PLACA DE DESENVOLVIMENTO DE SISTEMAS ELETRÔNICOS BASEADOS EM ARM E FPGA COM INTERFACES ETHERNET E GSM

GESNER ORLANDI PASSOS JUNIOR

Trabalho de Conclusão de Curso apresentado à Escola de Engenharia de São Carlos, da Universidade de São Paulo.

Curso de Engenharia Elétrica com Ênfase em Eletrônica.

Orientador: Prof. Dr. José Roberto Boffino de Almeida Monteiro

São Carlos
2007

Dedicatória

Ao meu pai,
com quem aprendi
o que é realmente
essencial

Agradecimentos

A Deus pelo dom da vida.

Ao meu orientador, Prof. Dr. José Roberto Monteiro, por sua impressionante dedicação e atenção para ensinar e orientar, o qual me deu o exemplo de busca incansável por melhoria continua, em todos os trabalhos e projetos que assume.

Ao Pedro Zorzenon com quem aprendi o que é projetar *hardware*.

Aos meus colegas do LACEP e do curso por toda a ajuda e suporte que me deram, inclusive pela companhia nas noites em claro a terminar o projeto.

Às minhas irmãs, sempre dispostas a me auxiliarem nos mais diversos assuntos. Aos meus pais, meus verdadeiros conselheiros.

Aos meus irmãos, pela preocupação e carinho com a minha saúde.

Sumário

Lista de Figuras **ix**

Lista de Abreviaturas e Siglas **xi**

Resumo **xiii**

Abstract **xv**

I Introdução **1**

1 Introdução **3**

1.1 ARM: Principais características 5

1.1.1 Características Gerais do Processador 5

1.1.2 *System on Chip* 8

1.1.3 *Hardwares* especializados 9

1.2 FPGA: *HARDWARE* REPROGRAMÁVEL 9

1.3 Considerações Básicas de *layout* de PCB 10

1.3.1 Interferências Eletromagnéticas 10

1.3.2 *LAYOUT* da Placa 11

1.3.3 *Debug* de *Hardware* 12

1.4 Diagrama de Blocos do Projeto 13

II Desenvolvimento **17**

2 ARM 7 : Processador Central **19**

2.1 STR712FR2 : O ARM da STMicroeletronics 19

2.2 Ambiente de Desenvolvimento 20

2.3 O Esquemático do Circuito 22

2.3.1 Circuito de Relógio 22

2.3.2 Circuito de Alimentação 23
2.3.3 Interface JTAG 23
2.3.4 *Jumpers* 24
2.3.5 *WatchDog* 25
2.3.6 SPI 26
2.3.7 *Interface Serial-IN / Parallel-OUT*, Módulo de Controle 27
2.3.8 Interface Completa entre o ARM e os outros Periféricos 28

3 Modem GSM **31**
3.1 O circuito de Alimentação do Modem 33
3.2 Análise do Áudio do Modem 34
3.3 Interface de Comunicação com ARM: UART 36
3.4 Restrições e Características Gerais 37
3.5 Interface do Modem 38
3.6 Resumo das Recomendações 39

4 FPGA **41**
4.1 Ambiente de Desenvolvimento 43
4.2 Interfaces da FPGA 44
4.3 A Interface Externa 45
4.4 I^2C 47

5 Ethernet **51**

6 Outras Interfaces da Plataforma **57**
6.1 RS232 - UART 57
6.2 CAN - *Controller Area Network* 58
6.3 Circuito para o CAN 60

7 Implementações **63**
7.1 Integração com uma Placa de Aquisição de Dados 63
7.2 Controle e Acionamento de Motores 64
7.3 *CallBox* 65
7.4 Celular 66

III Conclusões **69**

8 Conclusões **71**

IV Apêndices **73**

A Esquemáticos da Placa **75**

B *Layout* da Placa **83**

V Referências **85**

Referências Bibliográficas **87**

Lista de Figuras

FIGURA 1.1 Formato das Instruções de Processamento de Dados do ARM. Extraído de [7] 7

FIGURA 1.2 Diagrama de Blocos do *Hardware* 14

FIGURA 2.1 Circuito temporizador do ARM 22

FIGURA 2.2 Circuito de Alimentação do ARM 23

FIGURA 2.3 Interface JTAG do ARM 24

FIGURA 2.4 Jumpers de Configuração do ARM 24

FIGURA 2.5 Circuito de Supervisão (*WatchDog*) 25

FIGURA 2.6 Temporização do Sinal SPI no modo 00 27

FIGURA 2.7 *Serial-IN/Parallel-OUT* e Decodificador 28

FIGURA 2.8 Mapeamento dos Pinos do ARM 29

FIGURA 3.1 Diagrama de Bloco da Organização Interna do Modem. Extraído de [12] 32

FIGURA 3.2 Circuito exemplo fornecido pelo fabricante do regulador chaveado. Extraído de [14] 34

FIGURA 3.3 Formato do Código PCM para transmissão áudio simples. Extraído de [12] 36

FIGURA 3.4 Diagrama da conexão de um DCE e um DTE segundo a norma RS-232. Extraído de [12] 37

FIGURA 3.5 Circuito abaixador de tensão 38

FIGURA 4.1 Arquitetura interna da FPGA 41

FIGURA 4.2 Diagrama Lógico de um CLB. Extraído de [28] 42

FIGURA 4.3 Conector de interface externa para a placa 46

FIGURA 4.4 *Driver* para proteção e separação entre sinais internos e interface externa 46

FIGURA 4.5 Formato das mensagens I^2C 48

FIGURA 5.1 Camadas de Rede definida no padrão Ethernet 51

FIGURA 5.2 Diagrama de Blocos do Controlador de Ethernet ENC28J60. Figura extraída de [9] 53

FIGURA 5.3 Ligações necessárias ao ENC28J60. Figura extraída de [9] . . 54

FIGURA 6.1 Diagrama esquemático do Optoacoplador. Extraído de [6] . . 60

FIGURA 6.2 Circuito para Acionamento do acoplador óptico 61

FIGURA 7.1 Diagrama de blocos do módulo de aquisição de dados 63

FIGURA 7.2 Diagrama esquemático da integração com a placa de aquisição de dados 64

FIGURA 7.3 Diagrama esquemático do controlador de motores trifásicos . . 65

FIGURA 7.4 Diagrama esquemático do *CallBox* 66

FIGURA 7.5 Diagrama esquemático do Celular 67

FIGURA A.1 Esquemático da núcleo central da Placa 76

FIGURA A.2 Circuito de Alimentação - Retifificadores 77

FIGURA A.3 Saídas e Circuito Supervisor 78

FIGURA A.4 Esquemático do Modem 79

FIGURA A.5 Esquemático da FPGA 80

FIGURA A.6 Esquemático demais periféricos 81

FIGURA B.1 PCB da placa 84

Lista de Abreviaturas e Siglas

ABS: *Anti-lock Breaking System*

AD: *analog to digital*

ADC: *AD convert*

ARM: *Advanced RISC Machines*

BSPI: *BUFFERED SPI*

CAN: *Controller Area Network*

CI: Circuito Integrado

CLB: *Cell Logic Block*

CLK: *clock*

CPLD: *Complex Programmable Logic Device*

CRC: *Cyclic Redundancy Check*

CS: *Chip-Select*

DMA:*Direct Memory Access*

DVD: *Digital Versatile Disc* EESC: Escola de Engenharia de São Carlos

FIFO: *First-in/First-out*

FIQ: *Fast Interrupt Request*

FPGA: *Field Programmable Gate Array*

GCK: *Global Input*

GPS: *Global Positioning System*

GSM: *Global System for Mobile Communications*

I^2C:*Inter-Integrated Circuit*

IO: *Inputs/Outputs*

IRQ: *Interrupt Request*

JTAG: *Join Test Action Group*

LACEP: *Laboratório de Controle e Eletrônica de Potência*

MAC: *Medium Access Control*

MISO:*Master Input - Slave Output*

MOSI:*Master Output - Slave Input*

PCB: *Print Circuit Board*

PCM: *Pulse Code Modulation*

RAM: *Randomic Access Memory*

RISC: *Reduced Instruction Set Computer*

RF : Onda Eletromagnética de Rádio Frequência (30MHz a 3GHz)

SPI: *Serial Peripheral Interface*

TCP/IP: *Transmission Control Protocol/Internet Protocol*

UDP/IP: *Universal Datagram Protocol/Internet Protocol*

VHDL: *Very-high Hardware Description Language*

WDG: *WatchDog*

Resumo

Passos Jr., G. *Placa de Desenvolvimento de Sistemas Eletrônicos Baseados em ARM e FPGA com interfaces Ethernet e GSM.* São Carlos, 2007. 100p. Monografia – Escola de Engenharia de São Carlos, Universidade de São Paulo.

Este documento descreve a plataforma de desenvolvimento baseado em microprocessador ARM e *hardware* reconfigurável, (FPGA). O contexto das tecnologias atuais é discutido e justificada a escolha do ARM para processamento central considerando-se a arquitetura interna do processador. Trata-se também de um trabalho de integração de tecnologias expoentes por reunir em uma única placa, além das duas opções de módulos de processamento (ARM, FPGA), interfaces de rede de importância global, são elas Ethernet, CAN e GSM. Finalmente, este é um projeto de *hardware* e por isso discute técnicas de descrição e *layout* de circuito bem como barramentos e interfaces de interligação entre componentes.

Palavras-chave: ARM, FPGA, HARDWARE, GSM, ETHERNET, PCB

Abstract

Passos Jr., G. *Development Board of System of Electronic Circuits Based on ARM and FPGA with interfaces Ethernet and GSM.* São Carlos, 2007. 100p. Monografia – Escola de Engenharia de São Carlos, Universidade de São Paulo.

This document describes an evaluation board based on ARM microprocessor and reconfigurable hardware, FPGA. It discuss the actual technologies and justifies the decision of choosing the ARM as the central processing through an explanation about its internal architecture. It is a work of integrating main technologies by joining together in a board two processing modules (ARM and FPGA) and also, some of the most important networks: Ethernet, CAN and GSM. Finally, it is a project of hardware, it present the ways of describing and implementing circuits with their connections and interfaces between components.

Keywords: ARM, FPGA, HARDWARE, GSM, ETHERNET, PCB

Parte I

Introdução

Capítulo 1

Introdução

É inquestionável que os sistemas baseados em computadores tiveram um enorme desenvolvimento nas últimas décadas. Neste nicho, tem grande destaque os sistemas embarcados. Por sistema embarcado entende-se aparelhos, objetos ou mesmo sistemas em geral que possuem capacidade de processamento, mas não são especificamente computadores. Suas aplicações são as mais variadas possíveis e sua área de atuação está em franca expansão. Exemplos destes sistemas podem ser encontrados em instrumentos médicos e de medidas, controle de robôs e motores, video-games, DVDs, diversos aparelhos eletrônicos e muitos outros.

A ARM Ltd. [2] afirma que metade dos carros produzidos em 2007 sairão de fábrica com aproximadamente 70 microprocessadores internos. Estes processadores estão nos *players*, no ar-condicionado, em monitoramento dos sensores anti-colisão, no referenciamento geoestacionário (GPS), nos displays internos, no controle dos freios (ABS), na ignição e outras diversas funções. Por fazer parte de um único sistema, estes dispositivos microprocessados não são independentes; antes é preciso que haja integração e intercomunicação entre os diversos dispositivos. Por este motivo, o que se observa atualmente é a tendência da migração dos sistemas embarcados dos processadores de 8 e 16 *bits* para os processadores de 32 *bits*. Com os microprocessadores de 32 *bits* é possível uma padronização de *cores* e periféricos que facilitam a integração e a compatibilização dos instrumentos. Além disso, 32 *bits* é o padrão entendido por todos os PCs, e também de vários barramentos. A tendência é marcante, até mesmo a Microchip, líder do mercado de microcontroladores de 8 *bits* passou a investir no desenvolvimento de microcontroladores de 32*bits* [13].

É neste contexto, que o ARM aparece como o principal expoente em processador para sistemas embarcados baseados em arquitetura de 32 *bits*. Fabricado pela VLSI Tecnology Inc., California, e apresentado em Abril de 1985, já na década de 90 apresentava a liderança no mercado de alta performance com baixo consumo e custo nas aplicações de *hardware* embarcado. Hoje é fabricado por mais de 10 empresas e tem o melhor suporte internacional dentro da categoria. O *site* da ARM Ltd. apresenta mais de 40 empresas de renome que estão associadas à corporação para desenvolvimentos de *softwares*, e aplicativos para o ARM. Neste mesmo *site*, a empresa firma o compromisso de disponibilizar "*every thing you need to create an innovative product design based on industry*" [2].

É seguindo esta tendência do mercado, e na expectativa de trazer o que há de novo tecnologicamente que se insere o presente projeto. Por se tratar de uma placa de desenvolvimento, com interfaces e aplicações em aberto, abre boas oportunidades para que os alunos e pesquisadores da Engenharia Elétrica de São Carlos, especificamente os do Laboratório de Controle e Eletrônica de Potência (LACEP) possam ser treinados, familiarizados e passem a desenvolver soluções de sistemas baseados em ARM que os colocarão em sintonia com as exigências do novo mercado de sistemas embarcados e proporcionará ao laboratório visibilidade e referência para ser consultor nesta tecnologia para as empresas da região de São Carlos, onde a absorção do ARM é ainda incipiente. Além disso, permite a sua inserção do LACEP no contexto de pesquisa e desenvolvimento global de sistemas baseados em ARM.

Esta monografia apresenta todo o projeto do *hardware* implementado e também é o manual da placa. Aqui são definidos os principais periféricos, os barramentos, as interconexões, as restrições e as normas que devem ser observadas para desenvolvimento de aplicações para a plataforma.

Este capítulo introdutório está organizado de forma a fornecer um entendimento geral das tecnologias envolvidas na plataforma, mas também algumas considerações a respeito de habilidades e conhecimentos requeridos no desenvolvimento de *hardware*. Assim, a próxima seção apresenta o ARM e algumas de suas características que lhe possibilitaram o destaque obtido. Além do ARM, foi incorporado também uma FPGA no circuito da placa. Suas características e o motivo de fazer parte do projeto são explicados na Seção 1.2.

Por se tratar de um projeto de desenvolvimento de *hardware* vários conceitos de eletrônica e *design* de placas de circuito impresso (PCB) são importantes. Este tema é apresentado e discutido na Seção 1.3. Ao final dessa seção, discute-se sucintamente o assunto de identificação de falhas e mal funcionamento de dispositivos, apresentando uma estratégia de abordagem destes tipos de problemas. Após estas considerações iniciais, a placa com seus principais blocos funcionais é apresentada.

1.1 ARM: Principais características

A primeira definição importante no que se refere ao ARM é que não se trata de um microcontrolador produzido por uma empresa. Na verdade, ele é o *core*, o núcleo funcional de um microprocessador, que define os registros internos, o conjunto de instruções, algumas unidades funcionais e sua arquitetura. A ARM Ltd., responsável pelo ARM, licencia este *core* para fabricantes que montam seus microcontroladores com os periféricos que julgam mais importante para atender ao mercado. Os periféricos e as funcionalidades específicas do ARM escolhido é o assunto do capítulo 2. Esta seção apresenta apenas as características deste núcleo central (*core*) do ARM.

1.1.1 Características Gerais do Processador

O ARM é um processador RISC (*Reduced Instruction Set Computer*). A outra possibilidade para processadores é CISC (*Complex Instruction Set Computer*). A estratégia das arquiteturas CISC é permitir que programas fiquem menores, com alta densidade de código, pois uma única instrução é capaz de realizar funcionalidades de alto grau de complexidade. Porém, para poder realizar estas instruções mais complexas, o *core* dos processadores CISC são maiores e mais complexos. Por outro lado, os processadores RISC buscam um desempenho ligado à capacidade de execução de mais instruções por segundo, definindo instruções mais simples. O preço que estes processadores pagam é a necessidade de maior capacidade de armazenamento para os programas. No que diz respeito ao desempenho, as arquiteturas RISC têm se mostrado mais eficientes executando rotinas completas mais rapidamente.

O ARM não apresenta uma arquitetura RISC simples. Na verdade, ele combina a simplicidade do *hardware* baseado em idéias "RISC" e agrega algumas características

essenciais do desempenho "CISC", conseguindo assim melhor densidade de código que uma arquitetura RISC pura e maior velocidade e desempenho que uma arquitetura "CISC". Isso proporciona ao ARM alto desempenho com um *core* reduzido e grande eficiência no consumo de energia [7].

É preciso conhecer um pouco da estrutura interna do processador para compreender sua capacidade. Seu processamento é feito através de *pipeline*. O ARM7 utilizado no projeto apresenta 3 estágios (*Fetch, Decode, Execute*) para carregar, decodificar e executar a instrução. A técnica de *pipeline* permite ao processador operar em maiores frequências, pois o processamento necessário em cada estágio é reduzido, e também aumenta o *throughput*, isto é, a quantidade de instrução finalizada num período. Além disso, por ser RISC, trata-se de uma estrutura *load-store*, o que significa que todas as operações lógicas e aritméticas não são realizados diretamente com a memória, antes, apenas seus 16 registros internos participam destas operações. O conjunto de instrução do ARM 7 pode ser dividido em 3 grupos:

- Processamento de Dados:

  Operações aritméticas e lógicas que podem ser realizadas sobre os registros.

- Transferência de Dados:

  Operações de transferência de valores entre registros e memória e entre memória e registros.

- Controle de Fluxo:

  Operações de saltos e chamadas de funções condicionadas ao *status* do sistema.

A figura 1.1 apresenta a estrutura de um registro de instrução do grupo de processamento de dados. Uma característica interessante é o fato de todas as operações poder ser condicionadas (primeiros 4 *bits*). O condicionamento é baseado no valor de algumas variáveis de *status* do processador. Além disso, é possível determinar se uma determinada operação deve ou não alterar o valor destas variáveis de *status* (*bit* S). Há um *bit* que identifica se a operação é somente sobre registros ou também sobre valores imediatos, fornecidos junto com a instrução. Segue-se o *opcode* da instrução, o destino e o primeiro operando. Esta também é uma característica interessante, cada instrução no processamento de dados pode envolver até 3 registros diversos.

Se a instrução operar sobre um valor imediato, este não pode ter qualquer valor de 32 *bits*. Como se pode observar, há apenas 12 *bits* para escrevê-lo. Os valores possíveis para valores imediatos nestas operações é: $(0 \rightarrow 250) \times 2^{2n}$. Se a operação for realizada sobre seus registros, há a possibilidade de se determinar um *shift* lógico dos dados numa mesma instrução.

