



UNIVERSIDADE LUTERANA DO BRASIL

PRÓ-REITORIA DE GRADUAÇÃO

DEPARTAMENTO DE ENGENHARIA ELÉTRICA



Aquisição de dados por uma Rede sem Fio

Reginaldo Marin

Canoas, Junho de 2007



Reginaldo Marin
Matrícula: 052000371-8

Aquisição de dados por uma Rede sem Fio

Trabalho de Conclusão de Curso
apresentado ao Departamento de
Engenharia Elétrica da ULBRA como um
dos requisitos obrigatórios para a
obtenção do grau de Engenheiro
Eletricista

Professor Orientador:

MSc. Eng. Eletr. Paulo César Cardoso Godoy – CREA-RS: 11.682-2

Comissão Avaliadora:

MSc. Eng. Eletr. Augusto Alexandre Durgante de Mattos – CREA-RS: 88.003-D

MSc. Eng. Eletr. Dalton Luiz Rech Vidor – CREA-RS: 079.005-D

Autor
Reginaldo Marin

Orientador
Paulo César Cardoso Godoy

Avaliador
Augusto Alexandre Durgante
de Mattos

Avaliador
Dalton Luiz Rech Vidor

Relatório Aprovado em:



DEDICATÓRIA

Dedico aos meus pais Reinaldo e Regiane, à minha namorada Aline, e agradeço a todos pelo apoio e compreensão.



AGRADECIMENTOS

A todos que colaboraram direta ou indiretamente na elaboração deste trabalho, o meu reconhecimento.

Aos professores Eng. Eletr. Luis Fernando Espinosa Cocian, o Eng. Eletr. Paulo César Godoy, o Eng. Eletr. Augusto Alexandre Durgante de Mattos e o Eng. Eletr. Dalton Luiz Rech Vidor pelo estímulo, dedicação e esforço pessoal proporcionado.

Aos inúmeros colegas pelas sugestões e observações valiosas.

Aos meus pais pelo incentivo, colaboração e apoio dado para vencer mais esta etapa.

Aos colegas de empresa Eng. Eletr. Gustavo Adriano Rocha, Eng. Eletr. Gelson Batista e ao Eng. Mec. Volnei Alfredo Giacomini pelas contribuições de grande valia.



RESUMO

O projeto consiste em realizar a implementação em hardware e software do sistema de aquisição de dados sem fio de um conjunto de grandezas analógicas. A transmissão é feita via um link de radiofrequência a um microcomputador para a visualização e armazenamento dos dados. Este sistema é composto de três módulos. No módulo transmissor (microprocessador + Módulo TRF-2.4G) o sinal é convertido pelo microcontrolador e enviado ao módulo TRF-2.4G. O sinal é então recebido pelo módulo receptor e novamente adequado ao padrão RS232.

Palavras chave: hardware, software, sistema de aquisição sem fio, radiofrequência, microprocessador, Módulo TRF-2.4G.



ABSTRACT

The project consists of accomplishing the implementation in hardware and software of a wireless data acquisition system of a group of analog signals. The transmission is made by a radio frequency link, to a computer to visualize and store data. This system consists of three modules. In the transmitter module (microcontroller + TRF-2.4G module) the signal is converted by the microcontroller and sent to the TRF-2.4G module. The signal is received by the receiver and once again adapted to the standard RS232.

Keywords: hardware, software, acquisition system wireless, radio frequency, microcontroller, module TRF-2.4G.



LISTA DE ILUSTRAÇÕES

Ilustração 2-1 – Curva Resistência x Temperatura de um PT100 $\rightarrow \alpha = 0.00385$	15
Ilustração 2-2 – Curva Tensão x Temperatura de alguns termopares com junção fria a 0°C	18
Ilustração 2-3 – Conector DB9 Macho.....	21
Ilustração 2-4 – Canais Simplex, Half-Duplex e Full-Duplex	22
Ilustração 3-1 – Arquitetura do PIC16F877A.....	27
Ilustração 3-2 – Esquema de organização da memória do PIC16F877A.....	30
Ilustração 3-3 – Arquitetura do PIC16F877A.....	31
Ilustração 3-4 – Estrutura do ADCON0.....	31
Ilustração 3-5 – Estrutura do ADCON1.....	32
Ilustração 3-6 – Tela MPLAB.....	32
Ilustração 3-7 – Tela Software BootLoader.....	33
Ilustração 3-8 – Módulo TRF-2.4G.....	35
Ilustração 3-9 – Pinagem do Módulo TRF-2.4G.....	35
Ilustração 3-10 – Diagrama demonstrativo de ligação entre o PIC e o Módulo.....	36
Ilustração 3-11 – Modo de Implementação dos Módulos TRF-2.4G.....	37
Ilustração 3-12 – Fonte de Alimentação de 3V.....	38
Ilustração 3-13 – Circuito de conversão de sinais.....	38
Ilustração 3-14 – Circuito <i>Half-Duplex</i>	39
Ilustração 3-15 – Diagrama de Tempo para Configuração.....	41
Ilustração 3-16 – Fluxograma de Configuração.....	42
Ilustração 3-17 – Fluxograma de montagem da Estrutura de Envio de Dados no Módulo TRF-2.4G.....	47
Ilustração 3-18 – Fluxograma de Transmissão do Módulo TRF-2.4G.....	50
Ilustração 3-19 – Fluxograma de Recepção do Módulo TRF-2.4G.....	51
Ilustração 3-20 – Lógica de Funcionamento do Módulo de Aquisição.....	52
Ilustração 3-21 – Lógica de Funcionamento do Módulo de Controle.....	54
Ilustração 3-22 – Tela Gráfica.....	55
Ilustração 3-23 – Circuito teste do A/D.....	56
Ilustração 5-1 – Diagrama do Tpd2cfgm.....	119
Ilustração 5-2 – Diagrama do Tpd2a.....	120
Ilustração 5-3 – Diagrama dos Tempos de Recepção no Modo <i>ShockBurst</i>	121
Ilustração 5-4 – Diagrama dos Tempos do Modo Configuração.....	122
Ilustração 5-5 – Diagrama dos Tempos de Transmissão no Modo <i>ShockBurst</i>	123



LISTA DE TABELAS

Tabela 2-1 – Coeficientes de Callendar-Va Dusen para os RDTs.....	16
Tabela 2-2 – Características de alguns termopares comerciais	18
Tabela 2-3 – Coeficientes do Polinômio para conversão de Temperatura em Tensão	19
Tabela 3-1 – Modos Principais do Módulo TRF-2.4G.....	41
Tabela 3-2 – Configuração dos bits RF_PWR.	44
Tabela 3-3 – Palavra de Configuração do Módulo TRF-2.4G.	45
Tabela 3-4 – Estrutura de Envio de Dados	45



LISTA DE SÍMBOLOS

°C – Graus Celsius

Kbps - kilobit por segundo

Kbytes – kilobyte por segundo

Mbps – megabit por segundo

MHz – megahertz



LISTA DE ABREVIATURAS E SIGLAS

A/D: *Analog/ Digital*

ALU: *Arithmetic and Logic Unit*

ASCII: *American Standart Code for Information Interchange*

CPU: *Central Processing Unit*

CRC: *Cyclic Redundancy Checksum*

DCE: *Data Communication equipment*

DTE: *Data Terminal equipment*

EEPROM: *Electric Erasable Programmable Read Only Memory*

GPR: *General Purpose Registers*

IEEE: *Instituto de Engenheiros Elétricos e Eletrônicos*

IPSec: *IP Secure*

ISM: *Industrial Scientific and Medical*

LAN: *Local Area Network*

PC: *Personal Computer*

PIC: *Peripheral Interface Controller*

RAM: *Random Access Memory*

RDT: *Resistance Temperature Detector*

RF: *RadioFrequência*

RISC: *Reduced Instruction Set Computer*

SFR: *Special Function Registers*

USB: *Universal Serial Bus*

VPN: *Virtual Private Network*

WAN: *Wide Area Network*

WEP: *Wired Equivalent Privacy*

WLAN: *Wireless Local Area Network*



SUMÁRIO

1. INTRODUÇÃO	12
1.1. Motivação	12
1.2. Objetivos	12
1.3. Estrutura do Trabalho	13
2. REFERENCIAL TEÓRICO.....	14
2.1. Sensores	14
2.2. Padrões para Comunicação Digital.....	19
2.3. Canais de Comunicação	21
2.4. Espalhamento Espectral	22
2.5. Aquisição de Dados	24
3. MATERIAIS E MÉTODOS.....	26
3.1. Microcontrolador PIC16F877A	26
3.2. Ambiente de Programação e BootLoader.....	32
3.3. Módulo de Comunicação TRF-2.4G	33
3.4. Testes e Validação	51
4. CONCLUSÃO.....	57
5. REFERÊNCIAS	58
OBRAS CONSULTADAS	59
GLOSSÁRIO	60
APÊNDICE A – ROTINAS DO SOFTWARE PIC.....	62
APÊNDICE B – ROTINAS DO SOFTWARE BUILDER.....	79
ANEXO A – ESQUEMA ELÉTRICO.....	86
ANEXO B – MANUAL DA LAIPAC - MÓDULO TRF-2.4G	88
ANEXO C – TEMPORIZAÇÃO - MÓDULO TRF-2.4G.....	119

1. INTRODUÇÃO

O controle e a medição de processos industriais têm avançado muito nas últimas décadas, tornando-se cada vez mais importante e indispensável em uma planta industrial. Buscando sempre novas tecnologias para a melhoria e a facilidade de acesso às informações, novos dispositivos e tecnologias estão surgindo a fim de realizar de diversas maneiras a transmissão de dados.

Esta demanda é causada pela necessidade de um controle mais confiável e apurado para que os processos sejam mais eficientes e com um custo final menor.

1.1. *Motivação*

A busca pela redução de custos nas empresas vem sendo constantemente requerida pelos administradores. Os engenheiros são cobrados para que os custos da construção de uma planta industrial juntamente com a manutenção das instalações, o tempo e o número de paradas sejam minimizados ao máximo.

Tendo em vista todos os fatores que influenciam a construção e o funcionamento da planta industrial, este trabalho foi desenvolvido com o objetivo de otimizar o sistema de aquisição de dados.

A instrumentação influencia diretamente na construção e o funcionamento da planta industrial, e seus custos estão ligados à instalação elétrica e automação das plantas. Uma parcela considerável dos custos envolvidos na instrumentação de uma planta ou fábrica está ligada à instalação de cabos, leitos, eletrodutos e mão de obra.

1.2. *Objetivos*

O trabalho tem como objetivo implementar um sistema de aquisição de dados sem fio, para sinais analógicos, utilizando módulos híbridos com transmissão e recepção de dados. Desta forma um módulo ficará na base de controle e os demais ficarão próximos aos pontos de controle.



Também serão desenvolvidas e analisadas técnicas de aquisição e tratamento de amostras e/ou fenômenos físicos de natureza analógica. Serão utilizados microcontroladores, um sistema de aquisição analógico através de uma rede sem fio com o menor custo possível, o uso de módulos híbridos de RF para a transmissão e recepção e as técnicas para o interfaceamento com os computadores.

Contudo, o projeto facilitará e agilizará a aquisição de dados, sem que haja a necessidade da passagem de cabos e que exista a possibilidade de mais de um ponto de controle fixo ou móvel. O projeto faz o uso de um canal em RadioFrequência a fim de enviar os dados do coletor à base de controle para a visualização pelo usuário através de uma interface homem-máquina (IHM).

1.3. Estrutura do Trabalho

O segundo capítulo apresenta os instrumentos utilizados para a aquisição dos dados, padrões de comunicação e interfaceamento e protocolos de comunicação utilizados neste projeto.

O terceiro capítulo apresenta a descrição do microcontrolador PIC 16F877A, portas I/O, Conversores A/D, também é apresentado o Módulo TRF-2.4G, sua configuração, funcionamento, transmissão e recepção de dados.

2. REFERENCIAL TEÓRICO

2.1. Sensores

2.1.1. PT100

O PT100 é um sensor de temperatura, do tipo RTD (*Resistance Temperature Detector*), que é baseado na variação de resistência elétrica. Os termômetros de resistência funcionam com base no fato de que a resistência dos metais varia em função da temperatura. O PT100 é bastante utilizado industrialmente e também são muito confiáveis, fazendo suas medidas com erros muito pequenos. (2001, National Instruments)

O metal do qual é construído o PT100 é a platina.

As principais características são:

- Condutor Metálico (platina).
- São dispositivos de alta linearidade.
- Faixa de operação (-200°C a 850°C).
- Apresentam baixíssima tolerância de fabricação (0,06% a 0,15%).

Os PT100 são considerados sensores de alta precisão. Os mesmos são confeccionados com um fio de metal de alto grau de pureza, que também são construídos depositando-se um filme metálico em um substrato cerâmico. A platina é utilizada, por ser um material inerte e conservar as suas características em altas temperaturas, além de poder trabalhar em altas temperaturas devido ao seu alto ponto de fusão. (Balbinot & Brusamarello, 2006)

Há três motivos para que a platina seja escolhida para a construção do PT100:

- Dentro da faixa, a relação resistência x temperatura é bastante linear.
- Apresenta uma boa repetibilidade.
- Sua faixa de linearidade é a maior entre os metais.

A ilustração 2-1 mostra a curva de resistência elétrica em função da temperatura para um PT100.

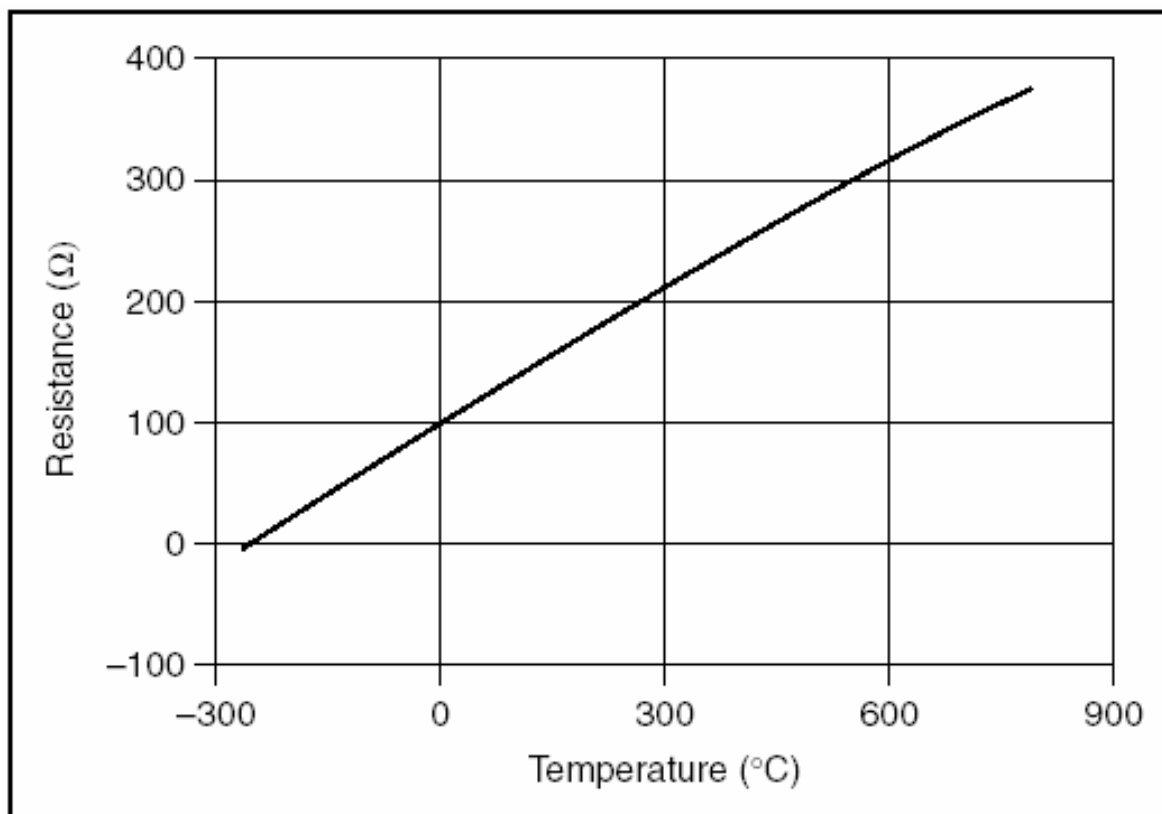


Ilustração 2-1 – Curva Resistência x Temperatura de um PT100 → $\alpha = 0.00385$

Fonte: Measuring Temperature with an RTD or Thermistor – National Instruments

O PT100 é o mais usado e mais conhecido dentre as outras termoresistências de platina que são: PT-25,5Ω, PT-120Ω, PT-130Ω, PT-500Ω e PT-1000Ω. A sigla significa o metal utilizado (PT = platina) seguida pela resistência à temperatura de 0°C. (Balbinot & Brusamarello, 2006)

Além da temperatura, impurezas e tensões mecânicas influenciam nas características resistência x temperatura dos elementos. Outros fatores que influenciam na sensibilidade são as contaminações químicas que reduzem a vida útil dos sensores. O valor desta sensibilidade está relacionada com o coeficiente de temperatura da resistência. (Balbinot & Brusamarello, 2006)

Definindo a variação de resistência do metal em função da variação de temperatura temos: (National Instruments – Application Note 046, 2001)

$$R_T = R_0[1 + \alpha T + \beta T^2 + \chi T^3(T - 100)]$$

Sendo:

R = Resistência à temperatura T.

R₀ = Resistência nominal.

α , β e x = Constantes usadas pela escala do RTD.

Os coeficientes α , β e x são utilizados de acordo com o material padrão utilizado na fabricação do RDT, conforme pode ser visto na tabela 2-1.

Tabela 2-1 – Coeficientes de Callendar-Va Dusen para os RDTs

Fonte: Measuring Temperature with an RTD or Thermistor – National Instruments

Standard	Temperature Coefficient (α)	A	B	C
DIN 43760	0.003850	3.9080×10^{-3}	-5.8019×10^{-7}	-4.2735×10^{-12}
American	0.003911	3.9692×10^{-3}	-5.8495×10^{-7}	-4.2325×10^{-12}
ITS-90	0.003926	3.9848×10^{-3}	-5.8700×10^{-7}	-4.0000×10^{-12}
*For temperatures below 0 °C only. C = 0.0 for temperatures above 0 °C				

2.1.2. Termopar

Os Termopares geram um sinal elétrico a partir de uma diferença de temperatura sem necessitar de alimentação. Muitas vezes são denominados de transdutores elétricos, pois fornecem tensão ou corrente elétrica em resposta ao estímulo. (Balbinot & Brusamarello, 2006)

O Termopar é um circuito fechado, formado por dois metais diferentes, que é percorrido por uma corrente elétrica quando as junções estão expostas a uma diferença de temperatura. Se o circuito está aberto, uma força eletromotriz termelétrica aparece e depende somente da natureza dos metais e das temperaturas das junções do termopar. Por isto ele pode ser usado como um sensor de temperatura ou como uma fonte de energia elétrica. Eles são geralmente usados como sensor de temperatura, pois estes apresentam um baixíssimo rendimento. (Balbinot & Brusamarello, 2006)

As principais características desejáveis dos termopares comerciais são: (National Instruments – Application Note 043, 2001)

- Resistência à oxidação e corrosão conseqüentes do meio e altas temperaturas.
- Linearidade dentro do possível.
- Ponto de fusão superior a maior temperatura à qual o termopar é submetido.
- Sua força eletromotriz deve ser suficiente para ser medida com exatidão razoável.
- Sua força eletromotriz deve aumentar continuamente com o aumento da temperatura.

- Os metais devem ser homogêneos.
- Suas resistências elétricas não devem apresentar valores que limitem seu uso.
- Sua força eletromotriz deve ser estável durante a calibração e o uso dentro de limites aceitáveis.
- Sua força eletromotriz não deve ser alterada consideravelmente por mudanças químicas, físicas, ou pela contaminação do ambiente.
- Deve ser facilmente soldado pelo usuário.

O número de características desejadas limita a escolha dos materiais para a formação do termopar. Alguns termopares comerciais são citados abaixo: (2001, National Instruments)

- Os termopares do Tipo J são constituídos de ferro e constantã. Estes são versáteis, de baixo custo e indicados para atmosferas inertes ou redutoras. Devem ser utilizados com tubos de proteção e não são indicados para operarem em ambientes oxidantes. Muitas vezes utilizados em fornos elétricos e processos de recozimento.
- Os termopares do tipo K são constituídos de cromel e alumel. Este tem uma melhor faixa de medição em ambientes oxidantes. Apresentam uma boa resistência mecânica a altas temperaturas e não são indicados para atmosferas redutoras. Muitas vezes são utilizados em tratamentos térmicos, fornos, processos de fundição e banhos.
- Os termopares do Tipo T são constituídos de cobre e constantã. Estes são resistentes à corrosão e úteis em ambientes excessivamente úmidos. Resistentes a atmosferas redutoras e oxidantes, são muito utilizados em temperaturas negativas. A principal desvantagem está relacionada à oxidação do cobre em altas temperaturas. Muitas vezes são utilizados em estufas, banhos, fornos elétricos para baixas temperaturas.
- Os termopares do Tipo E são constituídos de cromel e constantã. Estes apresentam alta sensibilidade e resistem a processos corrosivos inferiores a 0°C e a ambientes oxidantes.
- Os termopares do Tipo N são constituídos de nicrosil e nisil. Estes apresentam alta resistência à oxidação e são estáveis em altas temperaturas.

Tabela 2-2 – Características de alguns termopares comerciais

Fonte: Measuring Temperature with Thermocouples – National Instruments

Termopares (Tipo)	Condutor		Faixa de Temperatura (°C)	Faixa de Tensão (mV)
	Positivo	Negativo		
E	Cromel	Constantã	-270 a 1000	-9,835 a 76,358
J	Ferro	Constantã	-270 a 1200	-8,096 a 69,536
K	Cromel	Alumel	-270 a 1372	-6,548 a 54,874
T	Cobre	Constantã	-270 a 400	-6,258 a 20,869
S	Platina – 10% Ródio	Platina	-50 a 1768	-0,236 a 18,698
R	Platina – 13% Ródio	Platina	-50 a 1768	-0,226 a 21,108

As tabelas padrão fornecem a tensão de saída correspondente a temperatura quando a junção fria está a 0°C. A ilustração 2-2 mostra algumas curvas dos termopares comerciais existentes.

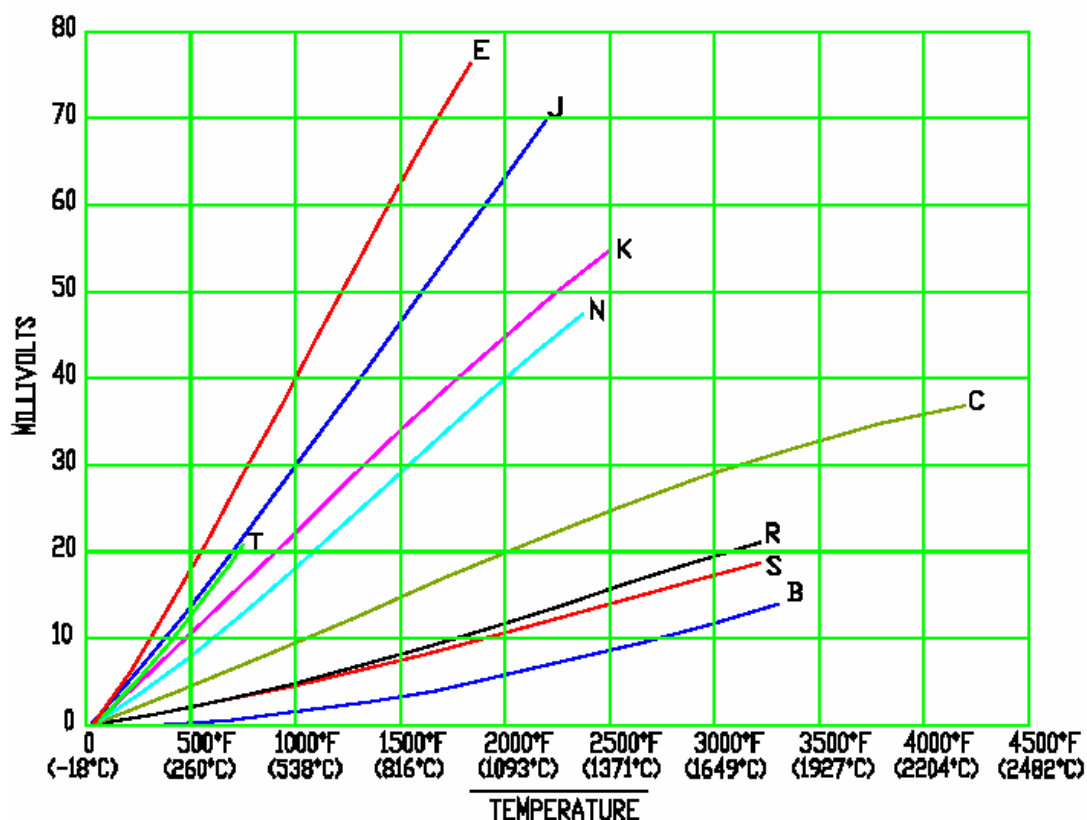


Ilustração 2-2 – Curva Tensão x Temperatura de alguns termopares com junção fria a 0°C

Fonte: How to Determine Temperature by Measuring the Output Millivoltage of a Thermocouple – Delta Controls Corp

A polaridade e a magnitude da tensão dependem da temperatura das junções e do tipo de material que constitui o termopar. Para isto existe uma equação matemática que transforma a temperatura mensurada em tensão, em mV, que é apresentada da seguinte forma: (2001, National Instruments)

$$V = c_0 + c_1T + c_2T^2 + \dots + c_nT^n$$

A tabela 2-3 mostra os coeficientes do polinômio para a conversão de temperatura em tensão de acordo com o tipo termopar.

Tabela 2-3 – Coeficientes do Polinômio para conversão de Temperatura em Tensão

Fonte: Measuring Temperature with Thermocouples – A Tutorial – National Instruments

	Thermocouple Type					
	E	J	K	R	S	T
Range	0 to 1000 °C	-210 to 760 °C	0 to 1372 °C	-50 to 1064 °C	-50 to 1064 °C	0 to 400 °C
c_0	0.0	0.0	-17.600413686	0.0	0.0	0.0
c_1	58.665508710	50.38118782	38.921204975	5.28961729765	5.40313308631	38.748106364
c_2	4.503227558E-2	3.047583693E-2	1.85587700E-2	1.3916658978E-2	1.2593428974E-2	3.32922279E-2
c_3	2.890840721E-5	-8.56810657E-5	-9.9457593E-5	-2.388556930E-5	-2.324779687E-5	2.06182434E-4
c_4	-3.30568967E-7	1.322819530E-7	3.18409457E-7	3.5691600106E-8	3.2202882304E-8	-2.18822568E-6
c_5	6.50244033E-10	-1.7052958E-10	-5.607284E-10	-4.62347666E-11	-3.314651964-11	1.09968809E-8
c_6	-1.9197496E-13	2.09480907E-13	5.6075059E-13	5.007774410E-14	2.557442518E-14	-3.0815759E-11
c_7	-1.2536600E-15	-1.2538395E-16	-3.202072E-16	-3.73105886E-17	-1.25068871E-17	4.54791353E-14
c_8	2.14892176E-18	1.56317257E-20	9.7151147E-20	1.577164824E-20	2.714431761E-21	-2.7512902E-17
c_9	-1.4388042E-21		-1.210472E-23	-2.81038625E-24		
c_{10}	3.59608995E-25		NOTE A			
NOTE A: The equation for type K is $v = c_0 + c_1T + c_2T^2 + \dots + c_9T^9 + 118.5976e^{(-1.183432E-4)(T - 126.9686)^2}$						

2.2. Padrões para Comunicação Digital

2.2.1. Padrão EIA RS-232

O RS-232, também conhecido por EIA RS-232C, é um padrão para troca série de dados binários entre um DTE (terminal de dados, de *Data Terminal equipment*) e um DCE (comunicador de dados, de *Data Communication equipment*). É o protocolo mais antigo, o mais conhecido e também o mais comum usado nas portas seriais dos PCs.

Este padrão foi originalmente usado para conectar um equipamento eletromecânico de comunicação assíncrona que usava código ASCII a um modem. Quando terminais eletrônicos começaram a ser usados, eram projetados para serem intercambiáveis com os telégrafos, e também suportavam RS-232. Deste modo, foi



utilizado em diversos tipos de comunicação remota, especialmente por modems. Posteriormente os microcomputadores começaram a utilizar este padrão para comunicação com equipamentos já existentes. Por muitos anos o padrão para comunicação serial em quase todos os computadores era algum tipo de porta RS-232. Continuou sendo utilizado em grande escala até o fim dos anos 90. Durante este período esta foi a maneira padrão para a conexão de modems.

Hoje, o protocolo de comunicação RS-232 vem sendo gradualmente substituído pelo USB para comunicação local. O protocolo USB é mais rápido, possui conectores mais simples de usar e tem um melhor suporte para software. Mesmo assim, o protocolo RS-232 continua sendo utilizado em periféricos para pontos de venda (caixas registradoras, leitores de códigos de barra ou fita magnética) e na área industrial (dispositivos de controle remoto). Por essas razões, computadores para estes fins continuam sendo produzidos com portas RS-232. Como alternativa, existem adaptadores para portas USB que podem ser utilizados para conectar uma ou mais portas seriais.

No protocolo de comunicação RS-232, caracteres são enviados um a um como um conjunto de bits. A codificação frequentemente usada é o "*start-stop assíncrono*" que usa um bit de início, seguido por sete ou oito bits de dados, possivelmente um bit de paridade, e um ou dois bits de parada, sendo então, necessários 10 bits para enviar um único caractere. Tal fato acarreta a necessidade em dividir por um fator de dez a taxa de transmissão de bits para obter a velocidade de transmissão de caracteres. A alternativa mais comum ao "*start-stop assíncrono*" é o HDLC. O padrão define os níveis elétricos correspondentes aos níveis lógicos um e zero, a velocidade de transmissão padrão e os tipos de conectores.

O padrão especifica 20 diferentes sinais de conexão, e um conector em forma de D é comumente usado. São utilizados conectores machos e fêmeas - geralmente os conectores dos cabos são machos e dos dispositivos são fêmeas.

No conector a maioria dos pinos não são utilizados nos mais diversos dispositivos, sendo comum para que as máquinas economizem espaço e dinheiro. O conector padrão em forma de D contém apenas 9 pinos. A ilustração 2-3 apresenta um conector de 9 pinos e seus sinais.

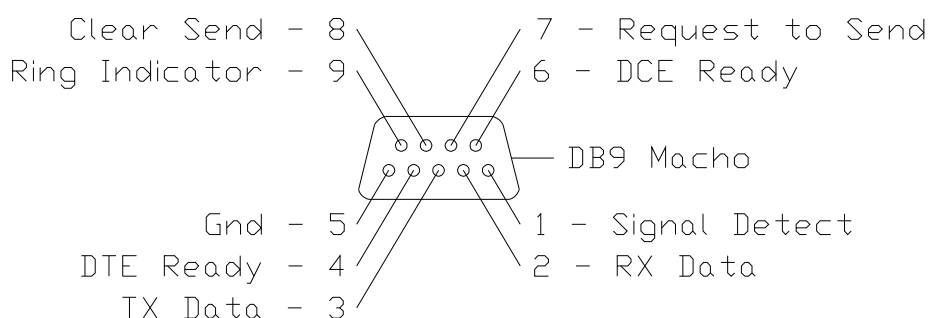


Ilustração 2-3 – Conector DB9 Macho

O RS-232 é recomendado para conexões curtas (quinze metros ou menos). Os sinais variam de 3 a 15 volts positivos ou negativos, valores próximos de zero não são sinais válidos. O nível lógico um é definido por ser voltagem negativa, a condição de sinal é chamada marca e tem significado funcional de OFF (desligado). O nível lógico zero é positivo, a condição de sinal é espaço, e a função é ON (ligado). Níveis de sinal ± 5 , ± 10 , ± 12 e ± 15 são vistos comumente, dependendo da fonte elétrica disponível.

