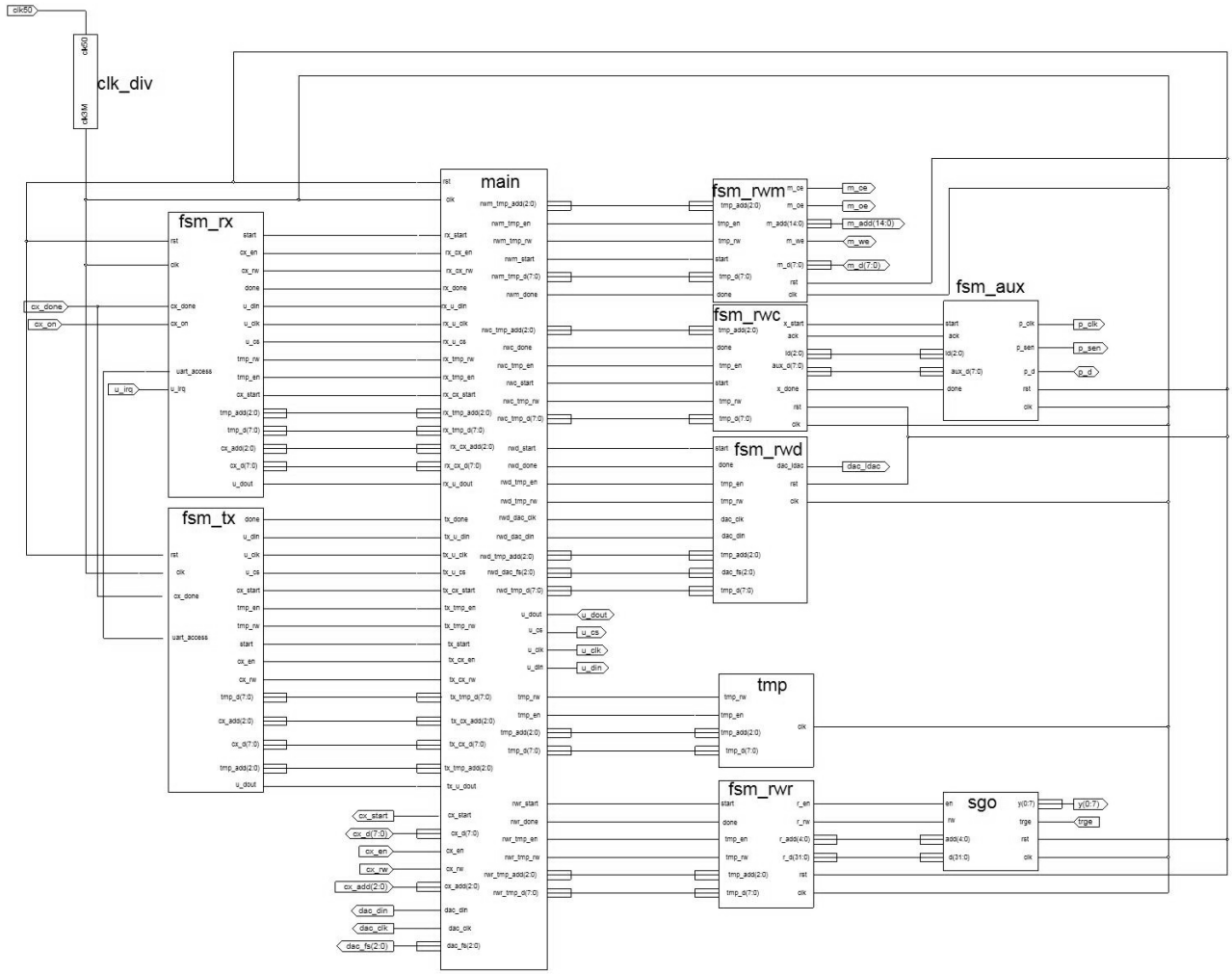


Figura 48 – Esquema de topo de nível de controladora