Alguns exemplos ilustrativos são mostrados para auxiliar a compreensão do potencial destas instruções. Por exemplo, a multiplicação de um registro por 5 pode ser feito numa única instrução da seguinte maneira:

ADD r0, r0,r0, LSL #2 ; r0=r0+2r0

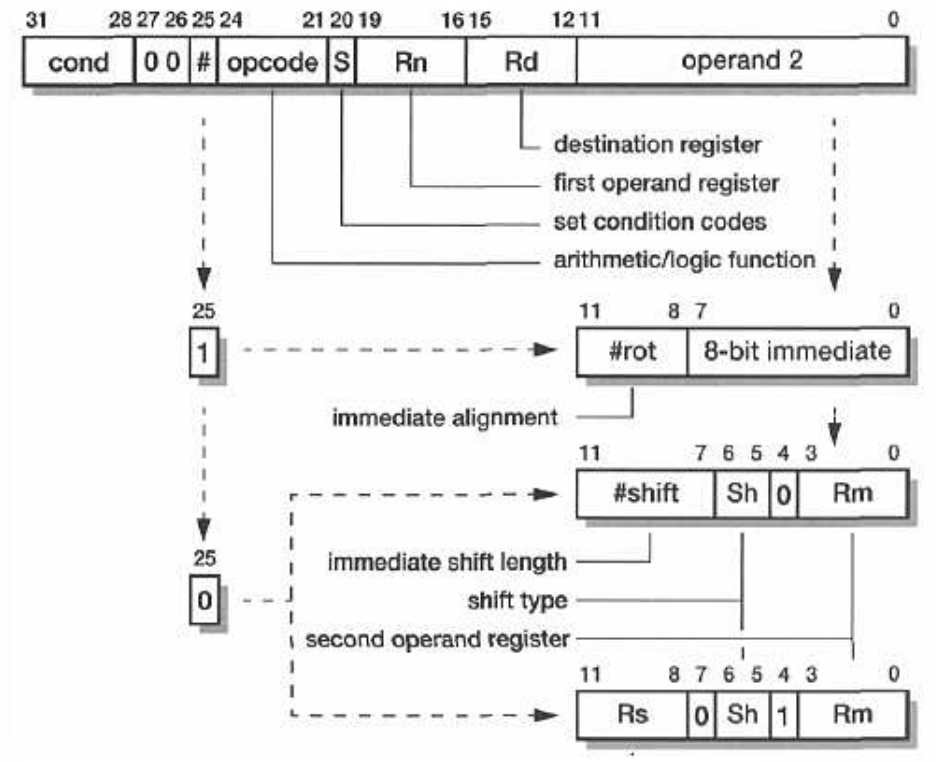

Figura 1.1: Formato das Instruções de Processamento de Dados do ARM. Extraído de [7]

A capacidade de fazer com que cada instrução seja condicionada, também permite grande redução no volume de código. Por exemplo, considere a instrução $if(a == b)\&\&(c = d))e++$, ela poderia ser escrita assim:

```
CMP r0,r1
CMPEQ r2,r1 ;      EQ diz para executar função apenas se a última foi igual
ADDEQ r4,r4,#1   ;
```

Há ainda instruções ligadas à multiplicação e também instruções do tipo MAC, multiplicar e acumular, muito úteis aos algoritmos de processamento digital de sinais. Além disso, é possível realizar operação de multiplicação com dois registros de 32 *bits* e definir posicionamento para guardar o resultado de 64 *bits* em dois registros, não sendo portanto necessário seccionar o resultado.

1.1.2 *System on Chip*

Há algumas características do processador que lhe permite ter um sistema operacional seguro no seu interior. Ele define dois modos de operação, o modo normal de usuário e um modo privilegiado. A instrução SWI (*software interrupt*) é um exemplo disso. Através desta instrução o *hardware* é mudado para o modo supervisor e passa a executar as funções a partir do endereço 0x08. Como em modo de usuário não é possível alterar os registros especiais ou as entradas e saídas do microcontrolador, isto garante a segurança do sistema, impondo que áreas protegidas só possam ser acessadas por chamadas ao sistema.

Outra característica importante é a instrução de SWAP. Esta instrução troca os valores entre uma posição de memória e um registro. Ela é essencial para que um sistema possa definir semáforos e controles de exclusão mútua que possibilitam a programação concorrente, requisito fundamental de sistemas operacionais.

Outra instrução útil, embora não seja essencial, é a capacidade que o ARM possui de transferência de blocos completos de registradores para a memória e também o contrário. Através dela , é possível ao processador implementar mais facilmente um *task-manager*, que é o responsável por partilhar os recursos de processador entre os processos ativos numa máquina. Pois, chamadas às interrupções podem guardar, com apenas esta instrução, todo o *status* do sistema quando a interrupção ocorreu, através da colocação do valor dos registros numa posição de memória.

Essas são algumas das características que tornam o desenvolvimento de sistema operacional para ARM atraente. Os *sites*

[3] e [4] mostram vários sistemas operacionais portáveis para o ARM, bem como uma lista de 28 desenvolvedores destes sistemas que são parceiros no desenvolvimento do *core* do ARM.

1.1.3 *Hardwares* especializados

O núcleo do ARM define ainda uma interface JTAG que é um protocolo serial desenvolvido inicialmente para teste de condutividade de circuitos, mas também muito útil para programação e também *debug* dos microprocessadores. Além disso, há também uma máquina de estado que implementa um DEBUG sobre a interface JTAG. O *Debug* especializado permite definições de *breakpoints*, *watchpoints* e mudanças de estado definidas pelo usuário no processo de depuração do programa ou *software* da máquina.

1.2 FPGA: *HARDWARE* REPROGRAMÁVEL

Além do ARM, está presente também neste projeto a FPGA XCS05XL, a SPARTAN da Xilinx. Por ser também uma unidade de processamento, pode ser configurada para auxiliar o ARM em aplicações. A grande característica das FPGAs é a reprogramabilidade do seu *hardware*. Isso significa, que elas podem ser modificadas, em função das aplicações para funcionarem como *driver* especializados de periféricos.

A motivação inicial para acrescentar ao projeto a FPGA diz respeito ao interfaceamento entre um modem GSM e o processador ARM para transferência de áudio digital. A transferência deste áudio digital, PCM, é através de um protocolo síncrono definido por quatro fios. Não há no ARM um *driver* especializado que decodifique este protocolo. Assim, colocar no circuito apenas o modem e o ARM exigiria do *software* implementar um algoritmo para esta transferência de dados. Com a inclusão da FPGA, pode-se descrever um *hardware* que transfere dados do barramento PCM para o barramento SPI, para o qual há *driver* especializado no ARM. Desta forma, a comunicação com o modem ocorre com procedimento de *software* muito mais simples e o processador do ARM fica livre para executar outras operações.

Além deste benefício considerado, a inclusão da FPGA torna muito vasta a faixa de aplicações possíveis para a plataforma desenvolvida e algumas possibilidades são apresentadas no capítulo 7.

1.3 Considerações Básicas de *layout* de PCB

Esta seção trata do aspecto relacionado ao desenvolvimento de *hardware* propriamente dito. Considerações sobre circuitos eletrônicos, requisitos de sistemas, fenômenos elétricos ligado a uma plataforma eletrônica, ruídos, capacitâncias parasitas, e outras questões são abordadas. O que se apresenta aqui é de caráter geral e diz respeito ao procedimento adotado para o desenvolvimento da placa. No final desta seção há um espaço dedicado ao tema de identificação de falhas e estratégias de abordagem do problema na operação com sistemas embarcados.

A primeira observação que se faz, é que em projeto de *hardware*, é possível, e até mesmo comum a ocorrência de mal funcionamento do circuito, embora este possa aparentemente apresentar um projeto de circuito perfeito. Muitas vezes, o responsável por ocorrências deste tipo é o ruído eletromagnético, que é o tema que segue.

1.3.1 Interferências Eletromagnéticas

Catsoulis em [5] apresenta algumas causas para o problema do ruído bem como dicas úteis na disposição, roteamento e confecção de placas de circuito impresso para amenizar as interferências. Primeiramente, têm-se 2 tipos de ruídos presentes em qualquer circuito eletrônico, os causados externamente e que interferem no circuito e os que são gerados internamente pelo funcionamento do próprio sistema. No primeiro grupo encontram-se os reatores, chaves e relés, transmissores de RF, indutâncias mútuas parasitas nos fios da alimentação, descargas elétricas e os motores. Ruídos internos ocorrem em geral por conta de fontes de tensões inadequadas ou mal projetadas, capacitores de desacoplamento insuficientes, capacitâncias parasitas entre trilhas paralelas, chaveamento em altas frequências além do ruído randômico térmico intrínseco a todos os processos.

Muitos destes efeitos, como por exemplo a indução magnética e as capacitâncias parasitas, são tanto mais prejudiciais quanto maior for o percurso que os sinais que os geram percorrem a placa. Por isso, uma estratégia aconselhável no *design* de circuitos impressos é a disposição de malhas de terra onde for possível. Estas malhas, que são o caminho de retorno de todos os sinais na placa diminuem as correntes de *loops* no circuito e consequentemente o ruído que estas correntes geram.

Um problema surge com as linhas de alimentação. Não é possível minimizar o percurso destas uma vez que elas tem que chegar a todos os CIs, para energizar todos os pontos em que há necessidade. Para contornar esta limitação, sugere-se que para cada CI na placa, sejam colocados capacitores de desacoplamento o mais próximo possível da alimentação. Estes capacitores, além de garantir um caminho para o terra de baixa impedância para o ruído, ainda ajudam a fornecer as correntes necessárias para vencer as capacitâncias internas e de junção dos CIs nas mudanças de estado lógico dos mesmos. Outra prática sugerida é a utilização de reguladores na própria placa que permitem assim a isolação dos circuitos internos de ruídos das fontes externas.

Há também um outro efeito que pode causar estrago relacionado às descargas eletrostáticas. Um ambiente seco, carpete, sapatos e outros isolantes podem facilmente fazer com que haja um potencial eletrostático num individuo de até centenas de Volts. Esta tensão tem baixíssima capacidade de corrente, porém pode causar dano irreversível aos CIs e outros componentes eletrônicos. Principalmente as arquiteturas baseadas na tecnologia CMOS, que apresentam um película de óxido isolante no *gate* que é facilmente rompida em virtude destas descargas e assim destrói a característica de amplificação do componente.

Uma alternativa para este problema é a introdução de CIs de *buffers* que isolem os componentes críticos do circuito. No projeto, pretendia-se ligar 6 pinos mais uma SPI do ARM ao barramento externo, com o objetivo de expansões e algumas aplicações. Por motivo de proteção, foi colocado entre eles o *buffer*, 74HC541 [15]. Além da maior resistência do *buffer* às descargas eletrostáticas, mesmo que uma linha sua queime serve como "fusível" que protege os pinos do ARM.

1.3.2 *LAYOUT* da Placa

Em circuitos integrados, principalmente os que operam em altas frequências, há efeitos que precisam ser considerados. As linhas que ligam o circuito precisam ser entendidas como linhas de transmissão sujeitas a ocorrência de reflexões e outros efeitos prejudiciais. Por este motivo, o *layout* da placa precisa de uma atenção especial. Por este motivo, apesar de existir *softwares* de roteamento automático de trilhas para circuito impresso, capazes de fornecer grande redução no tempo de projeto de placa, muitos desenvolvedores de *hardware* aconselham que o roteamento seja feito manual-

mente com o intuito de garantir melhor desempenho e menor interferência das ligações [5].

As práticas aconselháveis consistem na indicação de atenção especial aos sinais de maior frequência, em especial os sinais de *clock*. Sugere-se que o caminho do sinal de *clock* ao processador seja o menor possível, e ainda que uma malha de terra envolva o mesmo. Outra questão importante é evitar curvas de 90º nas trilhas. Curvas acentuadas, fazem com que os sinais nas linhas "enxerguem" uma terminação de linha e pode ocorrer por isso a reflexão. Por fim, é aconselhável isolação de terras de circuito que podem interferir em outros circuitos. Neste aspecto, atenção especial foi dada aos circuitos de CAN e Áudio Analógico do Modem GSM e são apresentados em capítulos posteriores [5].

1.3.3 *Debug* de *Hardware*

Um assunto muito relevante no desenvolvimento de sistema embarcados é a capacidade de identificação de problemas e mal funcionamento do circuito. Diferentemente de um *software* para o qual existem ferramentas especializadas em *debug* que permitem execução passo a passo, monitoramento de variáveis e controle dos processos; problemas relacionados aos sistemas embarcados apresentam em geral múltiplas possibilidades de causas e poucas ferramentas sistemáticas de identificação. Esta seção dedica-se a apresentar em linhas gerais o assunto. O intuito é mostrar princípios de procedimento que podem ser decisivos no sucesso da identificação e correção de erros de *hardware*.

Em [5], Catsoulis apresenta um exemplo interessante relacionado ao *debug* de *hardware*. Apresenta uma situação hipotética de mal funcionamento da interface RS-232. As possibilidades são múltiplas, pode ser erro no código do programa do PC ou do microprocessador, os cabos podem estar deteriorados, o *transceiver* pode ter queimado, o oscilador ou a alimentação podem não estar adequados, ou até mesmo o próprio processador pode estar deteriorado ou estar constantemente sendo reiniciado por conta do circuito supervisor (*WatchDog*). Linhas podem estar rompidas dentre outras muitas possibilidades. Isso mostra como não se trata de algo trivial identificar problemas relacionados aos sistemas embarcados.

A estratégia indicada é "dividir para conquistar". Isso quer dizer que se deve propor múltiplos testes que consigam identificar e isolar o problema. Por exemplo,

para identificar o problema acima, os seguintes passos poderiam ser tomados:

1. Para testar o *software* no PC, conectar o Rx e o Tx do computador e verificar se ocorre o "eco" das palavras enviadas para a porta RS232.
2. Verificar os cabos
3. Fazer um programa simples capaz de acender LED. Este programa pode fazer também o uso de um *timer* interno do dispositivo. Este teste avalia a alimentação do circuito, os circuito supervisor, *clock*, e funcionalidade do microprocessador.
4. Fazer um programa que escreve uma única palavra ininterruptamente e depois, com um osciloscópio, identificar se há atividade dos sinais nos pinos dos componentes.

Através desta estratégia há grandes possibilidades de sucesso na identificação de erros. Mas, há ainda uma atitude que é, sem dúvida, a mais importante. O desenvolvedor de *hardware* deve ser atencioso em todos os detalhes técnicos fornecidos pelo fabricante na integração dos componentes. Uma grande fonte de erros é o desenvolvedor supor que algo deve funcionar de um jeito, mas não verificar se o fabricante implementou daquela maneira. Por ser para ele uma suposição "óbvia" geralmente será o problema mais dificilmente identificado [5]. Por exemplo, há uma restrição imposta pelo fabricante do Modem GSM usado, que para poder alterar o áudio do modem de analógico (*default*) para digital, envia-se um comando AT, mas antes disto é preciso que a interface digital esteja desconectada. Na montagem do circuito, esta restrição implica na existência de *buffers* 3-*state* ou chaves analógicas entre o PCM (áudio digital) do Modem e a FPGA. Negligenciar este fato impedirá por completo a utilização do áudio digital do dispositivo.

1.4 Diagrama de Blocos do Projeto

Esta seção descreve o *hardware* de maneira geral. Detalhes são apresentados nos capítulos posteriores. Primeiramente, o circuito foi pensado para ser uma plataforma de desenvolvimento. Assim, buscou-se ao máximo permitir interfaces que facilitem e viabilizem diversas aplicações com o mesmo *hardware*. O objetivo desta escolha é que

uma plataforma de desenvolvimento permite o aprendizado continuado da tecnologia envolvida.

Porém, esta não é uma plataforma que simplesmente disponibiliza os pinos do processador para que o desenvolvedor faça suas próprias aplicações, como é o caso das *Evaluation Boards* padrões. Antes, algumas possibilidades de aplicações foram pensadas desde o desenvolvimento da placa, para que pudesse proporcionar projetos de grande interesse para o laboratório e de tecnologia avançada. Estas possibilidades são discutidas no capítulo 7.

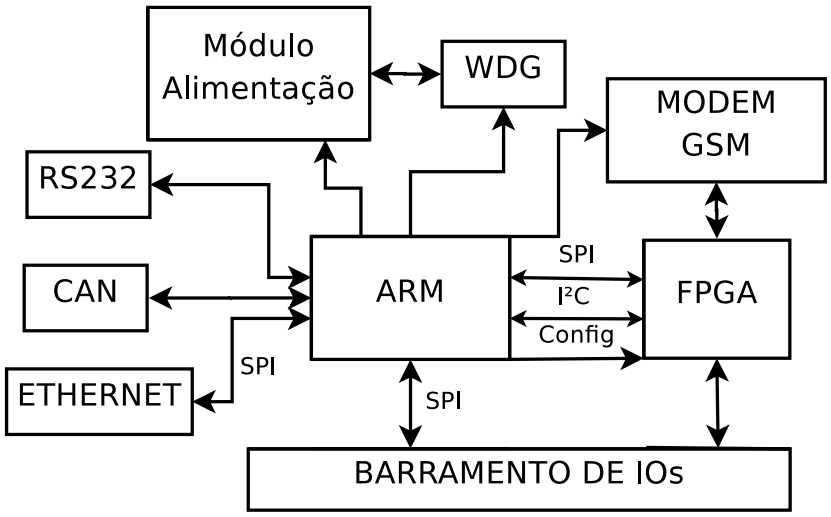

Figura 1.2: Diagrama de Blocos do *Hardware*

A figura 1.2 mostra um diagrama de blocos esquemático representativo do projeto. As principais características são:

- O módulo de Alimentação da Placa recebe tensão de 5 e 12 *Volts* e alimenta os CIs. Há um regulador separado para o ARM e também possibilidade de que o ARM desligue os outros reguladores. O motivo disto é satisfazer requisitos de operação em *low-power*. Garantindo o funcionamento continuado do ARM para aplicações que utilizem o *Real-Time*.
- Uma interface RS-232 simples foi implementada.
- Para interligação em rede estão presentes o CAN e a Ethernet. Para o CAN existe um *driver* especializado do próprio fabricante do ARM escolhido. Para a Ethernet foi utilizado um controlador da Microchip (ENC28J60) que comunica com o ARM via SPI.

- Um circuito supervisor externo (*WatchDog*) foi também adicionado para impedir que o mal funcionamento do sistema o deixe bloqueado.
- Um Modem G24 da Motorola para dar acesso à rede GSM, permite possibilidade de envio e recebimento de SMS, efetuar chamadas e também acesso ao áudio analógico e digital nas chamadas.
- Uma FPGA que atua como um co-processador ou como *driver* especializado para aumentar a capacidade de desempenho e possibilidades da placa.
- Interface JTAG está presente tanto para programação e *debug* do ARM como para programação externa para a FPGA.
- Conector para interfaceamento externo com um *driver* dedicado de SPI, 6 linhas de aplicação geral de saída do ARM, e mais de 40 pinos de Entrada e Saída (IO) da FPGA.

Este capítulo introdutório apresentou a motivação para o projeto do *hardware* realizado bem como as potencialidades que este apresenta. Os próximos capítulos dedicam-se a detalhar os blocos que compõem o sistema como um todo. O capítulo 2 diz respeito ao ARM escolhido para o projeto. Apresenta os periféricos do mesmo, o ambiente de desenvolvimento de aplicações e as interfaces presentes na placa desenvolvida. No capítulo 3 o modem GSM é discutido. A FPGA e suas interfaces são definidas no capítulo 4. O capítulo 5 é para a rede Ethernet, ao passo que a rede CAN é apresentada no capítulo 6. Por fim, sugestões de implementações e as capacidades da placa desenvolvida é o tema do capítulo 7. O anexo A mostra os esquemáticos dos blocos do circuito implementado e o anexo B apresenta a placa projetada.

Parte II

Desenvolvimento

Capítulo 2

ARM 7 : Processador Central

Na introdução foi apresentado o *core* do ARM7 com algumas das características que o tornaram o maior expoente da categoria de processadores de 32 *bits* da atualidade. Neste capítulo, será feita uma abordagem específica em relação ao processador escolhido para o projeto, o STR712FR2 da STMicroeletronics. Uma análise das capacidades e periféricos deste processador, ferramentas de desenvolvimento e suporte fornecido pelo fabricante serão apresentados.

Este documento não tem por objetivo apresentar todos os detalhes do *hardware*, antes, apenas os aspectos específicos relacionados com o circuito realizado. Por este motivo, novos desenvolvedores de *firmware* para esta placa devem analisar cuidadosamente a documentação deste microprocessador, em especial o *datasheet* [22] e o manual de referência [19].

2.1 STR712FR2 : O ARM da STMicroeletronics

Apesar da existência de mais de 10 fabricantes de processadores com o núcleo ARM, escolheu-se a empresa STMicroeletronics. O requisito inicialmente observado foi a alta imunidade ao ruído apresentado pelo ARM desta empresa. Como espera-se que a placa sirva como plataforma de desenvolvimento para motores em ambientes industriais, ou mesmo para aplicações com o modem, o qual também gera mais espúrios, é realmente determinante que o processador tenha a maior robustez possível.