2.3. Canais de Comunicação

Existem três maneiras de se interligar os sistemas digitais.

2.3.1. Canal Simplex

O Canal simplex é o canal no qual a direção de transmissão é inalterada. Por exemplo, uma estação de rádio é um canal simplex porque ela sempre transmite o sinal para os ouvintes e nunca é permitido a transmissão inversa.

2.3.2. Canal Half-Duplex

O Canal half-duplex é um canal físico simples no qual a direção pode ser revertida. As mensagens podem fluir nas duas direções, mas nunca ao mesmo tempo. Em uma chamada telefônica, uma parte fala enquanto a outra escuta. Depois de uma pausa, a outra parte fala e a primeira escuta. Falar simultaneamente resulta em sons que não podem ser compreendidos.

2.3.3. Canal Full-Duplex

O Canal full-duplex permite que mensagens sejam trocadas simultaneamente em ambas as direções. Ele pode ser visto como dois canais simplex, um canal direto e um canal reverso, conectados nos mesmos pontos.

É mostrado na ilustração 2-4, exemplos simples dos tipos de interligação.

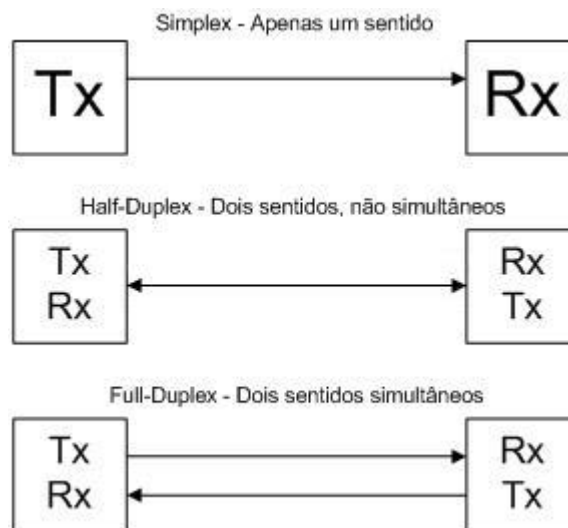


Ilustração 2-4 – Canais Simplex, Half-Duplex e Full-Duplex

2.4. Espalhamento Espectral

A técnica do espalhamento espectral foi desenvolvida no final da segunda guerra devido a necessidade do sigilo nas suas comunicações. Com o espalhamento espectral pode-se notar uma redução de densidade de energia, melhora do alcance para alta resolução e múltiplo acesso, e isso porque a técnica do espalhamento espectral utiliza uma largura de faixa muito maior que a necessária para transmitir informação.

Para um sistema ser considerado espalhamento espectral deve ter alguns requisitos, tais como:

- Possuir uma banda muito maior que a necessária para sua transmissão;
- O espelhamento espectral é obtido por código, que deve ser independente da mensagem;
- Para o receptor, a recuperação do sinal é obtida com uma réplica sincronizada do sinal de código utilizado para espalhar informação.

Anteriormente a técnica usada nos sistemas do espalhamento do espectro era o TR (referência transmitida), mas tinha algumas desvantagens como:

- Como o código era transmitido através do meio, estava disponível para qualquer um;
- Facilmente atrapalhado por um jammer (interferência internacional);
- Sua performance caía para sinais baixos;

A literatura aborda a existência de três técnicas de espalhamento espectral, cabendo destacar: Espalhamento Espectral por Seqüência Direta, Espalhamento Espectral por Salto em Frequência e Espalhamento Espectral por Salto no Tempo.

2.4.1. Espalhamento Espectral por Seqüência Direta

A técnica de espalhamento espectral por seqüência direta representa um dos principais segmentos no estudo de telefonia celular CDMA, padronizada pelo IS-95. Nesta técnica, os bits de dados são multiplicados por uma seqüência pseudo aleatória (PN) de comprimento N. Desta forma, o espectro do sinal de saída ocupa uma largura de faixa muito maior do que a largura de faixa ocupada pelo sinal original.

Após, o sinal recebido deve ser multiplicado pela seqüência PN em fase, para que o sinal possa ser recuperado. Caso as seqüências estejam defasadas, o sinal não é recuperado sendo interpretado como ruído pelo circuito de decisão.

2.4.2. Espalhamento Espectral por Salto em Frequência

O Espalhamento Espectral por Salto em Frequência representa outra técnica de espalhamento espectral que vem sendo bastante utilizada para transmissão de sinais digitais em meios perturbados. Neste caso, o Oscilador Controlado por Tensão tem uma frequência de operação definida por uma seqüência PN, ou seja, a cada instante de tempo, a frequência utilizada para modular o sinal a ser transmitido é definida por uma seqüência PN.

2.4.3. Espalhamento Espectral por Salto no Tempo

No Espalhamento Espectral por Salto no Tempo, o slot de tempo que o usuário ocupa em cada quadro é definido por uma seqüência PN.

Os sistemas de transmissão que utilizam técnicas de espalhamento espectral possuem um melhor desempenho referente a interferências propositas. Esta melhoria de desempenho é denominada de Ganho de Processamento. Com este sistema vários usuários utilizam a mesma faixa espectral ao mesmo tempo. Isso é possível pois cada usuário possui um código de espalhamento único, ou seja,



cada usuário utiliza uma sequência de espalhamento conhecida apenas pelo transmissor e pelo receptor.

Uma das grandes vantagens que se pode observar é que este tipo de análise das técnicas de espalhamento espectral pode ser adotado utilizando diferentes tipos de arquiteturas de rádios digitais, além de verificar o desempenho contra interferências intencionais de faixa estreita e ruído de faixa larga.

2.4.4. Modulação GFSK

A modulação GFSK (*Gaussian Frequency Shift Keying*), que é uma variação da modulação FSK aonde um filtro gaussiano é adicionado ao sinal antes dele ser modulado.

Já na modulação FSK, a modulação da envoltória se dá de maneira constante, o que diminui a complexidade dos circuitos de RF usados no projeto;

Como no FSK a modulação se dá através da variação da frequência, a mudança da amplitude do sinal causada por interferências não afetaria a confiabilidade do sinal transmitido.

Na modulação GFSK o bit 1 é representado por uma variação positiva da frequência e o bit 0, por uma variação negativa da frequência.

2.5. Aquisição de Dados

A aquisição de dados, através do recurso dos computadores pessoais ou industriais, é o processo pelo qual um fenômeno físico real é transformado num sinal elétrico proporcional e convertido num formato digital para posterior visualização, armazenamento, processamento e análise.

Em muitas aplicações, a aquisição de dados não se restringe apenas à aquisição, mas também compreende ações de controle sobre os sistemas. Os sinais digitais de controle correspondentes ao processo provenientes dos computadores são convertidos em sinais apropriados para atuar em diversos equipamentos de controle: atuadores, relés, válvulas, moduladores, etc.

Os sensores e transdutores fornecem a ligação direta entre o mundo real e o sistema de aquisição de dados convertendo sinais de grandezas físicas em sinais elétricos (tensões ou correntes) apropriados para os condicionadores de sinais e/ou os equipamentos de aquisição de dados. Atualmente, existem transdutores disponíveis para a medição da maioria das grandezas físicas existentes.

Os sinais elétricos gerados nos sensores e transdutores muitas vezes necessitam ser convertidos numa forma apropriada para o equipamento de aquisição.

O condicionamento do sinal é a parte mais importante da aquisição de dados. Para isto, existem dispositivos chamados condicionadores de sinais, que são circuitos eletrônicos que adequam os sinais analógicos para a conversão digital.

Os principais sub-componentes dos condicionadores são os amplificadores, filtros e isoladores. Através dos amplificadores, o sinal analógico é amplificado para ajustar-se à faixa de entrada do conversor A/D, e quando necessário, o amplificador responsabiliza-se também pela alimentação dos sensores. Os filtros reduzem os ruídos do sinal analógico, ou seja, diminuem eventuais interferências que podem ser originadas por diversas fontes: radiofrequência, rede elétrica, aterramento, etc. Os isoladores, quando presentes, têm a função de proteger os outros módulos contra eventuais sobrecargas de tensão e corrente, as quais podem causar danos irreversíveis aos circuitos eletrônicos digitais.

O hardware de medição é o responsável pelas entradas e saídas de sinais na cadeia de medida. Assim, ele pode executar qualquer uma das seguintes funções:

- Entrada, processamento e conversão para o formato digital, usando conversores digitais, de sinais analógicos provenientes do meio de medição. Os dados após convertidos são transferidos para o computador para visualização, armazenamento ou análise;
- Entrada de sinais digitais que contêm informação acerca de um sistema ou processo;
- Processamento, conversão para um formato analógico, utilizando conversores analógicos de sinais digitais do computador para controle de processos;
- Saída de sinais de controle digitais.

O equipamento de aquisição de dados existe em diversas plataformas provenientes de diversos fabricantes podendo dividir-se em placas de inserção que são ligadas diretamente no interior dos computadores e sistemas exteriores de comunicação. Na opção entre este tipo de equipamentos existem vantagens e desvantagens, sendo a sua seleção feita com base em diversos parâmetros, tais como: a capacidade de expansão, o ruído elétrico, o preço, as taxas de aquisição pretendidas, a possibilidade de configurar individualmente cada sinal, etc.

3. MATERIAIS E MÉTODOS

Para o processamento dos dados e o controle dos Módulos TRF-2.4G, foi adotado como plataforma de desenvolvimento o Microcontrolador PIC16F877A da Microchip Technology Inc.

3.1. *Microcontrolador PIC16F877A*

A família PIC possui um elevado número de microcontroladores, que se diferem pelo número de portas I/O, o tipo e o tamanho da memória, melhoria da performance dos circuitos periféricos e a flexibilidade dos dispositivos que não fazem parte da arquitetura dos microcontroladores, como os comparadores analógicos e os conversores A/D.

O PIC 16F877A contém todos os componentes necessários para o completo desenvolvimento de um sistema digital eletrônico programável. Este possui arquitetura RISC (*Reduced Instruction Set Computer*), 8 bits, até 20 MHz de *clock*, possui 8 Kbytes de memória FLASH programável, 368 bytes de memória RAM, conversor analógico / digital com 8 canais multiplexados, 32 entradas / saídas digitais e 3 timers para geração de PWM e contadores.

O PIC16F877A é um objeto mais complexo, com um tratamento completo, o qual foi definido para o escopo deste projeto. Portanto, nos parágrafos seguintes, nos limitaremos a descrever a arquitetura geral e nos aprofundarmos na funcionalidade da utilização para a nossa aplicação e o motivo pelo qual este dispositivo foi escolhido. (Microchip - PIC16F877A Data Sheet, 2003)

3.1.1. **Arquitetura**

A arquitetura RISC do PIC tem um número relativamente reduzido de instruções. A sua arquitetura mostrada na figura 3-1, com a memória e o barramento de dados separados da memória e do barramento do programa. Particularmente, quando o barramento de dados é de 8 bits, aquele para as instruções é de 14 bits, e é dimensionado de modo a poder capturar cada instrução com um único acesso à memória do programa. Além disso, esta estrutura permite

[illegible]

Fonte: PIC16F877A Data Sheet – Microchip

- Os pinos externos são organizados em 5 portas de I/O;
- A ALU (*Arithmetic and Logic Unit*), realiza as funções de cálculo e de elaboração dos dados durante a execução do programa;
- O barramento de programa e o barramento de dados;
- A memória de programa (*Flash Program Memory*), a memória de dados e uma terceira memória, a Data EEPROM;



- Alguns registros particulares, como o *Program Counter*, o *Status Register*, registros W e a pilha de 8 níveis (8 *level stack*);
- Diversos temporizadores: *Watch Dog Timer* *Timer0*, *Timer1* e *Timer2*;
- Conversor A/D de 10 bit;
- Outros periféricos, como comparadores analógicos e a porta serial síncrona.

3.1.2. Organização da Memória

O PIC16F877A tem três blocos de memória: a memória de programa, memória de dados e a EEPROM.

A memória de programa é uma *FLASH* composta de 8k de palavras de 14 bits. O *program counter* contém 13 bits e durante a execução do programa é automaticamente incrementado. Se ocorrer a chamada a uma sub-rotina ou há uma solicitação de interrupção o *program counter* direciona ao endereço da primeira instrução da sub-rotina, ao invés de informar todas as vezes o endereço da instrução sucessiva.

A memória de dados é uma RAM estática formada por 4 bancos de 128 bytes cada. Os registros localizados na parte inferior da memória são chamados de *Special Function Registers* (SFR), e os localizados na parte superior da memória são chamados de *General Purpose Registers* (GPR). A organização da memória é apresentada na figura 3-2.

Os SFR são registros utilizados pela CPU para o controle dos periféricos e a gestão das operações que o dispositivo exige. Portanto, todas as localizações da memória (1 byte) constituem um registro dedicado de uma função específica. Entre os registros mais importantes, os quais são replicados em todos os bancos de memória, são mostrados abaixo: (Microchip - PIC16F877A Data Sheet, 2003)

- *STATUS REGISTER* - Contém o bit de seleção dos bancos de memória de dados, os *flag's* são indicados no estado aritmético da ALU;
- O registro *OPTION_REG* - Contém o bit de controle das instruções externas, do *PULL-UP* da Porta B (*PORTB*) e o *TMR0 Prescaler / WDT Postscaler*;
- O registro *INTCON* - Contém vários *flag's* para a gestão das instruções;
- O registro *PCLATH* - Permite que o *locus* da memória possa ser selecionado;

- Finalmente, no microcontrolador existe também uma Data EEPROM, que não é mapeada no *File Register*, mas no qual é possível o acesso com o endereçamento indireto. (Microchip - PIC16F877A Data Sheet, 2003)

Reginaldo Marin - *Aquisição de dados por uma Rede sem Fio*
Universidade Luterana do Brasil

Ilustração 3-2 – Esquema de organização da memória do PIC16F877A.

Fonte: PIC16F877A Data Sheet – Microchip

3.1.3. Portas de I/O

Como pode ser visto na figura 3-3, os pinos externos do microcontrolador são organizados em 5 portas: uma de 6 bits, três de 8 bits e uma de 3 bits. Também encontramos pinos que são dedicados a funções particulares como a alimentação e a conexão do cristal. Pode-se notar que quase todos os pinos estão disponíveis para mais de uma função, que são selecionadas nos registros internos a que são dedicados.

O microcontrolador utiliza dois registros para a administração das portas de I/O: os registros PORTA, PORTB, PORTC, PORTD e PORTE, reportam os valores elétricos da porta (ver figura 3-1), e os registros TRISA, TRISB, TRISC, TRISD e TRISE, todos associados a uma porta e nos quais os valores de cada bit determinam se os pinos estão selecionados para uma porta de entrada ou de saída.

Neste trabalho, as portas de I/O foram configuradas da seguinte forma:

- PORTA: no módulo de leitura de dados, todos os pinos desta porta foram configurados como entradas analógicas. No módulo de recepção, os pinos analógicos não são utilizados;
- PORTB: tanto no módulo de leitura como no módulo de recepção, os pinos desta porta são utilizados para a comunicação e o controle dos módulos de transmissão;
- PORTC: Não é utilizada no módulo de leitura, nem no módulo de recepção;
- PORTD: Não é utilizada no módulo de leitura, nem no módulo de recepção;
- PORTE: os pinos RE0, RE1 e RE2 são utilizados como entradas analógicas no módulo de leitura, e no módulo de recepção não são utilizados.

Na figura 3-3 podemos notar que a maior parte dos pinos é possível associar mais de uma função em cada pino.

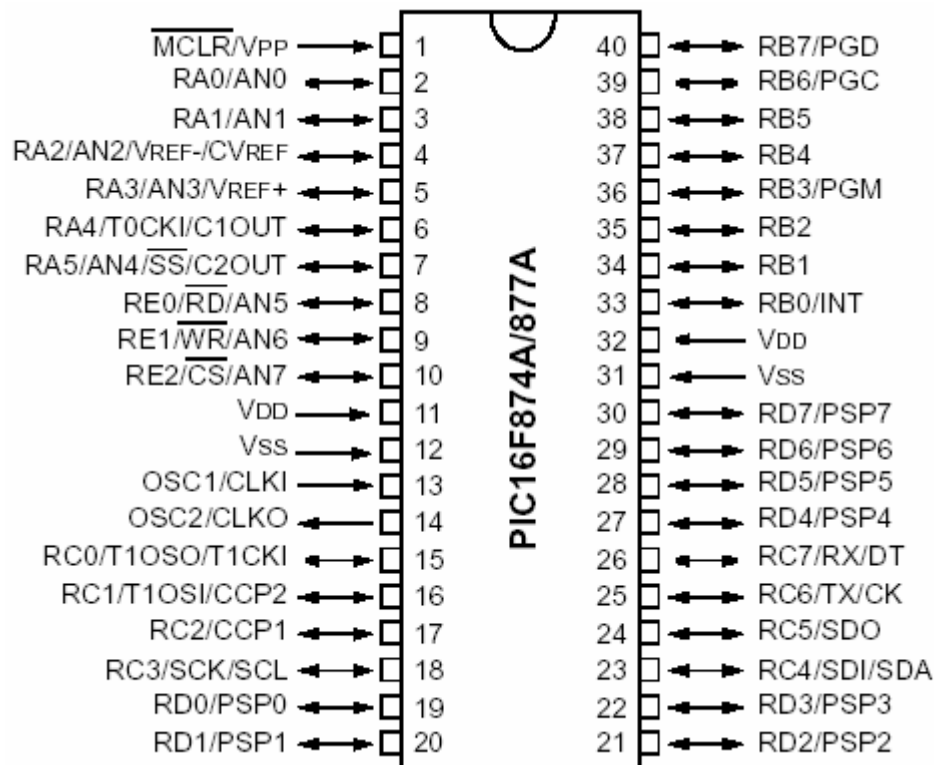


Ilustração 3-3 – Arquitetura do PIC16F877A.

Fonte: PIC16F877A Data Sheet – Microchip

3.1.4. Conversores A/D

Interno ao PIC16F877A está disponível um conversor A/D de 10 bits com oito canais de entrada multiplexadas, posicionados em seus pinos correspondentes na porta PORTA (exceto no pino RA4) e PORTE.

Os registros dedicados aos conversores A/D são quatro:

- ADRESH é a parte alta ou mais significativa da conversão do A/D;
- ADRESL é a parte baixa ou menos significativa da conversão do A/D. O ADRESH e ADRESL somados formam os 10 bit que constituem o resultado da conversão do A/D;
- ADCON0, controla a habilitação e o *clock* dos conversores e a multiplexação dos canais, e indica se uma conversão está sendo executada. A estrutura do ADCON0 pode ser vista na figura 3-4;

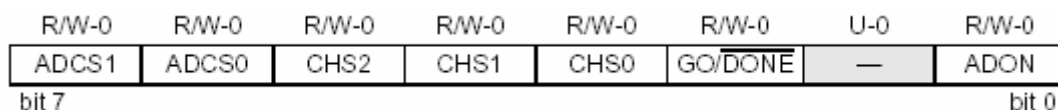


Ilustração 3-4 – Estrutura do ADCON0.

Fonte: PIC16F877A Data Sheet – Microchip

- ADCON1, configura a função das portas PORTA e PORTE, determina o alinhamento do resultado da conversão nos registros ADRESH e ADRESL, e ainda administra o *clock* dos conversores. A estrutura do ADCON0 pode ser vista na figura 3-5.

R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0
bit 7				bit 0			

Ilustração 3-5 – Estrutura do ADCON1.

Fonte: PIC16F877A Data Sheet – Microchip

3.2. Ambiente de Programação e BootLoader

3.2.1. MPLAB IDE v7.50

O MpLab é um ambiente integrado de desenvolvimento (I.D.E.: Integrated Development Environment). No mesmo ambiente o usuário pode executar todos os procedimentos relativos ao desenvolvimento de um software para o PIC, tornando o trabalho do projetista mais produtivo. O ambiente de programação do MpLab pode ser visto na figura 3-6.

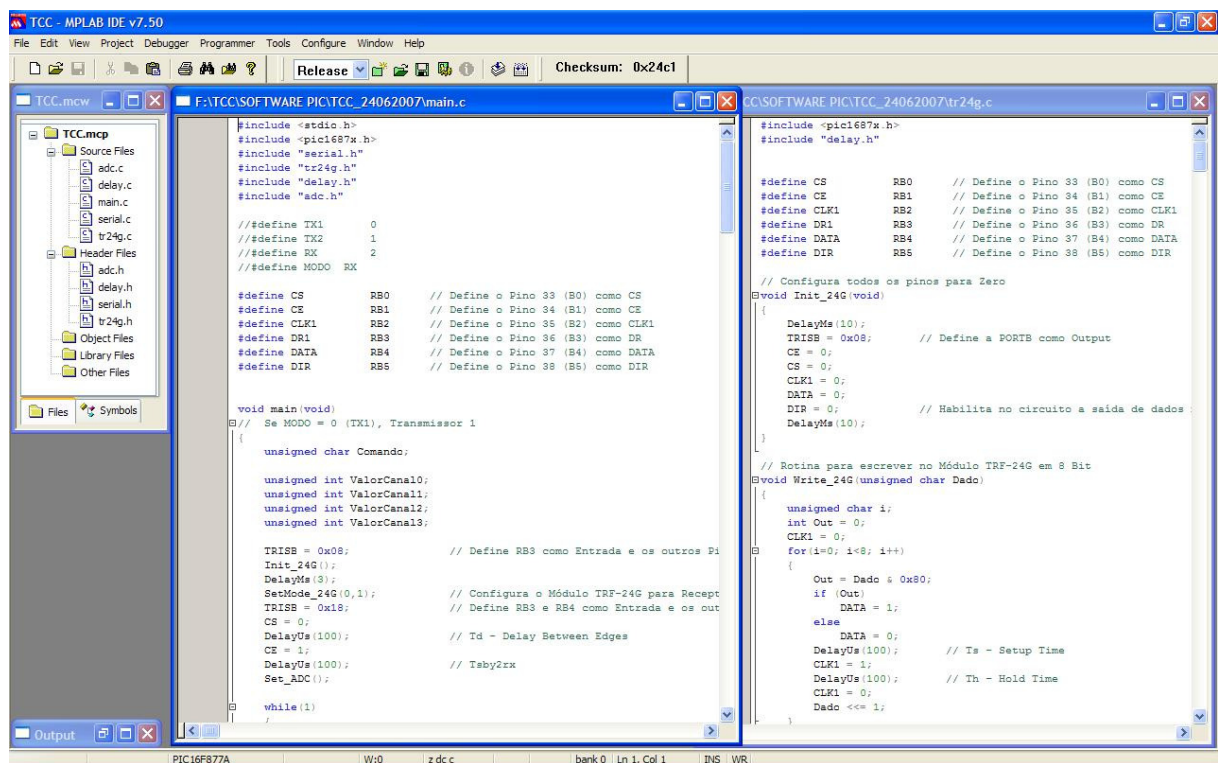


Ilustração 3-6 – Tela MPLAB.

Integrado ao MpLab foi utilizado o software HI-TECH, que possibilita que toda a programação seja feita na linguagem C.

Na compilação gera um arquivo com extensão .hex (hexadecimal) a partir dos arquivos de código fonte (.asm) e de projeto (.pjt). É o conteúdo do arquivo hexadecimal que é gravado na memória de programa do PIC.

3.2.2. BootLoader

A transferência de programas para os microcontroladores da família PIC é normalmente efetuada através de um dispositivo gravador específico. Este possui um inconveniente, que é a necessidade de a cada alteração do software e nova gravação o processador deve ser retirado do circuito e colocado no gravador.

Para facilitar e agilizar o processo de gravação do PIC, inicialmente foi gravado um software BootLoader. Assim que o pino do PIC, previamente configurado no BootLoader, é colocado em nível lógico 1 o BootLoader fica aguardando a chegada de um novo software pelos pinos seriais do PIC. Após o termino do download, o PIC começa a executar novo software.

Desta forma, não é necessária a retirada do PIC do circuito a cada nova programação, a necessidade de adquirir um gravador e possibilita o envio do software através da porta serial do PC. Na figura 3-7 podemos ver a tela do software de download.

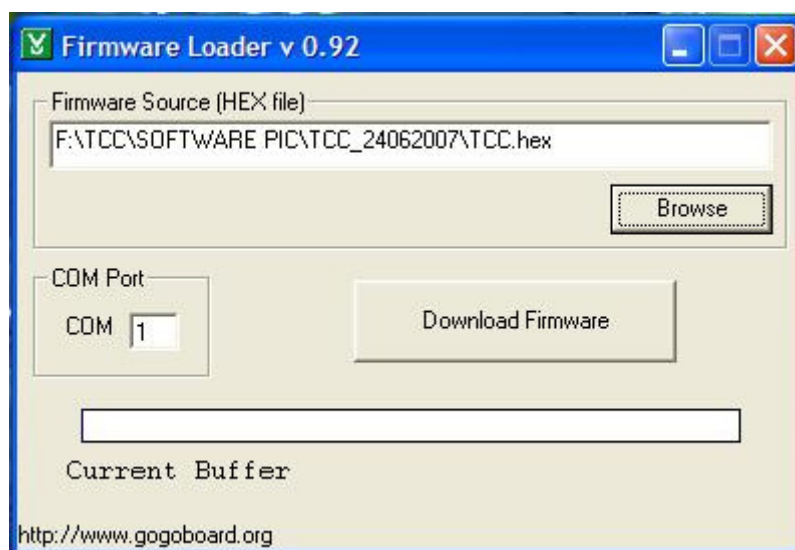


Ilustração 3-7 – Tela Software BootLoader.

3.3. Módulo de Comunicação TRF-2.4G

O módulo RF utilizado para a comunicação entre os módulos de aquisição e o módulo de controle é o Módulo TRF-2.4G.

O TRF-2.4G é um módulo da Laipac Tech Inc. que opera em uma banda de frequência reservada para o uso comercial ISM (*Industrial Scientific and Medical*) de 2.4 GHz e usa o chip da Nordic nRF2401VFSI com um cristal oscilador de 16MHz,

uma antena dipolo embutida e um amplificador de potência. As vantagens deste módulo transmissor são:

- Conter em um só chip o módulo de recepção e transmissão;
- Usar modulação GFSK e sua comunicação é *Full Duplex*;
- Possuir dois canais de operação;
- O hardware gera o código CRC (*Cyclic Redundancy Checksum*) e confere se há erros;
- Ter uma alta velocidade na transmissão no modo *ShockBurst*;
- Ter um alcance de 280 metros para uma taxa de transferência de 250Kbps e de 150 metros se a taxa de transferência é de 1Mbps;
- Ter um baixo consumo de energia.
- Ter um codificador, decodificador e “*data Buffer*”;
- Pelos mesmos pinos o módulo é configurado e os dados são enviados e adquiridos.

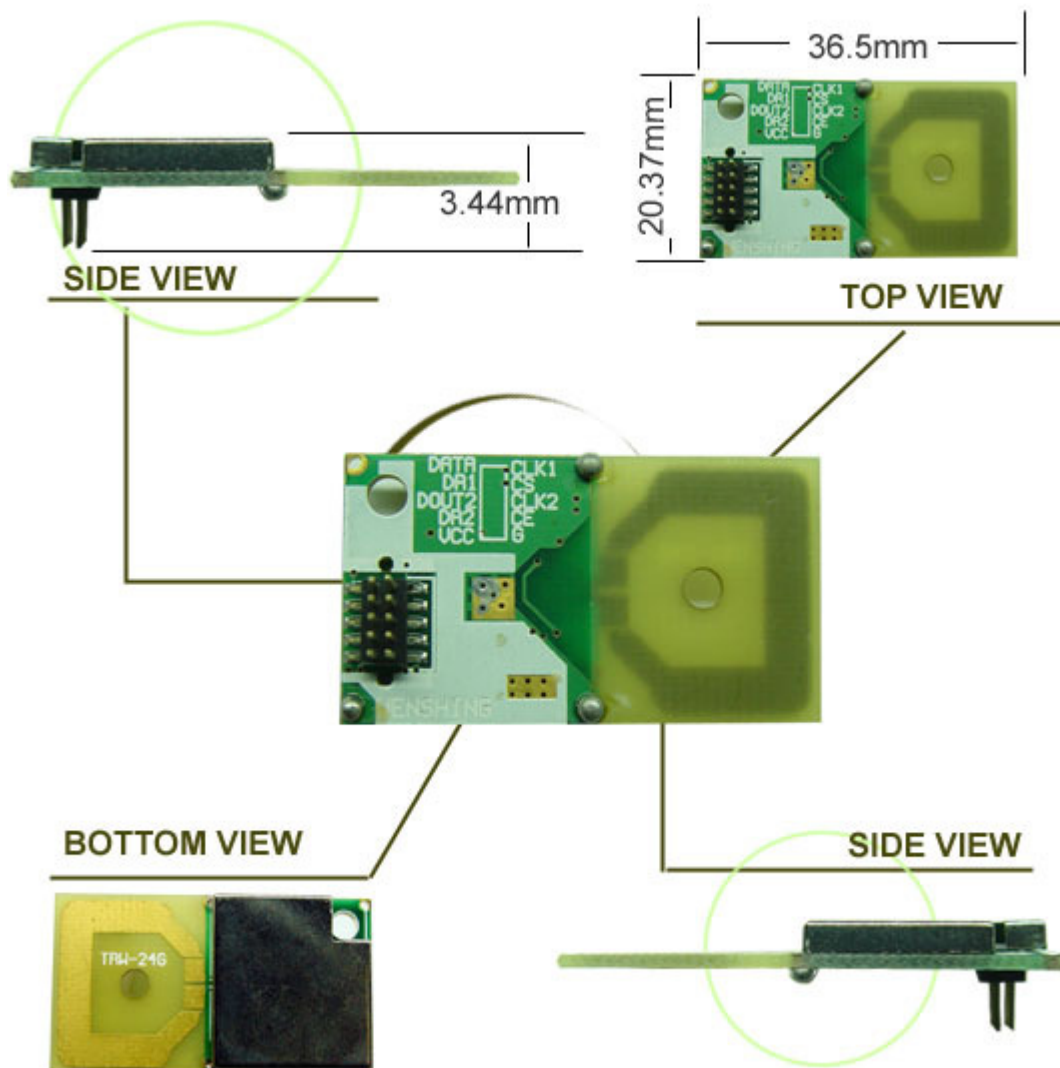


Ilustração 3-8 – Módulo TRF-2.4G.

Fonte: www.wenshing.com.tw

3.3.1. Pinos do Módulo TRF-2.4G

Os pinos do Módulo TRF-2.4G são descritos abaixo: (2004, Laipac Technology Inc.)

- **Pino 1 – GND** – *Ground* (0V);
- **Pino 2 – CE** – *Chip Enable*, ativo no modo recepção ou transmissão;
- **Pino 3 – CLK2** – *Clock output / input* para o modo recepção do canal 2 (não utilizado neste projeto);
- **Pino 4 – CS** – *Chip Select*, ativo durante a configuração do Módulo TRF2.4G;
- **Pino 5 – CLK1** – *Clock Input* (transmissão) e I/O (recepção) do canal 1;
- **Pino 6 – DATA** – Recepção do canal 1 e transmissão para ambos os canais;
- **Pino 7 – DR1** – Sinaliza, no modo recepção, quando os dados são recebidos pelo canal 1, somente no modo *ShockBurst*;
- **Pino 8 – DOUT2** – Recepção do canal 2 (não utilizado neste projeto);
- **Pino 9 – DR2** – Sinaliza, no modo recepção, quando os dados são recebidos pelo canal 2, somente no modo *ShockBurst* (não utilizado neste projeto);
- **Pino 10 – VCC** – Power Supply (+3V).