Anexo 2 – Esquema eléctrico

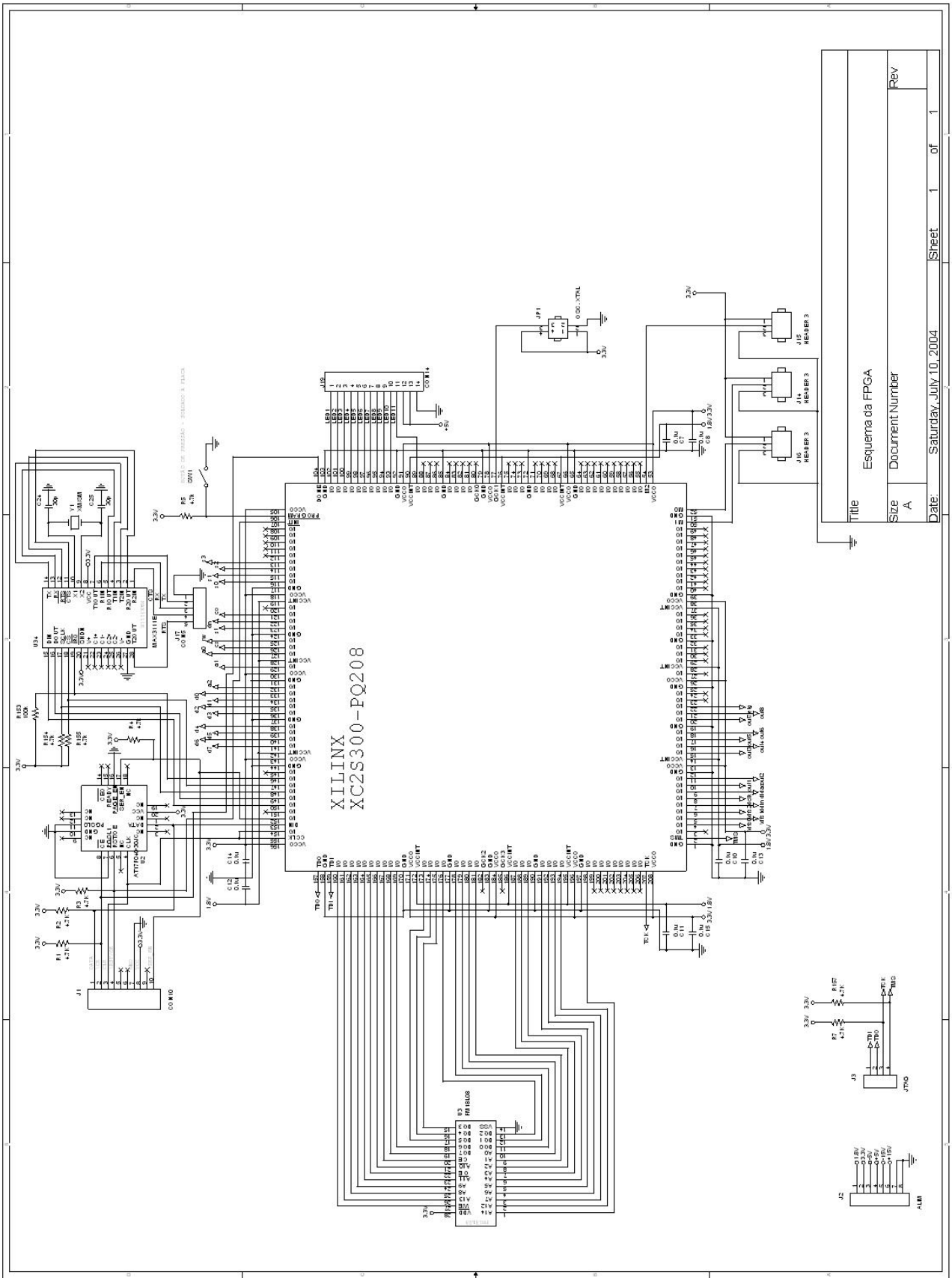


Figura 49 – Esquema das ligações da FPGA