Além desta característica intrínseca da arquitetura, as principais características do

processador escolhido [22] úteis para a plataforma montada são:

- Processador RISC ARM7TDI 32-bit
- Operação em até 66MHz
- Memória Flash de programa: $256Kbytes$; Memória Flash de dados: $16Kbytes$ e Memória RAM de $64Kbytes$.
- *Real Time Clock*
- Modos de operação para baixo consumo
- Controlador Vetorial de Interrupções interno com 16 níveis de prioridades para 32 canais de interrupções padrões (IRQ), mais dois canais privilegiados de interrupções (FIQ).
- 5 *timers*
- 1 I^2C
- 2 UARTs
- 2 BSPIs
- CAN (2.0B Ativa)

Além destas características do processador, o fabricante oferece um vasto conjunto de *application notes*, *softwares*, manuais e *firmwares* para auxiliar o desenvolvedor [23]. Entre os documentos encontrados ali, destaca-se o pacote do *software library* [21] o qual contém um conjunto de arquivos, escritos em C, que mapeam os registros do processador e também contém funções que se constituem nos *drivers* dos periféricos deste. A tabela 2.1 apresenta os principais cabeçalhos desta biblioteca. Foi preciso fazer algumas alterações nas definições destas bibliotecas para compatibilizá-las com o ambiente de desenvolvimento utilizado, o GNU-LINUX.

2.2 Ambiente de Desenvolvimento

Para o desenvolvimento de *firmwares* utilizou-se GNU *softwares*. O compilador responsável por transformar os programas escritos na linguagem C para o código de

Tabela 2.1: Principais Módulos da Biblioteca para Programação do ARM - STR7

Arquivo	**Descrição**
flash.h	Unidade de Controle para Escritas e Leituras na Memória Flash
gpio.h	Unidade de Configuração dos Pinos IO
rccu.h	Unidade de Controle do *Clock* e *Reset*
pcu.h	Gestão do Modo de Energia
wdg.h	Configuração do *WatchDog*
bspi.h	Gestão do *Driver* SPI
i2c.h	Gestão do *Driver* I^2C
uarth.h	Gestão da UART
rtc.h	Unidade de Configuração do Real-Time
eic.h	Unidade de Gestão das Interrupções (*Enhance Interrupt Controller*)
tim.h	Controle e Gestão dos *Timers*
can.h	Gestão do *Driver* CAN
xti.h	Controle das Interrupções Externas

máquina do ARM foi o *arm-elf-gcc*. Como sugere, trata-se de uma especialização do gcc para o ARM. Os arquivos e toda a documentação necessária para instalação do compilador podem ser encontrados em [8]. Este *site* contém os utilitários, o compilador, a biblioteca e até mesmo o *insight*, *software* útil para programação e *debug* de *firmware*. Ali também é possível encontrar *links* para outros *sites* com ferramentas úteis para desenvolvedores de aplicativos para o processador ARM.

Além do compilador, faz-se necessário o programador. Foi escolhido um programador JTAG, o OpenOCD (*Open On-Chip Debugger*). Este *software* foi desenvolvido inicialmente por Dominic Rath [18]. Através deste aplicativo é também possível desenvolver *debugging* do *software* 'rodando' no processador de um PC.

Porém, para ser capaz de projetar aplicativos para a plataforma é necessário conhecê-la antes nos seus detalhes e interfaces. Por isso, a partir da próxima seção, as interfaces entre o ARM e os outros periféricos são apresentadas. Este capítulo aborda ainda detalhes de alguns periféricos e barramentos, porém os periféricos mais importantes são detalhados nos capítulos seguintes.

2.3 O Esquemático do Circuito

No anexo A, encontra-se o diagrama esquemático do ARM (fig.A.1) e de todos os componentes que estão relacionados com ele. Os blocos funcionais que se podem identificar naquele circuito são:

- Processador ARM
- Circuito de Relógio (*Clock* e RTC)
- Circuito de Alimentação
- Interface JTAG
- *Jumpers* para definição do modo de operação do processador.
- Circuito Supervisor (*WatchDog*)
- Uma interface *serial-in/parallel-out* responsável por permitir interfaceamento do ARM com periféricos mais "lentos" do circuito.
- Decodificador para aumentar os canais de comunicação com a SPI Interna (74HC138)

2.3.1 Circuito de Relógio

A figura 2.1 mostra a configuração dos circuitos temporizadores relacionados ao processador. Para o RTC, é suficiente apenas o cristal de 32kHz, ao passo que para o *clock* interno do ARM exige-se o projeto do oscilador completo. Observe que o *clock* da FPGA também vem deste circuito.

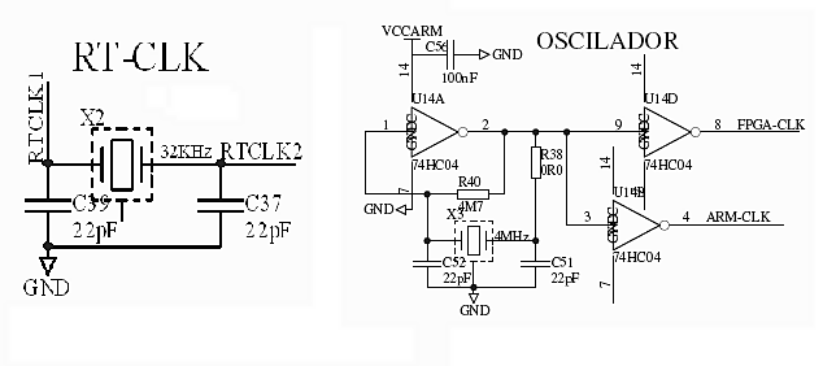

Figura 2.1: Circuito temporizador do ARM

2.3.2 Circuito de Alimentação

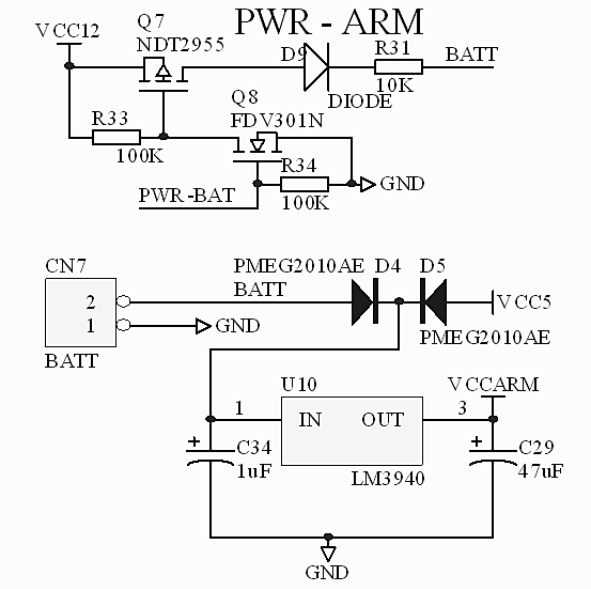

Figura 2.2: Circuito de Alimentação do ARM

O circuito de alimentação é constituído de um regulador de baixo *drop-out* para fornecer 3,3 *Volts* a partir da tensão de entrada de 5 *Volts*. À entrada do regulador há 2 diodos que conectam duas fontes independentes, de 5 *Volts* da alimentação e uma bateria que estará também no circuito. O propósito da bateria é garantir que o ARM não precise ser desligado. Ele poderá operar em *low-power mode* garantindo o funcionamento do RTC interno para que o sistema sempre mantenha a hora atualizada. Há um circuito auxiliar de carregamento da bateria que funciona para PWR-BAT = '1'.

2.3.3 Interface JTAG

O conector JTAG permite que novas programações e até *debug* possa ser realizado no ARM. Como já foi dito, o *software* que faz acesso à porta paralela para programar o ARM com a interface JTAG é o OpenOCD. O circuito foi implementado de forma a permitir que seja dado comando de *reset* para o ARM através da interface JTAG, requisito importante para o funcionamento dos *softwares* de *debug*.

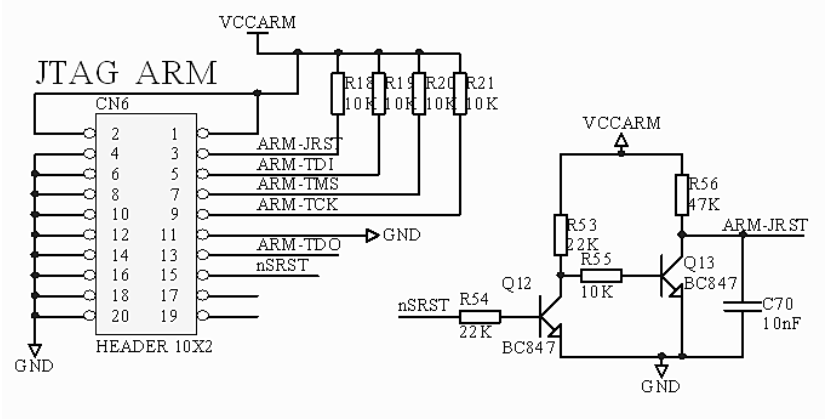

Figura 2.3: Interface JTAG do ARM

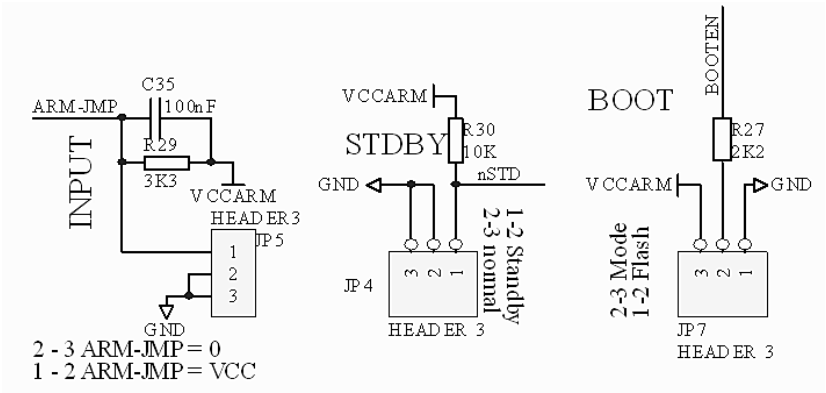

Figura 2.4: Jumpers de Configuração do ARM

2.3.4 *Jumpers*

Há 3 *jumpers* principais: *StandBy* (nSTD), BOOTEN e ARM-JMP. O primeiro serve para colocar o ARM em *low-power mode*. De acordo com o manual de referência do dispositivo, há duas maneiras possíveis de se fazer isso, uma é através de conectar o pino STDBY ao GND, a outra é através de rotinas de *software* para alterar registros que configuram o modo de operação. Em *low-power mode*, todas as interfaces são desligadas, ficando apenas o circuito regulador de *back-up* para funcionalidades essenciais do processador, como é o caso do RTC.

Outro *jumper*, BOOTEN, define qual o endereço das primeiras rotinas do processador. Se o valor de BOOTEN for 0, então o sistema sempre será inicializado pela sua memória Flash. Porém, se o BOOTEN for 1, então a inicialização do sistema depende dos valores dos campos BOOT[0:1] pertencentes ao registro PCU_BOOTCR. De acordo com a programação destes registros o ARM pode inicializar pela Flash, pela RAM ou

por uma memória externa.

Por fim, o ARM-JMP é um *jumper* de propósito geral a ser definido pela aplicação do programador do *firmware* do processador. Trata-se de um pino de *input* para o ARM e que pode servir para qualquer finalidade.

2.3.5 *WatchDog*

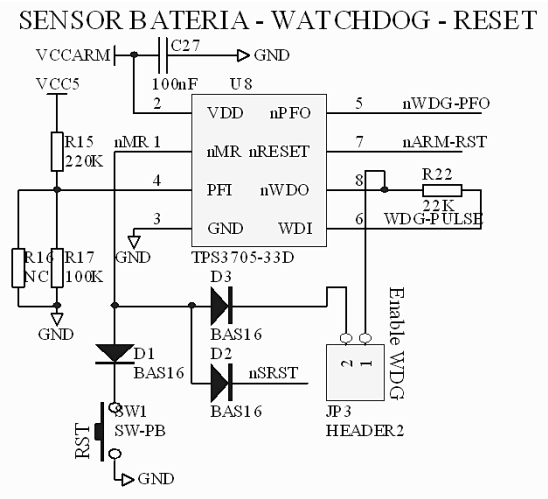

Figura 2.5: Circuito de Supervisão (*WatchDog*)

O circuito supervisor, ou *WatchDog*, TPS3705-33 [25], tem a capacidade de reinicializar o processador e tem as seguintes funcionalidades:

- Inicialização do Circuito
- *Reset* Manual
- *Reset* vindo do JTAG
- Indicação de queda na tensão de alimentação (PFO - *Power-Fail*)
- Temporizador para controlar se o sistema entrou em *deadlock* com período de 1.6s.

O objetivo deste circuito é garantir que o sistema funcione adequadamente e com segurança. Inicialmente, ao ligar o circuito, o CI mantém o sinal de nRESET do ARM em zero até que a alimentação tenha alcançado nível de 2,8V aproximadamente, para então permitir que o microcontrolador ligue. Além disso, há um temporizador interno

que a cada 1,6 segundos envia pulso de *reset* para o ARM a menos que este seja reinicializado, o que é feito através de enviar um pulso para o pino WDI (*WatchDog Input*). É possível desabilitar seu funcionamento desconectando o pino WDI, através da retirada do *jumper* junto ao WDG. Através desse mecanismo de *reset*, garante-se que o processador não esteja em *loop* infinito, ou tenha "travado" (*deadlock*).

Se a tensão de alimentação estiver abaixo de 4 *Volts*, então o regulador já não pode operar satisfatoriamente. Neste caso, a tensão em nPFO vai a zero. Frequentemente o ARM pode monitorar o valor dessa tensão, e caso ocorra essa queda, ele pode desligar todos os periféricos e entrar no modo STANDBY para economizar energia e manter assim o RTC sempre ativo.

2.3.6 SPI

Antes de apresentar o módulo *serial-in/parallel-out* é preciso apresentar de maneira geral o que é uma SPI, pois esse módulo faz uso dela. SPI é um acrônimo para *Serial Peripheral Interface*, que define uma interface para comunicação de um processador central e periféricos. Trata-se de um protocolo *master-slave*, de múltiplos pontos. É uma interface simples definida por 4 fios:

- CLK (*Clock*)
- MOSI (*Master Output / Slave Input*)
- MISO (*Master Input / Slave Output*)
- nCS (*Chip-Select*)

De maneira geral, o mestre (*master*) decide com quem vai se comunicar colocando o CS deste dispositivo em '0'. Depois disso, a cada pulso de *Clock* são transferidos os *bits* entre o *Master* e o *Slave*, e deste para o mestre, entrando em um *shift-register*. Há quatro modos de operação possíveis dependendo se os dados podem ser alterados enquanto o *clock* for 0 ou 1, e se os *bits* são amostrados na subida ou descida do *clock*. A figura 2.6 mostra o modo 00, ou seja, a interface é livre para alterações dos *bits* enquanto o *clock* está em nível 0 e a amostragem dos valores é feita na subida do pulso de *clock*.

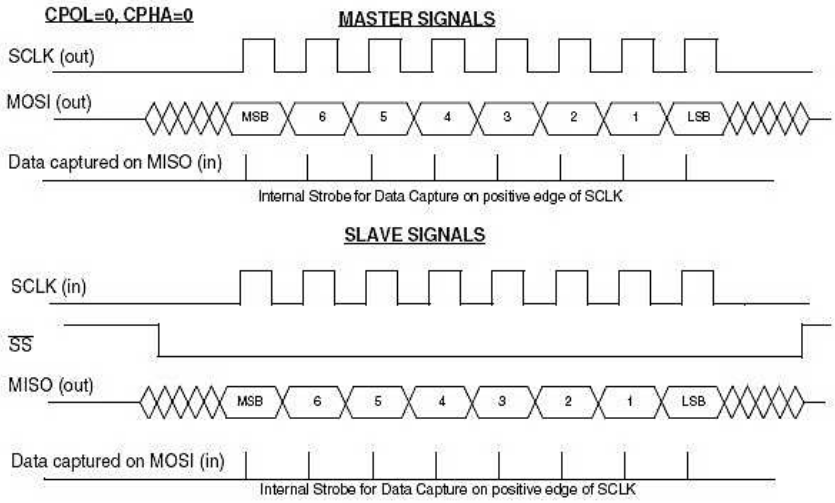

Figura 2.6: Temporização do Sinal SPI no modo 00

O ARM STR7 tem dois periféricos SPI. Através destes periféricos, a transferência serializada é implementada automaticamente pelo *driver* do periférico. Após a correta configuração dos registros destes *drivers* o *core* do processador é capaz de se comunicar com outros dispositivos com rotinas simples de escrita e leitura de registros.

A única garantia necessária é que o ARM deve ser capaz de fornecer o sinal de *Chip-Select* ao dispositivo com o qual deseja conversar. Na plataforma desenvolvida, o ARM comunica com três dispositivos diferentes através da SPI, são eles: o controlador da Ethernet, a FPGA e a interface *Serial-in/Parallel-out* ou Módulo de Controle. Para fornecer os sinais de *Chip-Select* escolheu-se utilizar um decodificador (74HC138 - fig. 2.7), que recebe três sinais e permite até 8 canais de CS. Um canal é para Ethernet, outro para a interface de saída e 5 são para a FPGA, assim facilita-se a divisão de canais para diferentes aplicações que podem estar ocorrendo na FPGA.

2.3.7 *Interface Serial-IN / Parallel-OUT*, Módulo de Controle

O módulo *Serial-IN / Parallel-OUT* é constituído por dois CIs 74HC595 [17]. Na entrada deste CI há os sinais MOSI e CLK da SPI. Embora ele não seja um dispositivo SPI propriamente dito, o comportamento de *shift-register* permite que se utilize o *driver* SPI para enviar os *bytes* de configuração deste dispositivo. O *chip* só transfere os dados para a saída no instante de subida do pino de RCLK. Portanto, não há nenhum inconveniente de se usar o mesmo protocolo de transmissão e recepção de dados via SPI tradicional.

Além disso, este CI apresenta a funcionalidade de ser *tri-state*, estado controlado através do pino OUT-RDY do ARM. Através dele, é possível garantir a pré-inicialização dos pinos do dispositivo antes que os seus valores estejam válidos. Note que isto é muito importante, visto que algumas das saídas deste CI são controladores de linhas de alimentação e controle de outros periféricos do circuito. Pelo fato da maioria destes sinais estarem relacionados ao controle de ON/OFF e habilitações de outros periféricos é que também é referenciado por módulo de controle.