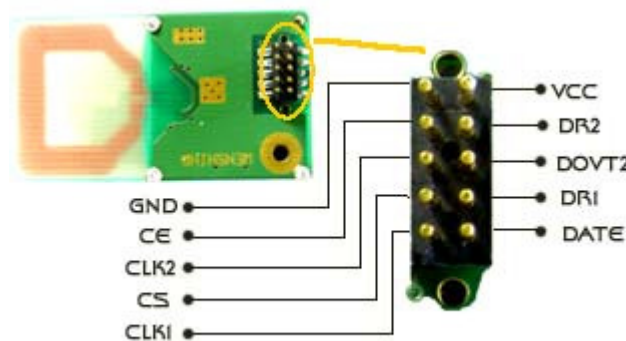


Ilustração 3-9 – Pinagem do Módulo TRF-2.4G.

Fonte: www.wenshing.com.tw

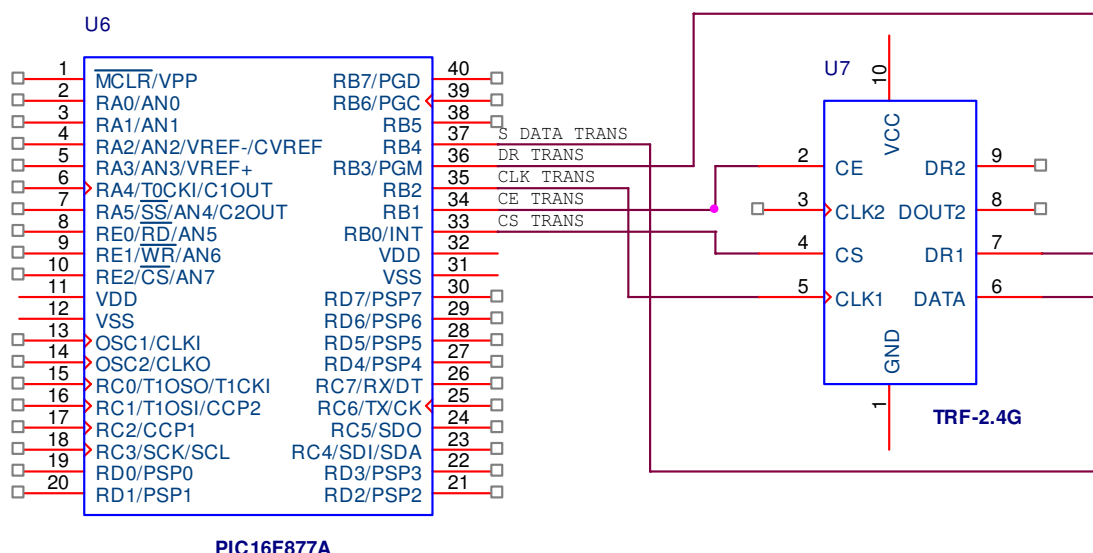


Ilustração 3-10 – Diagrama demonstrativo de ligação entre o PIC e o Módulo.

3.3.2. Implementação

Para o envio dos bits ao Módulo TRF2.4G está sendo utilizado neste projeto as portas de I/O digital do microcontrolador PIC16F877A. Este está conectado a um circuito que converte os níveis de tensão adequados para o módulo e para o PIC. Os pinos conectados são o CE, CS, CLK e DATA. Os bits são enviados ao pino DATA em grupos de 8 bits, bit a bit e após cada bit é dado um pulso de *clock* no pino CLK do módulo.

Neste projeto estamos usando três Módulos TRF2.4G, um módulo de controle conectado ao PC e outros dois módulos de aquisição que estão conectados aos sensores, para que possam fazer as leituras dos mesmos e enviá-las ao módulo receptor.

O módulo de controle inicialmente é configurado para selecionar e após transmitir os dados de comando para o módulo de aquisição escolhido. Assim que os dados de comando são enviados, o módulo de controle é configurado para o modo de recepção e fica monitorando o ar a fim de receber os dados adquiridos pelos módulos de aquisição e enviá-los ao PC.

Da mesma forma que acontece com o módulo de controle, os módulos de aquisição são configurados para recepção e transmissão. Inicialmente o módulo de aquisição é configurado para o modo de recepção e fica monitorando o ar a espera dos comandos enviados pelo módulo de controle. Assim que o módulo de aquisição recebe os comandos, os interpreta e após é configurado para o modo de transmissão para que possa enviar dos dados solicitados pelo módulo de controle. Após o envio dos dados, o módulo de aquisição é configurado para o modo de

recepção para que o mesmo fique monitorando o ar a espera dos comandos do módulo de controle.

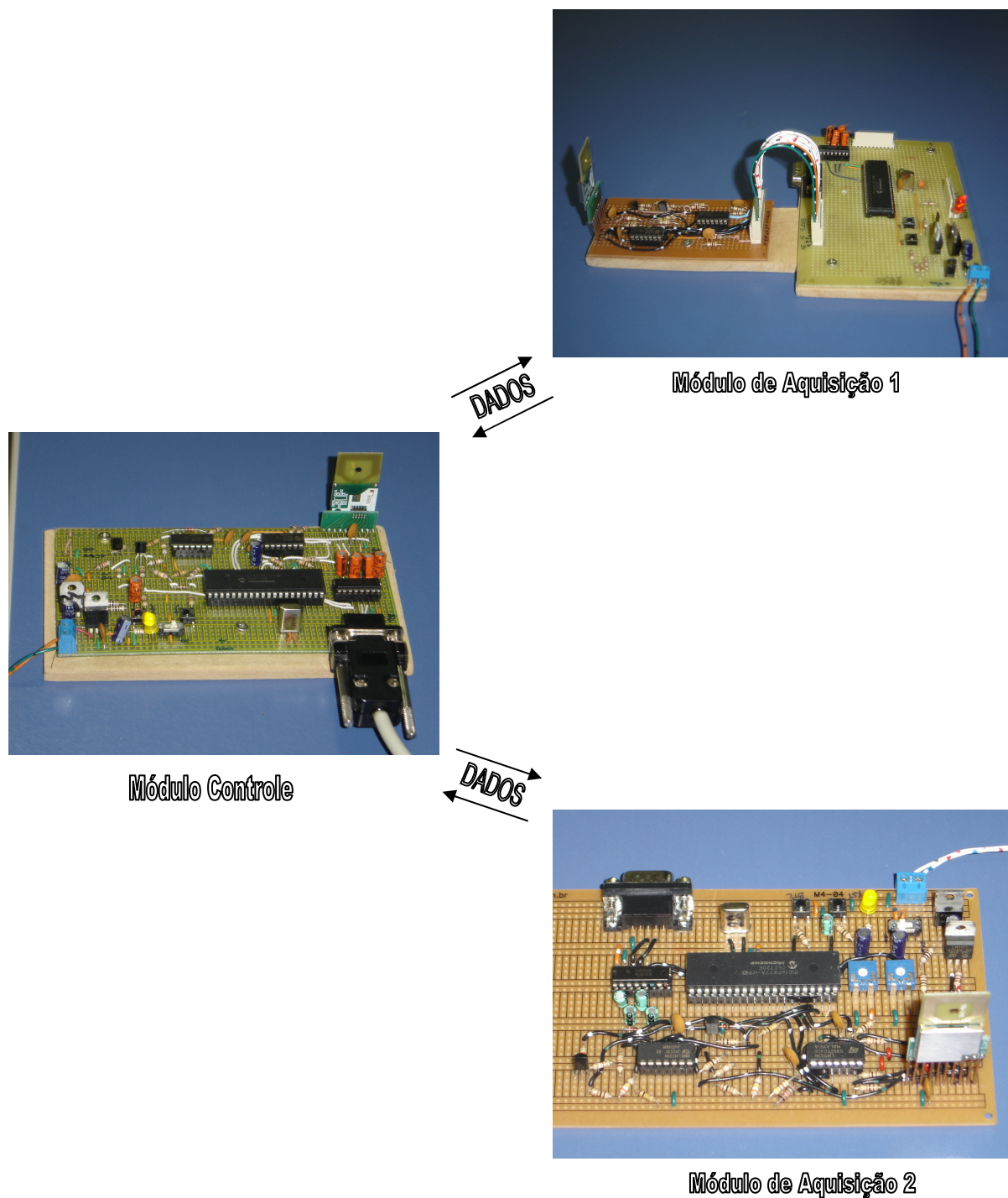


Ilustração 3-11 – Modo de Implementação dos Módulos TRF-2.4G.

3.3.3. Alimentação

O Módulo TRF-2.4G opera em baixa tensão, isso requer uma tensão V_{cc} de alimentação e os níveis lógicos iguais a $3,0 \pm 0,3$ V. Para a alimentação do módulo

utilizou-se uma fonte baseada no CI regulador de tensão LM317, conforme figura 3-12.

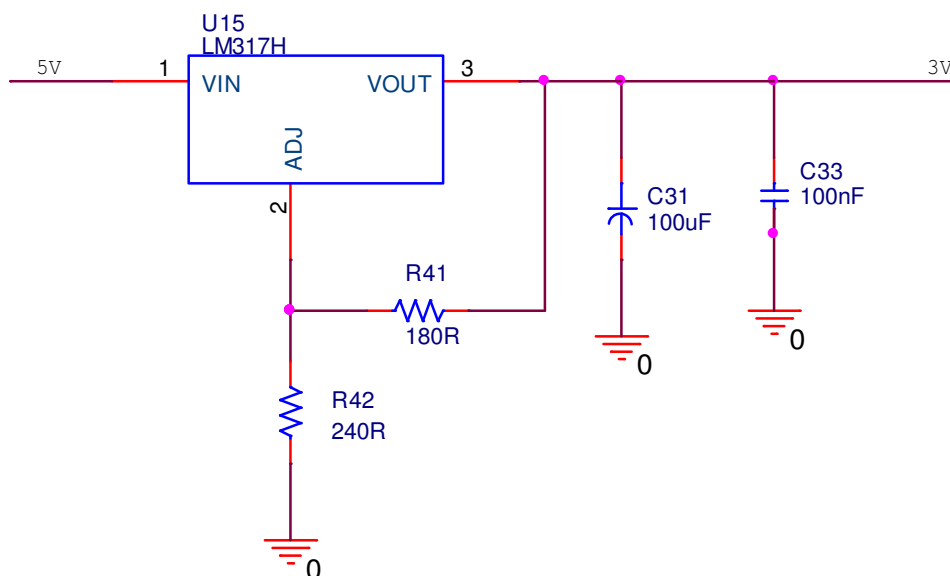


Ilustração 3-12 – Fonte de Alimentação de 3V.

A fim de conectar o Módulo TRF-2.4G com o Microcontrolador PIC16F877A, os níveis de tensão de suas saídas devem passar por um conversor do nível que transformará os 5V da saída do PIC em 3,0V. O dispositivo utilizado para a conversão dos níveis de tensão é o comparador LM339. Este converte os níveis lógicos de tensão de 5V para 3,0V quando os sinais são enviados do PIC para o módulo e de 3,0V para 5V quando os sinais são enviados pelo módulo para o PIC, conforme a figura 3-13.

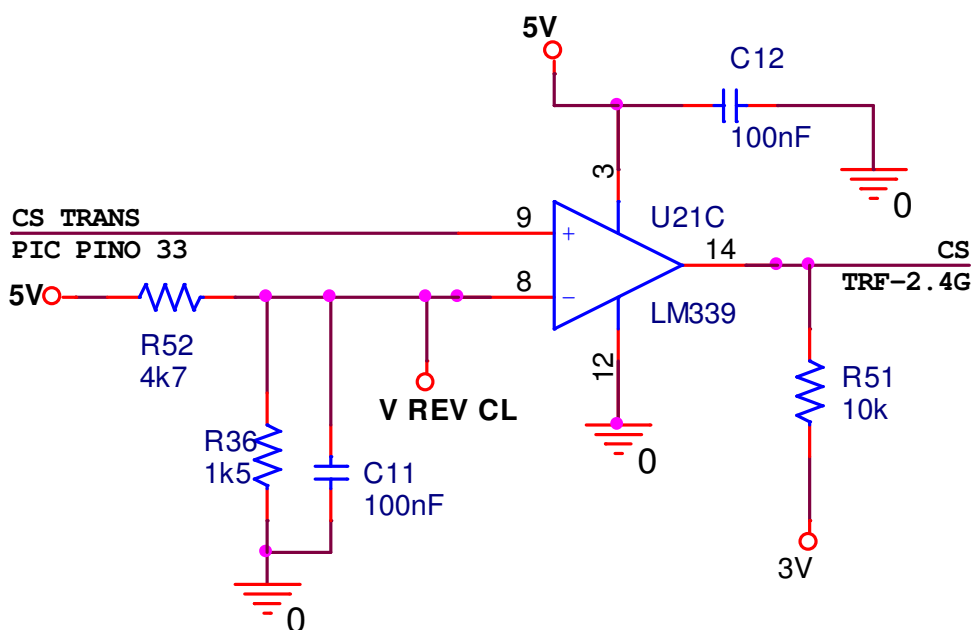


Ilustração 3-13 – Circuito de conversão de sinais.

Neste projeto, é utilizado um circuito *Half-Duplex* para o envio dos dados do Microcontrolador PIC16F877A para o Módulo TRF-2.4G e recebimento dos dados do módulo pelo PIC. Este circuito possibilita que seja utilizado apenas um pino do PIC para o envio e recebimento dos dados, conforme figura 3-14.

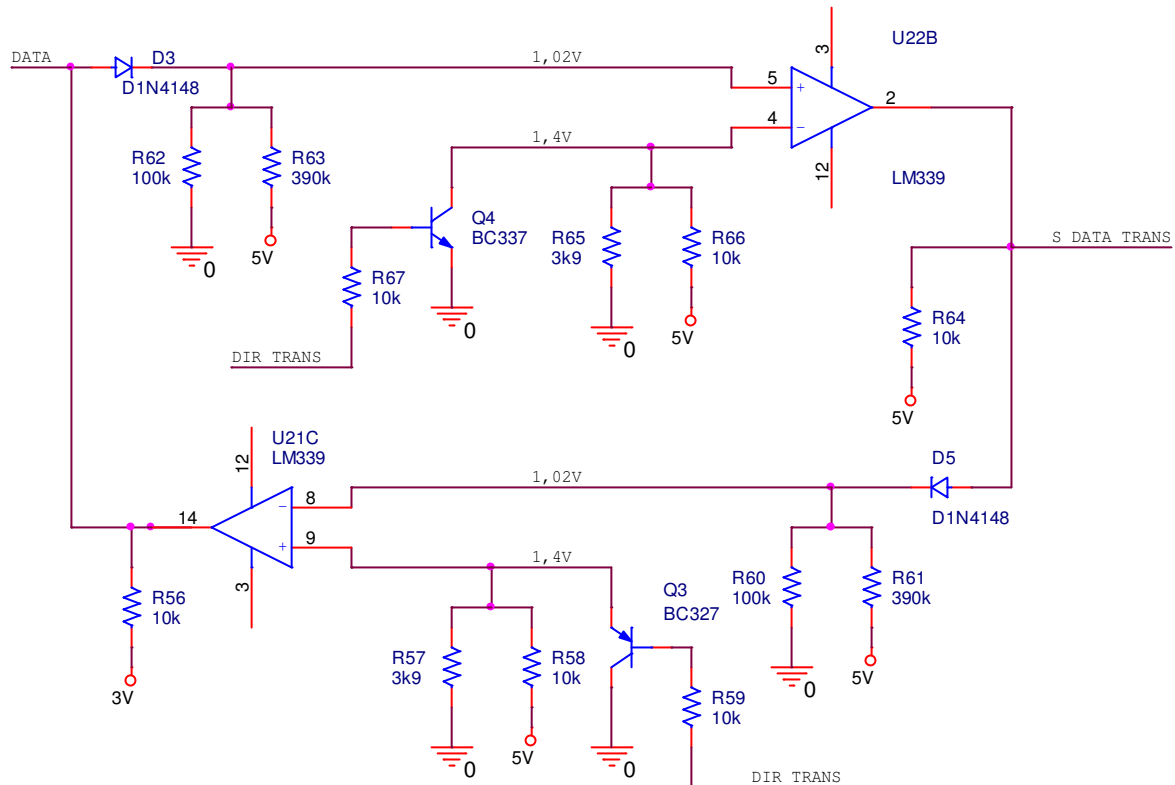


Ilustração 3-14 – Circuito *Half-Duplex*.

Para o envio dos dados do PIC ao módulo, o PIC coloca o pino DIR para nível lógico 1, possibilitando que os dados trafeguem somente em uma direção e bloqueando a chegada de dados ao pino DATA do PIC. Para o recebimento dos dados do módulo pelo PIC, o PIC coloca o pino DIR para nível lógico 0, possibilitando somente o recebimento dos dados.

O circuito completo utilizado, pode ser visto no **Anexo A** deste trabalho.

3.3.4. Modos de Operação

O Módulo TRF-2.4G pode operar em dois modos: (2004, Laipac Technology Inc.)

- *ShockBurst* – Este modo de funcionamento utiliza um *buffer* interno para armazenar a informação recebida do Microcontrolador PIC16F877A através do pino DATA. Uma vez que o Módulo TRF-2.4G recebe todo o pacote de dados que deverá enviar, este calcula o CRC, insere o *preamble* e transmite todo o pacote por RF com a velocidade

que foi definida. Esta velocidade de envio dos dados pode ser de 250kpbs ou de 1Mbps. Observa-se que não é necessária uma alta velocidade de comunicação entre o PIC e o módulo para que se possa utilizar a máxima velocidade de transmissão do módulo. Por outro lado, a transmissão dos dados em alta velocidade reduz tanto o risco de colisão no ar, como o tempo que permanecem os níveis altos de potência, minimizando consideravelmente o consumo médio de corrente. No modo *ShockBurst*, quando um pacote é recebido, o dispositivo notifica o PIC através do pino DR1, e os dados permanecerão armazenados no buffer interno do módulo até que o PIC receba todos os dados de forma ordenada.

- Modo Direto Transmissão / Recepção – Neste modo, o Módulo TRF-2.4G trabalha no modo tradicional de RF. Os dados são enviados em 1Mbps, ou em 250kpbs com uma taxa de dados baixa, para que o receptor possa detectar os sinais. No Modo Direto, o módulo age enquanto os dados que estão sendo recebidos são modulados. O microcontrolador tem que criar o *preamble*, monitorar as ondas do ar e verificar o erro de software.

Foi escolhido para este projeto o modo *ShockBurst* para que o PIC possa executar e processar outras tarefas, os dados lidos pelo A/D possam ser enviados com mais segurança, para que minimize os erros de transmissão e para que se torne mais ágil a aquisição de dados dos módulos transmissores.

3.3.5. Configuração

O Módulo TRF-2.4G é configurado com uma palavra de 144 bits (18 bytes). Nestes 144 bits são configurados o tamanho dos dados do canal 2, o tamanho dos dados do canal 1, o endereço do canal 2, o endereço do canal 1, o tamanho do endereço, o tamanho do CRC, o número de canais de recepção, modo de operação, velocidade de transmissão, frequência de clock, potência de transmissão, frequência de transmissão e o modo de operação do Módulo TRF2.4G.

Enviar os dados ao Módulo TRF-2.4G é razoavelmente simples. A palavra da configuração é emitida ao dispositivo em série (primeiro MSB) através do pino DATA pelo microprocessador PIC e este dá os pulsos de *clock*, no pino CLK1, de forma ordenada sucessivamente após cada bit enviado. É muito importante respeitar as exigências de temporização entre os dados enviados e o *clock* para que o Módulo

TRF2.4G compreenda os dados enviados a ele. O Módulo TRF2.4G requer um *delay* mínimo de 500ns entre o envio do dado e o pulso de *clock*.

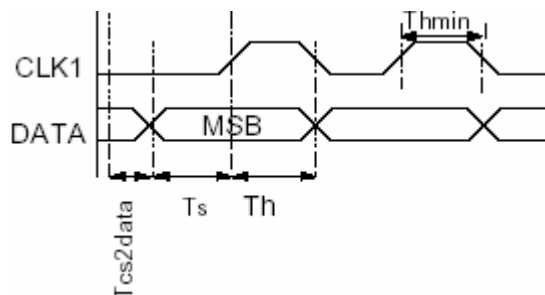


Ilustração 3-15 – Diagrama de Tempo para Configuração.

Fonte: Nordic Semiconductor ASA - Single Chip 2.4 GHz Transceiver nRF2401

Antes de configurar o Módulo TRF-2.4G os pinos de CE e CS devem ser colocados para o nível lógico 0. Somente após 5 μ s o módulo poderá ser colocado em modo de configuração, colocando o pino CE para nível lógico 0 e o pino CS para nível lógico 1, conforme tabela 3-2. Estes comandos serão dados pelo PIC através das suas saídas digitais.

Tabela 3-1 – Modos Principais do Módulo TRF-2.4G

	CE	CS
Modo Ativo	1	0
Configuração	0	1
Standby	0	0

Após o Módulo TRF2.4G ser colocado em modo de configuração, o mesmo já está pronto para receber os bits de configuração. Assim que o módulo receber todos os bits de configuração, o pino CS deve ser colocado para o nível lógico 0 e o pino CE deve ser colocado em nível lógico 1 para que o módulo fique monitorando o ar. Se o mesmo foi configurado para recepção, ou para que o módulo possa transmitir, se o mesmo foi configurado para transmissão. O fluxograma da rotina de configuração é demonstrada na figura 3-16.

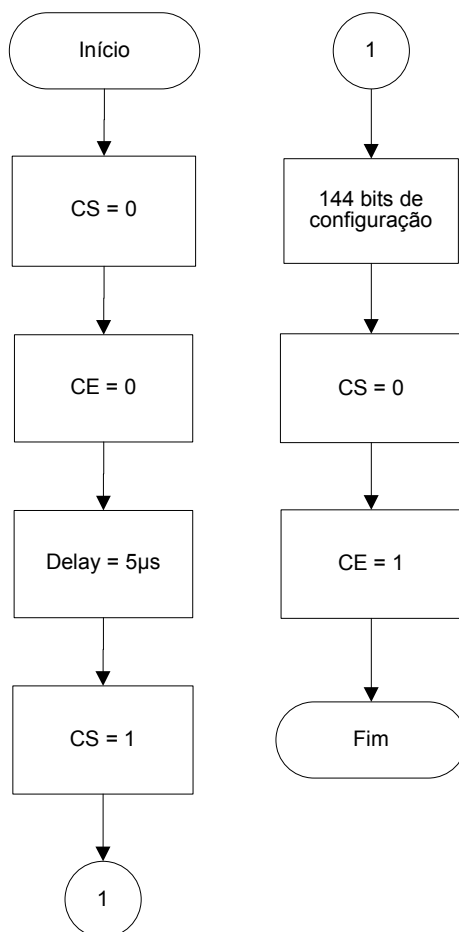


Ilustração 3-16 – Fluxograma de Configuração.

Para este projeto, o Módulo TRF2.4G foi configurado para operar em uma frequência de 2410MHz, modo ShockBurst, velocidade de transmissão de 1Mbps, potência de transmissão 0dB, endereço de 40-bit, dados 16 bit e 8 bit e comprimento do CRC de 16 bit.

Os 144 bits de configuração do Módulo TRF-2.4G são divididos e descritos da seguinte forma:

- Os Bits de Teste, do bit 143 ao bit 120, estes são reservados para testes do módulo. Estes bytes são os mais significativos, consequentemente são os primeiros a serem enviados pelo microcontrolador PIC16F877A para o módulo.
- Os Bits de Tamanho dos Dados do Canal 2, do bit 119 ao bit 112, configuram o tamanho dos dados que são enviados pelo canal 2 do Módulo TRF2.4G. Neste projeto não será utilizado o canal 2.
- Os Bits de Tamanho dos Dados do Canal 1, do bit 111 ao bit 104, configuram o tamanho dos dados que são enviados pelo canal 1 do Módulo TRF2.4G. Neste projeto utilizamos 8 bits de tamanho de dados para enviarmos os comandos aos módulos e 16 bits para a

transmissão dos dados lidos nos canais A/D do microprocessador PIC16F877A.

- O tamanho da palavra de configuração do endereço do canal 2 do Módulo TRF-2.4G é de 5 bytes, do bit 103 ao bit 64. Neste projeto não será utilizado o canal 2.
- O tamanho da palavra de configuração do endereço do Canal 1 do Módulo TRF-2.4G é de 5 bytes, do bit 63 ao bit 24. Neste projeto serão utilizados dois módulos para a transmissão dos dados e somente um módulo para a recepção dos dados. Desta forma a cada envio de comando ou leitura de um dos módulos teremos que configurar o módulo receptor para o endereço do módulo que queremos nos comunicar.
- Os Bits de ADDR_W & CRC configuram o tamanho do endereço do Módulo TRF2.4G, o tamanho do CRC e a habilitação ou não do CRC. Estes bits somente serão utilizados se os módulos estiverem comunicando no modo *ShockBurst*. Os bits 23 a 18 são reservados para o ADDR_W, isto é, o tamanho do endereço do módulo. O tamanho máximo do endereço é 40 bits (5 bytes). O bit 17, CRC_L, é reservado para o tamanho do CRC que será calculado pelo módulo após o envio do último bit de dados, sendo os dados enviados do microcontrolador PIC16F877A ao módulo. Assim que o módulo receptor receber os dados, o mesmo irá retirar o CRC e enviar ao PIC somente os dados. Se este bit estiver com nível lógico 0, será configurado um tamanho de CRC de 8 bit e se o bit estiver com nível lógico 1, será configurado um tamanho de CRC de 16 bits. O bit 16, CRC_EN, é reservado para a habilitação ou não da geração do CRC quando o dado é enviado pelo módulo Transmissor e para a verificação no módulo Receptor. Se este bit estiver com nível lógico 0, o CRC não será habilitado quando o módulo for configurado e se este bit estiver com nível lógico 1, o CRC será habilitado quando o módulo for configurado. Neste projeto será utilizado 40 bits de endereço, 16 bits de CRC e o CRC está habilitado.
- Os Bits de Programação configuram o número de canais de recepção, o modo de transmissão, a velocidade de transmissão, a frequência de oscilação do cristal e a potência de saída. O bit 15, RX2_EN, é reservado para a configuração do número de canais de recepção do

Módulo TRF2.4G. Se este bit estiver com o nível lógico 0, será configurado somente um canal de recepção, se este bit estiver com o nível lógico 1, serão configurados dois canais de recepção. O bit 14, CM, é reservado para a configuração do modo de transmissão. Se este bit estiver com nível lógico 0, o módulo será configurado para a transmissão no Modo Direto, se este bit estiver com nível lógico 1, o módulo será configurado para a transmissão no modo *ShockBurst*. O bit 13, RFDR_SB, é reservado para a configuração da velocidade de transmissão dos dados. Se este bit estiver com nível lógico 0, o Módulo TRF2.4G será configurado para transmitir a uma velocidade de 250kbps, se este bit estiver com nível lógico 1, o Módulo TRF2.4G será configurado para transmitir a uma velocidade de 1Mbps. Os bits 12 a 10, XO_F, são reservados para a configuração da frequência do cristal. A frequência do cristal padrão é 16MHz. Os bits 9 e 8, RF_PWR, são reservados para a configuração da potência de saída, conforme tabela 3-3. Neste projeto estaremos usando um canal de recepção, o modo de transmissão é o *ShockBurst*, a velocidade de transmissão é de 1Mbps, a frequência do cristal é 16MHz e a potência de saída é 0dB.

Tabela 3-2 – Configuração dos bits RF_PWR.

Potência de Saída		
D9	D8	P (dB)
0	0	-20
0	1	-10
1	0	-5
1	1	0

- Os Bits de Frequência de Transmissão e Direção configuram a frequência de transmissão do canal e a direção do Módulo TRF-2.4G. Os bits 7 a 1, RF_CH#, são reservados para a configuração da frequência de operação do módulo, transmissão e recepção. Neste projeto será utilizado a mesma frequência para a comunicação dos módulos, e para a recepção dos dados do módulo de transmissão 1 e do módulo de transmissão 2, somente alterando o endereço dos módulos para a recepção individual de cada módulo. O bit 0, RXEN, é reservado para a configuração da direção do módulo. Se este bit estiver com nível lógico 0, o módulo será configurado para transmissor, e se o bit estiver com nível lógico 1, o módulo será

configurado para receptor. Neste projeto será utilizado uma frequência de transmissão e recepção igual a 2410MHz.

A tabela 3-3 demonstra toda a palavra de configuração do Módulo TRF-2.4G.

Tabela 3-3 – Palavra de Configuração do Módulo TRF-2.4G.

D143 - D136	D135 - D128	D127 - D120
0x8E	0x08	0x1C
D119 - D112	D111 - D104	D103 - D96
0x10	0x10	0xC0
D95 - D88	D87 - D80	D79 - D72
0xAA	0x55	0xAA
D71 - D64	D63 - D56	D55 - D48
0x55	0xA0	0x55
D47 - D40	D39 - D32	D31 - D24
0xAA	0x55	0xAA
D23 - D16	D15 - D8	D7 - D0
0xA3	0x6F	0x14

3.3.6. Temporização

O Módulo TRF-2.4G solicita que sua temporização seja respeitada diante de um dado ou comando enviado a ele. Este procedimento é de grande importância para que o módulo interprete corretamente os comandos e dados que são enviados pelo Microcontrolador PIC16F877A.

Devido à insuficiência de informações contidas no manual do módulo fornecido pela Laipac, que pode ser visto no **Anexo B**, foi detalhada a temporização no **Anexo C**.

3.3.7. Estrutura de Envio dos Dados

O Módulo TRF-2.4G envia os dados de uma forma estruturada. A estrutura é formada do *Preamble*, Endereço, *Payload* e CRC, conforme tabela 3-4. O pacote Endereço + Payload + CRC pode ter o tamanho máximo de 256 bits. Quando o outro módulo recebe o pacote, automaticamente retira o *Preamble*, o Endereço e o CRC e envia apenas o *Payload* para o Microcontrolador PIC16F877A.

Tabela 3-4 – Estrutura de Envio de Dados

PREAMBLE	ENDEREÇO	PAYLOAD	CRC
----------	----------	---------	-----

Ao longo deste projeto é citada a estrutura de envio de dados que será descrita abaixo: (Wenshing - TRW-24G High Frequency Transceiver Module (GFSK), 2004)

- *Preamble* – É automaticamente calculado e inserido ao pacote de dados no modo *ShockBurst*, pelo Módulo TRF-2.4G, antes do envio. O *Preamble* tem o tamanho de 8 bits.
- Endereço – O tamanho do endereço pode ter no máximo 5 bytes, é adicionado no modo *ShockBurst*, para que o Módulo TRF-2.4G de destino saiba que o pacote foi enviado a ele.
- *Payload* – Este é o dado a ser enviado. O tamanho máximo do pode variar, levando em consideração o tamanho do endereço e do CRC que são enviados juntamente no pacote. O tamanho máximo do *Payload* é calculado da seguinte forma: $Payload = 256 - \text{Tamanho do Endereço} - \text{CRC}$.
- CRC – O CRC, *Cyclic redundancy check*, é um código detector de erros. É calculado e anexado à informação que será transmitida e verificada após a recepção, para confirmar se não ocorreram alterações. O CRC é popular por ser simples de implementar em hardware binário, simples de ser analisado matematicamente, e pela eficiência em detectar erros típicos causados por ruído em canais de transmissão. Pode ter o tamanho de 8 ou 16 bits.