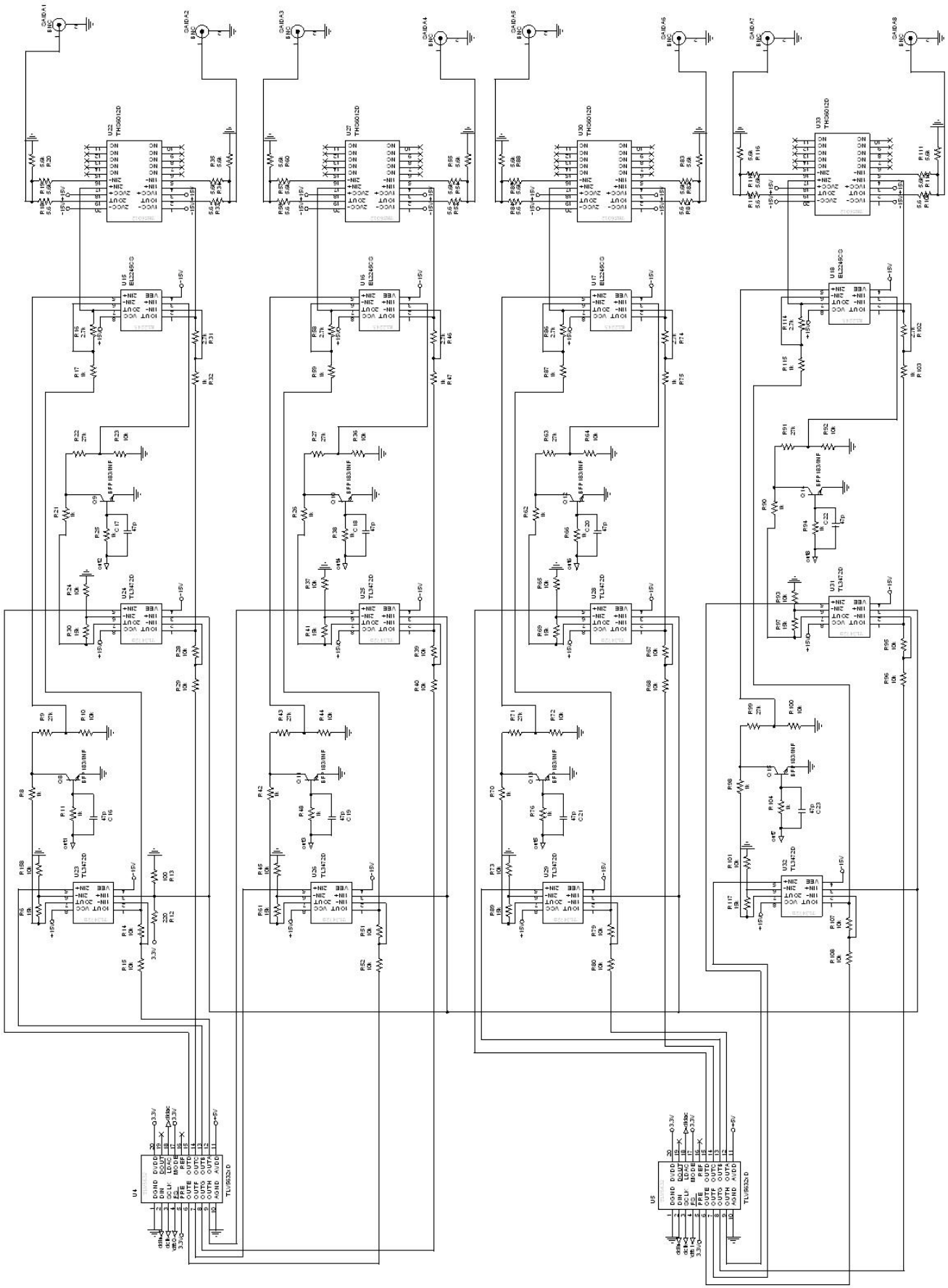


Figura 50 – Esquema dos andares de saída.