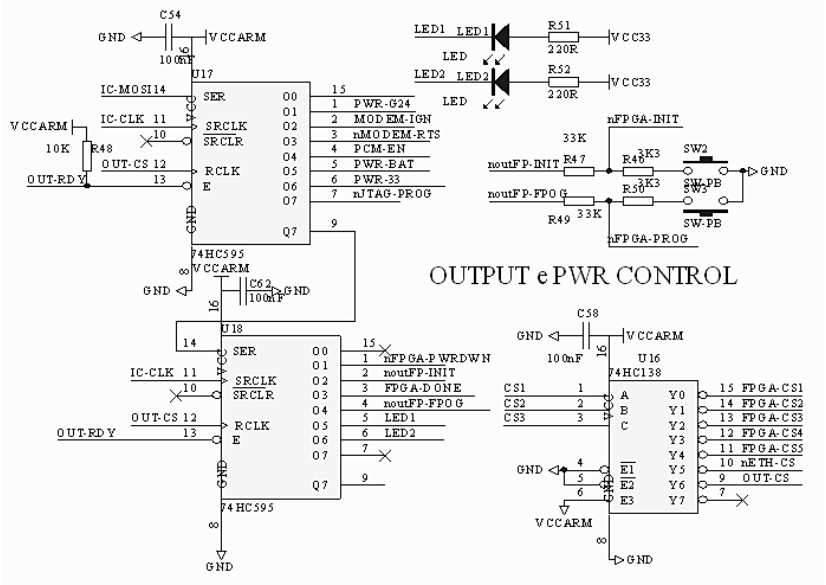

Figura 2.7: *Serial-IN/Parallel-OUT* e Decodificador

2.3.8 Interface Completa entre o ARM e os outros Periféricos

A figura 2.8 mostra o processador com as conexões de alimentação e também todos os pinos de interface. Além disso, a tabela 2.2 apresenta a função definida para estes pinos na plataforma desenvolvida, a identificação dos mesmos no esquemático da figura e uma descrição abreviada das funcionalidade, que serão expandidas nos próximos capítulos.

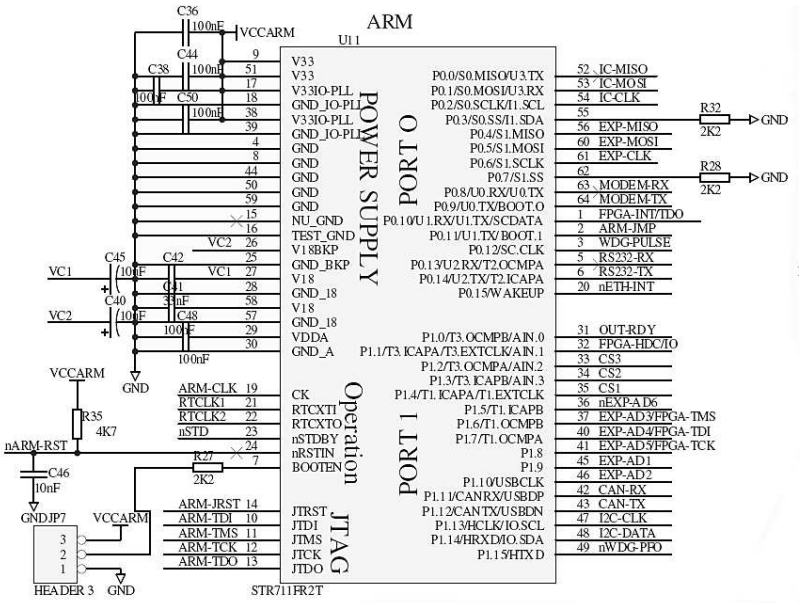

Figura 2.8: Mapeamento dos Pinos do ARM

Tabela 2.2: Definição da Interface do ARM no projeto

Definição	**Identificação**	**Pino**	**Descrição**
SPI INTERNA	IC-MISO IC-MOSI IC-CLK	P0.0 P0.1 P0.2	Interface para ENC28J60, Módulo Controle e FPGA
SPI EXTERNA	EXP-MISO EXP-MOSI EXP-CLK	P0.4 P0.5 P0.6	Interface SPI para conexão externa
UART0	MODEM-RX MODEM-TX	P0.8 P0.9	Interface para comunicação com o modem GSM
INTERRUPÇÃO FPGA	FPGA-INT/TDO	P0.10	Interrupção externa provocada pela FPGA
JUMPER	ARM-JMP	P0.11	Propósito Geral
CIRCUITO SUPERVISOR	WDG-PULSE	P0.12	Atualiza supervisor indicando o funcionamento normal do processador
RS-232	RS232-RX RS232-TX	P0.13 P0.14	Interface de comunicação
INTERRUPÇÃO ETHERNET	nETH-INT	p0.15	Interrupção externa provocada por ENC28J60
ENABLE OUTPUT	OUT-RDY	P1.0	Permite atualizações na interface *PARALLEL-OUT*
HDC	FPGA-HDC/IO	P1.1	HDC: configuração FPGA ou IO
CHIP SELECT	CS1 CS2 CS3	P1.2 P1.3 P1.4	Expansão das linhas de nCS para uso da SPI interna
INTERFACE SAIDA	nEXP-AD6 EXP-AD1 EXP-AD2	P1.5 P1.8 P1.9	Função Geral
INTERFACE SAÍDA ou JTAG-FPGA	AD3/TMS AD4/TDI AD5/TCK INT/TDO	P1.5 P1.6 P1.7 P1.10	Funções Multiplexadas: Programação FPGA ou Interface Saída
CAN	CAN-RX CAN-TX	P1.11 P1.12	Rede CAN
I^2C	I^2C-CLK I^2C-SDA	P1.13 P1.14	I^2C para interface com FPGA
ENERGIA BAIXA	nWDG-PFO	P1.15	nWDG-PFO=0 indica que a tensão caiu a baixo de 4V

Capítulo 3

Modem GSM

A plataforma desenvolvida tem o modem G24 da Motorola [12] capaz de se comunicar na rede GSM. A presença deste modem, que em linhas gerais tem as mesmas funcionalidades de um celular, permite desenvolvimento de aplicações diversas relacionadas ao envio e recebimento de dados ou voz na rede GSM. O modem é controlado por comandos AT e o conjunto de instruções válidas, além de atender requisitos para modens em geral, executa funções especiais projetadas pela Motorola. O tutorial dos comandos AT pode ser encontrado em [11].

As principais características do G24 são [12]:

- Pode operar nas 4 Bandas de Comunicação GSM/GPRS/EDGE (850/900/1800/1900)MHz
- Alta performance e desempenho na comunicação em RF
- Alimentação de 3,3 a 4,2 V
- Potência de Transmissão:
  - 2W (850-900 MHz)
  - 1W (1800-1900MHz)
- Alta sensibilidade na recepção, capaz de receber sinais à -106dBm
- Interface Serial
  - 2 UART
  - 1 USB

- Áudio Digital e Analógico
- Conversores AD
- *Real Time Clock*
- Regulador de Tensão 2,7V

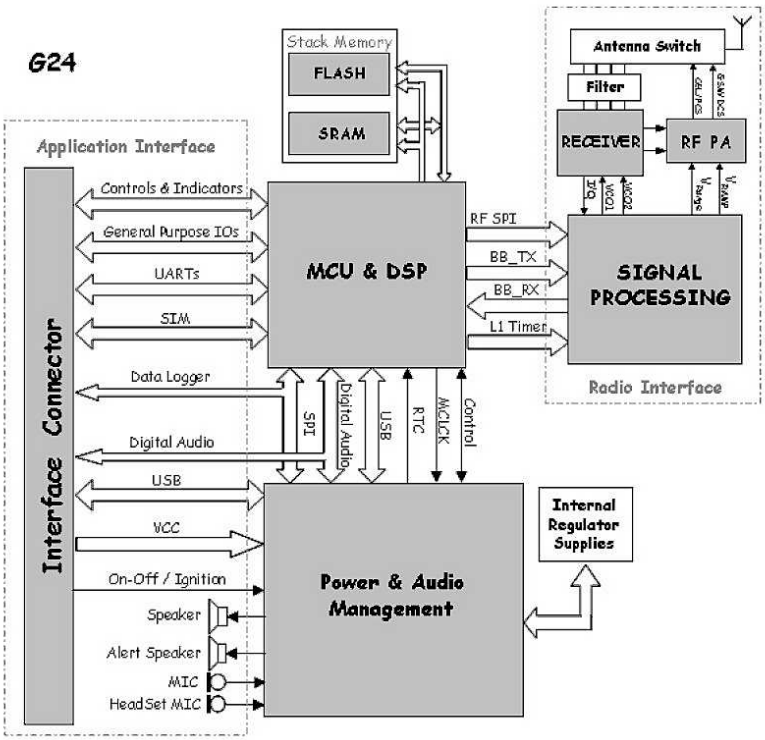

Figura 3.1: Diagrama de Bloco da Organização Interna do Modem. Extraído de [12]

A figura 3.1 apresenta em diagramas de blocos a estrutura interna do Modem com sua interface e organização estrutural. Os seguintes blocos estão presentes:

- **Bloco Digital**
  - Unidade de Processamento (MCU)
  - Módulo de processamento digital de sinal para voz e dados (DSP)
  - Interfaces Seriais
    - USB *driver slave*
    - UART1
    - UART2

    - SIM *card*
  - Barramento de Áudio Digital (PCM)
  - 8 pinos de IO

- **Bloco Analógico**
  - *Power Manangement* IC (PMIC)
    - Regulador interno
    - Regulador externo para uso geral
  - Interface Áudio Analógica
  - Conversores AD para uso geral
    - AD padrão
    - Sensor de Tensão
    - Sensor de Temperatura
  - RTC

- **Bloco GSM *Transceiver***
  - Amplificador de RF de 3 estágios
  - Receptor de RF
  - Processamento de sinal embarcado para transmissão e recepção de dados
  - Módulo *Front End* - Inclui a Antena e o filtro específico
  - Filtro para escolher a banda de operação entre as 4 possíveis

3.1 O circuito de Alimentação do Modem

Por se tratar de um circuito com potência alta em comparação com os outros módulos presentes, com potência de transmissão de até 4W, além de picos de corrente de até 2 A; e pelo fato de ser um emissor de RF, preocupações adicionais com ruído devem ser tomadas.

A este respeito, a primeira coisa importante diz respeito ao circuito de alimentação do modem. Utilizar reguladores lineares neste caso é proibitivo por causa do seu baixo rendimento. Por isso, escolheu-se um regulador chaveado, o LM2576, cuja eficiência

média é de 75%. O ganho na eficiência advém do chaveamento o qual gera muitos ruídos espúrios na rede. Assim, requer-se que o filtro de linha e o roteamento deste bloco sejam bem projetados.

O *datasheet* do componente [14] apresenta dicas e normas para o projeto do filtro, bem como orientações a respeito do roteamento. A figura 3.2 mostra uma montagem possível apresentada pelo fabricante. Além disso, apresenta tabelas e fórmulas para estimação dos indutores e capacitores que compõe o filtro. Para se ter boa margem de segurança, determinou-se o uso de indutor de 100μH e capacitor de 2200μF.

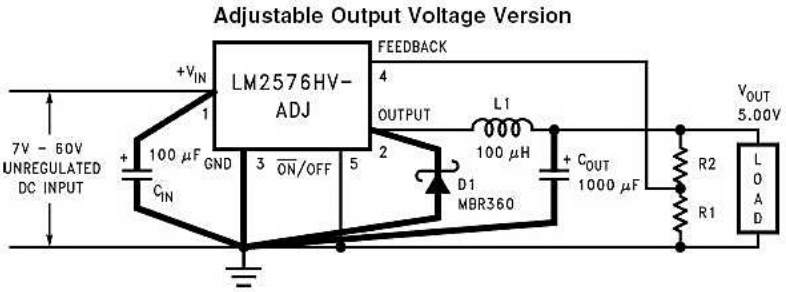

Figura 3.2: Circuito exemplo fornecido pelo fabricante do regulador chaveado. Extraído de [14]

Além do projeto do regulador cujo objetivo é evitar ruído externo, há também fontes de ruído internas no modem cujos efeitos precisam ser atenuados. Por esta causa, filtros capacitivos são postos na entrada da alimentação do modem, garantindo assim bom desempenho. Por orientação do fabricante, deve-se usar os seguintes capacitores:

- 10nF e 100nF - filtrar ruídos da interface digital
- 10pF - filtro para a banda de 1800 e 1900MHz
- 39pF - filtro para a banda de 850 e 900MHz

3.2 Análise do Áudio do Modem

Além do circuito de alimentação, atenção especial também foi dada à interface de áudio do G24, tanto a analógica quanto a digital (PCM - *Pulse Code Modulation*). Este modem apresenta uma conexão interna entre o áudio analógico o digital. Por isso, ao se usar o áudio analógico, a interface com o áudio digital deve estar desconectada. Por

default, o modem inicializa no áudio analógico e para mudá-lo para o áudio digital, deve enviar o comando AT+MDIGITAL. Antes porém de enviar o comando, é preciso garantir que a interface digital está desconectada do modem. Não atentar para isso pode causar estragos no modem.

Para garantir este requisito foi posto entre o G24 e a FPGA, que faz a interface com o PCM, uma chave analógica, o 74HC4066 ([16]). Através dele é possível garantir que a interface com o PCM do Modem está desconectada. O sinal PCM-EM controla a chave, quando em 1, a chave está fechada e abre para nível lógico 0. A tabela 3.1 apresenta o algoritmo recomendado para mudança entre o áudio digital e o analógico.

Tabela 3.1: Procedimento para mudança do áudio Analógico para o Áudio Digital

Alteração do Áudio: Analógico → Digital
PCM-EN neste momento deve ser 0!
Enviar AT+MADIGITAL=1 para o modem PCM-EN recebe 1
O áudio digital já está em funcionamento

Como foi dito, a interface do modem digital está conectada à FPGA. O objetivo desta ligação é permitir que seja desenvolvido um *driver* especial na FPGA para tratar do protocolo de comunicação PCM. Assim, a FPGA pode fazer o áudio trafegar entre o Modem e o ARM, pelos barramentos PCM e SPI ou I^2C respectivamente.

O PCM é produzido através da amostragem de um sinal de áudio num conversor AD com resolução de 13 *bits*, uma taxa de amostragem de 8kHz e na frequência de transmissão de 512kHz. Esta definição é válida para áudio simples, no caso de transmissão *stereo*, os tempos mudam e podem ser verificados em [12]. A interface PCM define 4 fios, sendo eles:

- PCM-CLK - *clock* de 512kHz
- PCM-FS - *frame sync* (marca de início de pacote) de 8kHz
- PCM-DOUT - 13 *bits* de áudio de saída com decodificação linear
- PCM-DIN - 13 *bits* de áudio de entrada com decodificação linear

O formato dos pulsos e a temporização são representados na figura 3.3.

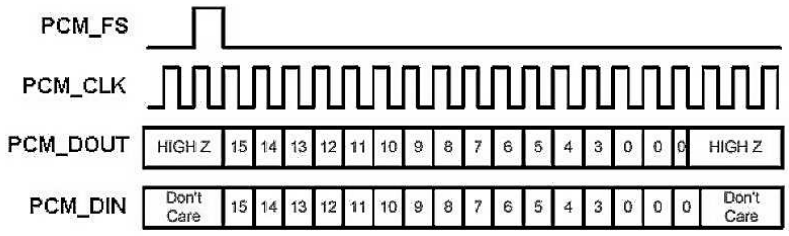

Figura 3.3: Formato do Código PCM para transmissão áudio simples. Extraído de [12]

Além do áudio digital, o projeto do circuito de áudio analógico também exige atenção. Na placa, está disponível apenas o sinal para fones de ouvido e microfone juntos, os *headseats*. Os ruídos das frequências dos sinais digitais, do modem e de outros periféricos podem deteriorar a qualidade do áudio. Para melhorar a relação sinal ruído, as seguintes recomendações foram observadas:

- Isolamento do terra do áudio - ligado ao AGND do modem
- Componentes do áudio analógico circundados pelo terra (AGND)
- Uso de capacitores de filtros
- Distanciamento da antena

3.3 Interface de Comunicação com ARM: UART

O modem G24 define 2 interfaces UARTs. Uma é destinada à comunicação, recebimento de comandos AT e transferência de dados. A outra é de aplicação livre, podendo estar associada a componentes com outras interfaces como *Bluetooth* ou GPS por exemplo; esta porém, está desconectada na plataforma desenvolvida.

A UART que faz interface com o ARM tem a arquitetura de um DCE e o processador deve ser um DTE de acordo com o padrão especificado pela norma RS-232. A figura 3.4 mostra como é definida a interface entre um dispositivo DCE e um DTE. É importante salientar que o modem nunca envia dados para a UART se os sinais nDTR ou nRTS forem iguais a 1. A plataforma não implementou no ARM um DTE. Antes, assumiu-se que o nível de transmissão de dados não é suficiente para que haja necessidade de interrupção da transmissão e conectou-se o DTR diretamente ao GND. Porém o

Tabela 3.2: Modos de Operação do Modem G24

Modo	Descrição	Características
Desligado	Alimentação Desconectada	O G24 está OFF e todos os sinais conectados à interface em 3-*state* ou 0
RTC	Alimentado RESET=0	A interface G24 está OFF. O processador PMIC opera em modo RTC. todos os sinais conectados à interface em 3-*state* ou 0
Idle	RESET=1 nCTS = nDSR = 1	Modo *default*, G24 está ativo.
Sleep	RESET=1 nCTS=0	G24 em *low-power* porém ainda monitora a rede GSM.
CSD call	RESET=1 TXEN ativo	Chamada em progresso. Após término, G24 volta ao modo anterior à chamada.

nRTS está sob o comando do ARM, para que se for preciso, ele possa interromper o recebimento de dados vindo do G24.

A configuração por *default* da UART do Modem é: 8 *bits* de dados, 1 *stop bit*, sem *bit* de paridade e detecção automática de *baud-rate*.

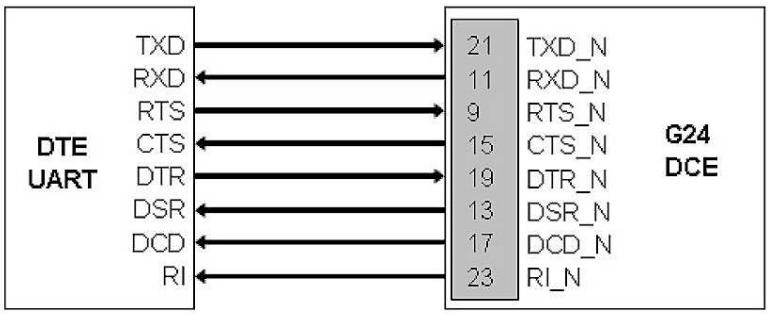

Figura 3.4: Diagrama da conexão de um DCE e um DTE segundo a norma RS-232. Extraído de [12]

3.4 Restrições e Características Gerais

Há 5 modos de operação definidos para o modem, descritos na tabela 3.2. Para cada modo de operação, algumas restrições são impostas, em especial, nos modos de baixo consumo, exige-se que a interface digital esteja desligada ou em nível zero.

Outra restrição imposta pelo fabricante é o nível de tensão da interface digital o

qual é tipicamente 2,7V, não podendo exceder 3V. Por isso, ligação direta aos pinos do ARM ou FPGA não é permitida. Este valor vale tanto para as entradas como para as saídas do modem. Para as saídas, não há nenhum problema, pois a partir de 2 *Volts* o ARM já distingue como nível lógico 1. A solução para o problema relacionado aos pinos de *input* foi utilizar um circuito abaixador de tensão (fig. 3.5).

Esta solução é possível pois há um regulador interno do modem com tensão nominal de 2,75V. Este sinal é posto na base do transistor abaixador, e garante que a tensão máxima no coletor, que é a entrada do modem, seja menor que 2,7V. Além disso, esta solução também garante que a interface estará desligada para os modos *low-power*. Isso, porque o regulador segue também o modo de funcionamento do Modem, e quando em *low-power*, não tem capacidade de corrente e mantém o transistor cortado a despeito da tensão no emissor.

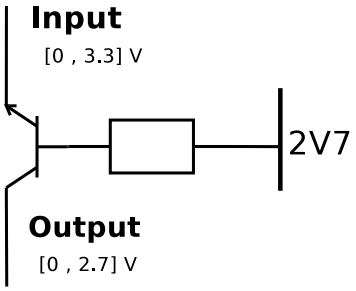

Figura 3.5: Circuito abaixador de tensão

Outra recomendação importante diz respeito ao roteamento que liga a interface SIM ao modem. De acordo com o fabricante, cuidado especial deve ser dado ao pinos SIM-CLK e SIM-DIO. A proximidade destes pinos pode provocar baixa relação sinal-ruído devido às capacitâncias parasitas. Por isso, um distanciamento no roteamento foi usado, além da separação destes dois sinais por uma linha terra que reduz o efeito de *cross-talk*.