A figura 3-17 mostra o fluxograma de como é montada a estrutura de envio de dados pelo Módulo TRF-2.4G.

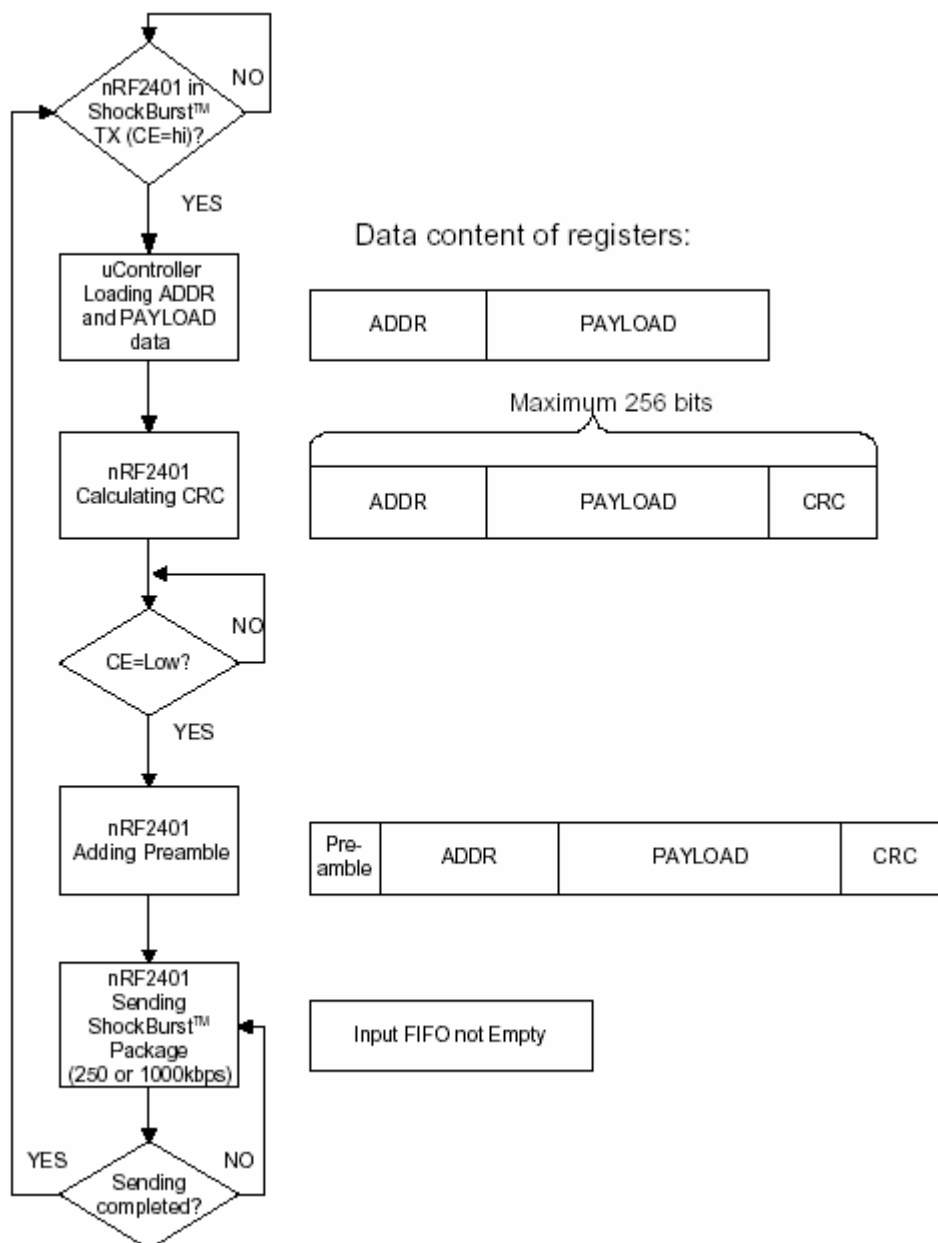


Ilustração 3-17 – Fluxograma de montagem da Estrutura de Envio de Dados no Módulo TRF-2.4G.

Fonte: Nordic Semiconductor ASA - Single Chip 2.4 GHz Transceiver nRF2401

3.3.8. Implementação no Software

Para o sucesso da configuração do Módulo TRF2.4G, devem ser enviados todos os 144 bits de configuração. A temporização e o tempo de *clock* com os bits devem ser perfeitos. Os bits de configuração devem ser enviados do mais significativo ao menos significativo. A forma em que os bits de configuração devem ser enviados ao Módulo TRF2.4G e a temporização dos dados com o pulso de *clock* podem ser vistos na rotina de software abaixo.

// Rotina de Configuração do Módulo TRF2.4G com o endereço do Módulo 1

```
// Transmitindo em 8 Bit
void SetMode_24G(unsigned char Mode, unsigned char freq)
{
    // Se Mode = 1 então TX
    // Se Mode = 0 então RX
    unsigned char Temp;

    TRISB = 0x08;          // Define a PORTB como Output
    DelayMs(1);
    DIR = 1;              // Habilita no circuito a entrada de dados no Módulo TRF-24G
    DelayUs(100);
    CE = 0;
    DelayUs(100);          // Td - Delay Between Edges
    CS = 1;
    DelayUs(100);          // Tcs2data - Delay from CS to Data
    Write_24G(0x8E);        // Reserved for testing
    Write_24G(0x08);        // Reserved for testing
    Write_24G(0x1C);        // Reserved for testing
    Write_24G(0x08);        // Length of Bit Ch 2 8 Bit
    Write_24G(0x08);        // Length of Bit Ch 1 8 Bit
    Write_24G(0xC0);        // Address 5 Byte Ch 2
    Write_24G(0xAA);
    Write_24G(0x55);
    Write_24G(0xAA);
    Write_24G(0x55);
    Write_24G(0xAA);        // Address 5 Byte Ch 1
    Write_24G(0x55);
    Write_24G(0xAA);
    Write_24G(0x55);
    Write_24G(0xAA);
    Write_24G(0xA3);        // Number of Address bit + CRC
    Write_24G(0x6F);        // RF Programming
    switch(freq)
    {
        case 1:
        {
            Temp = Mode==1?0x14:0x15; // Tx Mode e Rx Mode 2410MHz
            break;
        }
    }
    Write_24G(Temp);
    DelayUs(100);
    CS = 0;
}
```

Como pode ser visto na rotina, além de colocarmos o pino CE para o nível lógico 0 e o pino CS para o nível lógico 1, que coloca o Módulo TRF2.4G em modo de configuração. Sempre devemos garantir que ao enviar um bit de configuração, o pino de *clock* deve estar em nível lógico 0. Pois, assim que o bit é enviado pelo microcontrolador PIC16F877A para o módulo devemos dar um pulso de *clock*, para que o módulo entenda que recebeu um bit e assim sucessivamente até o último bit.

Para a escrita de um byte no Módulo TRF-2.4G é utilizada a rotina que converte o byte hexadecimal em 8 bits e envia ao módulo bit a bit, através do pino DATA, dando os pulsos de *clock* corretos. Esta rotina é listada abaixo.

```
// Rotina para escrever no Módulo TRF-24G em 8 Bit
void Write_24G(unsigned char Dado)
{

```



```
    unsigned char i;
    int Out = 0;
    CLK1 = 0;
    for(i=0; i<8; i++)
    {
        Out = Dado & 0x80;
        if (Out)
            DATA = 1;
        else
            DATA = 0;
        DelayUs(100);          // Ts - Setup Time - 100µs
        CLK1 = 1;
        DelayUs(100);          // Th - Hold Time - 100µs
        CLK1 = 0;
        Dado <<= 1;
    }
}
```

3.3.9. Transmissão

Para dar início à transmissão o pino CE deve ser colocado em nível lógico 1 e o pino CS em nível lógico 0. No modo *ShockBurst* o Módulo TRF2.4G recebe um pacote dos dados que inclui o endereço de destino e um *payload* do microcontrolador PIC16F877A. O pacote é transmitido em série (primeiro MSB), sempre dando os pulsos de *clock* temporizados pelo PIC, da mesma maneira que é feito quando o módulo é configurado.

Assim que todo o pacote foi enviado ao dispositivo, o pino CE deve ser colocado em nível lógico 0, para ativar o processo *onboard* no Módulo TRF2.4G. O módulo irá calcular e incluir, ao endereço e aos dados, o preamble e o CRC e só assim o módulo iniciará a transmissão dos dados. O fluxograma da rotina de transmissão é demonstrado na figura 3-18.

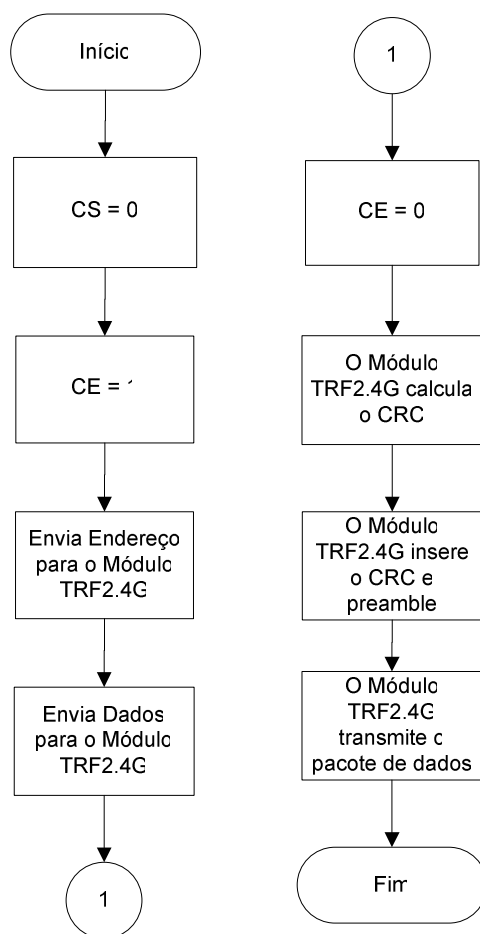


Ilustração 3-18 – Fluxograma de Transmissão do Módulo TRF-2.4G.

3.3.10. Recepção

Para dar início à recepção dos dados, o pino CE deve ser colocado em nível lógico 1 e o pino CS em nível lógico 0. No modo *ShockBurst* o Módulo TRF2.4G recebe um pacote de dados que inclui o endereço de destino, o *payload*, o *preamble* e o CRC. Assim que todo o pacote for recebido, o módulo retira automaticamente o endereço de destino, o *preamble* e o CRC. Acabado este processo ele coloca o pino DR1 para nível lógico 1, sinalizando que dados foram recebidos.

Assim que o microcontrolador PIC16F877A detectar que o pino DR1 do Módulo TRF2.4G está em nível lógico 1, o PIC deverá colocar o pino CE para nível lógico 0 durante a recepção dos dados pelo PIC. Porém o pino CE poderá ficar com o nível lógico em 1 durante a recepção dos dados pelo PIC, pois isso somente acarretará um maior consumo de corrente elétrica, aproximadamente 18mA.

O pacote de dados é lido, pelo microcontrolador PIC16F877A, em série (primeiro MSB), sempre sincronizando os pulsos de *clock* com o bit recebido. Assim que o Módulo TRF2.4G enviar todos os dados para o PIC, o pino DR1 do módulo

será levado para o nível lógico 0 e o PIC deverá colocar o pino CE para o nível lógico 1 para que o módulo comece a monitorar o ar a espera de novos pacotes de dados. O fluxograma da rotina de recepção é demonstrado na figura 3-19.

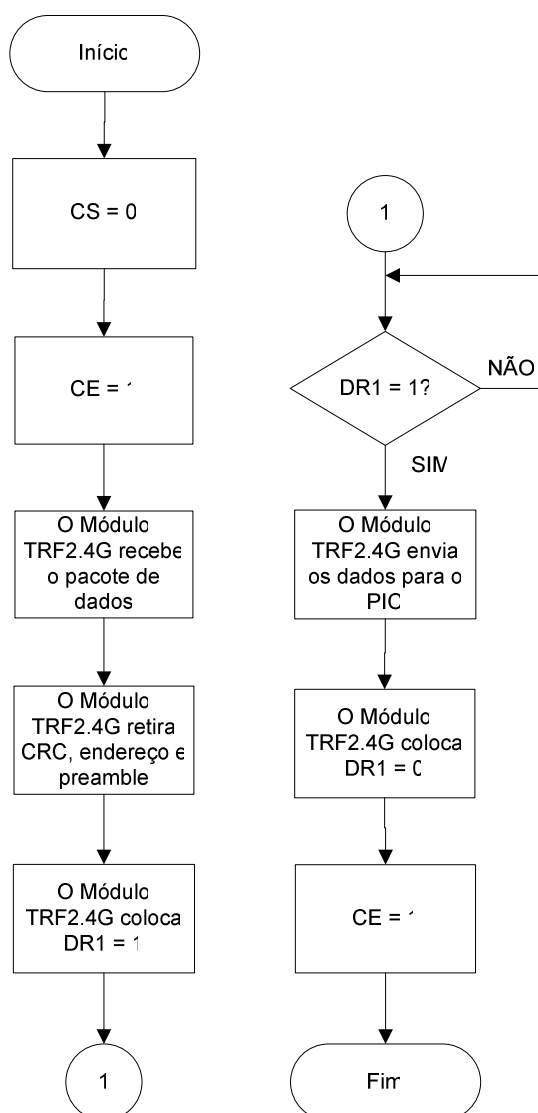


Ilustração 3-19 – Fluxograma de Recepção do Módulo TRF-2.4G.

3.4. Testes e Validação

3.4.1. Lógica de Funcionamento do Projeto

Para o bom entendimento da lógica de funcionamento do projeto é apresentado o fluxograma do módulo de controle na figura 3-21, e o fluxograma dos módulos de aquisição na figura 3-20.

A lógica de funcionamento do módulo de aquisição é a seguinte: o módulo recebe o comando enviado pelo módulo de controle, interpreta o comando, adquire o dado solicitado e envia ao módulo de controle. O fluxograma desta rotina é vista na figura 3-20.

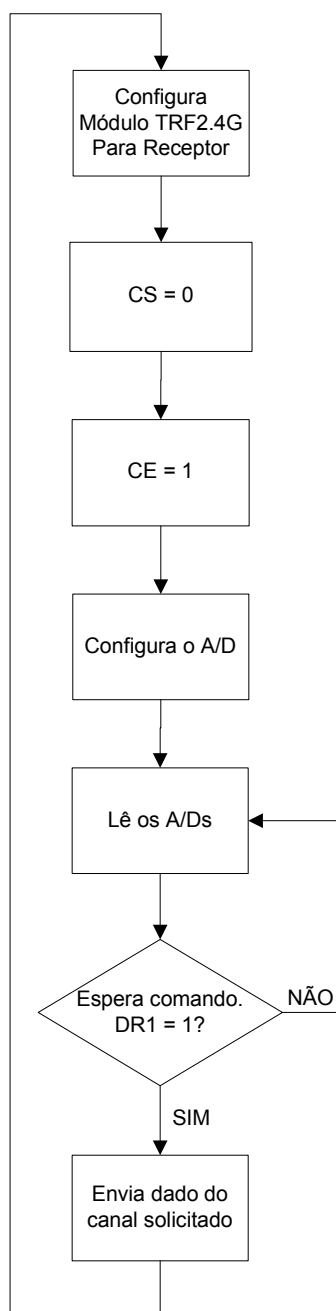


Ilustração 3-20 – Lógica de Funcionamento do Módulo de Aquisição.

A lógica de funcionamento do módulo de controle é a seguinte: o módulo recebe comandos do PC que são interpretados e enviados aos módulos de aquisição. Estes entendendo o comando a ser executado, respondem ao módulo de controle o que foi solicitado. Assim que o módulo de controle recebe os dados envia para o PC através da porta serial. O fluxograma desta rotina é vista na figura 3-21.

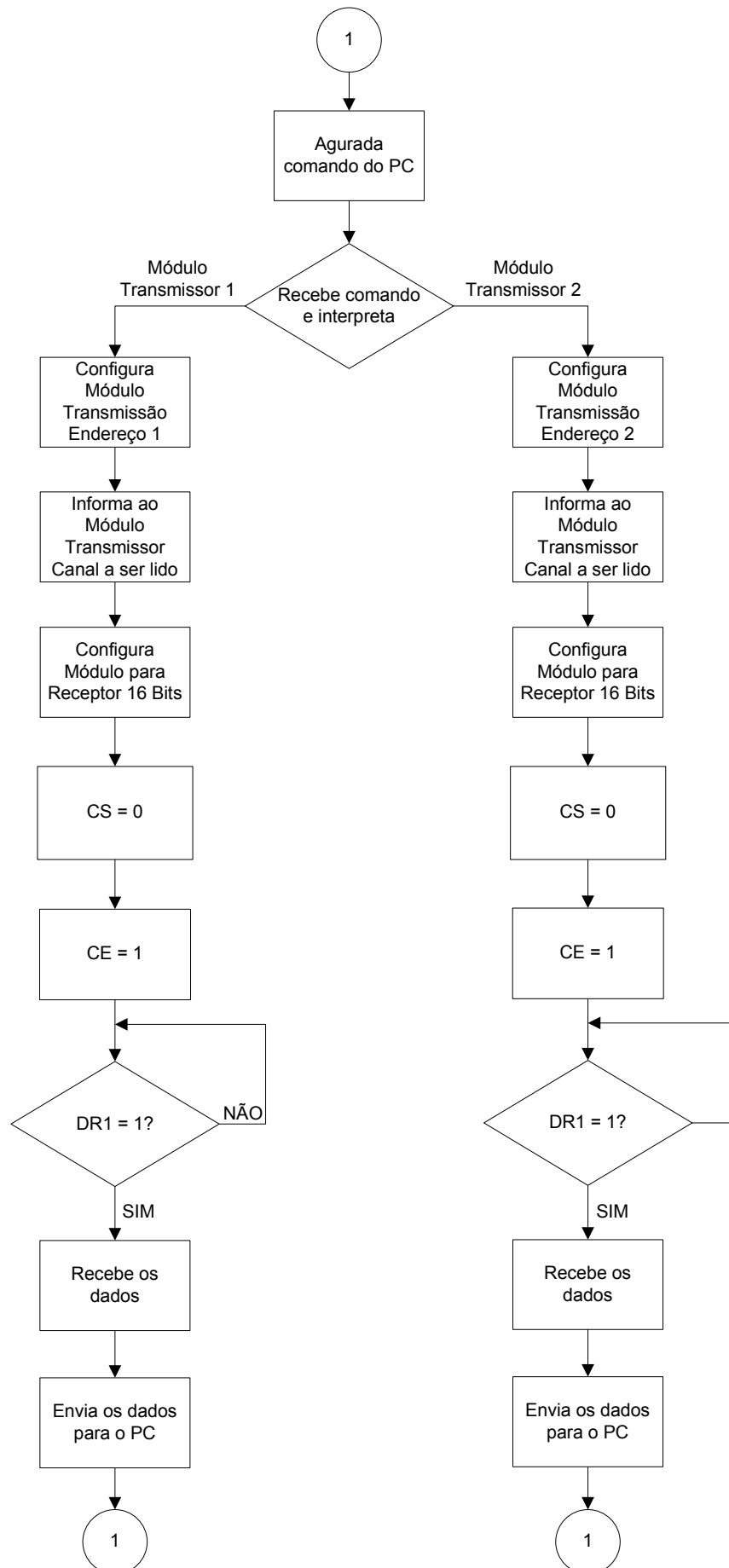


Ilustração 3-21 – Lógica de Funcionamento do Módulo de Controle.

3.4.2. Testes do Circuito

Inicialmente, os circuitos do Microcontrolador PIC16F877A e do Módulo TRF-2.4G foram montados em um protoboard. Foi implementada uma rotina no PIC para que os pinos de saída ligados ao módulo oscilassem. Com o auxílio do osciloscópio foram medidos os sinais que saiam do PIC até o módulo. Foram corrigidos alguns erros iniciais do projeto e certificou-se que os sinais que chegam ao módulo estavam com o nível de tensão correto, que deve ser de $3,0 \pm 0,3$ Vcc. Para o teste dos pinos de entrada do PIC foi colocado um sinal, no pino DATA do módulo, com nível de tensão igual a 3,0 Vcc. Desta forma, certificamos que o PIC e o circuito de conversão de nível de tensão estavam funcionando corretamente. Assim não havia um perigo maior de danificar o módulo, já que ele não suporta tensões maiores do que $3,0 \pm 0,3$ Vcc.

Da mesma maneira que certificamos a conversão dos níveis de tensão no circuito, foi feita a correção na rotina de *Delay*.

3.4.3. Validação da Transmissão

Para os testes de validação da transmissão foram montados dois circuitos com o um PIC e um Módulo TRF-2.4G, cada um. Foi desenvolvida a rotina de configuração dos módulos, transmissão e recepção. Um do módulo foi configurado apenas para transmitir sempre o mesmo dado e o outro para receber o dado e enviar para a tela do PC.

3.4.4. Funcionamento do Projeto

Para o controle dos Módulos TRF-2.4G e a apresentação dos dados adquiridos na tela do PC, foi desenvolvido um software no Borland C++ Builder.

Este software apresenta uma tela gráfica, figura 3-22, onde pode ser escolhido o módulo e o canal a qual se deseja obter os dados, o tempo de amostragem e o número de amostras.



Ilustração 3-22 – Tela Gráfica.

Para dar início a aquisição dos dados, deve ser selecionada a opção *Habilita Leitura*. Após, é escolhido o Módulo 1, Módulo 2 ou ambos e qual o canal a ser lido. O software informa ao PIC, pela porta serial, qual módulo e canal ele deseja ler e após o PIC envia os dados solicitados ao software que exibe na tela e plota no gráfico, correspondente ao módulo.

3.4.5. Teste do A/D

O PIC16F877A dispõe de um conversor A/D de 10 bits com oito canais de entrada multiplexadas. A tensão de referência utilizada é a mesma tensão de alimentação do PIC, que neste projeto é de 5 Vcc.

Para os teste e a validação do conversor A/D foi utilizado um circuito com um potenciômetro que varia a tensão no pino de entrada analógica de 0 a 5 Vcc, que pode ser visto na figura 3-23.

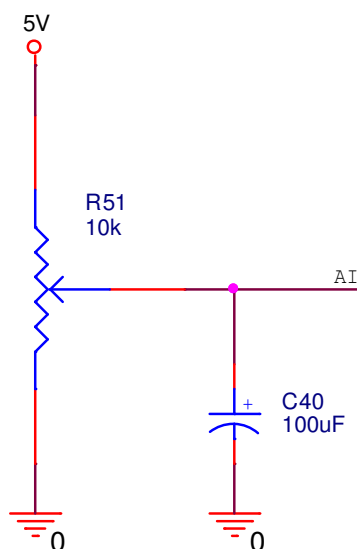


Ilustração 3-23 – Circuito teste do A/D.

Sabe-se que quando a tensão na entrada analógica do PIC é igual a 0 Vcc, o A/D coloca nos seus registradores ADRESH + ADRESL um valor igual a 0. Quando a tensão na entrada analógica é igual a 5 Vcc, o valor do ADRESH + ADRESL é igual a 1023. Assim, pode-se chegar a uma fórmula para o cálculo do valor de ADRESH + ADRESL em relação à tensão na entrada analógica.

$$N = \frac{(V \times 1023)}{5}$$

Sendo:

N = Valor da conversão de 10 bits do A/D (ADRESH + ADRESL).

V = Tensão na Entrada Analógica do PIC.

Utilizando esta fórmula e com o auxílio do osciloscópio, variamos a tensão da entrada analógica e comparamos ao valor do A/D impresso na tela no PC.



4. CONCLUSÃO

Este trabalho teve por objetivo desenvolver um sistema de aquisição de dados de instrumentação e a transmissão dos dados para uma estação de trabalho e controle, onde os mesmos podem ser visualizados na tela do computador.

A transmissão é feita através de um link de radiofrequência utilizando-se os módulos TRF-2.4G, os quais um foi conectado ao computador e outros dois estão posicionados em diferentes locais.

Os sensores estão conectados ao microcontrolador através de um circuito de implementação. No microcontrolador são feitas as conversões A/D de 10 bits e o envio das informações. Para tanto, o microcontrolador está programado com um software para constantemente ler, converter e enviar as informações para que sejam transmitidas pelo módulo TRF-2.4G.

Constatou-se que é possível a transmissão dos dados de instrumentação através de uma rede sem fio. Assim sendo, este trabalho pode ser implementado em uma planta industrial que tenha a necessidade da aquisição de dados sem fio.

O trabalho apresentou os resultados esperados e atendeu ao que inicialmente havia sido proposto a ser desenvolvido.



5. REFERÊNCIAS

BALBINOT, Alexandre & BRUSAMARELLO, Valner João - *Instrumentação e Fundamentos de Medidas* - Volume 1 - Rio de Janeiro:LTC, 2006

Laipac Technology Inc. - *TRF-2.4G Transceiver Data Sheet*

Microchip - *PIC16F877A Data Sheet*

National Instruments - *Measuring Temperature with Thermocouples – A Tutorial – Application Note 043*

National Instruments - *Measuring Temperature with na RTD or Thermistor – Application Note 046*

Nordic Semiconductor ASA - *Single Chip 2.4 GHz Transceiver nRF2401*

Wenshing - *TRW-24G High Frequency Transceiver Module (GFSK)*



OBRAS CONSULTADAS

ENGST, Adam & FLEISHMAN, Glenn - *Kit do Iniciante em Redes Sem Fio* - 2a. Ed. - São Paulo:Pearson Education do Brasil Ltda, 2005

SOUZA, David José de - *Desbravando o PIC* - Edição 1 - São Paulo:Editora Érica, 2000

PEREIRA, Fábio - *PIC - Programação em C* - 5º Edição - São Paulo:Editora Érica, 2006

HELD, Gilbert - *Comunicação de Dados* - 6a. Ed. - Rio de Janeiro:Editora Campus, 2000

HELGERT, Hermann J. - *Integrated Services Digital Networks* - Washington D.C:Addison-Wesley Publishing Company, 1991

HAYKIN, Simon - *Sistemas de Comunicação Analógicos e Digitais* - 4a. Ed. - Porto Alegre:Bookman, 2004

ZANCO, Wagner da Silva - *Microcontroladores PIC* - 1º Edição - São Paulo:Editora Érica, 2006

Delta Controls Corp - *How to Determine Temperature by Measuring the Output Millivoltage of a Thermocouple* – AN-HTP15

Honeywell Micro Switch Sensing and Control - *Reference and Application Data – Temperature Sensors*

www.microchip.com

www.protocols.com

www.tato.ind.br

www.wenshing.com.tw



GLOSSÁRIO

Analógico: Diz respeito à grandeza que varia continuamente e cujo valor não está dividido em unidades discretas.

ASCII: Código de 7 bits que representa dígitos decimais, letras do alfabeto, símbolos e caracteres de controle.

Binário: Sistema de numeração na base dois que usa e dígitos, 0 e 1.

Bit: Uma contração do dígito binário. Cada posição em um número binário é um bit.

Byte: Conjunto de oito bits.

Bps: Bits por segundo, uma medida de velocidade com que equipamentos digitais podem transferir dados, na forma de um bit de cada vez.

Clock: Uma forma de onda retangular e contínua usada para temporização.

CPU: Unidade Central de Processamento. A parte do computador que decodifica as instruções buscadas na memória, e as executa.

Decimal: Sistema de numeração de base 10 que usa os dígitos de 0 a 9.

Digital: Diz respeito ao dado que não é naturalmente contínuo. Ele muda em degraus separados. O dado é representado por zeros ou uns.

EEPROM: Memória apenas de leitura que é programada e apagada eletricamente. Memória não volátil que pode ser programada e apagada por sinais elétricos.

Frequência: O número de ciclos completos de uma forma de onda em 1 segundo. Ela é medida em Hertz.

Hexadecimal: Sistema de numeração de base 16 que usa os 16 dígitos, de 0 a 9 e de A à F.

LSB: Bit menos significativo. O bit mais à direita em um número binário.

Módulos Híbridos: Um par de módulos transmissor-receptor já montados com componentes eletrônicos em uma placa de circuito impresso.

MSB: Bit mais significativo. O bit mais à esquerda em número binário.



Protocolo: Conjunto de regras que governam a troca de informações entre dois ou mais processos.

RadioFrequência: Frequência de radiações eletromagnéticas utilizadas em radiotransmissão, e que está compreendida aproximadamente entre 10kHz e 100000MHz.

RAM: Memória de acesso aleatório. A memória de leitura e escrita em um circuito microprocessado.

Wireless: Sistema que utiliza meios de comunicação sem fio.