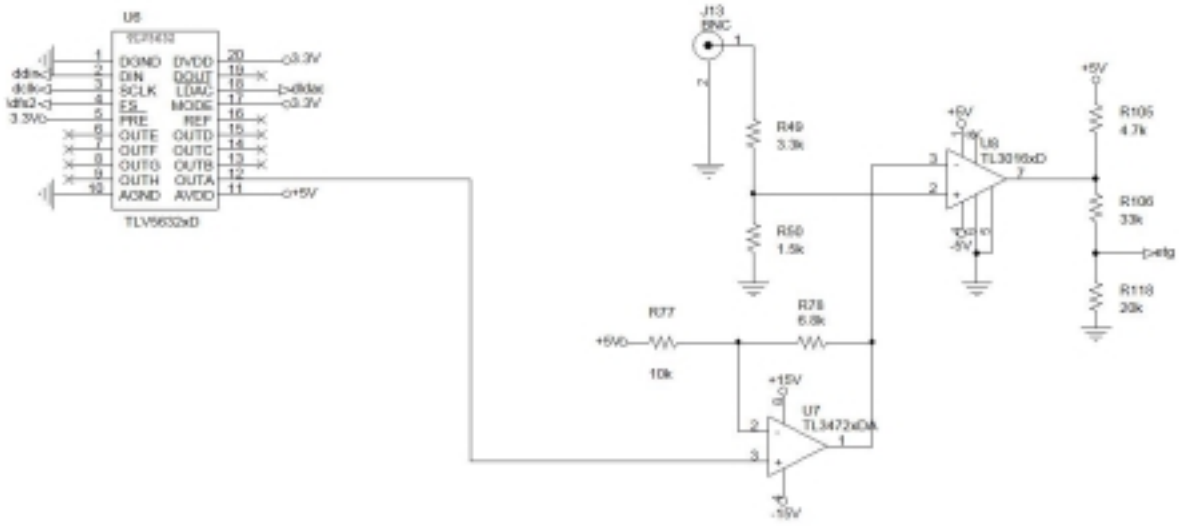


Figura 51 – Esquema do andar de entrada (*Trigger*).

Anexo 3 – *Software*

Desenvolvemos uma aplicação de linha de comandos para comunicar com a controladora e testar a funcionalidade da geração de ondas. A aplicação foi criada usando o *Visual Studio.NET 2003.*

Devido à sua simplicidade o programa é bastante fácil de utilizar recebendo todas as instruções directamente do teclado.

Para testar o programa foi necessário criar também um emulador da controladora, uma vez que o *debug* da controladora torna-se mais fácil se o software funcionar correctamente.