3.5 Interface do Modem

O esquemático completo da interface do G24 pode ser consultado na figura A.4 no anexo A. A tabela 3.3 apresenta os pinos da interface que foram utilizados na placa e a descrição sucinta dos mesmos.

Tabela 3.3: Definição da Interface do Modem G24

Definição	Identificação	Descrição
SIM-CARD	SIM-RESET SIM-DIO SIM-CLK	A interface de controle do SIM-CARD
PCM	PCM-DIN PCM-DOUT PCM-FS PCM-CLK	Interface de controle do áudio digital através da codificação PCM
Áudio Analógico	SPKR_N HDST_MIC	Interface de áudio analógico para dispositivos *headseat*
UART	MODEM-RX MODEM-TX nMODEM-CTS nMODEM-RTS	Interface de comunicação serial para controle via comandos AT e transmissão e recepção de dados

3.6 Resumo das Recomendações

Como foi apresentado neste capítulo, o modulo GSM apresenta algumas restrições de operação que devem ser respeitadas com pena de dano irreparável ao dispositivo. Por se tratar de um assunto importante, esta seção apresenta um sumário das principais recomendações para projeto envolvendo o modem:

- Manter o PCM-EN em nível 0, mude para nível 1 apenas após mudança do áudio para digital
- Não alterar a configuração por *default* do regulador de 2,75V.
- Manter sempre que possível os pinos de entrada do Modem em nível lógico 0.
- Deixar o pino nRTS em *low* para que o modem possa enviar dados ao ARM.

Por fim, para ligar o Modem, além da alimentação, deve-se colocar o IGN='1'. Após ligado pelo IGN, o modem permite ser ligado e desligado via comandos AT.

Capítulo 4

FPGA

Como já foi dito no capítulo 1, a FPGA foi acrescentada ao projeto da placa por ter a capacidade de ser reprogramável. A arquitetura interna da FPGA que permite esta capacidade pode ser observada na figura 4.1.

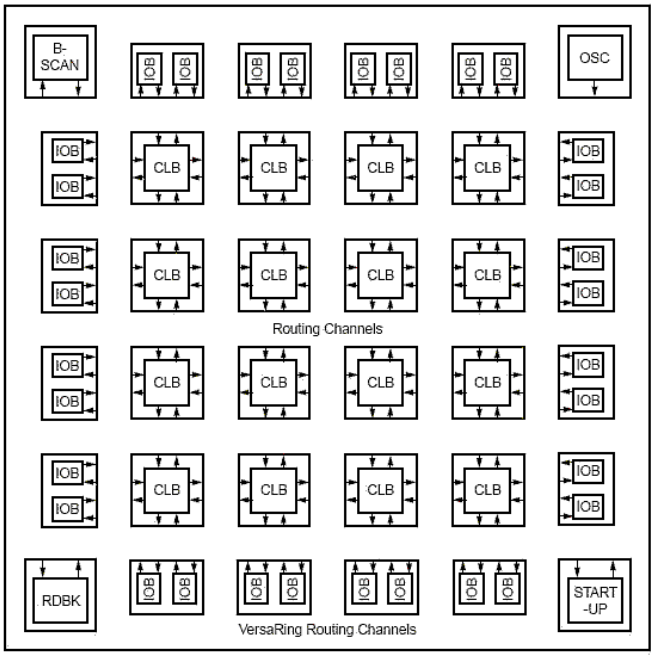

Figura 4.1: Arquitetura interna da FPGA

A capacidade da arquitetura está no fato de, primeiramente, não haver ligação pré-estabelecida entre os blocos, elas estão condicionadas a cada *hardware* descrito. Além disso, os CLBs (*Configurable Logic Block*) são blocos cuja função lógica não é estática. A figura 4.2 apresenta a estrutura interna de um CLB. Nele há uma memória de 16

posições que permite qualquer lógica de 4 entradas e 1 saída, *latches* para as saídas e multiplexadores. Cada CLB pode realizar uma das seguintes funções:

- Qualquer função de 4 variáveis, mais qualquer outra função de outras quatro variáveis não relacionadas, mais função qualquer com outras 3 variáveis
- Qualquer função de 5 variáveis
- Qualquer função de 4 variáveis mais algumas de 6 variáveis,
- Algumas funções de 9 variáveis

A versatilidade destes CLBs permite que várias funcionalidades sejam feitas com poucas destas estruturas e favorece o desempenho no quesito da velocidade de processamento.

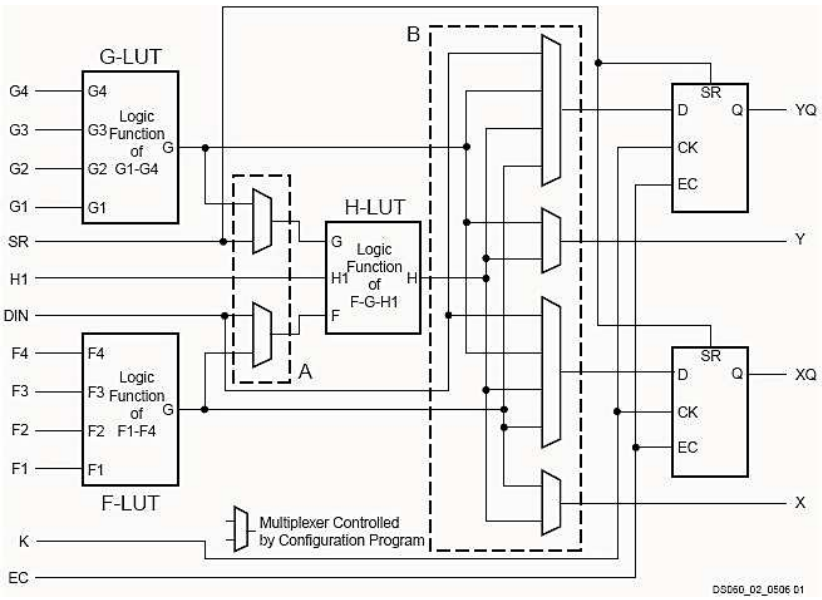

Figura 4.2: Diagrama Lógico de um CLB. Extraído de [28]

A FPGA SPARTAN XCS05XL, presente no projeto, contém um total de 100 CLBs que lhe dá a capacidade de até 5000 portas lógicas, com 360 *flip-flops*, 77 pinos de IO, e 3200 *bits* de memória RAM distribuídos na arquitetura. Pode operar em até 80MHz, as entradas e saídas têm funcionalidade *3-state* e é capaz de fornecer até 24mA por pino.

Há ainda uma característica muito importante na arquitetura da FPGA que são as linhas de GCK que são *Global Inputs*. Como o nome sugere, são linhas que podem

ser usadas como *clock* local pois podem interconectar diversos CLBs. A utilidade delas reside no fato de se poder descrever *drivers* síncronos para diversos *clocks* não relacionados. Assim, não se tem problema nenhum em descrever a SPI sincronamente com o sinal SPI-CLK e o PCM síncrono com PCM-CLK. A importância de descrever os hardwares sincronamente é que a descrição síncrona faz com que o *core* do *hardware* ocupe espaço muito menor na FPGA.

O que foi dito refere-se a apenas algumas das características da FPGA, o detalhamento completo desta tecnologia pode ser encontrada no *datasheet* do fabricante [28].

4.1 Ambiente de Desenvolvimento

Para desenvolvimento de aplicações com a FPGA, o fabricante fornece gratuitamente o *software* ISE [30]. Além do software, é possível encontrar a descrição de vários *cores* disponíveis para serem integrados na aplicação.

Há ainda outro *site* útil para obtenção de *cores* que pode ser integrado no desenvolvimento de aplicações, o www.opencores.org. Nele encontram-se drivers para SPI e I^2C dentre outros que facilitam a integração de blocos funcionais para rápido desenvolvimento das aplicações.

A programação da FPGA neste software se dá através dos seguintes meios: diagrama esquemático, no qual o usuário procura componentes na biblioteca do software e monta a aplicação como se estivesse interconectando componentes físicos, e através de linguagens de descrição de *hardware* como VHDL ou Verilog.

Além da possibilidade de projetar o *hardware*, o software também permite a configuração da FPGA através da interface JTAG. O protocolo e o procedimento necessários para a configuração utilizando esta interface são disponibilizados pelo fabricante em [26]. A Xilinx também desenvolveu uma interface simplificada para configuração da FPGA. O modelo desenvolvido tem por finalidade permitir ela seja configurada através de uma memória PROM junto a ela, ou por um processador. O fabricante disponibiliza um *application note* com descrição detalhada de procedimento para configurar a FPGA além de um programa escrito em C para microcontrolador da família 8051 [27] e [29]. Os pinos da interface requeridos estão na tabela 4.1.

Ainda sobre a interface, o pino DONE da FPGA é pino de *output* em modo normal, mas no período de configuração ele é *input*. É necessário utilizá-lo no período da configuração somente. Assim, escolheu-se colocá-lo em contato com uma saída de *output* da interface SERIAL-IN/PARALLEL-OUT. Isso é possível porque o DONE é implementado como coletor aberto e assim, basta-se colocar um resistor entre os pinos e não há nenhum risco de dano à interface.

Tabela 4.1: Pinos necessários para configuração SPARTAN-XL. Extraído de [27]

Signal	Type	Direction	Description
M0,M1	*Mode select*	*Input*	*Select Slave or Master Mode*
DIN	*Data*	*Input*	*Write configuration data in the Spartan device*
DOUT	*Data*	*Output*	*Read configuration data from the Spartan device*
CCLK	*Clock*	*Input*	*Syncronizes data on the rising edge*
nPROGRAM	*Control*	*Input*	*Begin clearing the Spartan configuration memory*
nINIT	*Status*	*Open-drain Output*	*A transition from Low to High indicates that the Spartan configuration memory is clear and ready to receive the bitstream*
DONE	*Status*	*Open-drain Output*	*A High indicates that the configuration process is complete*
HDC	*Status*	*Output*	*High troughout configuration, until the I/Os go active*
nLDC	*Status*	*Output*	*Low troughout configuration, until the I/Os go active*

O interesse em apresentar os temas relacionados à configuração da FPGA é porque esta é internamente baseada em RAM estáticas e, por isso, apaga quando desligada. Assim, todas as vezes que o sistema é reinicializado, a FPGA precisa ser novamente configurada. É uma restrição muito forte para a maioria das aplicações, por isso, deve-se desenvolver no ARM um *firmware* capaz de reconfigurar a FPGA todas as vezes que o sistema for reinicializado. O ARM poderá guardar o arquivo de configuração da SPARTAN na região de dados da sua memória FLASH interna.

4.2 Interfaces da FPGA

Esta seção apresenta as interfaces da FPGA determinadas neste projeto, descrevendo-as quando necessário. A tabela 4.2 resume as funcionalidades das entradas e saídas da SPARTAN na placa.

Tabela 4.2: Pinos de Interface da SPARTAN-XL no placa

Interface	Identificação	Pino nro.	Descrição
I^2C	I^2C-CLK	73	Barramento Serial I^2C
	I^2C-SDA	72	
SPI	IC-CLK	99	Barramento SPI com 5 canais de seleção
	IC-MISO	98	
	IC-MOSI	97	
	FPGA-CS[1..5]	92..96	
PCM	PCM-DIN	18	Interface PCM com Modem
	PCM-DOUT	19	
	PCM-FS	20	
	PCM-CLK	21	
JTAG	FPGA-TDI	4	Interface JTAG para configuração da SPARTAN
	FPGA-TCK	5	
	FPGA-TMS	6	
	FPGA-TDO	76	
PROG	HDC/IO	28	Interface para Configuração da SPARTAN via protocolo desenvolvido pela XILINX
	nINIT	36	
	DONE	50	
	nPROG	52	
	DIN	72	
	DOUT	73	
ENERGIA	nPWRDDWN	26	Altera modo para *Low-Power*
IO [1]	GCK[4..7]	—	*Clock* local
	IO[2]	—	IO padrão

[1] Numeração dos pinos pode ser consultada em A.5

As próximas seções não se referem à FPGA especificamte, mas estão ligadas a ela pela aplicação. A seção 4.3 define a interface de entrada e saída da plataforma, e a seção 4.4 o barramento I^2C. O motivo delas serem discutidas neste capítulo é que o barramento I^2C só está ligando FPGA e ARM, e a interface de saída é principalmente composta de linhas da FPGA.

4.3 A Interface Externa

A figura 4.3 mostra os 96 pinos de conexão da placa com o exterior. São 3 os principais grupos funcionais desta interface:

- Módulo de Energia:
  - 12 V
  - 5 V

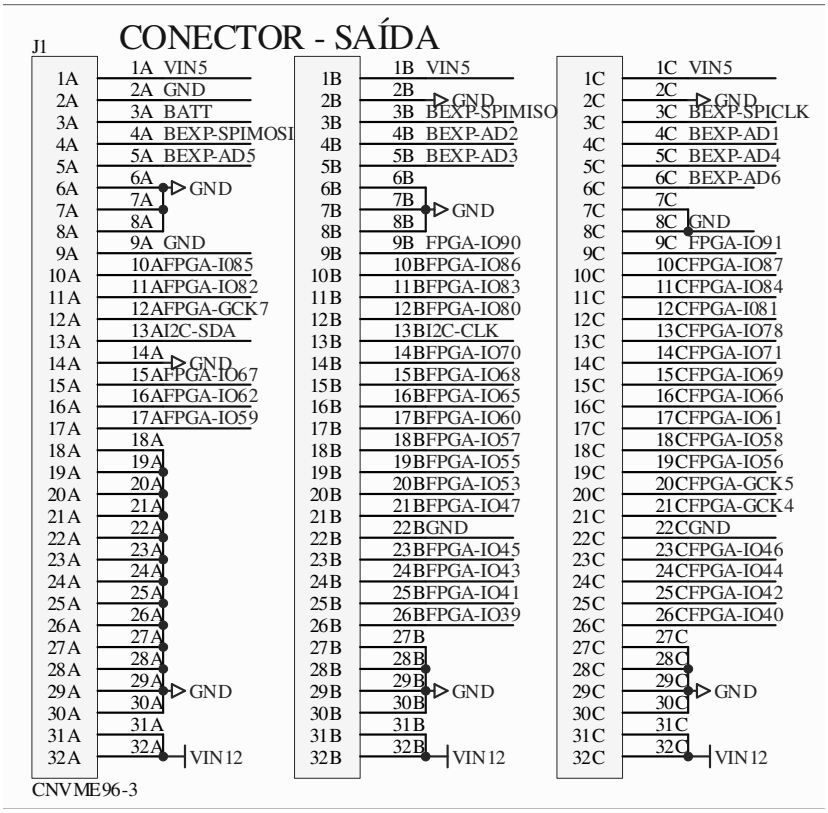

Figura 4.3: Conector de interface externa para a placa

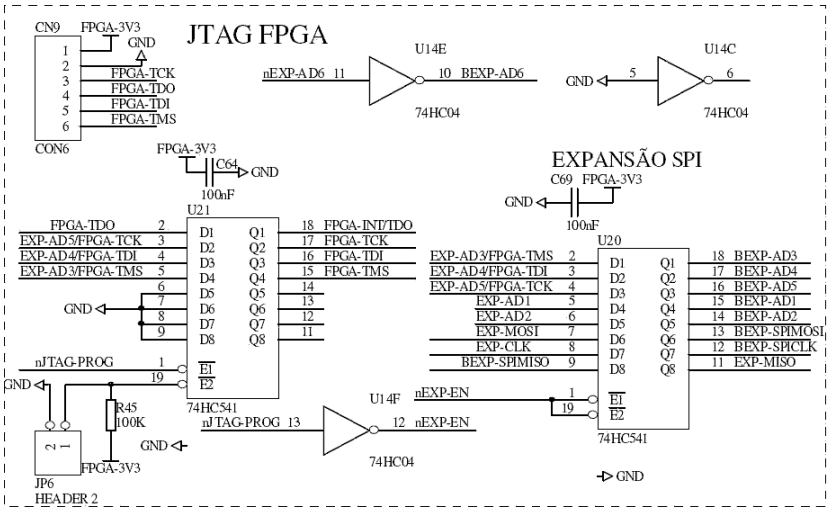

Figura 4.4: *Driver* para proteção e separação entre sinais internos e interface externa

  - GND

- Interface SPI com ARM:

  - Módulo SPI

  - 6 pinos de *output* do ARM com propósito geral.

Tabela 4.3: Modos de Operação dos *Buffers* de Saída

Modo de Operação	***Jumper***	**nJTAG-PROG**
Programação externa FPGA	com *jumper*	0
Barramento SPI	sem *jumper*	1
Não definido	Outras Combinações	

- Interface com a FPGA:
  - 39 pinos de IO de propósito geral
  - 4 pinos multiplexados IO/GCK, para linhas especiais de *clock*.

Como foi dito anteriormente, conexão direta entre os pinos do ARM e o exterior podem eventualmente danificar o processador em função de cargas estáticas. Para proteger os pinos do ARM, entre os pinos deste e a interface externa há um *buffer* 74HC541 [15]. Este *buffer* também tem pinos de controle que colocam a saída em alta impedância. Esta característica foi utilizada para permitir também a configuração externa da FPGA através da interface JTAG. Somente assim tanto o ARM como um PC podem ter acesso aos mesmo pinos de JTAG da FPGA. São dois os pinos de controle deste *buffer*, um controlado por *software* e outro controlado pelo *jumper* da placa (vide fig.4.4), os quais definem 2 modos de operação definidos na tabela 4.3.

4.4 I^2C

I^2C é um acrônimo para *Inter Integrated Circuit*, que é um barramento serial para computadores desenvolvido pela Phillips. As principais características deste barramento são:

- Barramento Síncrono Bidirecional
- 2 linhas de interface: *clock* (SCL) e dados (SDA)
- Cada CI tem um endereço único no barramento
- Sistema *multi-master*, cada master gera o seu próprio *Clock*.
- Baixa taxa de transmissão (máximo de 100Kbs)
- Detecção de Erro ou de Rede ocupada

Por ser um barramento *multi-master*, há necessidade de um protocolo que gerencie a utilização do barramento. É padronizado que antes de transmitir um dado, o CI primeiramente observa o estado da rede para verificar se está disponível (nível lógico 1). Caso esteja, então este transmite a sua mensagem. Caso ocorra colisão, os dois *masters* que transmitiram juntos interrompem a transmissão por um tempo aleatório e tentam retransmitir.

Além disso, para ter garantia de comunicação efetiva, para cada 8 *bits* enviados no barramento o mestre libera a linha para receber um ACK (*acknowledge*). Quando a linha está liberada ela deve estar em nível lógico 1. Para garantir isto, resistores de *pull-up* devem ser colocados no barramento.

No protocolo de envio de dados (fig. 4.5), há sempre 8 *bits* de endereçamento do CI a comunicar e depois vem os *bits* dados. Este pacote é precedido por um *start bit* e encerrado por um *stop bit*. A respeito do endereçamento dos CIs, o protocolo define o endereço 0x00, como o endereço para chamadas gerais. Estas chamadas são úteis para anunciar a entrada de um novo CI no barramento ou para que um CI possa identificar outros no mesmo barramento.