APÊNDICE A – ROTINAS DO SOFTWARE PIC

Rotina Main.c

```
#include <stdio.h>
#include <pic1687x.h>
#include "serial.h"
#include "tr24g.h"
#include "delay.h"
#include "adc.h"

#define TX1 0
#define TX2 1
#define RX 2
#define MODO RX

#define CS RB0 // Define o Pino 33 (B0) como CS
#define CE RB1 // Define o Pino 34 (B1) como CE
#define CLK1 RB2 // Define o Pino 35 (B2) como CLK1
#define DR1 RB3 // Define o Pino 36 (B3) como DR
#define DATA RB4 // Define o Pino 37 (B4) como DATA
#define DIR RB5 // Define o Pino 38 (B5) como DIR

#if MODO==0
void main(void)
// Se MODO = 0 (TX1), Transmissor 1
{
    unsigned char Comando;

    unsigned int ValorCanal0;
    unsigned int ValorCanal1;
    unsigned int ValorCanal2;
    unsigned int ValorCanal3;

    TRISB = 0x08; // Define RB3 como Entrada e os outros Pinos como Saída
    Init_24G();
    DelayMs(3);
    SetMode_24G(0,1); // Configura o Módulo TRF-24G para Receptor
    TRISB = 0x18; // Define RB3 e RB4 como Entrada e os outros como Saída
    CS = 0;
    DelayUs(100); // Td - Delay Between Edges
    CE = 1;
    DelayUs(100); // Tsby2rx
    Set_ADC();

    while(1)
    {
        ValorCanal0 = Le_ADC(0);
        ValorCanal1 = Le_ADC(1);
        ValorCanal2 = Le_ADC(2);
        ValorCanal3 = Le_ADC(3);

        if(DR1==1)
        {
```

```

Comando = Read_24G();
SetMode_24G_16Bit(1,1);      // Configura o Módulo TRF-24G para

Transmissor

DelayMs(10);
switch(Comando)
{
    case 0:
        Send_24G_16Bit(ValorCanal0);
        break;
    case 1:
        Send_24G_16Bit(ValorCanal1);
        break;
    case 2:
        Send_24G_16Bit(ValorCanal2);
        break;
    case 3:
        Send_24G_16Bit(ValorCanal3);
        break;
    default:
        Send_24G_16Bit(0xFFFF);
}
SetMode_24G(0,1);      // Configura o Módulo TRF-24G para

Receptor

TRISB = 0x18;      // Define RB3 e RB4 como Entrada e os
outros Pinos como Saída

CS = 0;
DelayUs(100);      // Td - Delay Between Edges
CE = 1;
DelayUs(100);      // Tsby2rx
}
}
}
#elif MODO==1
void main(void)
// Se MODO = 1 (TX2), Transmissor 2
{
    unsigned char Comando;

    unsigned int ValorCanal0;
    unsigned int ValorCanal1;
    unsigned int ValorCanal2;
    unsigned int ValorCanal3;

    TRISB = 0x08;      // Define RB3 como Entrada e os outros Pinos como Saída
    Init_24G();
    DelayMs(3);
    SetMode_24G_ADD2(0,1);      // Configura o Módulo TRF-24G para Receptor
    TRISB = 0x18;      // Define RB3 e RB4 como Entrada e os outros Pinos como Saída
    CS = 0;
    DelayUs(100);      // Td - Delay Between Edges
    CE = 1;
    DelayUs(100);      // Tsby2rx
    Set_ADC();

    while(1)
    {
        ValorCanal0 = Le_ADC(0);
        ValorCanal1 = Le_ADC(1);
        ValorCanal2 = Le_ADC(2);
        ValorCanal3 = Le_ADC(3);

        if(DR1==1)
        {
            Comando = Read_24G();
            SetMode_24G_16Bit_ADD2(1,1);      // Configura o Módulo TRF-
24G para Transmissor

```

```
        DelayMs(10);
        switch(Comando)
        {
            case 0:
                Send_24G_16Bit_ADD2(ValorCanal0);
                break;
            case 1:
                Send_24G_16Bit_ADD2(ValorCanal1);
                break;
            case 2:
                Send_24G_16Bit_ADD2(ValorCanal2);
                break;
            case 3:
                Send_24G_16Bit_ADD2(ValorCanal3);
                break;
            default:
                Send_24G_16Bit_ADD2(0xFFFF);
        }
        SetMode_24G_ADD2(0,1);           // Configura o Módulo TRF-
24G para Receptor
        TRISB = 0x18;                   // Define RB3 e RB4 como
Entrada e os outros Pinos como Saída
        CS = 0;
        DelayUs(100);                   // Td - Delay Between Edges
        CE = 1;
        DelayUs(100);                   // Tsby2rx
    }
}
#elif MODO==2
void main(void)
// Se MODO = 2 (RX), Receptor
{
    unsigned int tmp = 0;
    unsigned char Modulo , Canal;
    //Amostras = 0x32;
    //Intervalo = 0xC8;

    INIT_COMMS();
    Init_24G();
    DelayMs(3);
    while(1)
    {
        Modulo = getch();
        Canal = getch();

        switch(Modulo)
        {
            case 0x01:
            {
                SetMode_24G(1,1);        // Configura o Módulo TRF-24G para Transmissor
                Send_24G(Canal);
                DelayMs(1);
                SetMode_24G_16Bit(0,1);  // Configura o Módulo TRF-24G para Receptor
                TRISB = 0x18;            // Define RB3 e RB4 como Entrada e os outros Saída
                CS = 0;
                DelayUs(100);             // Td - Delay Between Edges
                CE = 1;
                DelayUs(100);             // Tsby2rx
                while(DR1==0);           // Espera o Módulo TRF-24G receber os dados
                tmp = Read_24G_16Bit();
                printf("I");
                putchar((char)((tmp&0xFF00)>>8));
                putchar((char)(tmp&0x00FF));
                printf("F");
                break;
            }
        }
    }
}
```



```
    }
    case 0x02:
    {
        SetMode_24G_ADD2(1,1);        // Configura o Módulo TRF-24G para
Transmissor
        Send_24G_ADD2(Canal);
        DelayMs(1);
        SetMode_24G_16Bit_ADD2(0,1);    // Configura o Módulo TRF-
24G para Receptor
        TRISB = 0x18;                // Define RB3 e RB4 como Entrada e os
outros Pinos como Saída
        CS = 0;
        DelayUs(100);                // Td - Delay Between Edges
        CE = 1;
        DelayUs(100);                // Tsby2rx
        while(DR1==0);                // Espera o Módulo TRF-24G receber os dados
        tmp = Read_24G_16Bit();
        printf("I");
        putch((char)((tmp&0xFF00)>>8));
        putch((char)(tmp&0x00FF));
        printf("F");
        break;
    }
    default:
        printf("I");
        putch(0xFF);
        putch(0xFF);
        printf("F");
    }
}
}
}
#endif
```

Rotina TRF-2.4G.c

```
#include <pic1687x.h>
#include "delay.h"
```

```
#define CS          RB0    // Define o Pino 33 (B0) como CS
#define CE          RB1    // Define o Pino 34 (B1) como CE
#define CLK1        RB2    // Define o Pino 35 (B2) como CLK1
#define DR1         RB3    // Define o Pino 36 (B3) como DR
#define DATA       RB4    // Define o Pino 37 (B4) como DATA
#define DIR         RB5    // Define o Pino 38 (B5) como DIR
```

```
// Configura todos os pinos para Zero
```

```
void Init_24G(void)
```

```
{
    DelayMs(10);
    TRISB = 0x08;        // Define a PORTB como Output
    CE = 0;
    CS = 0;
    CLK1 = 0;
    DATA = 0;
    DIR = 0;              // Habilita no circuito a saída de dados no Módulo TRF-24G
    DelayMs(10);
}
```

```
// Rotina para escrever no Módulo TRF-24G em 8 Bit
```

```
void Write_24G(unsigned char Dado)
```

```
{
    unsigned char i;
    int Out = 0;
    CLK1 = 0;
    for(i=0; i<8; i++)
    {
```



```
        Out = Dado & 0x80;
        if (Out)
            DATA = 1;
        else
            DATA = 0;
        DelayUs(100);          // Ts - Setup Time
        CLK1 = 1;
        DelayUs(100);          // Th - Hold Time
        CLK1 = 0;
        Dado <<= 1;
    }
}

// Rotina para escrever no Módulo TRF-24G em 16 Bit
void Write_24G_16Bit(unsigned int Dado)
{
    unsigned int i;
    int Out = 0;
    CLK1 = 0;
    for(i=0; i<16; i++)
    {
        Out = Dado & 0x8000;
        if (Out)
            DATA = 1;
        else
            DATA = 0;
        DelayUs(100);          // Ts - Setup Time
        CLK1 = 1;
        DelayUs(100);          // Th - Hold Time
        CLK1 = 0;
        Dado <<= 1;
    }
}

// Rotina de Configuração do Módulo TRF-24G com o endereço do Módulo 1
// Transmitindo em 8 Bit
void SetMode_24G(unsigned char Mode, unsigned char freq)
{
    // Se Mode = 1 então TX
    // Se Mode = 0 então RX
    unsigned char Temp;

    TRISB = 0x08;              // Define a PORTB como Output
    DelayMs(1);
    DIR = 1;                    // Habilita no circuito a entrada de dados no Módulo TRF-24G
    DelayUs(100);
    CE = 0;
    DelayUs(100);               // Td - Delay Between Edges
    CS = 1;
    DelayUs(100);               // Tcs2data - Delay from CS to Data
    Write_24G(0x8E);            // Reserved for testing
    Write_24G(0x08);            // Reserved for testing
    Write_24G(0x1C);            // Reserved for testing
    Write_24G(0x08);            // Length of Bit Ch 2 8 Bit
    Write_24G(0x08);            // Length of Bit Ch 1 8 Bit
    Write_24G(0xC0);            // Address 5 Byte Ch 2
    Write_24G(0xAA);
    Write_24G(0x55);
    Write_24G(0xAA);
    Write_24G(0x55);
    Write_24G(0xAA);            // Address 5 Byte Ch 1
    Write_24G(0x55);
    Write_24G(0xAA);
    Write_24G(0x55);
    Write_24G(0xAA);
    Write_24G(0xA3);            // Number of Address bit + CRC
}
```



```
Write_24G(0x6F);      // RF Programming
switch(freq)
{
    case 0:
    {
        Temp = Mode==1?0x0A:0x0B; // Tx Mode e Rx Mode 2405MHz
        break;
    }
    case 1:
    {
        Temp = Mode==1?0x14:0x15; // Tx Mode e Rx Mode 2410MHz
        break;
    }
    case 2:
    {
        Temp = Mode==1?0x1E:0x1F; // Tx Mode e Rx Mode 2415MHz
        break;
    }
    case 3:
    {
        Temp = Mode==1?0x28:0x29; // Tx Mode e Rx Mode 2420MHz
        break;
    }
    case 4:
    {
        Temp = Mode==1?0x32:0x33; // Tx Mode e Rx Mode 2425MHz
        break;
    }
    case 5:
    {
        Temp = Mode==1?0x3C:0x3D; // Tx Mode e Rx Mode 2430MHz
        break;
    }
    case 6:
    {
        Temp = Mode==1?0x46:0x47; // Tx Mode e Rx Mode 2435MHz
        break;
    }
    case 7:
    {
        Temp = Mode==1?0x50:0x51; // Tx Mode e Rx Mode 2440MHz
        break;
    }
    case 8:
    {
        Temp = Mode==1?0x5A:0x5B; // Tx Mode e Rx Mode 2445MHz
        break;
    }
    case 9:
    {
        Temp = Mode==1?0x68:0x69; // Tx Mode e Rx Mode 2452MHz
        break;
    }
    case 10:
    {
        Temp = Mode==1?0x6E:0x6F; // Tx Mode e Rx Mode 2455MHz
        break;
    }
    case 11:
    {
        Temp = Mode==1?0x78:0x79; // Tx Mode e Rx Mode 2460MHz
        break;
    }
    case 12:
    {
        Temp = Mode==1?0x82:0x83; // Tx Mode e Rx Mode 2465MHz
    }
}
```



```
        break;
    }
    case 13:
    {
        Temp = Mode==1?0x90:0x91; // Tx Mode e Rx Mode 2472MHz
        break;
    }
    case 14:
    {
        Temp = Mode==1?0x96:0x97; // Tx Mode e Rx Mode 2475MHz
        break;
    }
    case 15:
    {
        Temp = Mode==1?0xA0:0xA1; // Tx Mode e Rx Mode 2480MHz
        break;
    }
    }
    Write_24G(Temp);
    DelayUs(100);
    CS = 0;
}

// Rotina de Configuração do Módulo TRF-24G com o endereço do Módulo 1
// Transmitindo em 16 Bit
void SetMode_24G_16Bit(unsigned char Mode, unsigned char freq)
{
    // Se Mode = 1 então TX
    // Se Mode = 0 então RX
    unsigned char Temp;

    TRISB = 0x08;          // Define a PORTB como Output
    DelayMs(1);
    DIR = 1;               // Habilita no circuito a entrada de dados no Módulo TRF-24G
    DelayUs(100);
    CE = 0;
    DelayUs(100);          // Td - Delay Between Edges
    CS = 1;
    DelayUs(100);          // Tcs2data - Delay from CS to Data
    Write_24G(0x8E);       // Reserved for testing
    Write_24G(0x08);       // Reserved for testing
    Write_24G(0x1C);       // Reserved for testing
    Write_24G(0x10);       // Length of Bit Ch 2 16 Bit
    Write_24G(0x10);       // Length of Bit Ch 1 16 Bit
    Write_24G(0xC0);       // Address 5 Byte Ch 2
    Write_24G(0xAA);
    Write_24G(0x55);
    Write_24G(0xAA);
    Write_24G(0x55);
    Write_24G(0xAA);       // Address 5 Byte Ch 1
    Write_24G(0x55);
    Write_24G(0xAA);
    Write_24G(0x55);
    Write_24G(0xAA);
    Write_24G(0xA3);       // Number of Address bit + CRC
    Write_24G(0x6F);       // RF Programming
    switch(freq)
    {
        case 0:
        {
            Temp = Mode==1?0x0A:0x0B; // Tx Mode e Rx Mode 2405MHz
            break;
        }
        case 1:
        {
            Temp = Mode==1?0x14:0x15; // Tx Mode e Rx Mode 2410MHz
```

```
        break;
    }
    case 2:
    {
        Temp = Mode==1?0x1E:0x1F; // Tx Mode e Rx Mode 2415MHz
        break;
    }
    case 3:
    {
        Temp = Mode==1?0x28:0x29; // Tx Mode e Rx Mode 2420MHz
        break;
    }
    case 4:
    {
        Temp = Mode==1?0x32:0x33; // Tx Mode e Rx Mode 2425MHz
        break;
    }
    case 5:
    {
        Temp = Mode==1?0x3C:0x3D; // Tx Mode e Rx Mode 2430MHz
        break;
    }
    case 6:
    {
        Temp = Mode==1?0x46:0x47; // Tx Mode e Rx Mode 2435MHz
        break;
    }
    case 7:
    {
        Temp = Mode==1?0x50:0x51; // Tx Mode e Rx Mode 2440MHz
        break;
    }
    case 8:
    {
        Temp = Mode==1?0x5A:0x5B; // Tx Mode e Rx Mode 2445MHz
        break;
    }
    case 9:
    {
        Temp = Mode==1?0x68:0x69; // Tx Mode e Rx Mode 2452MHz
        break;
    }
    case 10:
    {
        Temp = Mode==1?0x6E:0x6F; // Tx Mode e Rx Mode 2455MHz
        break;
    }
    case 11:
    {
        Temp = Mode==1?0x78:0x79; // Tx Mode e Rx Mode 2460MHz
        break;
    }
    case 12:
    {
        Temp = Mode==1?0x82:0x83; // Tx Mode e Rx Mode 2465MHz
        break;
    }
    case 13:
    {
        Temp = Mode==1?0x90:0x91; // Tx Mode e Rx Mode 2472MHz
        break;
    }
    case 14:
    {
        Temp = Mode==1?0x96:0x97; // Tx Mode e Rx Mode 2475MHz
        break;
    }
```



```
    }
    case 15:
    {
        Temp = Mode==1?0xA0:0xA1; // Tx Mode e Rx Mode 2480MHz
        break;
    }
}
Write_24G(Temp);
DelayUs(100);
CS = 0;
}

// Rotina de Configuração do Módulo TRF-24G com o endereço do Módulo 2
// Transmitindo em 8 Bit
void SetMode_24G_ADD2(unsigned char Mode, unsigned char freq)
{
    // Se Mode = 1 então TX
    // Se Mode = 0 então RX
    unsigned char Temp;

    TRISB = 0x08;          // Define a PORTB como Output
    DelayMs(1);
    DIR = 1;               // Habilita no circuito a entrada de dados no Módulo TRF-24G
    DelayUs(100);
    CE = 0;
    DelayUs(100);          // Td - Delay Between Edges
    CS = 1;
    DelayUs(100);          // Tcs2data - Delay from CS to Data
    Write_24G(0x8E);        // Reserved for testing
    Write_24G(0x08);        // Reserved for testing
    Write_24G(0x1C);        // Reserved for testing
    Write_24G(0x08);        // Length of Bit Ch 2 8 Bit
    Write_24G(0x08);        // Length of Bit Ch 1 8 Bit
    Write_24G(0xC0);        // Address 5 Byte Ch 2
    Write_24G(0xAA);
    Write_24G(0x55);
    Write_24G(0xAA);
    Write_24G(0x55);
    Write_24G(0xA0);        // Address 5 Byte Ch 1
    Write_24G(0x55);
    Write_24G(0xAA);
    Write_24G(0x55);
    Write_24G(0xAA);
    Write_24G(0xA3);        // Number of Address bit + CRC
    Write_24G(0x6F);        // RF Programming
    switch(freq)
    {
        case 0:
        {
            Temp = Mode==1?0x0A:0x0B; // Tx Mode e Rx Mode 2405MHz
            break;
        }
        case 1:
        {
            Temp = Mode==1?0x14:0x15; // Tx Mode e Rx Mode 2410MHz
            break;
        }
        case 2:
        {
            Temp = Mode==1?0x1E:0x1F; // Tx Mode e Rx Mode 2415MHz
            break;
        }
        case 3:
        {
            Temp = Mode==1?0x28:0x29; // Tx Mode e Rx Mode 2420MHz
            break;
        }
    }
}
```



```
    }
    case 4:
    {
        Temp = Mode==1?0x32:0x33; // Tx Mode e Rx Mode 2425MHz
        break;
    }
    case 5:
    {
        Temp = Mode==1?0x3C:0x3D; // Tx Mode e Rx Mode 2430MHz
        break;
    }
    case 6:
    {
        Temp = Mode==1?0x46:0x47; // Tx Mode e Rx Mode 2435MHz
        break;
    }
    case 7:
    {
        Temp = Mode==1?0x50:0x51; // Tx Mode e Rx Mode 2440MHz
        break;
    }
    case 8:
    {
        Temp = Mode==1?0x5A:0x5B; // Tx Mode e Rx Mode 2445MHz
        break;
    }
    case 9:
    {
        Temp = Mode==1?0x68:0x69; // Tx Mode e Rx Mode 2452MHz
        break;
    }
    case 10:
    {
        Temp = Mode==1?0x6E:0x6F; // Tx Mode e Rx Mode 2455MHz
        break;
    }
    case 11:
    {
        Temp = Mode==1?0x78:0x79; // Tx Mode e Rx Mode 2460MHz
        break;
    }
    case 12:
    {
        Temp = Mode==1?0x82:0x83; // Tx Mode e Rx Mode 2465MHz
        break;
    }
    case 13:
    {
        Temp = Mode==1?0x90:0x91; // Tx Mode e Rx Mode 2472MHz
        break;
    }
    case 14:
    {
        Temp = Mode==1?0x96:0x97; // Tx Mode e Rx Mode 2475MHz
        break;
    }
    case 15:
    {
        Temp = Mode==1?0xA0:0xA1; // Tx Mode e Rx Mode 2480MHz
        break;
    }
    }
    Write_24G(Temp);
    DelayUs(100);
    CS = 0;
}
```



```
// Rotina de Configuração do Módulo TRF-24G com o endereço do Módulo 2
// Transmitindo em 16 Bit
void SetMode_24G_16Bit_ADD2(unsigned char Mode, unsigned char freq)
{
    // Se Mode = 1 então TX
    // Se Mode = 0 então RX
    unsigned char Temp;

    TRISB = 0x08;          // Define a PORTB como Output
    DelayMs(1);
    DIR = 1;              // Habilita no circuito a entrada de dados no Módulo TRF-24G
    DelayUs(100);
    CE = 0;
    DelayUs(100);          // Td - Delay Between Edges
    CS = 1;
    DelayUs(100);          // Tcs2data - Delay from CS to Data
    Write_24G(0x8E);        // Reserved for testing
    Write_24G(0x08);        // Reserved for testing
    Write_24G(0x1C);        // Reserved for testing
    Write_24G(0x10);        // Length of Bit Ch 2 16 Bit
    Write_24G(0x10);        // Length of Bit Ch 1 16 Bit
    Write_24G(0xC0);        // Address 5 Byte Ch 2
    Write_24G(0xAA);
    Write_24G(0x55);
    Write_24G(0xAA);
    Write_24G(0x55);
    Write_24G(0xA0);        // Address 5 Byte Ch 1
    Write_24G(0x55);
    Write_24G(0xAA);
    Write_24G(0x55);
    Write_24G(0xAA);
    Write_24G(0xA3);        // Number of Address bit + CRC
    Write_24G(0x6F);        // RF Programming
    switch(freq)
    {
        case 0:
        {
            Temp = Mode==1?0x0A:0x0B; // Tx Mode e Rx Mode 2405MHz
            break;
        }
        case 1:
        {
            Temp = Mode==1?0x14:0x15; // Tx Mode e Rx Mode 2410MHz
            break;
        }
        case 2:
        {
            Temp = Mode==1?0x1E:0x1F; // Tx Mode e Rx Mode 2415MHz
            break;
        }
        case 3:
        {
            Temp = Mode==1?0x28:0x29; // Tx Mode e Rx Mode 2420MHz
            break;
        }
        case 4:
        {
            Temp = Mode==1?0x32:0x33; // Tx Mode e Rx Mode 2425MHz
            break;
        }
        case 5:
        {
            Temp = Mode==1?0x3C:0x3D; // Tx Mode e Rx Mode 2430MHz
            break;
        }
    }
}
```



```
        case 6:
        {
            Temp = Mode==1?0x46:0x47; // Tx Mode e Rx Mode 2435MHz
            break;
        }
        case 7:
        {
            Temp = Mode==1?0x50:0x51; // Tx Mode e Rx Mode 2440MHz
            break;
        }
        case 8:
        {
            Temp = Mode==1?0x5A:0x5B; // Tx Mode e Rx Mode 2445MHz
            break;
        }
        case 9:
        {
            Temp = Mode==1?0x68:0x69; // Tx Mode e Rx Mode 2452MHz
            break;
        }
        case 10:
        {
            Temp = Mode==1?0x6E:0x6F; // Tx Mode e Rx Mode 2455MHz
            break;
        }
        case 11:
        {
            Temp = Mode==1?0x78:0x79; // Tx Mode e Rx Mode 2460MHz
            break;
        }
        case 12:
        {
            Temp = Mode==1?0x82:0x83; // Tx Mode e Rx Mode 2465MHz
            break;
        }
        case 13:
        {
            Temp = Mode==1?0x90:0x91; // Tx Mode e Rx Mode 2472MHz
            break;
        }
        case 14:
        {
            Temp = Mode==1?0x96:0x97; // Tx Mode e Rx Mode 2475MHz
            break;
        }
        case 15:
        {
            Temp = Mode==1?0xA0:0xA1; // Tx Mode e Rx Mode 2480MHz
            break;
        }
    }
    Write_24G(Temp);
    DelayUs(100);
    CS = 0;
}

// Rotina Envia os Dados para o Módulo TRF-24G no Endereço 1
// Enviando 8 Bit
void Send_24G(unsigned char Dado)
{
    DIR = 1;          // Habilita no circuito a entrada de dados no Módulo TRF-24G
    DelayUs(100);
    CS = 0;
    DelayUs(100);      // Td - Delay Between Edges
    CE = 1;
    DelayUs(100);      // Tcs2data - Delay from CS to Data
}
```

```
        Write_24G(0xAA);          // Address 5 Byte Ch 1
        Write_24G(0x55);
        Write_24G(0xAA);
        Write_24G(0x55);
        Write_24G(0xAA);
        Write_24G(Dado);          // Data 8 bit
        DelayUs(100);
        CE = 0;
        DelayUs(100);
        DelayUs(100);
        DIR = 0;                  // Habilita no circuito a saída de dados do Módulo TRF-24G
    }

// Rotina Envia os Dados para o Módulo TRF-24G no Endereço 2
// Enviando 8 Bit
void Send_24G_ADD2(unsigned char Dado)
{
    DIR = 1;                      // Habilita no circuito a entrada de dados no Módulo TRF-24G
    DelayUs(100);
    CS = 0;
    DelayUs(100);                 // Td - Delay Between Edges
    CE = 1;
    DelayUs(100);                 // Tcs2data - Delay from CS to Data
    Write_24G(0xA0);              // Address 5 Byte Ch 1
    Write_24G(0x55);
    Write_24G(0xAA);
    Write_24G(0x55);
    Write_24G(0xAA);
    Write_24G(Dado);              // Data 8 bit
    DelayUs(100);
    CE = 0;
    DelayUs(100);
    DelayUs(100);
    DIR = 0;                      // Habilita no circuito a saída de dados do Módulo TRF-24G
}

// Rotina Envia os Dados para o Módulo TRF-24G no Endereço 1
// Enviando 16 Bit
void Send_24G_16Bit(unsigned int Dado)
{
    DIR = 1;                      // Habilita no circuito a entrada de dados no Módulo TRF-24G
    DelayUs(100);
    CS = 0;
    DelayUs(100);                 // Td - Delay Between Edges
    CE = 1;
    DelayUs(100);                 // Tcs2data - Delay from CS to Data
    Write_24G(0xAA);              // Address 5 Byte Ch 1
    Write_24G(0x55);
    Write_24G(0xAA);
    Write_24G(0x55);
    Write_24G(0xAA);
    Write_24G_16Bit(Dado);        // Data 16 bit
    DelayUs(100);
    CE = 0;
    DelayUs(100);
    DelayUs(100);
    DIR = 0;                      // Habilita no circuito a saída de dados do Módulo TRF-24G
}

// Rotina Envia os Dados para o Módulo TRF-24G no Endereço 2
// Enviando 16 Bit
void Send_24G_16Bit_ADD2(unsigned int Dado)
{
    DIR = 1;                      // Habilita no circuito a entrada de dados no Módulo TRF-24G
    DelayUs(100);
    CS = 0;
```



```
    DelayUs(100);          // Td - Delay Between Edges
    CE = 1;
    DelayUs(100);          // Tcs2data - Delay from CS to Data
    Write_24G(0xA0);        // Address 5 Byte Ch 1
    Write_24G(0x55);
    Write_24G(0xAA);
    Write_24G(0x55);
    Write_24G(0xAA);
    Write_24G_16Bit(Dado);  // Data 16 bit
    DelayUs(100);
    CE = 0;
    DelayUs(100);
    DelayUs(100);
    DIR = 0;                // Habilita no circuito a saída de dados do Módulo TRF-24G
}

// Rotina que Lê os Dados enviados para o Módulo TRF-24G
// Dados Enviados - 8 Bit
unsigned char Read_24G(void)
{
    unsigned char i, Temp=0;
    DIR = 0;                // Habilita no circuito a saída de dados do Módulo TRF-24G
    DelayUs(100);
    CLK1 = 0;
    DelayUs(100);          // Td - Delay Between Edges
    CLK1 = 1;
    for(i=0; i<8; i++)
    {
        Temp = Temp <<= 1;
        CLK1 = 1;
        DelayUs(100);      // Thmin
        if (DATA == 1)
        {
            Temp = Temp + 0x01;
        }
        else
        {
            Temp = Temp + 0x00;
        }
        CLK1 = 0;
        DelayUs(100);
    }
    DIR = 1;                // Habilita no circuito a entrada de dados no Módulo TRF-24G
    return(Temp);           // Retorna o valor lido
}

// Rotina que Lê os Dados enviados para o Módulo TRF-24G
// Dados Enviados - 16 Bit
unsigned int Read_24G_16Bit(void)
{
    unsigned int i;
    unsigned int Temp=0;
    DIR = 0;                // Habilita no circuito a saída de dados do Módulo TRF-24G
    DelayUs(100);
    CLK1 = 0;
    DelayUs(100);          // Td - Delay Between Edges
    CLK1 = 1;
    for(i=0; i<16; i++)
    {
        Temp = Temp <<= 1;
        CLK1 = 1;
        DelayUs(100);      // Thmin
        if (DATA == 1)
        {
            Temp = Temp + 0x01;
        }
    }
}
```

```
        else
        {
            Temp = Temp + 0x00;
        }
        CLK1 = 0;
        DelayUs(100);
    }
    DIR = 1;          // Habilita no circuito a entrada de dados no Módulo TRF-24G
    return(Temp);      // Retorna o valor lido
}
```

Rotina Serial.c

```
#include <pic.h>
#include <conio.h>

// Prints some text via serial
//
// Copyright (C)2005 HI-TECH Software.
// Freely distributable.

// Comms setup
#ifndef BAUD
#define BAUD 9600
#endif

#define FOSC 16000000L
#define NINE 0 // Use 9bit communication? FALSE=8bit

#define DIVIDER ((int)(FOSC/(16UL * BAUD) - 1))
#define HIGH_SPEED 1

#if NINE == 1
#define NINE_BITS 0x40
#else
#define NINE_BITS 0
#endif

#if HIGH_SPEED == 1
#define SPEED 0x4
#else
#define SPEED 0
#endif

#define RX_PIN TRISC7
#define TX_PIN TRISC6

// Serial initialization
void INIT_COMMS(void)
{
    RX_PIN = 1;
    TX_PIN = 1;
    SPBRG = DIVIDER;
    RCSTA = (NINE_BITS | 0x90);
    TXSTA = (SPEED | NINE_BITS | 0x20);
}

// PUTC() - outputs a byte to the serial port
void putch(unsigned char byte)
{
    while(!TXIF); // set when register is empty
    TXREG = byte; // output one byte
}

// Output a string via the serial port
void puts(const char *s)
```



```
{
    while(s && *s)
        putchar(*s++);
}

unsigned char getch(void)
{
    while(!RCIF);
    return RCREG;
}

unsigned int getint(void)
{
    unsigned char c;
    unsigned int i, tmp=0;

    for(i=1; i<=10000; i*=10)
    {
        c = getch()-'0';
        tmp += c*i;
    }
    return tmp;
}
```

Rotina ADC.c

```
#include <stdio.h>
#include <pic1687x.h>
#include "delay.h"

void Set_ADC(void)
{
    ADCON0 = 0x41;
    ADCON1 = 0xC9;
}

unsigned int Le_ADC(unsigned char canal)
{
    unsigned int tmp;

    canal&=0x07;
    ADCON0&=0xC5;
    ADCON0|=(canal<<3);
    DelayMs(5);
    ADGO = 1;
    while(ADGO) continue;
    tmp = (ADRESH<<8)+ADRESL;
    return tmp;
}
```

Rotina Delay.c

```
/*
 * Delay functions
 * See delay.h for details
 *
 * Make sure this code is compiled with full optimization!!!
 */

#include "delay.h"

void DelayMs(unsigned char cnt)
{
    unsigned char i;
    do
```

```
    {
        i = 10;
        do
        {
            DelayUs(100);
        }
        while(--i);
    }
    while(--cnt);
}
```

Rotina TRF-2.4G.h

```
extern void Init_24G(void);
extern void SetMode_24G(unsigned char, unsigned char);
extern void SetMode_24G_ADD2(unsigned char, unsigned char);
extern void SetMode_24G_16Bit(unsigned char, unsigned char);
extern void SetMode_24G_16Bit_ADD2(unsigned char, unsigned char);
extern void Write_24G(unsigned char);
extern void Write_24G_16Bit(unsigned int);
extern void Send_24G(unsigned char);
extern void Send_24G_ADD2(unsigned char);
extern void Send_24G_16Bit(unsigned int);
extern void Send_24G_16Bit_ADD2(unsigned int);
extern unsigned char Read_24G(void);
extern unsigned int Read_24G_16Bit(void);
```

Rotina Serial.h

```
extern void putch(unsigned char);
extern void puts(const char *);
extern void INIT_COMMS(void);
extern unsigned int getint(void);
```

Rotina ADC.h

```
extern void Set_ADC(void);
extern unsigned int Le_ADC(unsigned char canal);
```

Rotina Delay.h

```
//
// Delay functions for HI-TECH C on the PIC
//
// Functions available:
//      DelayUs(x)      Delay specified number of microseconds
//      DelayMs(x)      Delay specified number of milliseconds
//
// Note that there are range limits: x must not exceed 255 - for xtal
// frequencies > 12MHz the range for DelayUs is even smaller.
// To use DelayUs it is only necessary to include this file; to use
// DelayMs you must include delay.c in your project.
//
//
#define      DelayUs(x)      {unsigned char _dcnt;\
                             _dcnt = (unsigned char)(1.333*x);\
                             while(--_dcnt != 0) continue;\
                             }

extern void DelayMs(unsigned char);
```



APÊNDICE B – ROTINAS DO SOFTWARE BUILDER

Rotina Principal

```
//-----  
#include <vcl.h>  
#pragma hdrstop  
USERES("LeituraRemota.res");  
USEFORM("Leitura.cpp", Form1);  
USEUNIT("SerialCom.cpp");  
//-----  
WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)  
{  
    try  
    {  
        Application->Initialize();  
        Application->CreateForm(__classid(TForm1), &Form1);  
        Application->Run();  
    }  
    catch (Exception &exception)  
    {  
        Application->ShowException(&exception);  
    }  
    return 0;  
}  
//-----
```