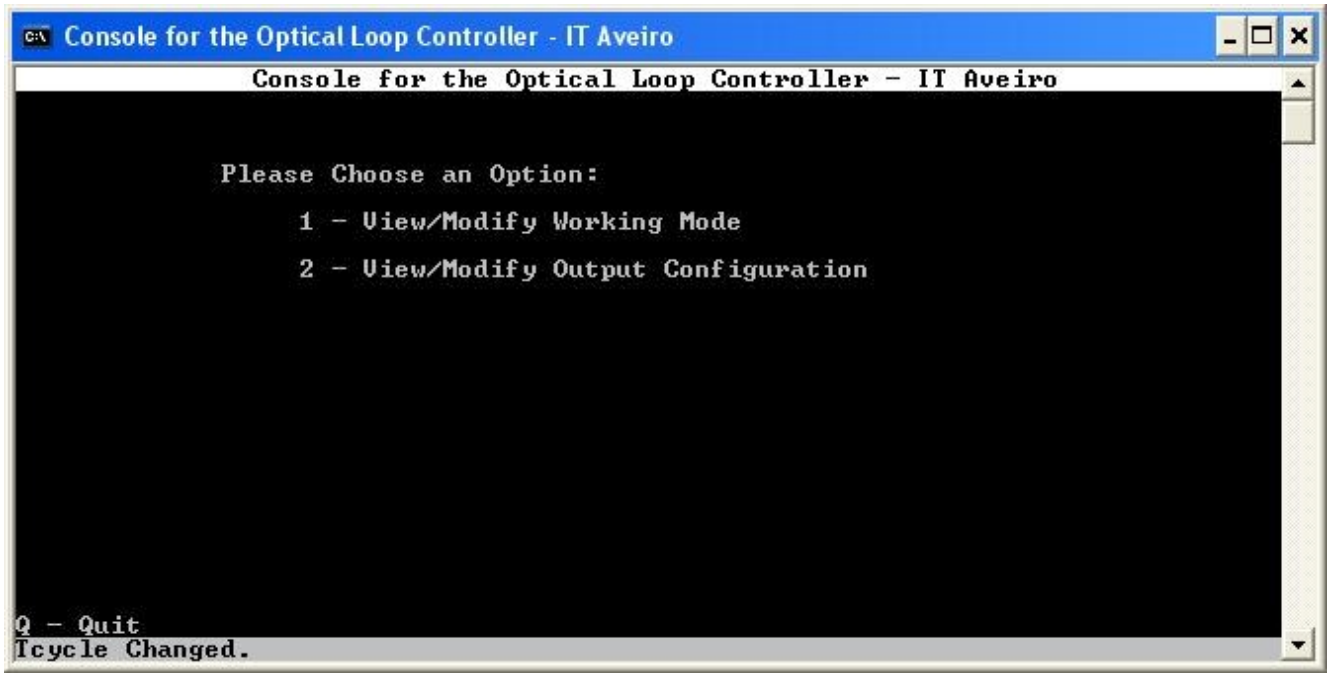


Figura 52 – Ecrã inicial

Figura 53 – Modo Master

No modo *master* podemos controlar o valor de $T_{cycle.}$

Figura 54 – Modo *Slave*

No modo *slave* temos controlo sobre o nível do trigger e o sentido da inclinação.

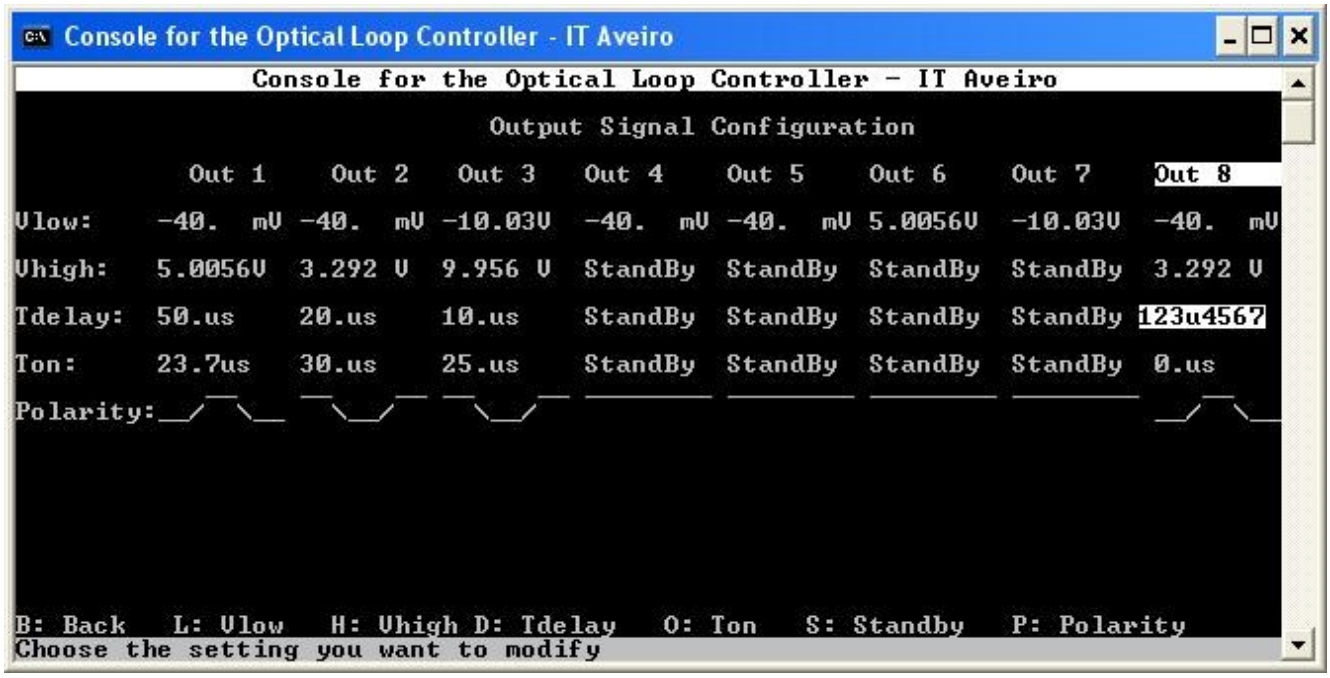


Figura 55 – Configuração das ondas de saída

Para além do '.' também podemos usar o 'm' e o 'u' como separador decimal para indicar grandezas mais pequenas.