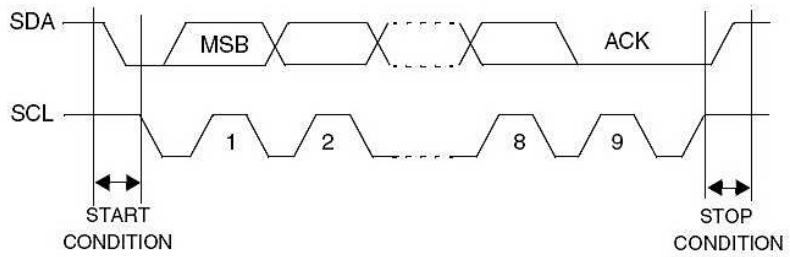

Figura 4.5: Formato das mensagens I^2C

Algumas considerações adicionais devem ser feitas para o *driver* I^2C implementado pelo ARM STR712. A primeira observação é que ele é capaz de operar em até 400kHz. O *hardware* é capaz de reconhecer seu próprio endereço e também chamadas gerais. Ele gera os ACKs autonomamente na recepção de dados. Pode ser configurado para interromper o processamento quando novos pacotes são recebidos ou em eventuais falhas na comunicação. Além disso, quando em transmissão, ele é capaz de reconhecer colisões, identifica a falta de ACK, e também o *loss arbitration* que é o caso quando o *master* tenta colocar a linha em nível 1 e não consegue, indicando que há outro CI segurando a linha em zero. Este é um sinal para o processador de que o *slave* com quem ele

está conversando é mais lento, ou simplesmente não está disponível para receber novas mensagens.

Escolheu-se colocar o barramento I^2C interligando FPGA e ARM para ser uma alternativa à comunicação via SPI. Assim, se houver uma aplicação com grande atividade sobre a *Ethernet* que impuser grande ocupação do barramento SPI, ainda se faz possível a comunicação entre o ARM e a FPGA. Para muitas aplicações, o desempenho do barramento I^2C não será limitante pois as duas tecnologias podem operar o barramento em 400kHz. Outro motivo para se propor a I^2C como alternativa é o fato de ser um barramento *multi-master*, o que permite à FPGA iniciar a comunicação sem a necessidade, como no caso da SPI, de uma linha a mais de interrupção.

Capítulo 5

Ethernet

A Ethernet é um barramento de rede baseado em *data frame*. Além da capacidade de criar uma rede local, através deste barramento é possível também ligar-se à rede *internet*. Assim, o sistema embarcado passa a ter a capacidade de ser monitorado, acionado e reprogramado em um computador distante conectado à *internet*.

A norma estabelece camadas (*layers*) que separam as diversas etapas da comunicação, desde a camada de aplicação até a camada física como ilustra a figura 5.1. Esta separação em camadas, permite que haja uma conexão virtual entre cada camada correspondente numa comunicação. Nesse caso, é como se camadas inferiores fossem transparentes. Cada uma destas etapas define a formação do *data frame*.

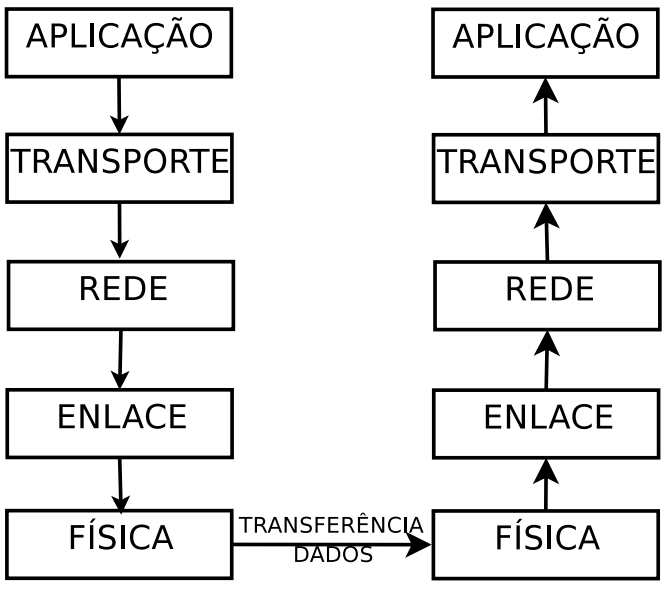

Figura 5.1: Camadas de Rede definida no padrão Ethernet

Para aplicações envolvendo a Ethernet é preciso que o desenvolvedor conheça as normas e a formação dos *data frames*. Para informações adicionais sobre redes de computadores e rede Ethernet aconselha-se a consulta ao Tanenbaum [24]. Além disso, a ST-Microeletronics disponibiliza um *application note* de uma implementação de TCP/IP, o protocolo de rede mais utilizado, para a família STR71x [20].

Para possibilitar o uso da rede Ethernet, utilizou-se o ENC28J60 [9] da Microchip que é um controlador de Ethernet autônomo com interface SPI. Abaixo, encontra-se uma lista das principais características deste controlador.

- Satisfaz os requisitos da norma IEE802.3
- Suporta uma porta 10BASE-T
- Implementa as camadas MAC (*Medium Access Controller*) e a camada física (PHY).
- Opera nos modos *Half* e *Full-Duplex*
- Tem *buffer* de 8 KBytes para recepção e transmissão de pacotes.
- Opções programáveis:
  - Detecção automática de erros e retransmissão de pacotes
  - Cálculo de CRC (*frame Check Sequence*), que são dois *bytes* calculados segundo um algoritmo que permite, na recepção do pacote, verificar se este não está corrompido. Pelo fato do algoritmo prejudicar o desempenho do ARM por causa da quantidade de cálculos que se tem de fazer, pode-se configurar o controlador para que o seu hardware especializado calcule este valor
  - Dimensão do *buffer* dedicado à recepção e o *buffer* dedicado à transmissão que são implementados como uma FIFO (*first-in first-out*) circular.
  - Registro de endereço MAC do dispositivo.
  - *Leds* que sinalizam a atividade e o *link* do dispositivo à rede Ethernet.
  - Seis linhas de interrupções
- Uma DMA (*Direct Memory Access*) que junto com o gerador automático de CRC facilitam a transmissão dos pacotes.

- Suporte para pacotes *Unicast*, *Multicast* e *Broadcast*, além da possibilidade de filtros para rejeitar pacotes indesejados automaticamente.

- Operação a 25MHz

A figura 5.2 mostra o diagrama de blocos esquemático do controlador da Ethernet. Os principais blocos funcionais são descritos a seguir:

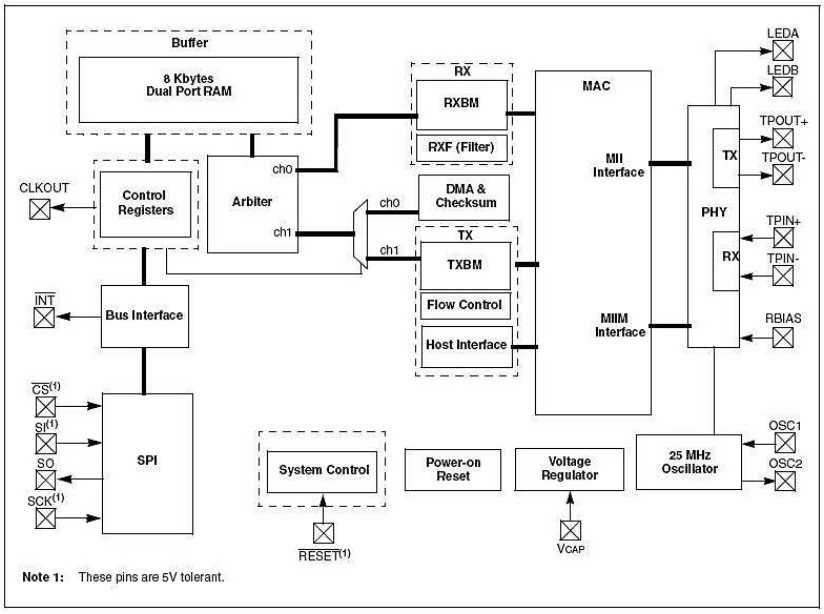

Figura 5.2: Diagrama de Blocos do Controlador de Ethernet ENC28J60. Figura extraída de [9]

- Uma interface SPI (modo 00) para comunicar com o controlador da Ethernet e o ARM.

- Vários registradores que configuram e determinam a operação do ENC28J60.

- RAM para recepção e transmissão dos *data frames*

- Circuito de Árbitro para determinar prioridade ao acesso da memória quando requisitada pela DMA, ou blocos de transmissão e recepção

- Uma interface que interpreta os comandos recebidos via SPI

- Módulo MAC (*Medium Access Control*) que implementa os requisitos da norma IEEE802.3

Tabela 5.1: Requisitos para o transformador de pulso da Ethernet. Extraído de [9]

Parâmetro	Min	Norm	Max	Unidade	Condições
Trafo RX	—	1:1	—	—	
Trafo TX	—	1:1	—	—	*Center Tap = 3.3V*
Inserção de Perda	0.0	0.6	1.1	dB	
Indutância do Primário	350	—	—	μH	Polarização: 8mA
Isolação	—	1.5	—	kV	
Rejeição de Modo Comum	40	—	—	dB	0.1 a 10MHz
Perda de Retorno	-16	—	—	dB	

- Uma camada física (PHY) que codifica e decodifica os dados presentes no par trançado

A rede Ethernet apresenta alta taxa de transmissão a longas distâncias, que podem produzir tensões induzidas e cargas estáticas capazes de causar danos aos componentes internos do circuito. Por isso, um transformador de isolação magnética deve ser posto logo na entrada, junto ao conector da rede. A Microchip disponibiliza as ligações necessárias para a utilização do controlador, que pode ser visto na figura 5.3. Além disso, há requisitos relacionados ao transformador de entrada que também são fornecidos pelo fabricante e foram copiados na tabela 5.1.

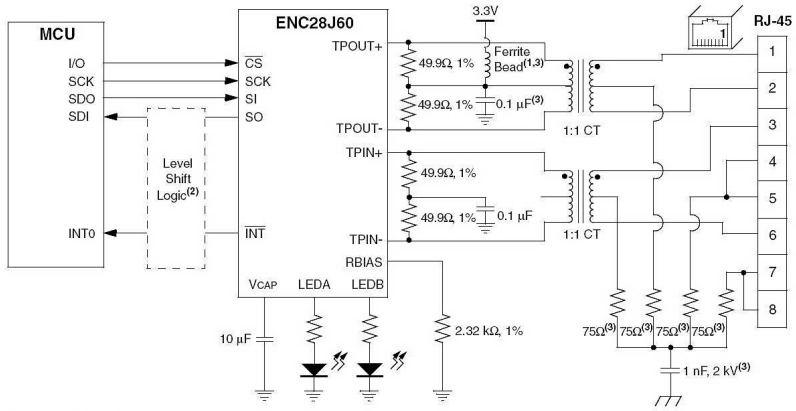

Figura 5.3: Ligações necessárias ao ENC28J60. Figura extraída de [9]

Por fim, a tabela 5.2 apresenta a interface entre o processador e o controlador, além disso, o esquemático da Ethernet pode ser observado na figura A.6 do anexo A.

Tabela 5.2: Interface do controle do ENC28J60

Linha	**Descrição**
IC-MOSI	*Master-Output Slave-Input*
IC-MISO	*Master-Input Slave-Output*
IC-CLK	*Clock* do barramento SPI
ETH-CS	*Chip-Select*
ETH-INT	Linha responsável por gerar interrupção no microprocessador.

Capítulo 6

Outras Interfaces da Plataforma

Este capítulo dedica-se a algumas interfaces de comunicação presentes na plataforma que ainda não foram apresentadas. A primeira seção apresenta interface RS-232, e a segunda seção é dedicada ao barramento CAN.

6.1 RS232 - UART

A interface RS232 é um padrão de troca de dados binários serialmente ponto a ponto entre um dispositivo DTE e um DCE. Na verdade o que foi implementado na plataforma é uma UART (*Universal Asynchronous receiver/transmiter*) usando a via de comunicação da RS232 do computador. Assim, a UART define o que enviar ao passo que o padrão RS232 se incumbe em como enviar.

Para ser capaz de utilizar a via RS232 foi necessário utilizar o *transceiver* MAX3232 que são responsáveis por impor os níveis adequados de tensão para a comunicação. Pois, para o padrão RS232, nível lógico 0 é tensão entre 5 e 25 V, e nível lógico 1 entre -5 e -25V, na transmissão. Esta transposição de níveis é feita pelo *transceiver*, que apresenta uma interface bem simples, e exige apenas que alguns capacitores sejam colocados em alguns de seus pinos. O circuito pode ser consultado na figura A.6 do anexo A. A tabela 6.1 apresenta a interface controlável pelo ARM da comunicação via RS-232.

Tabela 6.1: Interface entre RS232 e ARM

Linha	**Descrição**
RS232-RX	Recepção
RS232-TX	Transmissão

6.2 CAN - *Controller Area Network*

Num único carro é possível encontrar vários dispositivos com processamento. A ignição eletrônica, os freios ABS, a suspensão eletrônica, a transmissão eletrônica, luz automática, ar-condicionado, sistema de segurança, etc. Esses são apenas alguns exemplos de subsistemas processados internos a um carro. Mas, quando em operação, estes sistemas não são absolutamente independentes, antes é preciso integrá-los.

Há porém algumas particularidades num sistema veicular que são limitantes fortes para muitos barramentos de redes. Carros são ambientes extremamente ruidosos, os motores elétricos, o sistema de ignição que chega a consumir picos de corrente de 200A gerando tensões induzidas, e também a bateria do carro que chega a ter transientes de até 400V. Portanto, um sistema que interligue componentes eletrônicos do carro deve ser totalmente confiável por estar relacionado à questões de segurança dos usuários do veículo.

Para resolver este problema, a Bosch desenvolveu o *Controlle Area Network* (CAN), cujas características principais são:

- Alta imunidade ao ruído.
- Capacidade distribuída de detecção de erros e retransmissão de pacotes.
- Mensagens pequenas (Máximo de 8 *bytes*)
- Alta confiabilidade. A probabilidade de ocorrência de que um erro na recepção não seja detectado é menor que $4,7 \times 10^{-11}$.
- Sistema *multi-master*
- Barramento rápido (até 1 Mbps em 40 metros)
- Barramento hierárquico. Isso significa que o barramento define prioridade para os elementos da rede. Através deste sistema de hierarquia, módulos com maior prioridade são escutados primeiro. Isso é também uma exigência de um barramento que atenda as necessidades de um ambiente veicular. Pois é lógico que se deseja uma precedência num sistema onde possa ocorrer do AIR-BAG e do *player* do carro quererem usar o barramento ao mesmo tempo.

O sistema de prioridade se dá através da característica do barramento. O barramento é de característica AND, assim, 0 é dominante, ou seja, se 2 dispositivos quiserem colocar no barramento 1 e 0 ao mesmo tempo, no barramento terá 0. Assim, faz-se com que sistemas para os quais se queira maior prioridade tenham maior quantidade de 0s em seu endereço. Na transmissão, além de transmitir o dado, o dispositivo também verifica qual é o valor que ficou na rede, quando o que ele verifica e o que ele escreveu é diferente ele identifica que alguém com maior prioridade quer utilizar o barramento e o deixa livre, e voltará a transmitir apenas quando o barramento ¡estiver livre.

Estas características fizeram com que o CAN ganhasse os mais diversos ambientes, sendo o barramento presente na automação industrial, em trens e navios, sistemas de controles, instrumentos médicos, elevadores e etc. Pelo fato de se pensar na utilização desta plataforma para controle e acionamento de motores, escolheu-se este barramento.

O *driver* desenvolvido pela STMicroeletronics para o STR712 apresenta as seguintes características:

- Implementa protocolo 2.0 do CAN
- Implementa uma FIFO para envio de mensagens com até 32 níveis
- Retransmissão programável
- Detecção dos seguintes erros:
  - *Stuff Error* - Mais de 5 *bits* iguais em trechos da mensagem onde isso não é permitido
  - *Form Error* - Erro no formato da mensagem
  - *ACKError* - Falha no recebimento de ACK
  - *Bit1Error* - Identifica que outro de maior prioridade está usando a rede
  - *CRCError* - Erro na mensagem recebida
- 2 interfaces de Registros para evitar conflitos ente retransmissão e novas mensagens por escrever
- Administra até 32 mensagens para transmitir

Tabela 6.2: Interface entre rede CAN e ARM

Linha	**Descrição**
CAN-RX	Recepção
CAN-TX	Transmissão

6.3 Circuito para o CAN

A interface entre o ARM e o CAN é bastante simples e está descrita na tabela 6.2. Mas, para propor um circuito que satisfaça os requisitos necessários para a operação no barramento, é necessário observar algumas características. Por se tratar de um barramento de alta frequência a grandes distâncias, faz-se também necessário um circuito de isolação para o barramento. Diferentemente do circuito da Ethernet, para o qual o acoplamento magnético é norma, para o CAN isto não é definido. Na placa escolheu-se fazer o acoplamento ótico, proporcionado pelo HCPL0611 [6] (fig. 6.1). As principais características deste acoplador é listada abaixo, destacando-se a frequência de operação que satisfaz a necessidade de *baud-rate* do barramento CAN.

- Rejeição de Modo Comum: 10kV/μs para $V_{CM} = 50V$
- Alta frequência: 10Mbps
- Baixa corrente de polarização, alcançando 5mA.

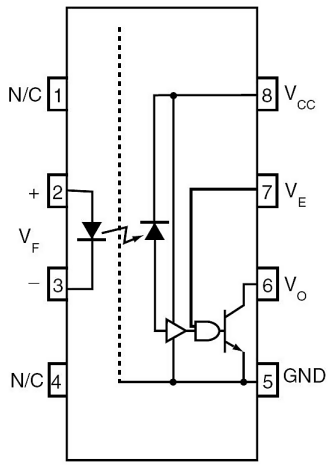

Figura 6.1: Diagrama esquemático do Optoacoplador. Extraído de [6]

Para um desempenho satisfatório, o acoplador drena deve drenar corrente de 10mA. Para evitar que esta corrente circule pelos pinos do ARM utilizou-se um transistor P-MOS (FDV302) na configuração de chave, ligado quando CAN-TX igual a 0 (fig. 6.2).

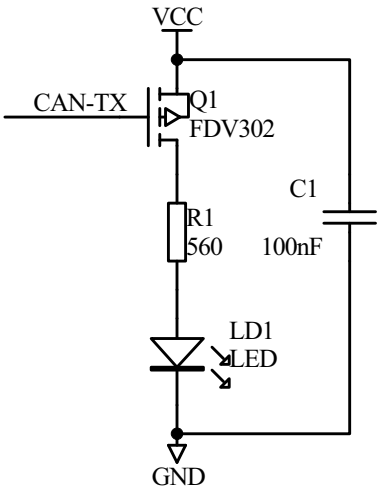

Figura 6.2: Circuito para Acionamento do acoplador óptico

Além do isolador, um *transceiver* é também necessário para atender aos níveis de tensão do padrão CAN. Utilizou-se o SN65HVD230. Para além de operação normal, nível lógico 1 no pino RS do CI o coloca em *sleep mode*.

Por fim, uma característica importante do barramento CAN é a exigência de resistores de terminação para casamento de impedância (120 Ω). Não há porém uma posição relativa preferencial para esta plataforma, assim, não foi incluído no projeto os resistores de terminação. Mas, ao operar o barramento, este requisito deve ser observado para o funcionamento satisfatório da rede.

Capítulo 7

Implementações

Neste capítulo são apresentadas algumas das possibilidades presentes na plataforma proposta que foram imaginadas no período de desenvolvimento. Estas propostas estão aqui para, por um lado mostrar as potencialidades da plataforma desenvolvida, e por outro para servirem de ponto de partida para os próximos alunos do laboratório que desejarem implementar *firmwares* para a plataforma.