Rotina Leitura

```
//-----  
#include <vcl.h>  
#pragma hdrstop  
  
#include "Leitura.h"  
#include "SerialCom.h"  
//-----  
#pragma package(smart_init)  
#pragma resource "*.dfm"  
DRVPAR Driver;  
int CommPort;  
  
int SampleCounter;  
  
TForm1 *Form1;  
//-----  
__fastcall TForm1::TForm1(TComponent* Owner)  
: TForm(Owner)  
{  
    Driver.cPort = (char)1; //porta COM1  
    Driver.wBaudRate = 9600;  
    if(OpenPort(&Driver)) {  
        CommPort = 0;  
    }  
}
```



```
        ShowMessage("Open COMM Port Error!\nReconfigure the Port.");
    }
    else{
    }
}
//-----
void __fastcall TForm1::FormDestroy(TObject *Sender)
{
    ClosePort();
}
//-----
void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
    unsigned int ValorLido=0;
    unsigned int *ptr;
    char Buffer[10];
    float ValoremTensao;

    // Leitura do módulo 1
    if(CheckBox2->Checked == true)
    {
        if(CheckBox4->Checked == true)
        {
            // Leitura do Módulo 1 Canal 0
            SendData("10");
            Sleep(StrToInt(Edit4->Text));
            Buffer[0] = ReadData();
            Buffer[1] = ReadData();
            Buffer[2] = ReadData();
            Buffer[3] = ReadData();
            ValorLido = Buffer[1]<<8;
            ValorLido = ValorLido | (0x00FF & (unsigned int)Buffer[2]);
            ValoremTensao = (float)ValorLido/1023*5;
            Edit2->Text = IntToStr(ValoremTensao);
            Series1->AddXY(SampleCounter, ValoremTensao,IntToStr(SampleCounter),clBlue);
        }
        if(CheckBox5->Checked == true)
        {
            // Leitura do Módulo 1 Canal 1
            SendData("11");
            Sleep(StrToInt(Edit4->Text));
            Buffer[0] = ReadData();
            Buffer[1] = ReadData();
            Buffer[2] = ReadData();
            Buffer[3] = ReadData();
            ValorLido = Buffer[1]<<8;
            ValorLido = ValorLido | (0x00FF & (unsigned int)Buffer[2]);
            ValoremTensao = (float)ValorLido/1023*5;
            Edit5->Text = IntToStr(ValoremTensao);
            Series2->AddXY(SampleCounter, ValoremTensao,IntToStr(SampleCounter),clRed);
        }
    }

    // Leitura do módulo 2
    if(CheckBox6->Checked == true)
    {
        if(CheckBox7->Checked == true)
        {
            // Leitura do Módulo 1 Canal 2
            SendData("20");
            Sleep(StrToInt(Edit4->Text));
            Buffer[0] = ReadData();
            Buffer[1] = ReadData();
            Buffer[2] = ReadData();
            Buffer[3] = ReadData();
            ValorLido = Buffer[1]<<8;
```



```
        ValorLido = ValorLido | (0x00FF & (unsigned int)Buffer[2]);
        ValoremTensao = (float)ValorLido/1023*5;
        Edit6->Text = IntToStr(ValorLido);
        Series3->AddXY(SampleCounter, ValoremTensao,IntToStr(SampleCounter),clRed);
    }
    if(CheckBox3->Checked == true)
    {
        // Leitura do Módulo 1 Canal 3
        SendData("21");
        Sleep(StrToInt(Edit4->Text));
        Buffer[0] = ReadData();
        Buffer[1] = ReadData();
        Buffer[2] = ReadData();
        Buffer[3] = ReadData();
        ValorLido = Buffer[1]<<8;
        ValorLido = ValorLido | (0x00FF & (unsigned int)Buffer[2]);
        ValoremTensao = (float)ValorLido/1023*5;
        Edit7->Text = IntToStr(ValorLido);
        Series4->AddXY(SampleCounter, ValoremTensao,IntToStr(SampleCounter),clRed);
    }
}
SampleCounter++;
if(SampleCounter > StrToInt(Edit1->Text))
CheckBox1->Checked = false;
}
//-----
void __fastcall TForm1::FormCreate(TObject *Sender)
{
    Chart1->BottomAxis->Automatic = true;
    Chart1->LeftAxis->Automatic = true;

}
//-----
void __fastcall TForm1::CheckBox1Click(TObject *Sender)
{
    Timer1->Interval = StrToInt(Edit3->Text);
    Timer1->Enabled = CheckBox1->Checked;
}
//-----
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    SampleCounter = 0;
    Series1->Clear();
    Series2->Clear();
    Series3->Clear();
    Series4->Clear();
}
//-----
```

Rotina Leitura

```
//-----
#include <vcl.h>
#pragma hdrstop

#include "SerialCom.h"

//Variáveis

Byte  BuffTx[25];           //Output Buffer
Byte  BuffRx[300];          //Input Buffer
Word  wRegCRC;              //CRC register accumulator for read/write
long  lNumCh;               //Number of bytes received on Input Buffer
char  cCodError;            //Error code
```



```
TDCB Dcb; // Estrutura DCB com os parâmetros de comunicação serial.
THandle ComFile; //Communication Device Handle

//-----
// Rotina: OpenPort
// Função: Conecta a porta serial
// Retorno: true->erro; false->OK
//-----

bool OpenPort(DRVPAR *Driver){
    bool bError;
    char szBuf[20]; // String para formatação de configuração de porta
    COMMTIMEOUTS CommTimeouts; //Timeout struct

    AnsiString DeviceName = "COM1"; // "LPT1";
    ComFile = (THandle)CreateFile("COM1" /*DeviceName.c_str()*/,
                                GENERIC_READ | GENERIC_WRITE,
                                0,
                                NULL,
                                OPEN_EXISTING,
                                FILE_ATTRIBUTE_NORMAL,
                                0);

    if ((HANDLE)ComFile == INVALID_HANDLE_VALUE){
        ShowMessage("CreateFile Error");
    }

    unsigned int RxBufferSize = 256;
    unsigned int TxBufferSize = 256;

    if (SetupComm((HANDLE)ComFile, RxBufferSize, TxBufferSize)==false){
        ShowMessage("SetupComm Error");
    }

    if (GetCommState((HANDLE)ComFile, &Dcb)== false){
        ShowMessage("GetCommState Error");
    }

    AnsiString ASConfig = "baud=9600 parity=n data=8 stop=1";

    if (BuildCommDCB(ASConfig.c_str(), &Dcb)==false){
        ShowMessage("BuildCommDCB Error");
    }

    if (SetCommState((HANDLE)ComFile, &Dcb)==false){
        ShowMessage("SetCommState Error");
    }

    CommTimeouts.ReadIntervalTimeout = 0;
    CommTimeouts.ReadTotalTimeoutMultiplier = 0;
    CommTimeouts.ReadTotalTimeoutConstant = 1000;
    CommTimeouts.WriteTotalTimeoutMultiplier = 0;
    CommTimeouts.WriteTotalTimeoutConstant = 1000;

    if (SetCommTimeouts((HANDLE)ComFile, &CommTimeouts)==false){
        ShowMessage("SetCommTimeouts Error");
    }
}
return (false);
}

//-----
// Rotina: ClosePort
// Função: Desconecta a porta serial
// Retorno: nenhum
//-----
```



```
void ClosePort(void){
    // Desliga o transmissor
    // DisableTX();

    // Zera todos os eventos
    // SetCommMask((HANDLE)ComFile, 0);

    // purge any outstanding reads/writes and close device handle
    // PurgeComm((HANDLE)ComFile, PURGE_TXABORT | PURGE_RXABORT |
PURGE_TXCLEAR | PURGE_RXCLEAR);
    // Encerra o dispositivo de comunicacao
    CloseHandle((HANDLE)ComFile);
}

//-----
void SendData(AnsiString ASData){

    unsigned long int BytesWritten;
    unsigned int result;
    unsigned char *cdata;
    cdata = ASData.c_str();
    unsigned int Length;
    Length = ASData.Length();
    result = WriteFile ((HANDLE)ComFile, cdata, Length,&BytesWritten, NULL);
    if (result == false){
        ShowMessage("Erro de Transmissão");
    }
}

//-----
char ReadData(void){

    char cData;
    unsigned long int BytesRead;

    if (ReadFile ((HANDLE)ComFile, &cData, sizeof(cData), &BytesRead, NULL)==false){
        ShowMessage("Erro de Recepção");
    }
    return 0;
}
if(BytesRead == 0) return 0;
return (cData);
}

//-----
// Rotina: GetIntervalTimeOut
// Função: Calcula o tempo de recepção entre 2 caracteres para dar timeout
// Retorno: Time-out, em mseg
//-----
short GetIntervalTimeOut(int iBaud){
    short sTout;

    switch(iBaud){
        case 1200:
            sTout = 640; //160*4
            break;
        case 2400:
            sTout = 320; //80*4
            break;
        case 4800:
            sTout = 160; //40*4
            break;
        case 9600:
            sTout = 80; //20*4 - PERFEITO!!!!
            break;
        case 19200:
            sTout = 40; //10*4
    }
}
```



```
        break;
    }

    return(sTout);
}

//-----
// Rotina: GetMultiplierTimeOut
// Função: Calcula o tempo de recepção de 1 byte, dependente do baudrate
// Retorno: Tempo de 1 byte, em mseg
//-----
short GetMultiplierTimeOut(int iBaud){
    short  sTout;

    switch(iBaud){
        case 1200:
            sTout = 32;
            break;
        case 2400:
            sTout = 16;
            break;
        case 4800:
            sTout = 8;
            break;
        case 9600:
            sTout = 4;
            break;
        case 19200:
            sTout = 4;
            break;
    }

    return(sTout);
}

//-----
// Rotina: GetConstantTimeOut
// Função: Calcula o tempo de recepção do cabeçalho fixo e mais o tempo de
//         resposta do aparelho
// Retorno: Tempo de recepcao do cabecalho e resposta, em mseg
//-----
short GetConstantTimeOut(int iBaud)
{
    short sTout;

    switch(iBaud)
    {
        case 1200:
            sTout = 48;
            break;
        case 2400:
            sTout = 24;
            break;
        case 4800:
            sTout = 12;
            break;
        case 9600:
            sTout = 6;
            break;
        case 19200:
            sTout = 3;
            break;
    }
    sTout += short(2000);
    return(sTout);
}
```



```
/*
    // Monta a estrutura de Dcb
    lstrcat(szBuf, ":9600,n,8,1");
    bError = BuildCommDCB(szBuf, &Dcb);
    if(bError==FALSE)    return true;

    // Seta os parametros gerais
    Dcb.BaudRate = (Driver->wBaudRate==1200) ? CBR_1200 :
                  (Driver->wBaudRate==2400) ? CBR_2400 :
                  (Driver->wBaudRate==4800) ? CBR_4800 :
                  (Driver->wBaudRate==9600) ? CBR_9600 : CBR_19200;

    Dcb.ByteSize = 8;
    Dcb.Parity = NOPARITY;
    Dcb.StopBits = ONESTOPBIT;

    // Parametros gerais
    Dcb.DCBlength = sizeof(Dcb);
    Dcb.fBinary = 1;
    Dcb.fParity = 0;
    Dcb.fOutxDsrFlow = 0;
    Dcb.fOutxCtsFlow = 0;
    Dcb.fDtrControl = DTR_CONTROL_ENABLE;
    Dcb.fRtsControl = RTS_CONTROL_TOGGLE;
    Dcb.fOutX = 0;
    Dcb.fInX = 0;
    Dcb.fErrorChar = 0;
    Dcb.fNull = 0;
    Dcb.EvtChar = 0;
    Dcb.ErrorChar = 0;

    // Seta os parametros de Dcb no hardware
    bError = SetCommState(hComm, &Dcb);
    if(bError==FALSE){
        unsigned int unError = GetLastError();
        return true;
    }

    // Seta para gerar o evento EV_RXCHAR (chegou byte na queue de recepção)
    // e o evento EV_TXEMPTY (queue de transmissao vazia => já enviou tudo)
    SetCommMask(hComm, EV_RXCHAR | EV_TXEMPTY);

    // Configura os buffer das filas de transmissão e recepção
    SetupComm(hComm, 300, 128);

    DisableTX();    //RTS=0

    // Inicializa o dispositivo (limpa buffers)
    PurgeComm(hComm, PURGE_TXABORT | PURGE_RXABORT | PURGE_TXCLEAR |
PURGE_RXCLEAR);

    CommTimeOuts.ReadIntervalTimeout=GetIntervalTimeOut(Dcb.BaudRate);
    CommTimeOuts.ReadTotalTimeoutMultiplier=GetMultiplierTimeOut(Dcb.BaudRate);
    CommTimeOuts.ReadTotalTimeoutConstant=GetConstantTimeOut(Dcb.BaudRate);
    CommTimeOuts.WriteTotalTimeoutMultiplier=0;
    CommTimeOuts.WriteTotalTimeoutConstant=0;
    SetCommTimeouts(hComm, &CommTimeOuts);
    return false;
*/
```



ANEXO A – ESQUEMA ELÉTRICO



ANEXO B – MANUAL DA LAIPAC - MÓDULO TRF-2.4G



TRF-2.4G Transceiver Data Sheet

**High frequency 2.4G Wireless Transceiver
Antenna, Codec and CRC built in
Data Rate up to 1Mbps**

High frequency TRF-2.4G Transceiver module

Specification

- Frequency Range: 2.4~2.524 GHz ISM band
- Modulate Mode: GFSK
- Data Rate: 1Mbps; 250Kbps
- Multi channel operation: 125 channels, Channel switching time<200uS, Support frequency hopping
- Emulated full duplex RF link due to the 1Mbps/s on the air data rate
- Simultaneous dual receiver
- Data slicer / clock recovery of data
- Including decoder, encoder and data buffer and CRC computation
- ShockBurst mode for ultra-low power operation and relaxed MCU performance
- Sensitivity: -90dBm
- Built in antenna
- Power supply range: 1.9 to 3.6 V
- Low supply current (TX), typical 10.5mA peak@ -5dBm output power
- Low supply current (RX), typical 18mA peak in receive mode
- Supply current in Power Down Mode: 1 uA
- Operating Temperature: -40~+85 Centigrade
- Size: 20.5*36.5*2.4mm
- 100% RF tested
- Competitive price

Applications

- Wireless mouse, keyboard, joystick
- Wireless data communication
- Alarm and security systems
- Home automation
- Wireless Earphone
- Telemetry
- Surveillance
- Automotive

GENERAL DESCRIPTION

Laipac TRF-2.4G Module is an easy to use radio transceiver for the world wide 2.4 - 2.5 GHz ISM band. The transceiver consists of an antenna, a fully integrated frequency synthesizer, a power amplifier, a crystal oscillator and a modulator. Output power and frequency channels are easily programmable by use of the 3-wire serial interface. Current consumption is very low, only 10.5mA at an output power of -5dBm and 18mA in receive mode. Built-in Power Down modes makes power saving easily realizable.

ELECTRICAL SPECIFICATIONS

Conditions: VCC = +3V, VSS = 0V, TA = - 40°C to + 85°C

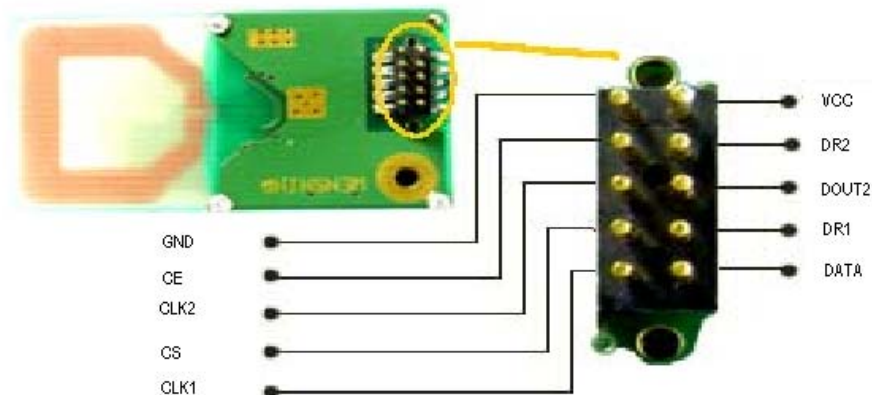
Symbol	Parameter (condition)	Notes	Min.	Ttp.	Max.	Units
Operating conditions						
VCC	Supply voltage		1.9	3.0	3.6	V
TEMP	Operating Temperature		-40	+27	+85	°C
Digital input pin						
VIH	HIGH level input voltage		VCC-0.3		VCC	V
VIL	LOW level input voltage		Vss		0.3	V
Digital output pin						
VOH	HIGH level output voltage (IOH=-0.5mA)		VCC-0.3		VCC	V
VOL	LOW level output voltage (IOL=0.5mA)		Vss		0.3	V
General RF conditions						
fOP	Operating frequency	1)	2400		2524	MHz
Δf	Frequency deviation			±156		kHz
RGFSK	Data rate ShockBurst		>0		1000	kbps
RGFSK	Data rate Direct Mode	2)	250		1000	kbps
FCHANNEL	Channel spacing			1		MHz
Transmitter operation						
PRF	Maximum Output Power	3)		0	+4	dBm
PRFC	RF Power Control Range		16	20		dB
PRFCR	RF Power Control Range Resolution				±3	dB
PBW	20dB Bandwidth for Modulated Carrier				1000	kHz
PRF2	2nd Adjacent Channel Transmit Power 2MHz				-20	dBm
PRF3	3rd Adjacent Channel Transmit Power 3MHz				-40	dBm
IvCC	Supply current @ 0dBm output power	4)		13		mA
IvCC	Supply current @ -20dBm output power	4)		8.8		mA
IvCC	Average Supply current @ -5dBm output power, ShockBurst	5)		0.8		mA
IvCC	Average Supply current in stand-by mode	6)		12		μA
IvCC	Average Supply current in power down			1		μA
Receiver operation						
IvCC	Supply current one channel 250kbps			18		mA
IvCC	Supply current one channel 1000kbps			19		mA
IvCC	Supply current two channels 250kbps			23		mA
IvCC	Supply current two channels 1000kbps			25		mA
RXSENS	Sensitivity at 0.1%BER (@250kbps)			-90		dBm
RXSENS	Sensitivity at 0.1%BER (@1000kbps)			-80		dBm

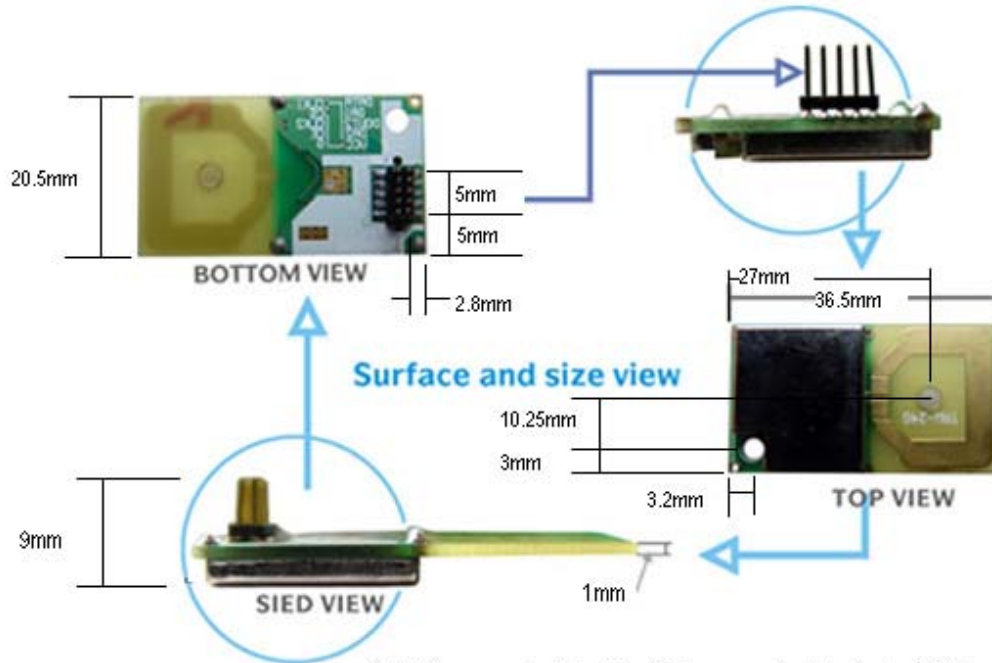
C/ICO	C/I Co-channel			6		dB
C/I1ST	1st Adjacent Channel Selectivity C/I 1MHz			-1		dB
C/I2ND	2nd Adjacent Channel Selectivity C/I 2MHz			-16		dB
C/I3RD	3rd Adjacent Channel Selectivity C/I 3MHz			-26		dB
RXB	Blocking Data Channel 2			-41		dB

- 1) Usable band is determined by local regulations
- 2) Data rate must be either 250kbps or 1000kbps.
- 3) De-embedded Antenna load impedance = 400
- 4) De-embedded Antenna load impedance = 400 . Effective data rate 250kbps or 1Mbps.
- 5) De-embedded Antenna load impedance = 400 . Effective data rate 10kbps.
- 6) Current if 4 MHz crystal is used.

Table 1 TRF-2.4G RF specifications

PIN ASSIGNMENT





Note: The connector pitch size is 1.25mm, mounting hole diameter is 2.8mm

PIN FUNCTIONS

Pin	Name	Pin funtion	Description
1	GND	Power	Gound (0V)
2	CE	Input	Chip Enable activates RX or TX mode
3	CLK2	I/O	Clock output/input for RX data channel 2
4	CS	Input	Chip Select activates Configuration mode
5	CLK1	I/O	Clock Input(TX)&I/O(RX) for data channel 1 3-wire interface
6	DATA	I/O	RX data channel 1/TX data input /3-wire interface
7	DR1	Output	RX data ready at data channel 1 (ShockBurst only)
8	DOUT2	Output	RX data channel 2
9	DR2	Output	RX data ready at data channel 2 (ShockBurst only)
10	VCC	Power	Power Supply (+3V DC)

Table 2 TRF-2.4G pin function

MODE OF OPERATION

TRF-2.4G can be set in the following main mode:

Mode	CE	CS
Active (RX /TX)	1	0
Configuration	0	1
Stand by	0	0

Table 3 TRF-2.4G main modes

TRF-2.4G has two active (RX /TX) modes:

- ShockBurst
- Direct Mode

The device functionality in these modes is decided by the content of a configuration word. This configuration word is presented in configuration section.

Absolute Maximum Ratings

Supply voltages

VCC.....- 0.3V to + 3.6V

VSS0V

Input/Output voltages

V_I.....- 0.3V to VCC + 0.3V

V_O.....- 0.3V to VCC + 0.3V

Total Power Dissipation

P_D (T_A=85°C).....90mW

Temperatures

Operating Temperature.... - 40°C to + 85°C

Storage Temperature..... - 40°C to + 125°C

ShockBurst Mode

The ShockBurst technology uses on-chip FIFO to clock in data at a low data rate and transmit at a very high rate thus enabling extremely power reduction.

When operating the TRF-2.4G in ShockBurst, you gain access to the high data rates (1 Mbps) offered by the 2.4 GHz band without the need of a costly, high-speed micro

controller (MCU) for data processing.

By putting all high speed signal processing related to RF protocol on-chip, the TRF-2.4G offers the following benefits:

- Highly reduced current consumption
- Lower system cost (facilitates use of less expensive micro controller)
- Greatly reduced risk of 'on-air' collisions due to short transmission time

The TRF-2.4G can be programmed using a simple 3-wire interface where the data rate is decided by the speed of the micro controller.

By allowing the digital part of the application to run at low speed while maximizing the data rate on the RF link, the nRF ShockBurst mode reduces the average current consumption in applications considerably.

ShockBurst principle

When the TRF-2.4G is configured in ShockBurst, TX or RX operation is conducted in the following way (10 kbps for the example only).

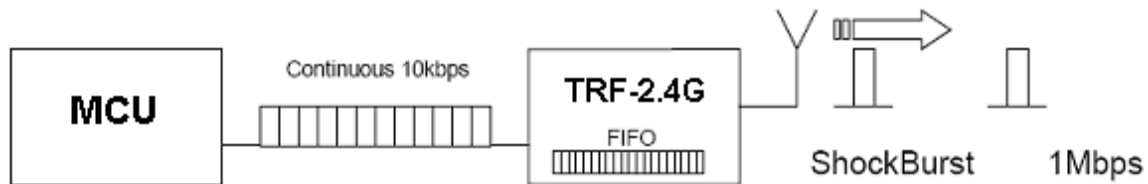


Figure 0 Clocking in data with MCU and sending with ShockBurst technology

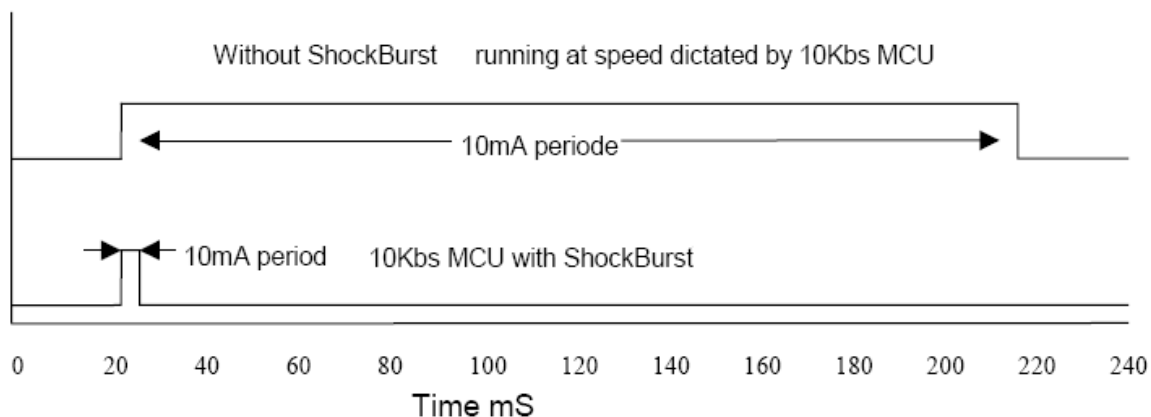
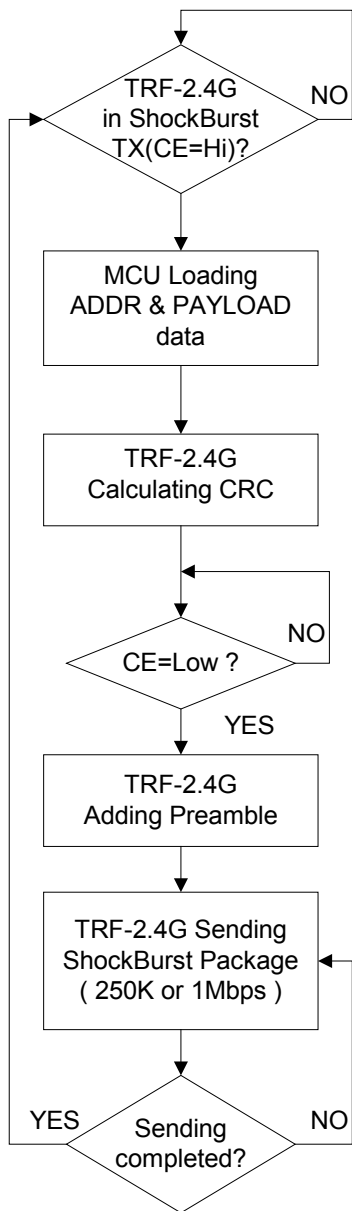


Figure 1 Current consumption with & without ShockBurst technology

TRF-2.4G ShockBurst Transmit:

MCU interface pins: CE, CLK1, DATA

1. When the application MCU has data to send, set CE high. This activates TRF-2.4G on-board data processing.
2. The address of the receiving node (RX address) and payload data is clocked into the TRF-2.4G. The application protocol or MCU sets the speed <1Mbps (ex: 10kbps).
3. MCU sets CE low, this activates a TRF-2.4G ShockBurst transmission.
4. TRF-2.4G ShockBurst:
 - RF front end is powered up
 - RF package is completed (preamble added, CRC calculated)
 - Data is transmitted at high speed (250 kbps or 1 Mbps configured by user).
 - TRF-2.4G return to stand-by when finished



Data content of registers:

ADDR	PAYLOAD
------	---------

ADDR	PAYLOAD	CRC
------	---------	-----

-----Maximum 256 bits-----

Pre- amble	ADDR	PAYLOAD	CRC
---------------	------	---------	-----

Input FIFO not Empty

Figure 2 Flow Chart ShockBurst Transmit of TRF-2.4G

TRF-2.4G ShockBurst Receive:

MCU interface pins: CE, DR1, CLK1 and DATA (one RX channel receive)

1. Correct address and size of payload of incoming RF packages are set when TRF-2.4G is configured to ShockBurst RX.
2. To activate RX, set CE high.
3. After 200 μ s settling, TRF-2.4G is monitoring the air for incoming communication.
4. When a valid package has been received (correct address and CRC found), TRF-2.4G removes the preamble, address and CRC bits.
5. TRF-2.4G then notifies (interrupts) the MCU by setting the DR1 pin high.
6. MCU may (or may not) set the CE low to disable the RF front end (low current mode).
7. The MCU will clock out just the payload data at a suitable rate (ex. 10 kbps).
8. When all payload data is retrieved TRF-2.4G sets DR1 low again, and is ready for new incoming data package if CE is kept high during data download. If the CE was set low, a new start up sequence can begin, see Figure 2.

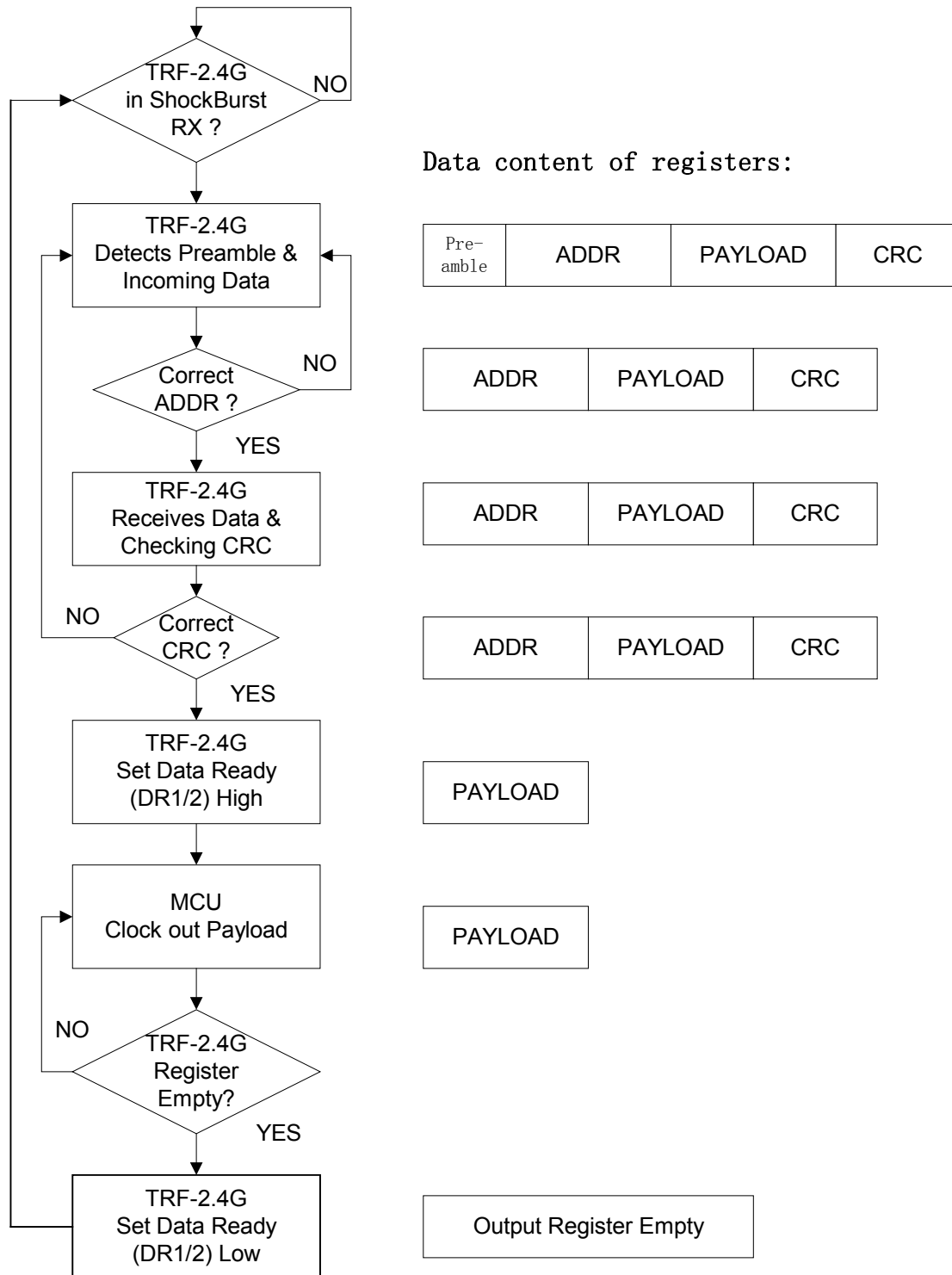


Figure 3 Flow Chart ShockBurst Receive of TRF-2.4G

TRF-2.4G Direct Mode:

In direct mode the TRF-2.4G works like a traditional RF device. Data must be at 1Mbps, or 250kbps at low data rate setting, for the receiver to detect the signals.