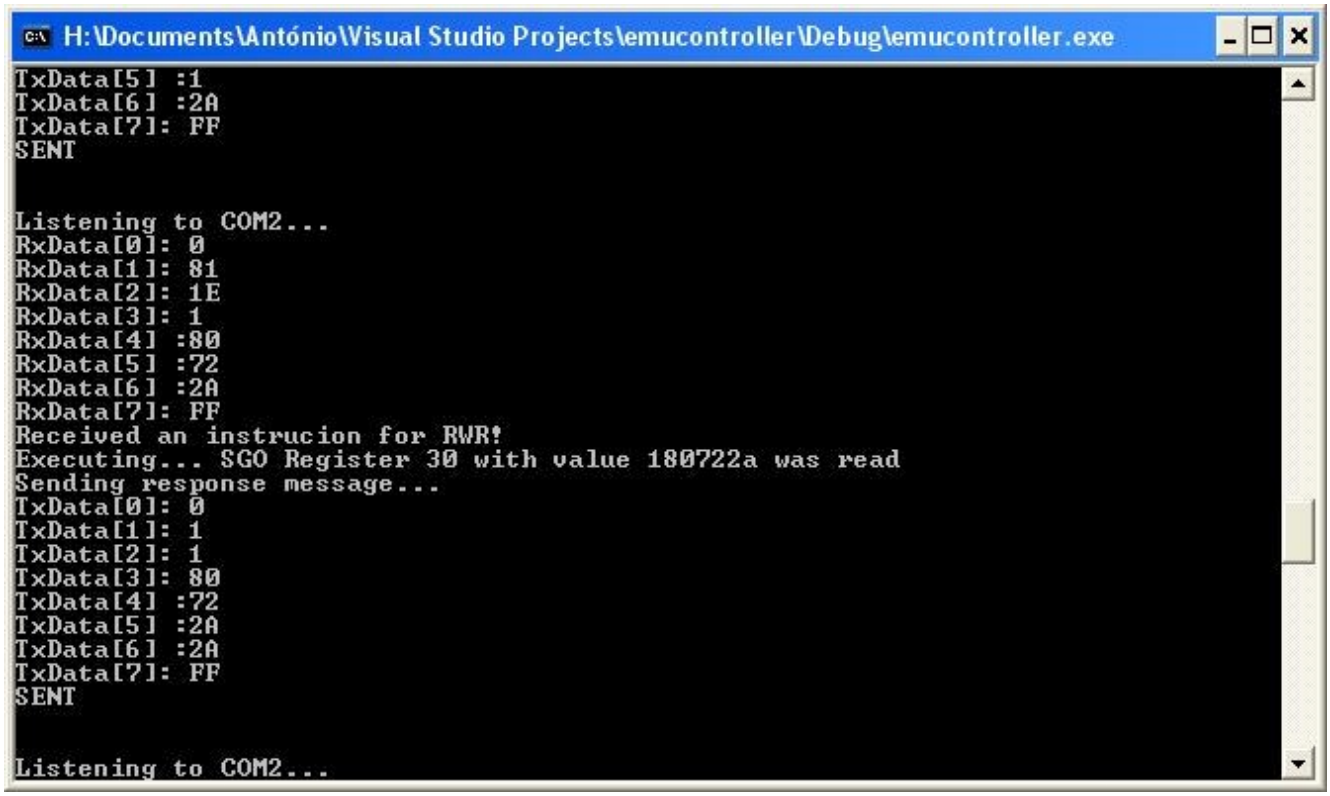


Figura 56 – Emulador da controladora

O emulador interpreta as instruções de acordo com o formato definido na especificação.

Para o software funcionar é necessário um cabo série adequado. A controladora está ligada como DTE, pelo que é preciso um cabo do tipo *Null-Modem* de 3 condutores.

Diagrama do cabo:

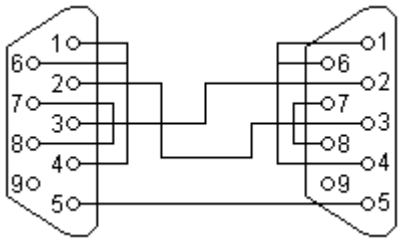


Figura 57 – Diagrama de ligações do cabo *Null-Modem*

Anexo 4 – Reparação da controladora antiga

Antes de começar a construção da controladora, reparámos a placa antiga que ainda tinha alguns problemas com a configuração.