7.1 Integração com uma Placa de Aquisição de Dados

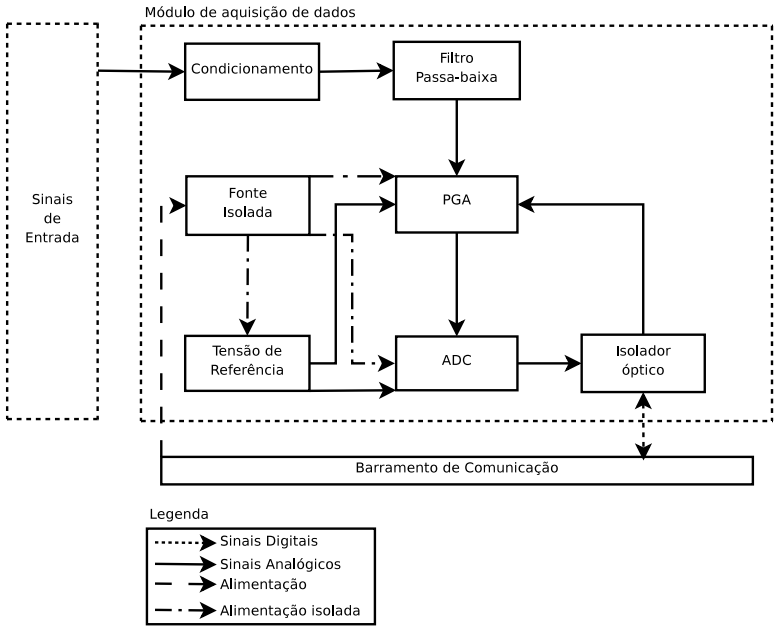

Figura 7.1: Diagrama de blocos do módulo de aquisição de dados

Foi desenvolvido pelo aluno Renato Machado Monaro da EESC-USP como trabalho de conclusão de curso, uma placa de aquisição de dados [10]. Esta placa é capaz de fazer leitura de tensão e corrente numa linha e a placa é interfaceada por um barramento SPI, o diagrama de blocos desta proposta é mostrado na Fig.7.1.

O processo de leitura ocorre da seguinte maneira: cada sinal de entrada (tensão ou corrente) é previamente filtrado, depois condicionado por um amplificador programável e convertido para digital pelo ADC, o sinal digital passa então por isoladores ópticos e é então enviado ao barramento de comunicação pela interface SPI.

Tanto o amplificador programável como o conversor usam o barramento SPI para comunicação, para cada placa, há 2 módulos de leitura e 4 canais de seleção.

A primeira aplicação possível para a plataforma desenvolvida é controlar alguns módulos de aquisição de dados. Por este motivo, a interface externa tem uma SPI dedicada e mais seis linhas de endereçamento que são capazes de endereçar até 32 canais, ou 8 módulos de leitura de corrente e tensão.

Esta aplicação é de grande interesse para o LACEP pois, por exemplo, possibilita a monitoração de correntes e tensões na linha de um motor trifásico. Isto requer três módulos de aquisição, e usa 12 canais de endereçamento. Controle e acionamento de motores é uma das grandes áreas de pesquisa do laboratório, e esta aplicação possibilita o levantamento de dados experimentais para auxiliar nas pesquisas.

A figura 7.2 apresenta o diagrama de bloco desta aplicação.

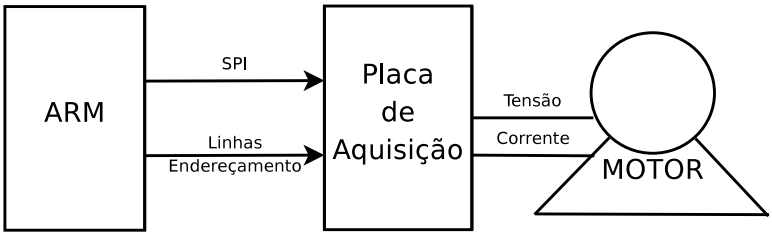

Figura 7.2: Diagrama esquemático da integração com a placa de aquisição de dados

7.2 Controle e Acionamento de Motores

Através da placa de aquisição de dados e de *hardwares* descritos na FPGA é possível desenvolver uma plataforma de controle de motores trifásicos para auxiliar validações

e testes das teorias desenvolvidas nas pesquisas do LACEP.

O projeto imaginado, cujo diagrama está na figura 7.3, conta com os seguintes blocos funcionais:

- Processamento central - ARM
- Tensões e Correntes da Linha - Placa de Aquisição de Dados
- Leitor de *Encoder* para ter a velocidade angular do motor - FPGA
- Gerador de PWM para o módulo de potência - FPGA

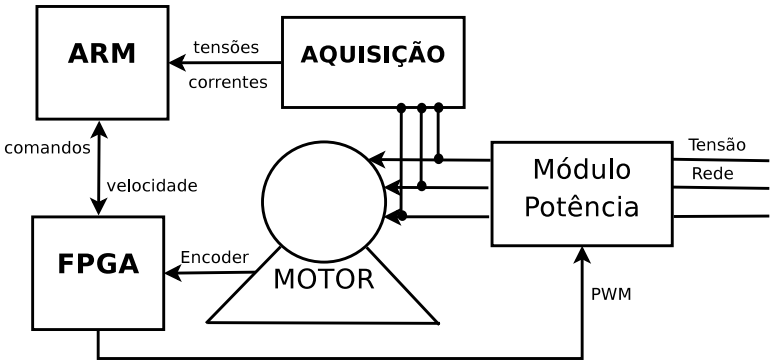

Figura 7.3: Diagrama esquemático do controlador de motores trifásicos

7.3 *CallBox*

Através do Modem G24 é possível receber ligações que usam a rede GSM. Esta plataforma pode receber as ligações e transferi-las para um computador através da rede Ethernet.

Esta implementação exige que as seguintes interfaces sejam desenvolvidas:

- Implementação de pilha UDP/IP através do ARM e do ENC28J60. É sugerido a camada de transporte UDP em substituição à TCP por dois motivos:
  - Maior simplicidade na implementação da camada de transporte UDP.
  - Em protocolos de comunicação de voz, há um interesse muito maior nas temporizações dos *data frames* do que na integridade dos dados. Isso é porque pequenos cortes e falhas numa conversa por telefone é muito mais aceitável do que atrasos na comunicação.

- Descrição do *driver* PCM na FPGA para transformar o protocolo de comunicação do modem com um protocolo válido para o ARM (SPI ou I^2C).
- Controle do modem para atender e realizar chamadas

A figura 7.4 mostra o diagrama esquemático desta implementação.

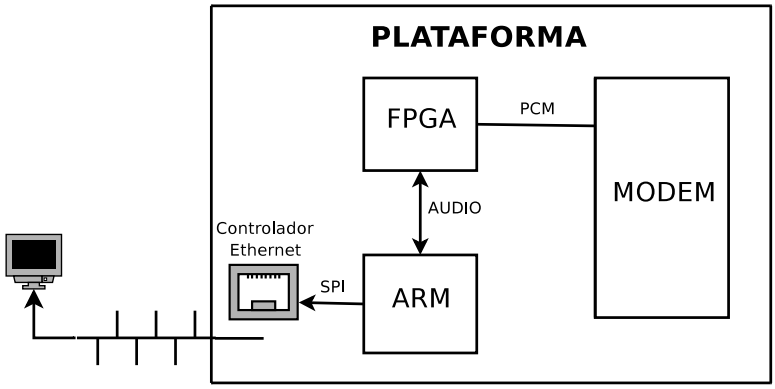

Figura 7.4: Diagrama esquemático do *CallBox*

7.4 Celular

Recentemente, foi lançado um projeto de Celular com código aberto. Uma aliança entre várias operadoras de celular, fabricantes de aparelhos, empresas de softwares e fabricantes de semicondutores foi estabelecida para o desenvolvimento de um celular *open core* sobre plataforma Linux. O projeto leva o nome de Android [1].

O compromisso estabelecido desta aliança é disponibilizar um conjunto completo de *softwares* abertos para aplicações para celular. Como foi dito no capítulo 3, o modem GSM integra todas as funcionalidades de um celular, assim, a placa também poderá ser utilizada para envolver desenvolvedores locais nesta aliança. As exigências para esta implementação são:

- Instalar o *kernel* Linux no ARM
- Configurar a FPGA para controlar *display* e teclado.

A figura 7.5 apresenta um diagrama esquemático de uma alternativa para esta implementação.

Estas são apenas algumas das funcionalidades que foram planejadas desde a fase de projeto da placa, mas é provável que muitas outras aplicações possam ser implementadas.

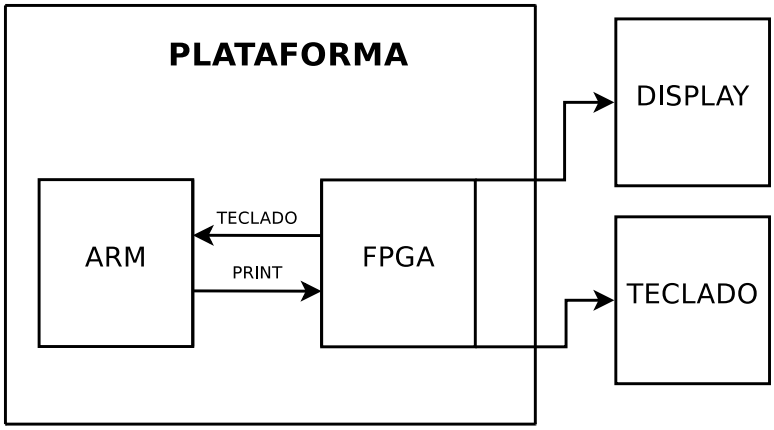

Figura 7.5: Diagrama esquemático do Celular

Parte III

Conclusões

Capítulo 8

Conclusões

A plataforma desenvolvida permite ao laboratório LACEP e, por extensão, ao SEL a oportunidade de estarem envolvidos com uma tecnologia de ponta e que é referência mundial na área de microcontroladores de 32 *bits*, o ARM. Como foi visto, a tendência atual das tecnologias de arquiteturas microprocessadas é a migração para microprocessadores de 32 *bits*.

Um dos motivos apresentados para a migração para as arquiteturas de 32 *bits* é que além do ganho de desempenho há um grande ganho na padronização de arquiteturas que facilitam a integração de subsistemas processados. Integração de sistemas implica em comunicação entre eles, e a plataforma permite o desenvolvimento em no mínimo 3 dos principais padrões de rede atualmente, são eles: CAN, Ethernet, e GSM. Além da mais tradicional RS232.

Embora seja uma placa de desenvolvimento e não tenha sido projetada para uma única aplicação, algumas possibilidades foram consideradas e a implementação de algumas destas possibilidades se constituirá num importante auxílio para o desenvolvimento das principais pesquisas do laboratório LACEP, que financia o projeto.

Outra característica importante no projeto desta plataforma foi o aprendizado em diretrizes gerais e teorias relacionadas ao projeto de *hardware*. Foi visto a importância de elaborar um projeto seguro que leve em conta ruído, isolação, dimensão de trilhas, técnicas de roteamento para garantir que o sistema final funcione como foi projetado. Além destes parâmetros, é da maior importância a verificação e consulta de todos os componentes do circuito e as normas relacionadas aos barramentos de comunicação

entre os módulos. A não observância desta regra geralmente compromete todo o desempenho do *hardware* com problemas de difícil identificação, podendo às vezes, até deteriorar permanentemente alguns componentes do circuito.