Direct Mode Transmit:

MCU interface pins: CE, DATA

1. When application MCU has data to send, set CE high
2. The TRF-2.4G RF front end is now immediately activated, and after 200 seconds settling time, data will modulate the carrier directly.
3. All RF protocol parts must hence be implemented in MCU firmware (preamble, address and CRC).

Direct Mode Receive:

MCU interface pins: CE, CLK1, and DATA

1. Once the TRF-2.4G is configured and powered up (CE high) in direct RX mode, DATA will start to toggle due to noise present on the air.
2. CLK1 will also start to toggle as TRF-2.4G is trying to lock on to the incoming data stream.
3. Once a valid preamble arrives, CLK1 and DATA will lock on to the incoming signal and the RF package will appear at the DATA pin with the same speed as it is transmitted.
4. To enable the demodulator to re-generate the clock, the preamble must be 8 bits toggling hi-low, starting with low if the first data bit low.
5. In this mode no data ready (DR) signals is available. Address and checksum verification must also be done in the receiving MC.

DuoCeiver Simultaneous Two Channel Receive Mode

In both ShockBurst & Direct modes the TRF-2.4G can facilitate simultaneous reception of two parallel independent frequency channels at the maximum data rate.

This means:

- TRF-2.4G can receive data from two 1 Mbps transmitters, 8 MHz (8 frequency channels) apart through one antenna interface.
- The output from the two data channels is fed to two separate MCU interfaces.
 - Data channel 1: CLK1, DATA, and DR1
 - Data channel 2: CLK2, DOUT2, and DR2
 - DR1 and DR2 are available only in ShockBurst.

The DuoCeiver technology provides 2 separate dedicated data channels for RX and replaces the need for two, stand alone receiver systems.

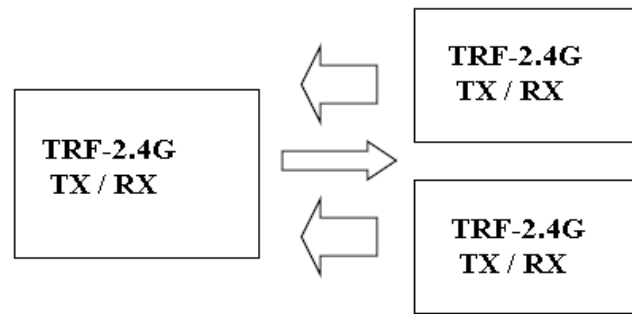
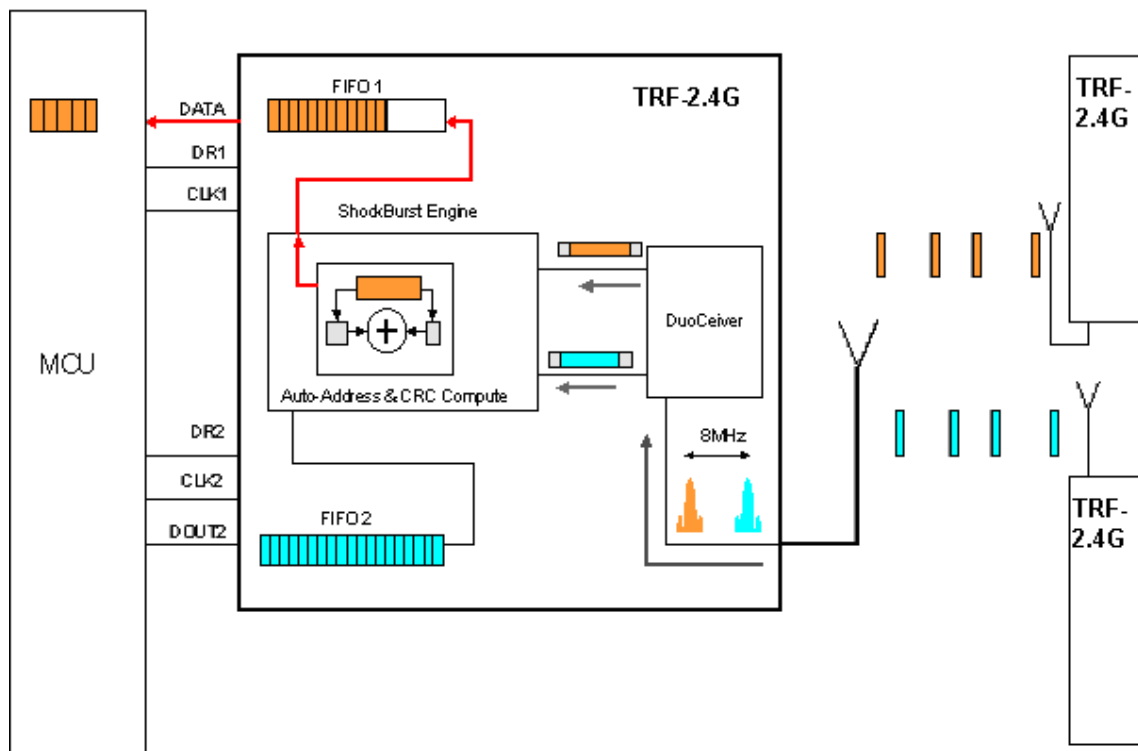


Figure 4 Simultaneous 2 channel receive on TRF-2.4G

There is one absolute requirement for using the second data channel. For the TRF-2.4G to be able to receive at the second data channel the frequency channel must be 8MHz higher than the frequency of data channel 1. The TRF-2.4G must be programmed to receive at the frequency of data channel 1. No time multiplexing is used in TRF-2.4G to fulfil this function. In direct mode the MCU must be able to handle two simultaneously incoming data packets if it is not multiplexing between the two data channels. In ShockBurst it is possible for the MCU to clock out one data channel at a time while data on the other data channel waits for MCU availability, without any lost data packets, and by doing so reduce the needed performance of the MCU.



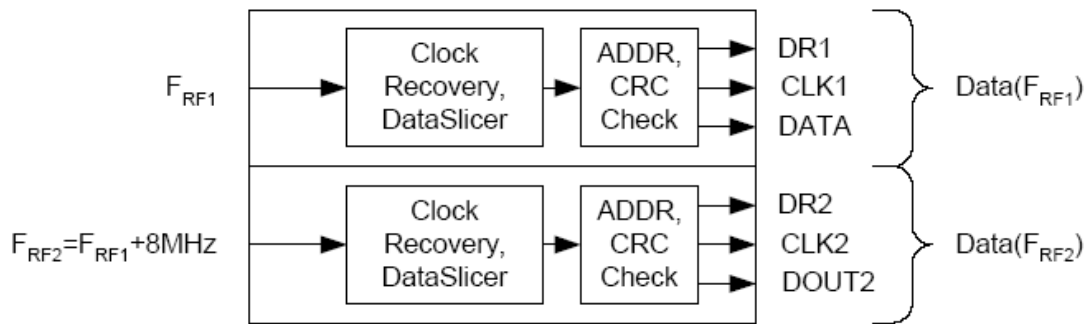


Figure 5 DuoCeiver with two simultaneously independent receive channels.

Configuration Mode

In configuration mode a configuration word of up to 15 bytes is downloaded to TRF-2.4G. This is done through a simple 3-wire interface (CS, CLK1 and DATA). For more information on configuration please refer to the TRF-2.4G Device configuration chapter on next 2nd page.

Stand-By Mode

Stand by mode is used to minimize average current consumption while maintaining short start up times. In this mode, part of the crystal oscillator is active. Current consumption is dependent on crystal frequency (Ex: 12uA @ 4 MHz, 32uA@ 16MHz). The configuration word content is maintained during stand by.

Power Down Mode

In power down the TRF-2.4G is disabled with minimal current consumption, typically less than 1 A. Entering this mode when the device is not active minimizes average current consumption, maximizing battery lifetime. The configuration word content is maintained during power down.

DEVICE CONFIGURATION

All configuration of the TRF-2.4G is done via a 3-wire interface to a single configuration register. The configuration word can be up to 15 bytes long for ShockBurst use and up to 2 bytes long for direct mode.

Configuration for ShockBurst operation

The configuration word in ShockBurst enables the TRF-2.4G to handle the RF protocol. Once the protocol is completed and loaded into TRF-2.4G only one byte, bit[7:0], needs to be updated during actual operation.

The configuration blocks dedicated to ShockBurst is as follows:

- Payload section width: Specifies the number of payload bits in a RF package. This enables the TRF-2.4G to distinguish between payload data and the CRC bytes in a received package.
- Address width: Sets the number of bits used for address in the RF package. This enables the TRF-2.4G to distinguish between address and payload data.
- Address (RX Channel 1 and 2): Destination address for received data.
- CRC: Enables TRF-2.4G on-chip CRC generation and de-coding.

NOTE:

These configuration blocks, with the exception of the CRC, are dedicated for the packages that a TRF-2.4G is to receive.

In TX mode, the MCU must generate an address and a payload section that fits the configuration of the TRF-2.4G that is to receive the data.

When using the TRF-2.4G on-chip CRC feature ensure that CRC is enabled and uses the same length for both the TX and RX devices.

PRE-AMBLE	ADDRESS	PAYLOAD	CRC
-----------	---------	---------	-----

Figure 6 Data packet set-up

Configuration for Direct Mode operation

For direct mode operation only the two first bytes (bit[15:0]) of the configuring word are relevant.

Configuration Word overview

	Bit position	Number of bits	Name	Function
ShockBurst configuration	143:120	24	TEST	Reserved for testing
	119:112	8	DATA2_W	Length of data payload section RX channel 1
	111:104	8	DATA1_W	Length of data payload section RX channel 1
	103:64	40	ADDR2	Up to 5 bytes address for channel 2
	63:24	40	ADDR1	Up to 5 bytes address for channel 1
	23:18	6	ADDR_W	Number of address bits(both RX channels)
	17	1	CRC_L	8 or 16 bits CRC
	16	1	CRC_EN	Enable on-chip CRC generation/checking
General device configuration	15	1	RX2_EN	Enable two channel receive mode
	14	1	CM	Communication mode (Direct or ShockBurst)
	13	1	RFDR_SB	RF data rate (1Mbps requires 16MHz crystal)
	12:10	3	XO_F	Crystal frequency (Factory default 16MHz crystal mounted)
	9:8	2	RF_PWR	RF output power
	7:1	7	RF_CH#	Frequency channel
	0	1	RXEN	RX or TX operation

Table 4 Table of configuration words.

The configuration word is shifted in MSB first on positive CLK1 edges. New configuration is enabled on the falling edge of CS.

NOTE.

On the falling edge of CS, the TRF-2.4G updates the number of bits actually shifted in during the last configuration.

Ex:

If the TRF-2.4G is to be configured for 2 channel RX in ShockBurst, a total of 120 bits must be shifted in during the first configuration after VCC is applied.

Once the wanted protocol, modus and RF channel are set, only one bit (RXEN) is shifted in to switch between RX and TX.

Configuration Word Detailed Description

The following describes the function of the 144 bits (bit 143 = MSB) that is used to configure the TRF-2.4G.

General Device Configuration: bit[15:0]

ShockBurst Configuration: bit[119:0]

Test Configuration: bit[143:120]

MSB	TEST																
D143	D142	D141	D140	D139	D138	D137	D136										
Reserved for testing																	
1	0	0	0	1	1	1	0	Default									

MSB	TEST																	
D135	D134	D133	D132	D131	D130	D129	D128	D127	D126	D125	D124	D123	D122	D121	D120			
Reserved for testing															Close PLL in TX			
0	0	0	0	1	0	0	0	0	0	0	1	1	1	0	0	Default		

DATA2 W															
D119	D118	D117	D116	D115	D114	D113	D112								
Data width channel#2 in # of bits excluding addr/crc															
0	0	1	0	0	0	0	0	Default							

DATA1 W															
D111	D110	D109	D108	D107	D106	D105	D104								
Data width channel#1 in # of bits excluding addr/crc															
0	0	1	0	0	0	0	0	Default							

ADDR2															
D103	D102	D101	D71	D70	D69	D68	D67	D66	D65	D64				
Channel#2 Address RX (up to 40bit)															
0	0	0	...	1	1	1	0	0	1	1	1	Default			

ADDR1															
D63	D62	D61	D31	D30	D29	D28	D27	D26	D25	D24				
Channel#1 Address RX (up to 40bit)															
0	0	0	...	1	1	1	0	0	1	1	1	Default			

ADDR_W															
D23	D22	D21	D20	D19	D18										
Address width in # of bits (both channels)															
0	0	1	0	0	0	Default									

CRC																	
D17								D16									
CRC Mode 1 = 16bit, 0 = 8bit								CRC 1 = enable; 0 = disable									
0								1								Default	

RF-Programming															LSB	
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	
Two Ch		BUF	OD	XO Frequency			RF Power		Channel selection						RXEN	
0	0	0	0	1	1	1	1	0	0	0	0	0	1	0	0	Default

Table 5 Configuration data word

The MSB bit should be loaded first into the configuration register.

ShochBurst configuration:

The section B[119:16] contains the segments of the configuration register dedicated to ShockBurst operational protocol. After VCC is turned on ShockBurst configuration is done once and remains set whilst VCC is present. During

operation only the first byte for frequency channel and RX/TX switching need to be changed.

DATAx_W

DATA2_W							
119	118	117	116	115	114	113	112

DATA1_W							
111	110	109	108	107	106	105	104

Table 6 Number of bits in payload.

Bit 119 – 112:

DATA2_W: Length of RF package payload section for receive-channel 2.

Bit 111 – 104:

DATA1_W: Length of RF package payload section for receive-channel 1.

NOTE:

The total number of bits in a ShockBurst RF package may not exceed 256!

Maximum length of payload section is hence given by:

$$DATAx_W(bits) = 256 - ADDR_W - CRC$$

Where:

ADDR_W: length of RX address set in configuration word B[23:18]

CRC: check sum, 8 or 16 bits set in configuration word B[17]

PRE: preamble, 4 or 8 bits are automatically included

Shorter address and CRC leaves more room for payload data in each package.

ADDRx

ADDR2											
103	102	101	71	70	69	68	67	66	65	64

ADDR1											
63	62	61	31	30	29	28	27	26	25	24

Table 7 Address of receiver #2 and receiver #1.

Bit 103 – 64:

ADDR2: Receiver address channel 2, up to 40 bit.

Bit 63 – 24: ADDR1

ADDR1: Receiver address channel 1, up to 40 bit.

NOTE!

Bits in ADDR_x exceeding the address width set in ADDR_W are redundant and can be set to logic 0.

ADDR_W& CRC

ADDR_W						CRC_L	CRC_EN
23	22	21	20	19	18	17	16

Table 8 Number of bits reserved for RX address + CRC setting.

Bit 23 – 18:

ADDR_W: Number of bits reserved for RX address in ShockBurst packages.

NOTE:

Maximum number of address bits is 40 (5 bytes). Values over 40 in ADDR_W are not valid.

Bit 17:

CRC_L: CRC length to be calculated by TRF-2.4G in ShockBurst.

Logic 0: 8 bit CRC

Logic 1: 16 bit CRC

Bit: 16:

CRC_EN: Enables on-chip CRC generation (TX) and verification (RX).

Logic 0: On-chip CRC generation/checking disabled

Logic 1: On-chip CRC generation/checking enabled

NOTE:

An 8 bit CRC will increase the number of payload bits possible in each ShockBurst data packet, but will also reduce the system integrity.

General device configuration:

This section of the configuration word handles RF and device related parameters.

Modes:

RX2_EN	CM	RFDR_SB	XO_F			RF_PWR	
15	14	13	12	11	10	9	8

Table 9 RF operational settings.

Bit 15:

RX2_EN:

Logic 0: One channel receive

Logic 1: Two channels receive

NOTE:

In two channels receive, the TRF-2.4G receives on two, separate frequency channels simultaneously. The frequency of receive channel 1 is set in the configuration word B[7-1], receive channel 2 is always 8 channels (8 MHz) above receive channel 1.

Bit 14:

Communication Mode:

Logic 0: TRF-2.4G operates in direct mode.

Logic 1: TRF-2.4G operates in ShockBurst mode

Bit 13:

RF Data Rate:

Logic 0: 250 kbps

Logic 1: 1 Mbps

NOTE:

Utilizing 250 kbps instead of 1Mbps will improve the receiver sensitivity by 10 dB. 1Mbps requires 16MHz crystal.

Bit 12-10:

XO_F: Selects the TRF-2.4G crystal frequency to be used:

XO FREQUENCY SELECTION			
D12	D11	D10	Crystal Frequency (MHz)
0	1	1	16
Factory default: 16MHz Crystal is used inside module			

Table 10 Crystal frequency setting.

Bit 9-8:

RF_PWR: Sets TRF-2.4G RF output power in transmit mode:

RF OUTPUT POWER		
D9	D8	P (dBm)
0	0	-20
0	1	-10
1	0	-5
1	1	0

Table 11 RF output power setting.

RF channel & direction

RF_CH#							RXEN
7	6	5	4	3	2	1	0

Table 12 Frequency channel + RX / TX setting.

Bit 7 – 1:

RF_CH#: Sets the frequency channel the TRF-2.4G operates on.

The channel frequency in **transmit** is given by:

$$Channel_{RF} = 2400MHz + RF_CH\# * 1.0MHz$$

RF_CH #: between 2400MHz and 2527MHz may be set.

The channel frequency in **data channel 1** is given by:

$$Channel_{RF} = 2400MHz + RF_CH\# * 1.0MHz \text{ (Reiceive at PIN\#8)}$$

RF_CH #: between 2400MHz and 2524MHz may be set.

NOTE:

The channels above 83 can only be utilized in certain territories (ex: Japan)

The channel frequency in **data channel 2** is given by:

$$Channel_{RF} = 2400MHz + RF_CH\# * 1.0MHz + 8MHz \text{ (Reiceive at PIN\#4) } ???$$

RF_CH #: between 2408MHz and 2524MHz may be set.

Bit 0:

Set active mode:

Logic 0: transmit mode

Logic 1: receive mode

DATA PACKAGE DESCRIPTION

PRE-AMBLE	ADDRESS	PAYLOAD	CRC
-----------	---------	---------	-----

Figure 7 Data Package Diagram

The data packet for both ShockBurst mode and direct mode communication is divided into 4 sections. These are:

1 PREAMBLE	<ul style="list-style-type: none"> • The preamble field is required in ShockBurst and Direct modes • Preamble is 8 (or 4) bits in length and is dependent of the first data bit in direct mode. <p>PREAMBLE 1st ADDR-BIT</p> <p>01010101 0</p> <p>10101010 1</p> <ul style="list-style-type: none"> • Preamble is automatically added to the data packet in ShockBurst and thereby gives extra space for payload. • In ShockBurst mode the preamble is stripped from the received output data, in direct mode the preamble is transparent to the output data.
2 ADDRESS	<ul style="list-style-type: none"> • The address field is required in ShockBurst mode. • 8 to 40 bits length. • Address automatically removed from received packet in ShockBurst mode. In Direct mode MCU must handle address.
3 PAYLOAD	<ul style="list-style-type: none"> • The data to be transmitted • In Shock-Burst mode payload size is 256 bits minus the following: (Address: 8 to 40 bits. + CRC 8 or 16 bits). • In Direct mode the payload size is defined by 1Mbps for 4ms: 4000 bits minus the following: (Preamble: 8 (or 4) bits. + Address: 8 to 40 bits. + CRC: 0, 8 or 16 bits).
4 CRC	<ul style="list-style-type: none"> • The CRC is optional in ShockBurst mode, and is not used in Direct mode. • 8 or 16 bits length • The CRC is stripped from the received output data.

Table 13 Data package description

IMPORTANT TIMING DATA

The following timing applies for operation of TRF-2.4G.

TRF-2.4G Timing Information

TRF-2.4G timing	Max.	Min.	Name
PWR_DWN => ST_BY mode	3ms		Tpd2sby
PWR_DWN =>Active mode (RX/TX)	3ms		Tpd2a
ST_BY => TX ShockBurst	195 μ s		Tsby2txSB
ST_BY => TX Direct Mode	202 μ s		Tsby2txDM
ST_BY => RX mode	202 μ s		Tsby2rx
Minimum delay from CS to data		5 μ s	Tcs2data
Minimum delay from CE to data		5 μ s	Tce2data
Minimum delay from DR1/2 to clk		50ns	Tdr2clk
Maximum delay from clk to data	50ns		Tclk2data
Delay between edges		50ns	Td
Setup time		500ns	Ts
Hold time		500ns	Th
Delay to finish internal GFSK data		1/data rate	Tfd
Minimum input clock high		500ns	Thmin
Set-up of data in Direct Mode	50ns		Tsdm
Minimum clock high in Direct Mode		300ns	Thdm
Minimum clock low in Direct Mode		230ns	Tldm

Table 14 Switching times for TRF-2.4G

When the TRF-2.4G is in power down it must always settle in stand-by (Tpd2sby) before it can enter configuration or one of the active modes.

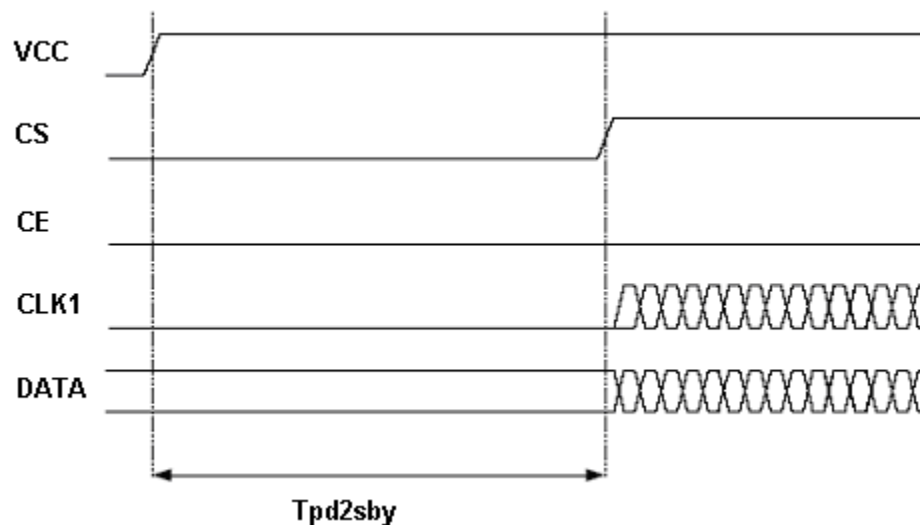


Figure 8 Timing diagram for power down (or VCC off) to stand by mode

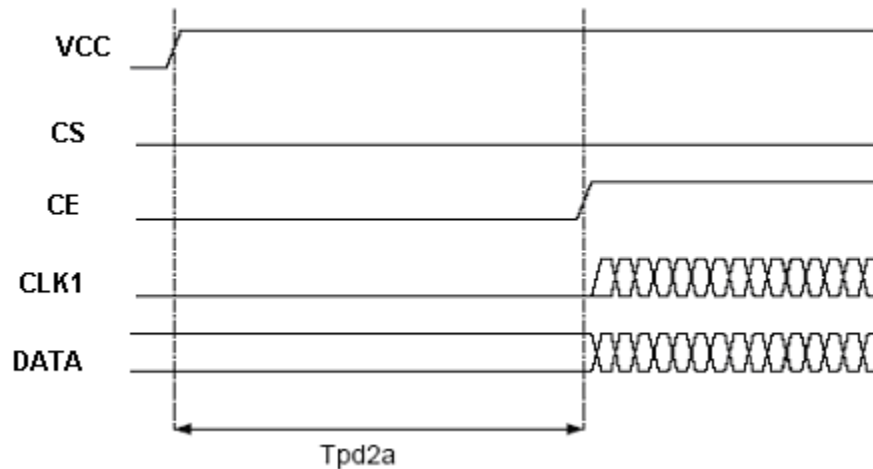


Figure 9 Power down (or VCC off) to active mode

Note that the configuration word will be lost when VCC is turned off and that the device then must be configured before going to one of the active modes. If the device is configured one can go directly from power down to the wanted active mode.

Note:

CE and CS may not be high at the same time. Setting one or the other decides whether configuration or active mode is entered.

Configuration mode timing

When one or more of the bits in the configuration word needs to be changed the following timing apply.

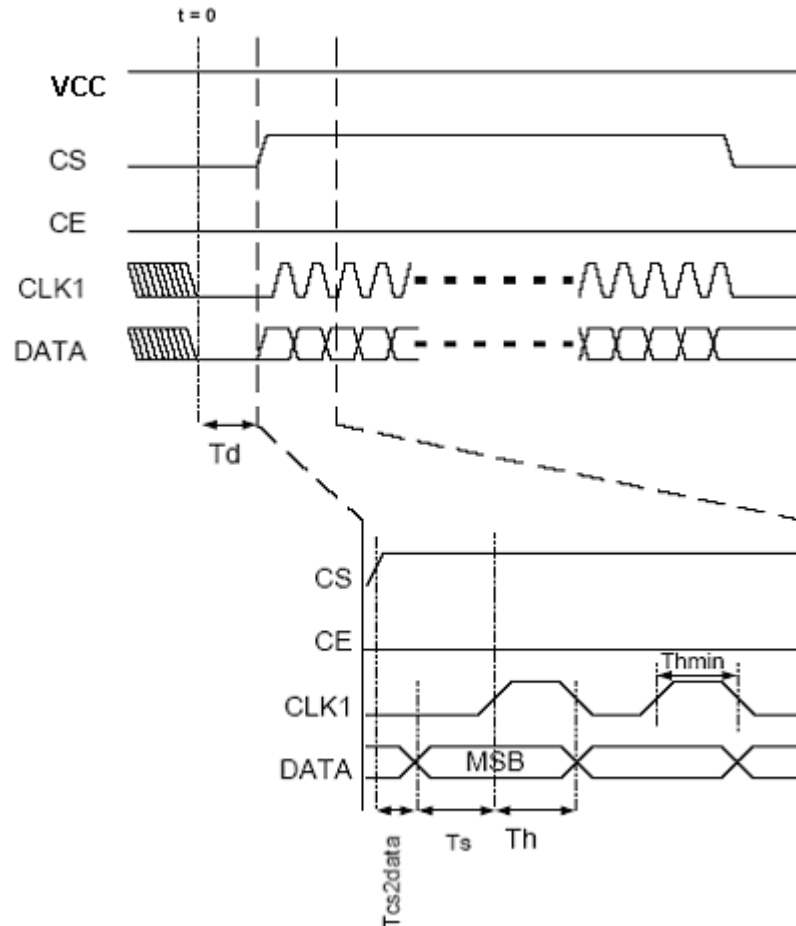


Figure 10 Timing diagram for configuration of TRF-2.4G
If configuration mode is entered from power down, CS can be set high after T_{pd2sby} as shown in Figure 10.

ShockBurst mode timing

ShockBurst TX:

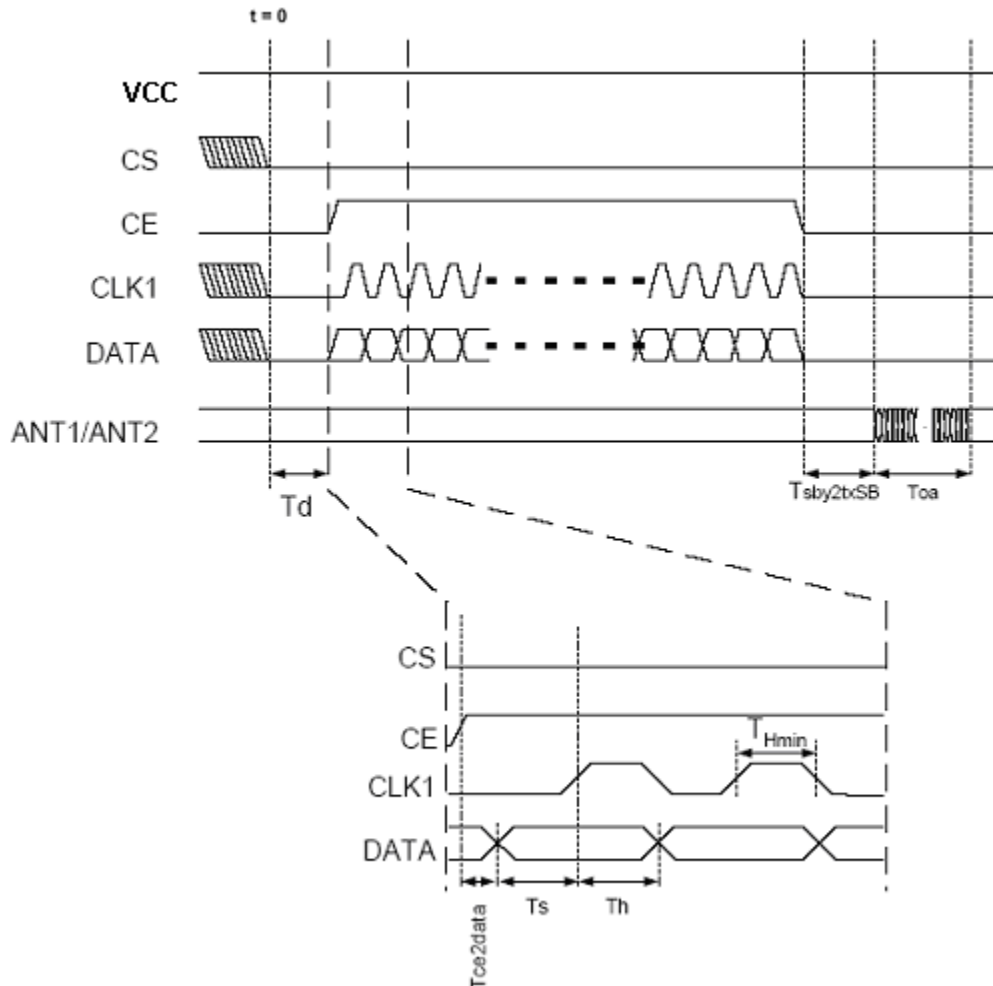


Figure 11 Timing of ShockBurst in TX

The package length and the data rate give the delay T_{oa} (time on air), as shown in the equation.

$$T_{OA} = 1/datarate * (\#databits + 1)$$

ShockBurst RX:

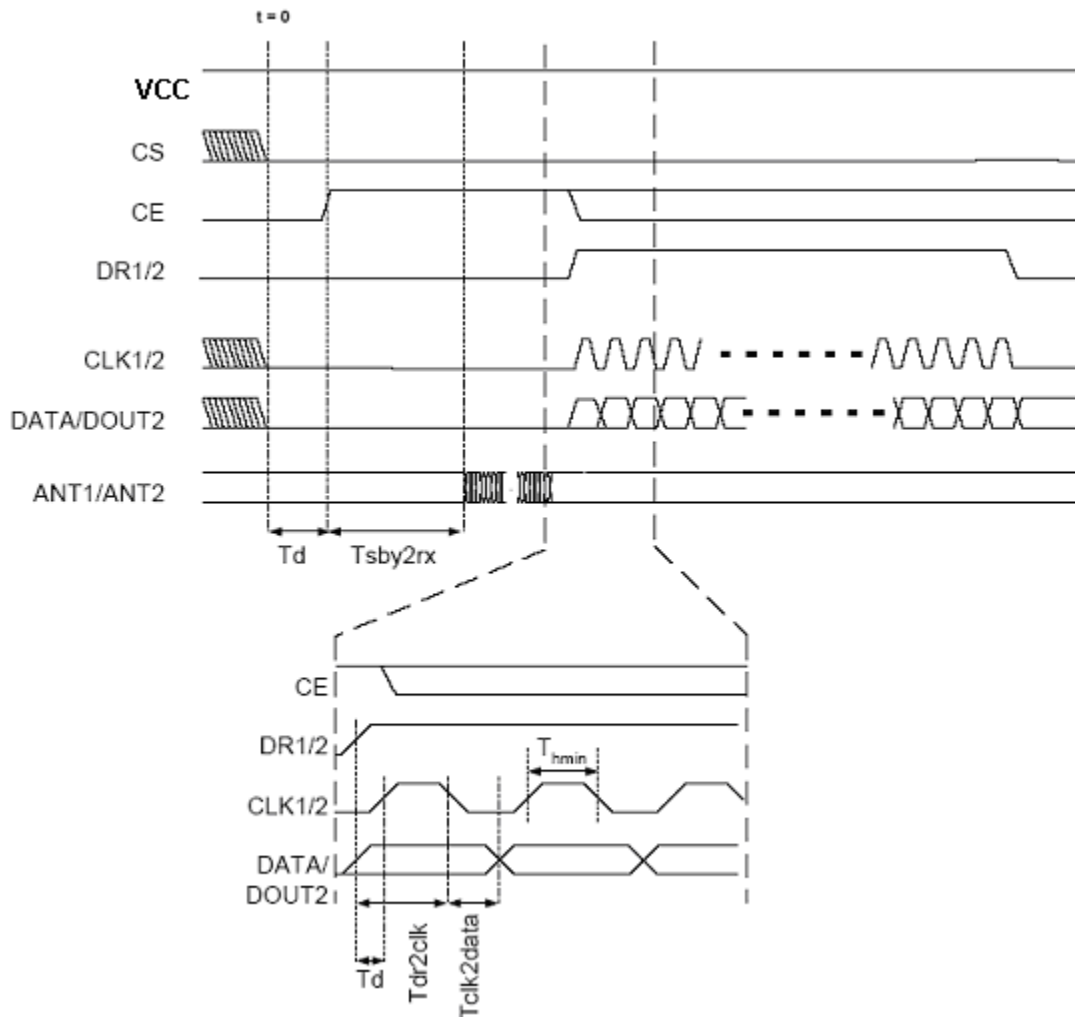


Figure 12 Timing of ShockBurst in RX

The CE may be kept high during downloading of data, but the cost is higher current consumption (18mA) and the benefit is no start-up time (200 μ s) after the DR1 goes low.