O primeiro passo foi a construção de um programador para escrever na memória de configuração, uma ATMEL AT17LV256. O programador é fornecido também pela ATMEL, sob a forma de uma aplicação e tem a referência ATDH2225. Construímos o programador segundo o esquema fornecido pela ATMEL e com os componentes que o fabricante nos indicou após contacto via *email*. O programador funciona sem problemas com o software CPS da ATMEL.

Para programar a memória é primeiro preciso criar um ficheiro .mcs usando o software da Xilinx, por exemplo o iMPACT. Este ficheiro .mcs é depois lido pelo CPS da ATMEL e permita configurar a memória sem problemas. É necessário ter em atenção que o reset da memória deve ser configurado activo a '0', para ser compatível com a polaridade do reset da FPGA XCS20.

Após conseguir escrever e ler da memória de configuração ainda não conseguíamos configurar a FPGA a partir da memória. Tivemos então de verificar o circuito entre a FPGA e a memória para detectar eventuais anomalias.

Analisámos a placa antiga e deparámo-nos com alguns problemas na sua construção. Inicialmente algumas pistas estavam partidas ou descoladas, devido a estarem a suportar demasiada força provocada por *jumpers* e ligações de fios. As linhas que ligavam a memória aos pinos de configuração da FPGA eram demasiado longas e provocavam distorção do sinal, isto é, o relógio tinha muitos espúrios e isso fazia com que as transições de relógio fossem mal detectadas, o que corrompia os dados da configuração.

Para solucionar o problema tivemos de cortar alguns fios que estavam soldados à placa e adicionar capacidades de desacoplamento (10 pF) à linha do relógio de configuração. Deste modo as componentes de frequência mais alta são desviadas para a massa e o sinal chega menos distorcido à FPGA.

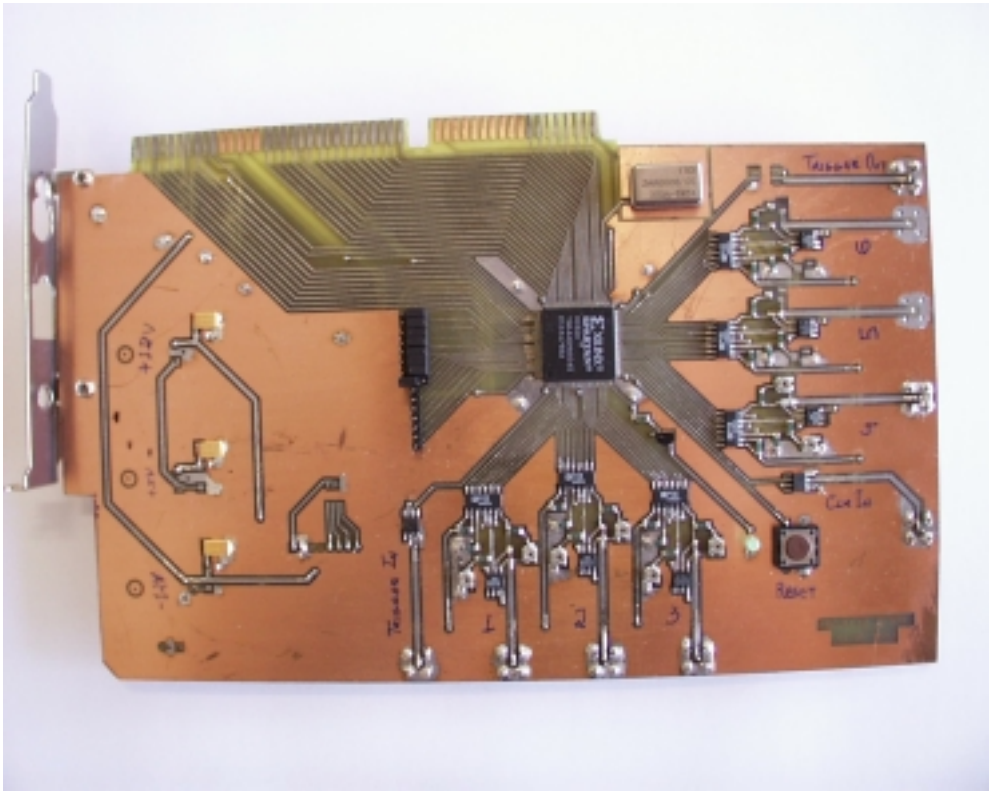


Figura 58 – Controladora antiga.