Parte IV

Apêndices

Apêndice A

Esquemáticos da Placa

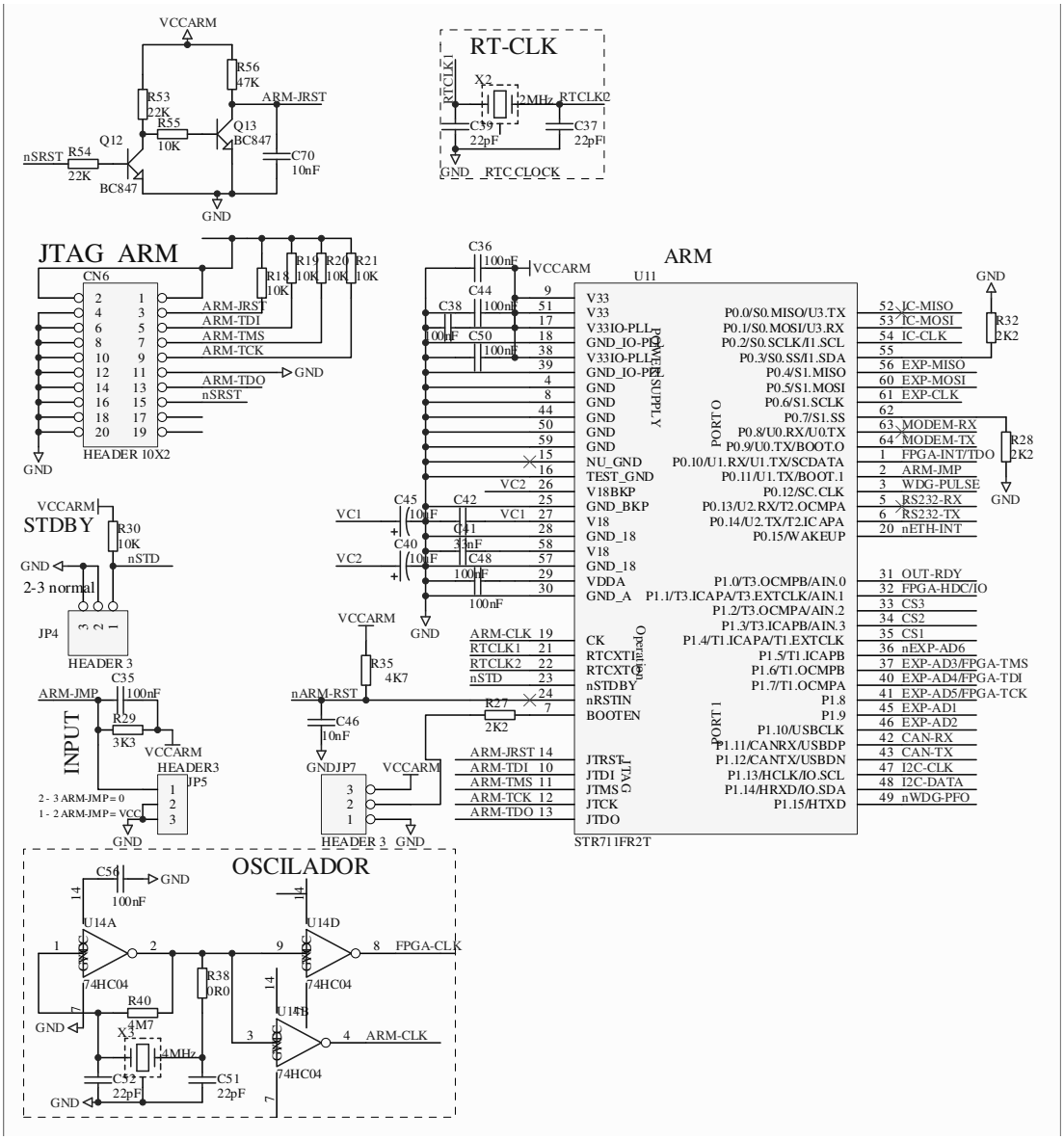

Figura A.1: Esquemático da núcleo central da Placa

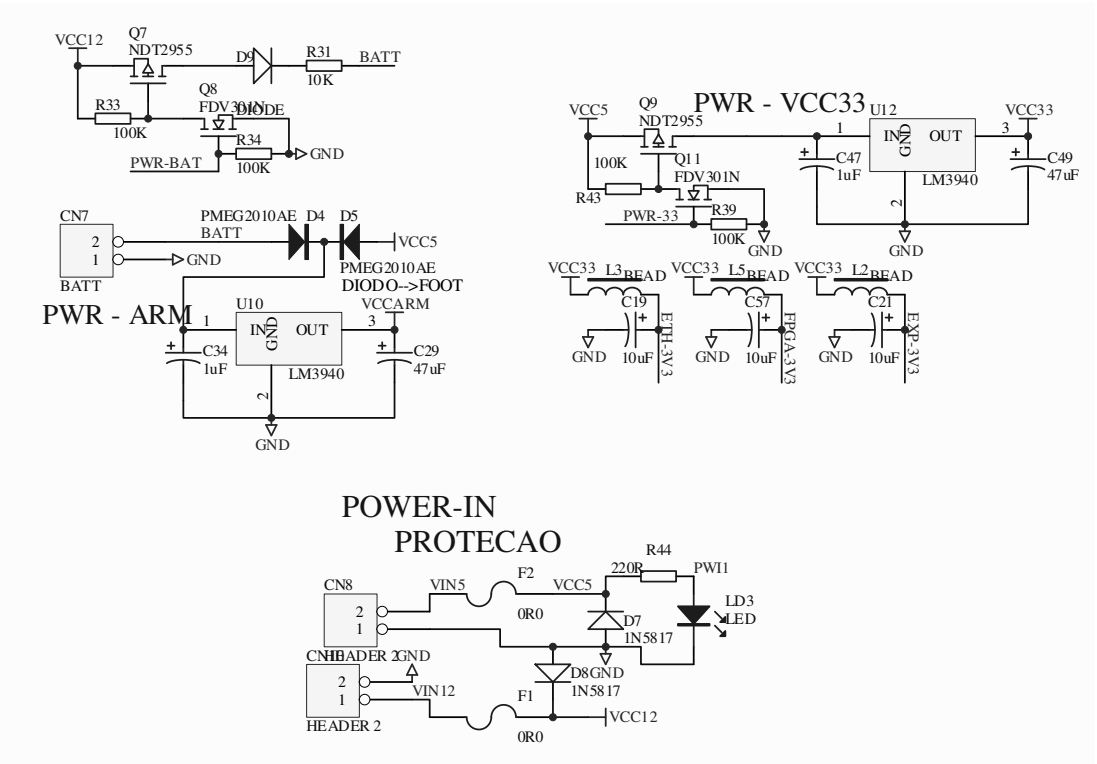

Figura A.2: Circuito de Alimentação - Retifificadores

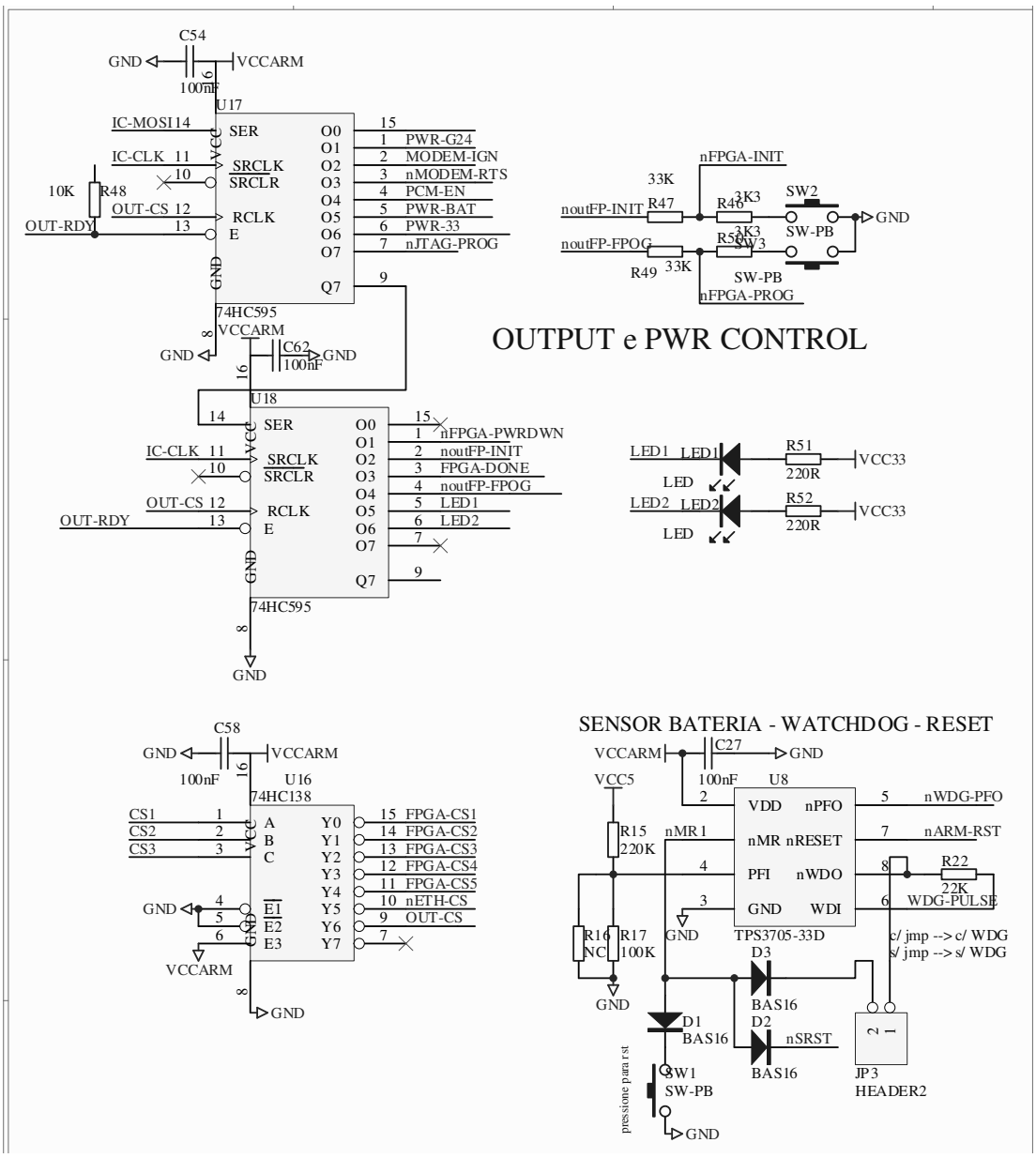

Figura A.3: Saídas e Circuito Supervisor

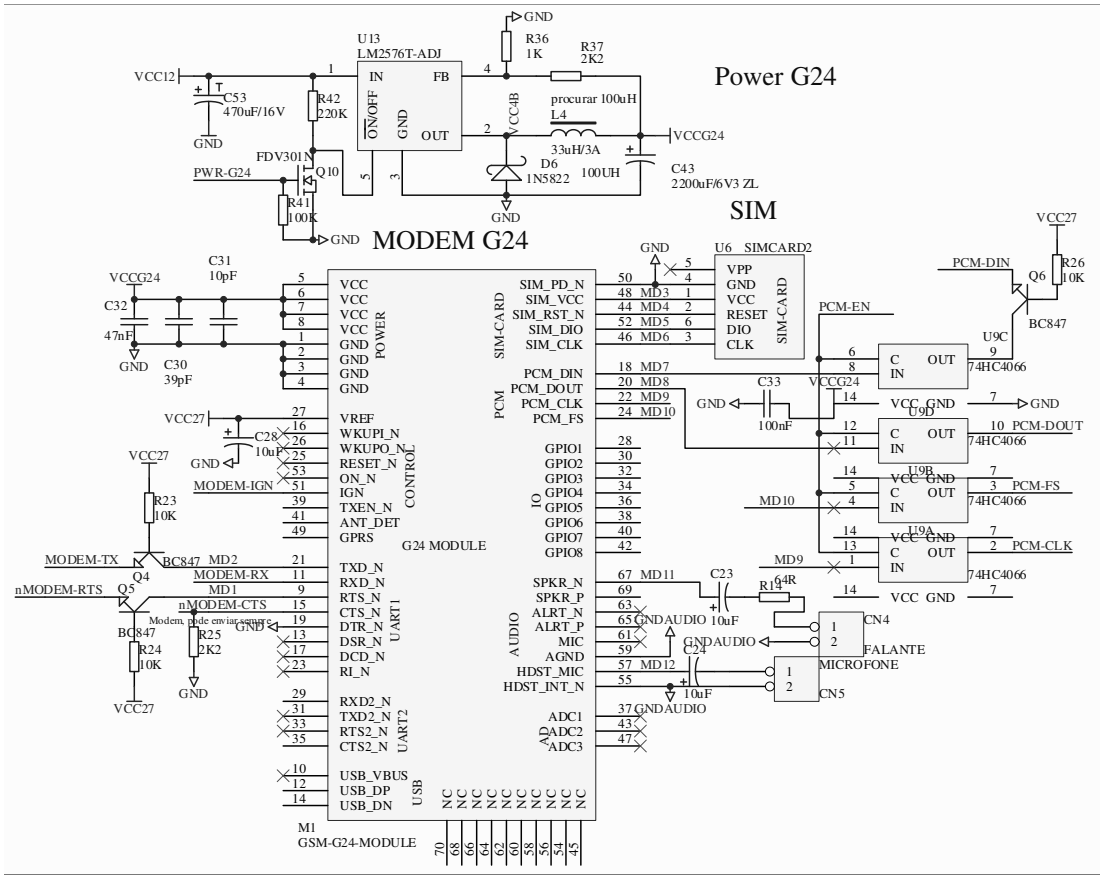

Figura A.4: Esquemático do Modem

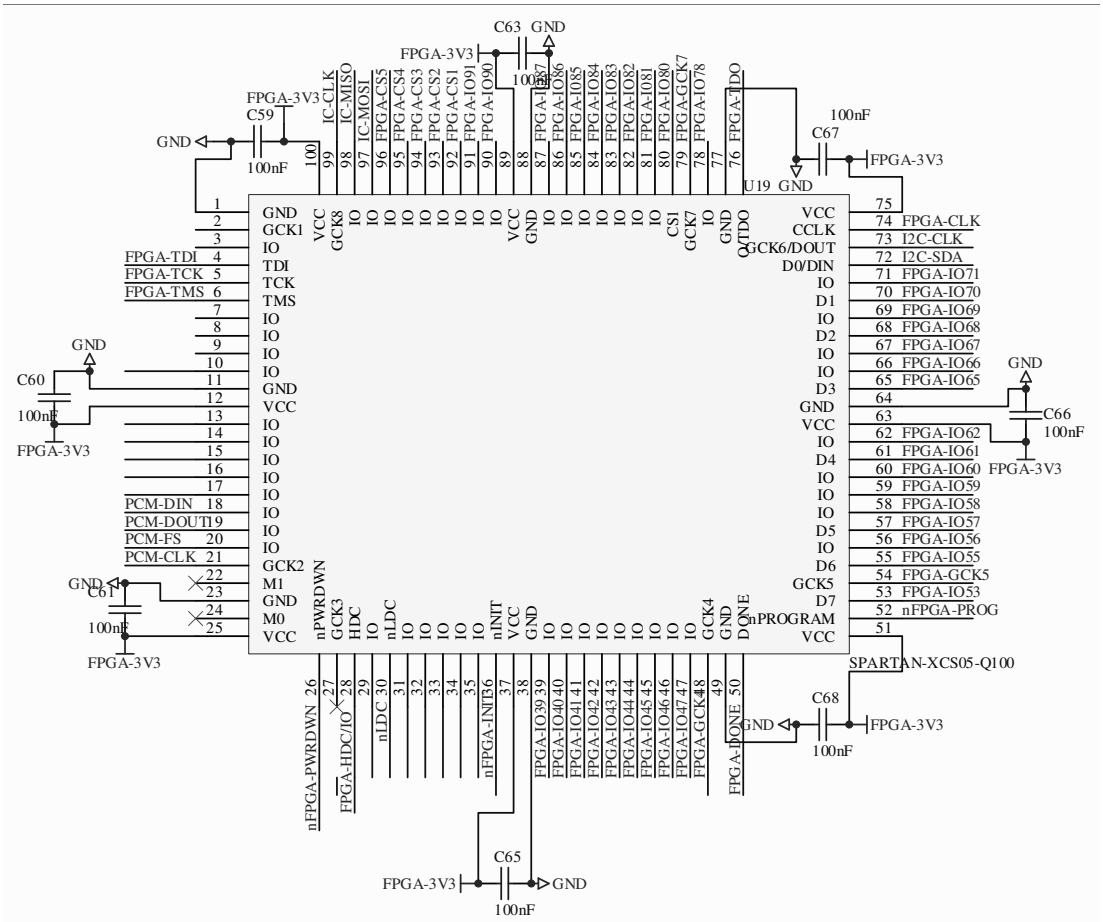

Figura A.5: Esquemático da FPGA

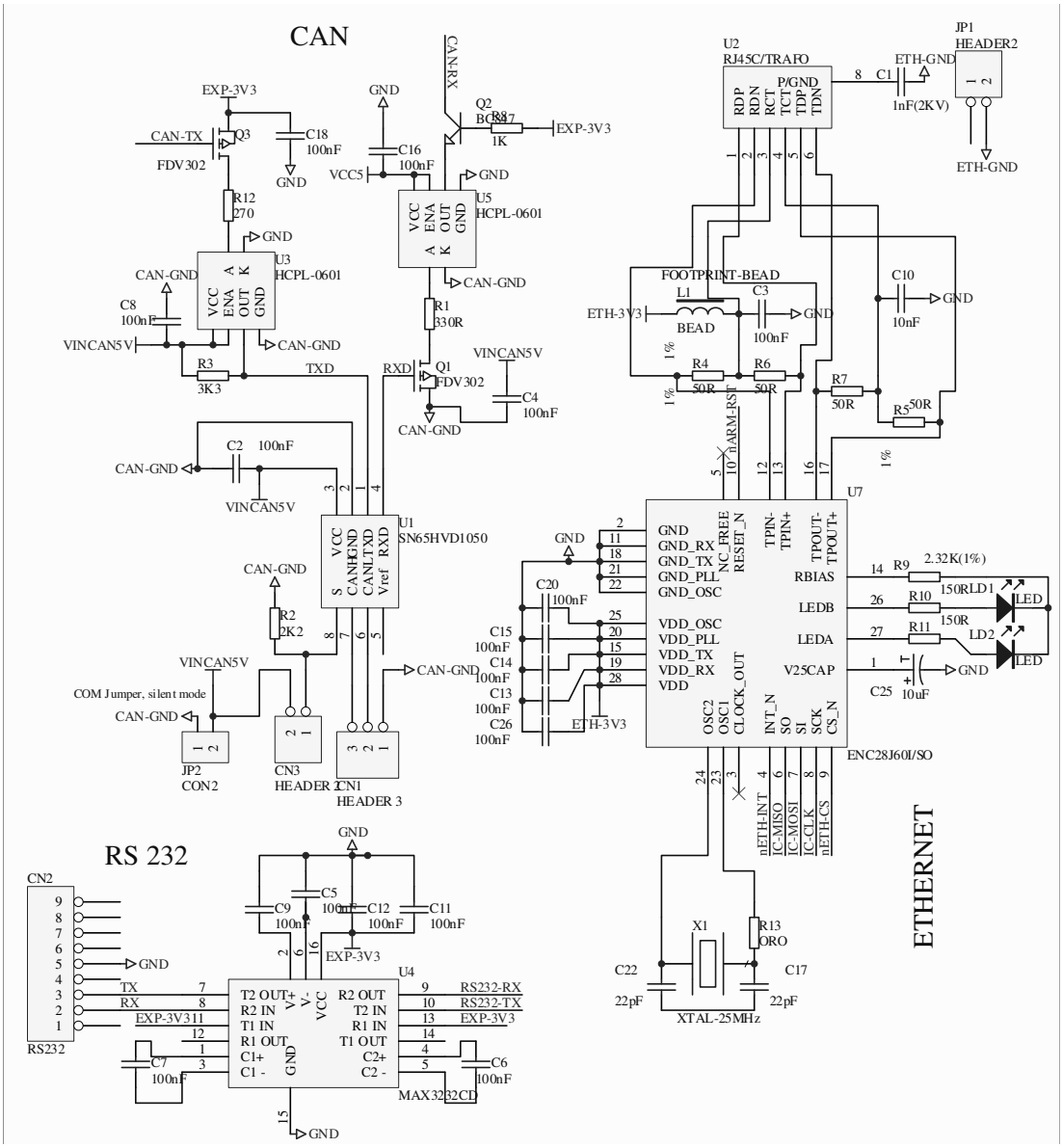

Figura A.6: Esquemático demais periféricos

Apêndice B

Layout da Placa

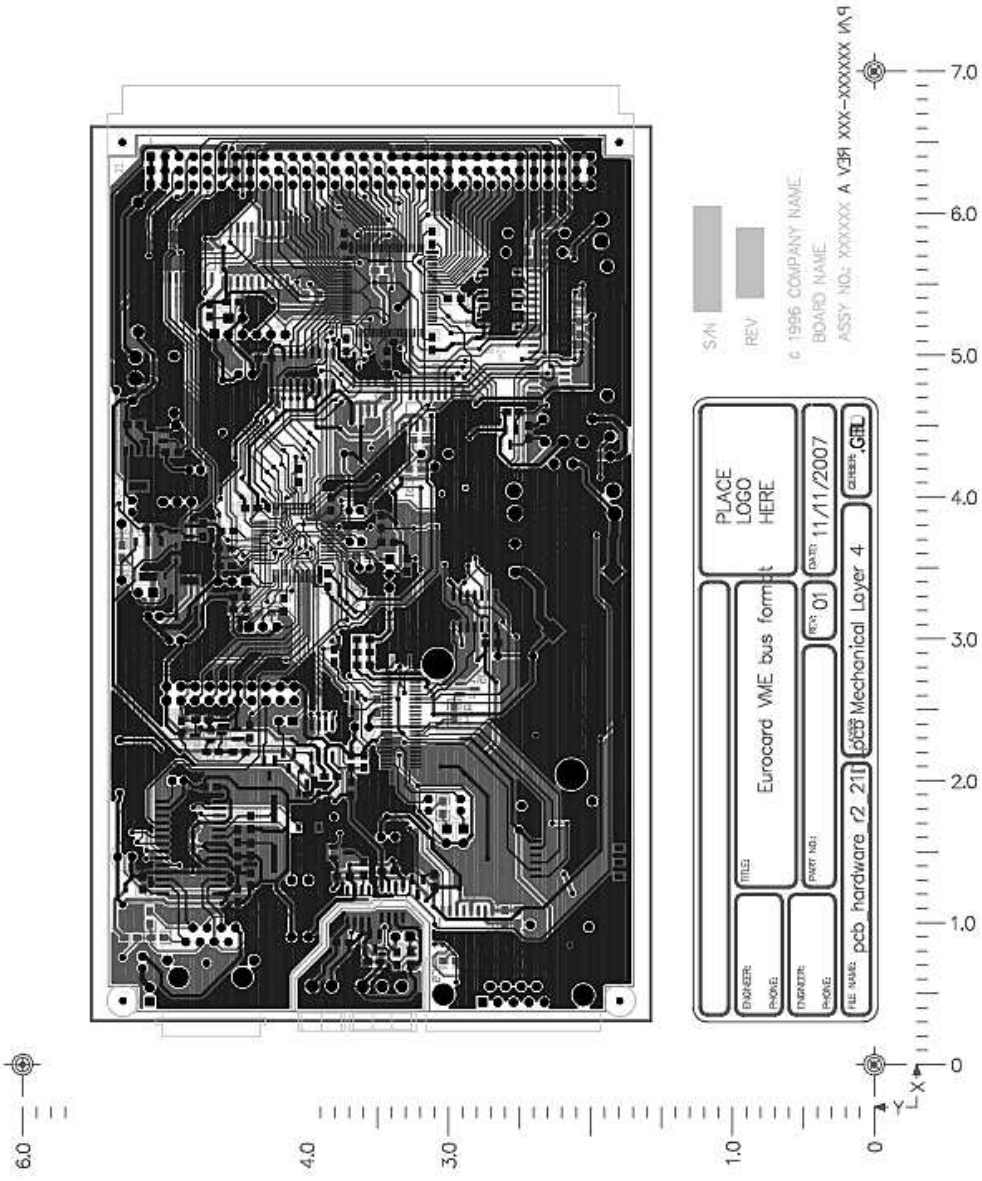

Figura B.1: PCB da placa

Parte V

Referências

Referências Bibliográficas

[1] Android^{TM}. Open handset alliance. URL: http://www.openhandsetalliance.com, 2007.

[2] ARM Ltd. Enterprise solutions. URL: http://www.arm.com/markets/. Acessado em Nov/07, 2005.

[3] ARM Ltd. Operating system support overview. URL: http://www.arm.com/-products/os/index.html. Acessado em Nov/07, 2006.

[4] ARM Ltd. Products & solutions home page. URL: http://www.arm.com/products/. Acessado em Nov/07, 2006.

[5] Catsoulis, J. *Designing Embedded Hardware*. O'Reilly, 2003.

[6] Fairchild. Hcpl0611 datasheet: High-speed 10mbps logic gate optocouplers. Technical report, 2007.

[7] Furber, S. *ARM system-on-chip architecture*. Addison-Wesley, 2000.

[8] Kocik, P.B. Gnu arm toolchain for cygwin, linux and macos. URL: http://www.gnuarm.com, 2006.

[9] Microchip. Enc28j60 datasheet: Stand-alone ethernet controller with spi interface. Technical report, 2006.

[10] Monaro, R.M. Sistema de aquisição de dados para um relé de proteção digital de baixo custo. 2007.

[11] Motorola. Motorola g24 developer's guide : At commands reference manual. Technical report, 2006.

[12] Motorola. Motorola g24 developer's guide : Module hardware description. Technical report, 2006.

[13] Nass, R. 8-bit mcu leader jumps to 32 bits. URL: http://www.embedded.com/-design/multicore/202802476?_requestid192089. Acessado em Dez/07, 2007.

[14] National Semiconductor. Datasheet: Lm2576 - simple switcher 3a step-down voltage regulator. Technical report, 2004.

[15] S. Philips. Datasheet 74hct541 - octal buffer driver; 3 state. Technical report, 1990.

[16] Philips, S. Datasheet: 74hc4066 - quad bilateral switches. Technical report, 1998.

[17] Phillips, S. . Datasheet: 74hc595/74hct595 - 8 bit serial-in, serial or parallel-out. Technical report, 2003.

[18] Rath, D. Open on-chip debbuger. URL: http://openfacts.berlios.de/index-en.phtml?title=Open_On-Chip_Debugger. Acessado em Set/07, 2007.

[19] STMicroeletronics. Datasheet 74hct541 - octal buffer driver; 3 state. Technical report, 1990.

[20] STMicroeletronics. Application note: Tcp/ip over ethernet conectivity with the str710. URL: http://www.st.com/stonline/products/literature/an/11948.pdf. Acessado em Nov/07, 2006.

[21] STMicroeletronics. Str71x firmware library. URL:http://www.st.com/stonline/products/literature/um/10780.pdf. Acessado em Nov/07, 2006.

[22] STMicroeletronics. Datasheet str71xf. Technical report, 2007.

[23] STMicroeletronics. Documents and files for family str7. URL: http://www.st.com/mcu/familiesdocs-86.html#Datasheet. Acessado em Dez/07, 2007.

[24] Tanenbaum,A.S. *Redes de Computadores.* Campus/Elsevier, 2003.

[25] Texas, I. Datasheet: Tps3705-33 - processor supervisory circuits with power-fail. Technical report, 1999.

[26] Xilinx. Boundary-scan in xc4000, spartan and xc5200 series devices. Technical report, 1999.

[27] Xilinx. The low-cost, efficiente serial configuration of spartan fpgas. Technical report, 1999.

[28] Xilinx. Datasheet: Spartan and spartan-xl families field programable gate arrays. Technical report, 2002.

[29] Xilinx. Xilinx in-system programming usin an embedded microcontroler. Technical report, 2004.

[30] Xilinx. Ise: Developing tool. URL: http://www.xilinx.com, 2007.