Direct Mode

Direct Mode TX

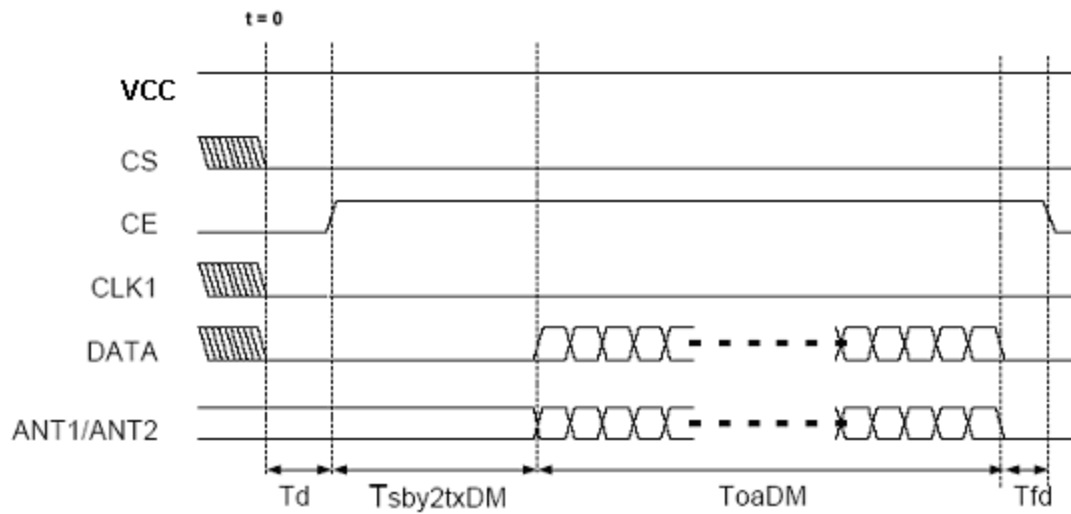


Figure 13 Timing of direct mode TX

In TX direct mode the input data will be sampled by TRF-2.4G and therefore no clock is needed. The clock must be stable at low level during transmission due to noise considerations. The exact delay $T_{sby2txDM}$ is given by the equation:

$$T_{sby2txDM} = 194\mu S + 1/F_{XO} * 14 + 2.25\mu S$$

Direct Mode RX

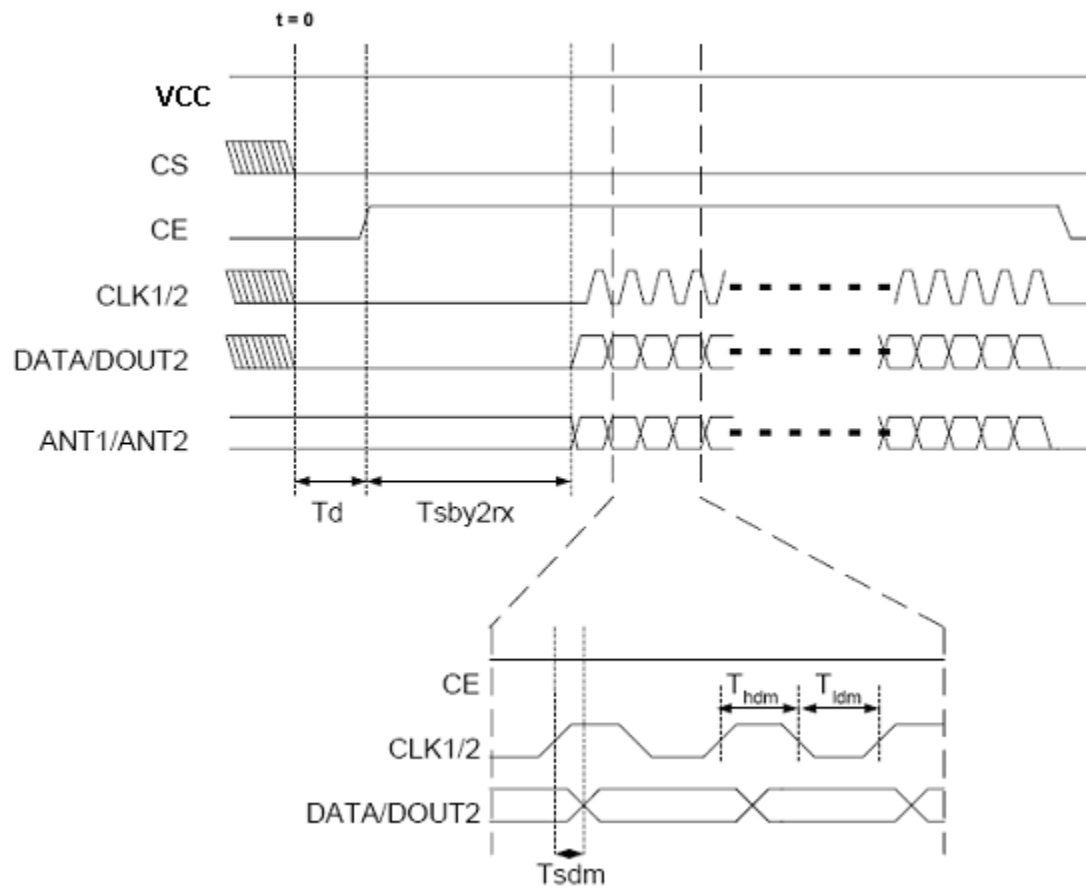


Figure 14 Timing of direct mode RX

T_{sby2rx} describes the delay from the positive edge of CE to the start detection of (demodulated) incoming data.

PERIPHERAL RFINFORMATION

Antenna output

The ANT1 & ANT2 output pins provide a balanced RF output to the antenna. The pins must have a DC path to VCC, either via a RF choke or via the center point in a dipole antenna. The load impedance seen between the ANT1/ANT2 outputs should be in the range 200-700 Ω . A de-embedded load impedance i.e. impedance seen at drain terminals of the output transistors of 400 Ω is recommended for maximum output power (0dBm). Lower load impedance (for instance 50 Ω) can be obtained by fitting a simple matching network.

Output Power adjustment

Power setting bits of Configuring word	RF output power	DC current consumption
11	0 dBm \pm 3dB	13.0 mA
10	-5 dBm \pm 3dB	10.5 mA
01	-10 dBm \pm 3dB	9.4 mA
00	-20 dBm \pm 3dB	8.8 mA

Conditions: VCC = 3.0V, VSS = 0V, TA = 27°C, Load impedance = 400 Ω .

Table 15 RF output power setting for the TRF-2.4G.

Configuration Word Example

1 Channel, Freq.: 2410MHz, 1Mbps and Transmit mode:

Bit143	Bit142	Bit141	Bit140	Bit139	Bit138	Bit137	Bit136
1	0	0	0	1	1	1	0
Bit135	Bit134	Bit133	Bit132	Bit131	Bit130	Bit129	Bit128
0	0	0	0	1	0	0	0
Bit127	Bit126	Bit125	Bit124	Bit123	Bit122	Bit121	Bit120
0	0	0	1	1	1	0	0
Bit119	Bit118	Bit117	Bit116	Bit115	Bit114	Bit113	Bit112
1	1	0	0	1	0	0	0
Bit111	Bit110	Bit109	Bit108	Bit107	Bit106	Bit105	Bit104
1	1	0	0	1	0	0	0
Bit103	Bit102	Bit101	Bit100	Bit99	Bit98	Bit97	Bit96
1	1	0	0	0	0	0	0
Bit95	Bit94	Bit93	Bit92	Bit91	Bit90	Bit89	Bit88
1	0	1	0	1	0	1	0
Bit87	Bit86	Bit85	Bit84	Bit83	Bit82	Bit81	Bit80
0	1	0	1	0	1	0	1
Bit79	Bit78	Bit77	Bit76	Bit75	Bit74	Bit73	Bit72
1	0	1	0	1	0	1	0
Bit71	Bit70	Bit69	Bit68	Bit67	Bit66	Bit65	Bit64
0	1	0	1	0	1	0	1
Bit63	Bit62	Bit61	Bit60	Bit59	Bit58	Bit57	Bit56
1	0	1	0	1	0	1	0
Bit55	Bit54	Bit53	Bit52	Bit51	Bit50	Bit49	Bit48
0	1	0	1	0	1	0	1
Bit47	Bit46	Bit45	Bit44	Bit43	Bit42	Bit41	Bit40
1	0	1	0	1	0	1	0
Bit39	Bit38	Bit37	Bit36	Bit35	Bit34	Bit33	Bit32
0	1	0	1	0	1	0	1
Bit31	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24
1	0	1	0	1	0	1	0
Bit23	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16
1	0	1	0	0	0	1	1
Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
0	1	1	0	1	1	1	1
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0	0	0	1	0	1	0	0

Table 16 Configuration Example

ANEXO C – TEMPORIZAÇÃO - MÓDULO TRF-2.4G

Ao longo deste projeto são citadas as temporizações a serem seguidas e que são descritas abaixo: (Wenshing - TRW-24G High Frequency Transceiver Module (GFSK), 2004)

- Tpd2cfgm – Tempo entre ligar o Módulo TRF-2.4G e o início do modo de configuração. Este tempo também deve ser respeitado quando o módulo está no modo *standby* e a configuração será iniciada, conforme figura 5-1. Tempo máximo de 3ms.

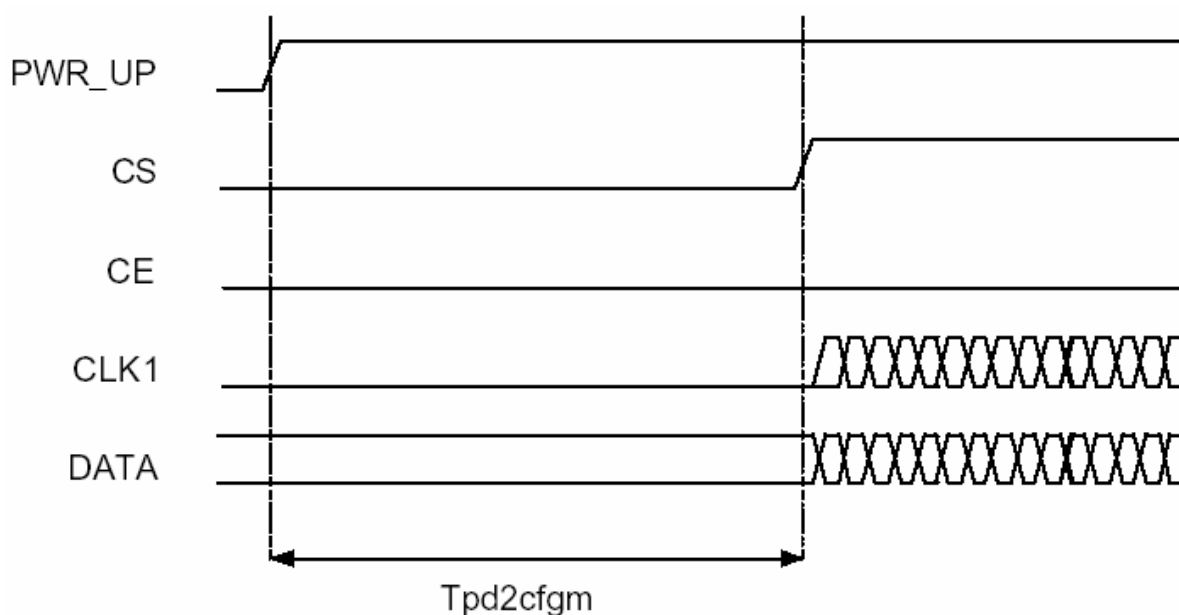


Ilustração 5-1 – Diagrama do Tpd2cfgm.

Fonte: Nordic Semiconductor ASA - Single Chip 2.4 GHz Transceiver nRF2401

- Tpd2a – Tempo entre ligar o Módulo TRF-2.4G e o início do modo ativo, a monitoração do ar, conforme figura 5-2. Tempo máximo 3ms.

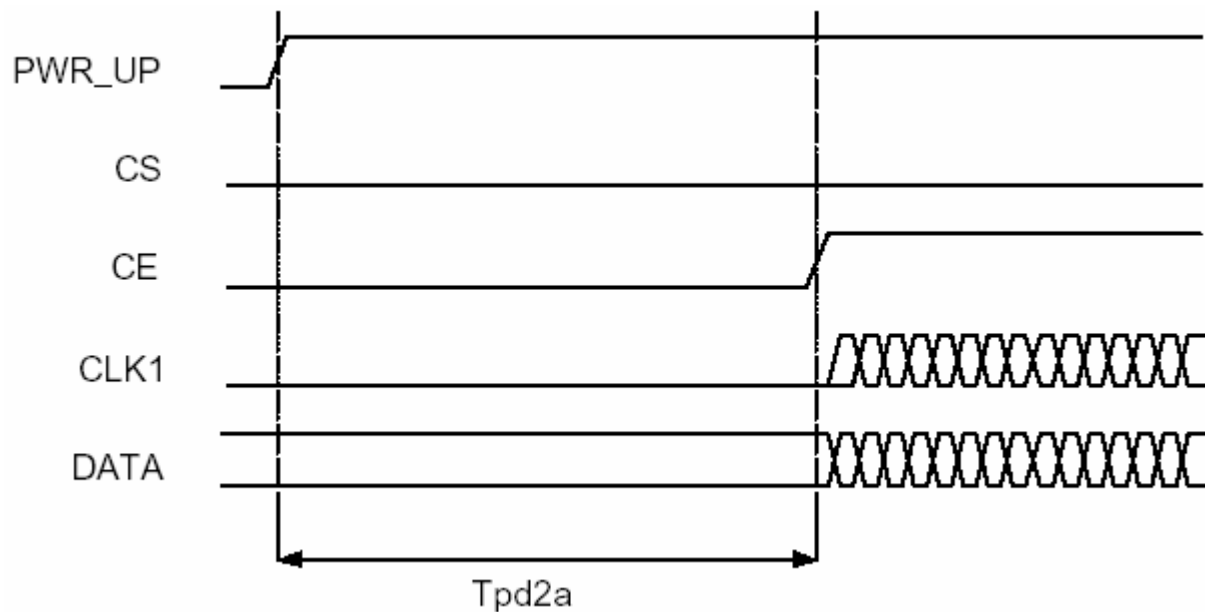


Ilustração 5-2 – Diagrama do Tpd2a.

Fonte: Nordic Semiconductor ASA - Single Chip 2.4 GHz Transceiver nRF2401

- Tsby2txSB – Tempo que o Módulo TRF-2.4G leva para transmitir os dados, no modo *ShockBurst*, após o pino CE ser levado de nível lógico 1 para nível lógico 0, conforme figura 5-5. Tempo mínimo 195 μ s.
- Tsby2rx – Tempo que o Módulo TRF-2.4G leva para começar a monitorar o ar, no modo *ShockBurst*, após o pino CE ser levado de nível lógico 0 para nível lógico 1, conforme figura 5-3. Tempo mínimo 202 μ s.

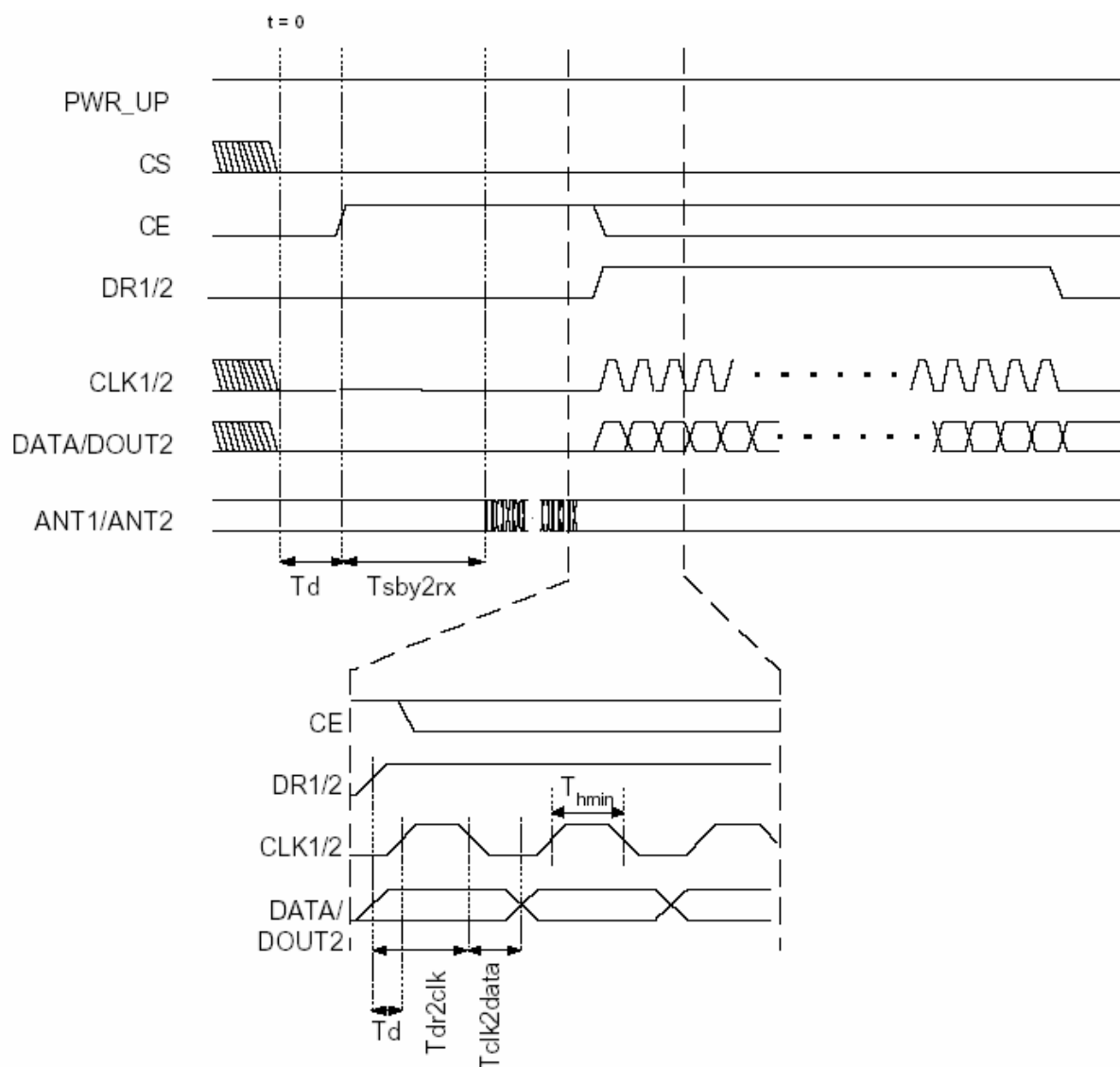


Ilustração 5-3 – Diagrama dos Tempos de Recepção no Modo *ShockBurst*.

Fonte: Nordic Semiconductor ASA - Single Chip 2.4 GHz Transceiver nRF2401

- $T_{cs2data}$ – Tempo de *delay* antes que a palavra de configuração seja enviada ao Módulo TRF-2.4G após o pino CS ser levado de nível lógico 0 para nível lógico 1, conforme figura 5-4. Tempo mínimo 5 μ s.

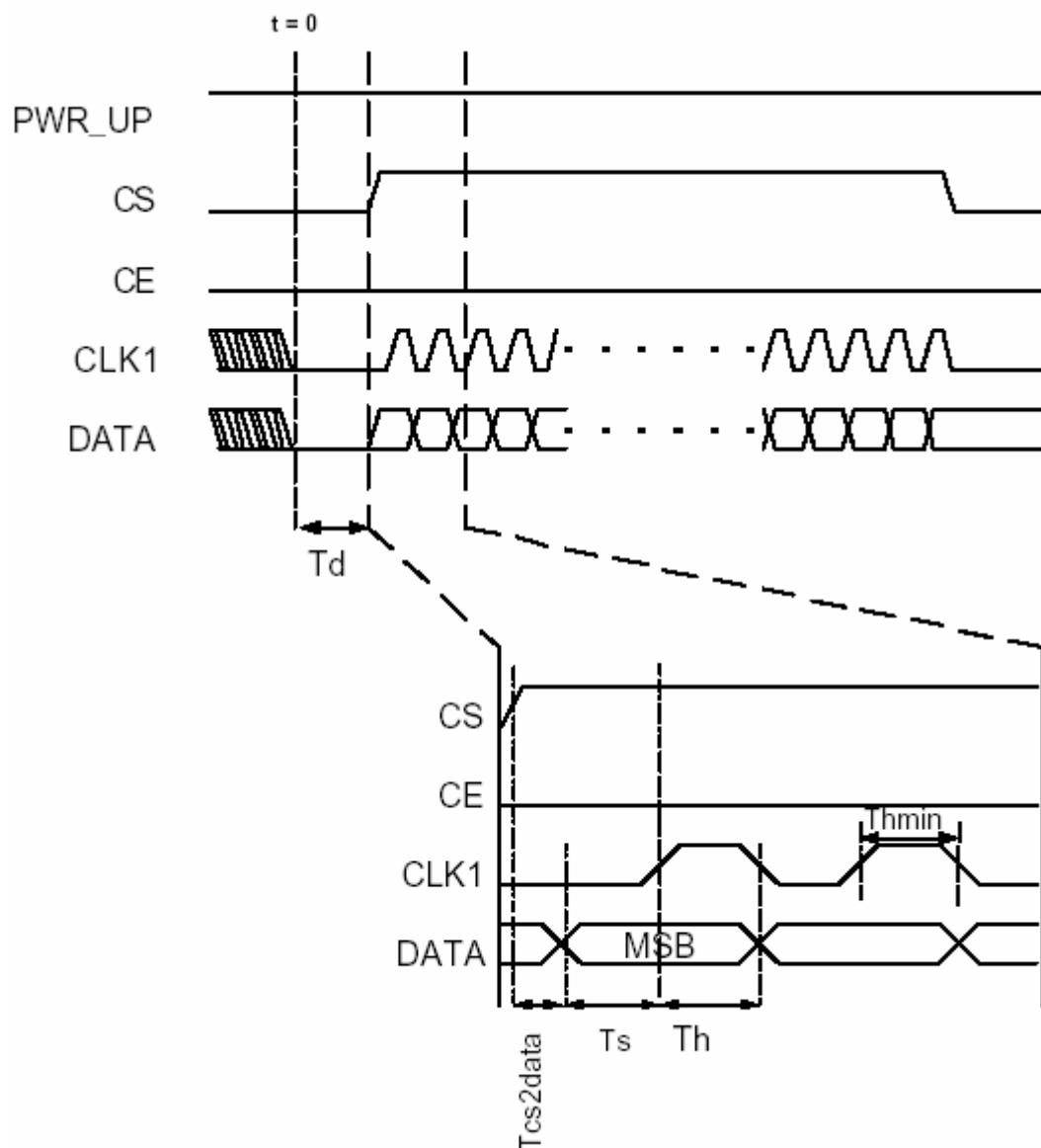


Ilustração 5-4 – Diagrama dos Tempos do Modo Configuração.

Fonte: Nordic Semiconductor ASA - Single Chip 2.4 GHz Transceiver nRF2401

- $T_{ce2data}$ – Tempo de *delay* antes que os dados sejam enviados ao Módulo TRF-2.4G após o pino CE ser levado de nível lógico 0 para nível lógico 1, conforme figura 5-5. Tempo mínimo 5 μ s.

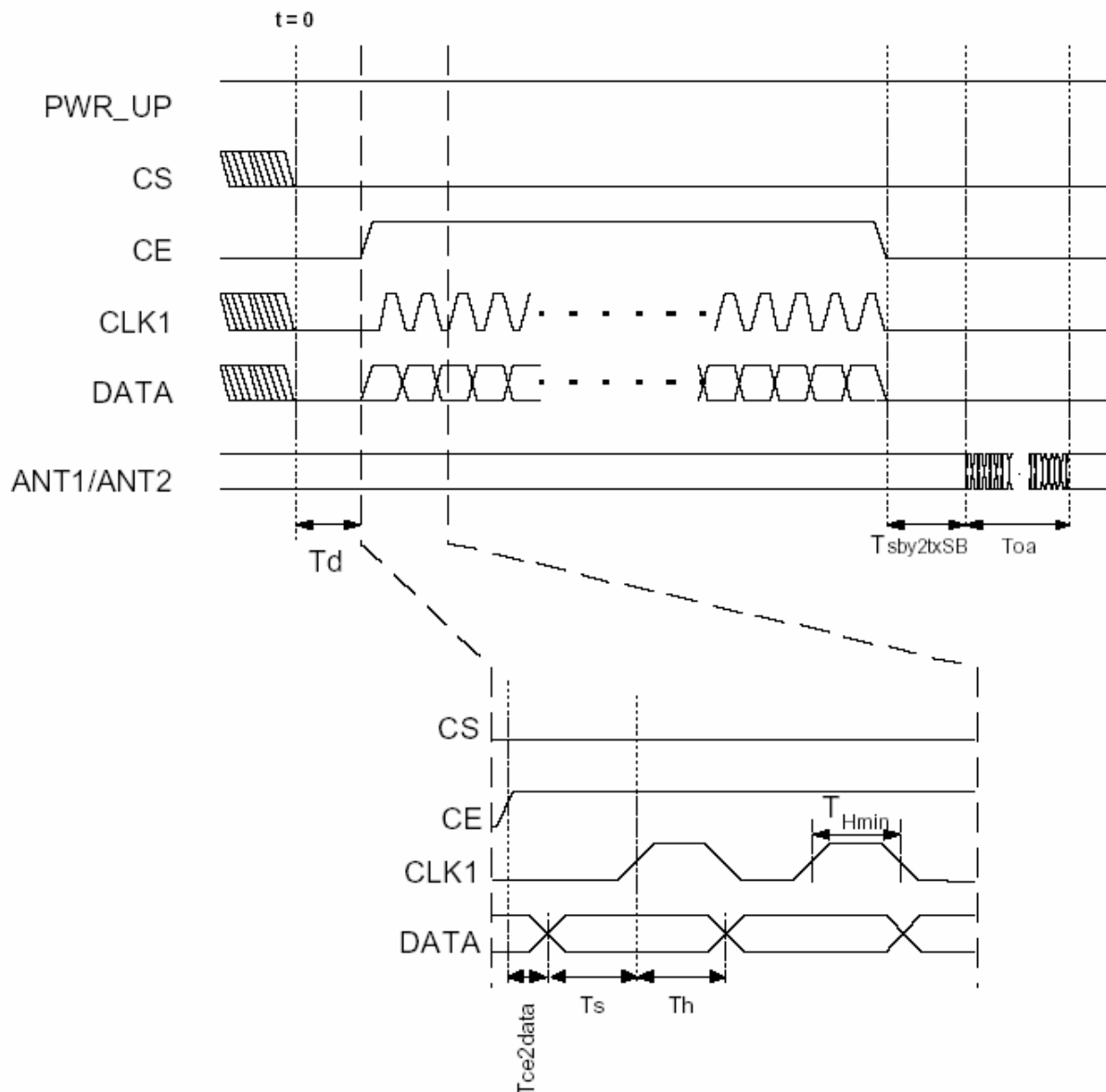


Ilustração 5-5 – Diagrama dos Tempos de Transmissão no Modo *ShockBurst*.

Fonte: Nordic Semiconductor ASA - Single Chip 2.4 GHz Transceiver nRF2401

- T_{dr2clk} – Tempo que o pino CLK, do Módulo TRF-2.4G, deve permanecer em nível lógico 1 durante a recepção de cada bit, conforme figura 5-3. Tempo mínimo 50ns.
- $T_{clk2data}$ – Tempo que o pino CLK, do Módulo TRF-2.4G, deve permanecer em nível lógico 0 durante a recepção de cada bit, conforme figura 5-3. Tempo mínimo 50ns.
- T_d – Tempo entre o pino CS ser levado de nível lógico 0 para nível lógico 1, do Módulo TRF-2.4G, com o pino CE estando em nível lógico 0, conforme figura 5-5. Os pinos CE e CS nunca podem ser levados para nível lógico 1 ao mesmo tempo. Tempo mínimo 50ns.



- T_s – Tempo que o pino CLK, do Módulo TRF-2.4G, deve permanecer em nível lógico 0 durante o envio do bit mais significativo ao módulo, conforme figura 5-5. Tempo mínimo 500ns.
- T_h – Tempo que o pino CLK, do Módulo TRF-2.4G, deve permanecer em nível lógico 1 durante o envio do bit mais significativo ao módulo, conforme figura 5-5. Tempo mínimo 500ns.
- T_{hmin} – Tempo que o pino CLK, do Módulo TRF-2.4G, deve permanecer em nível lógico 1 durante o envio de um bit ao módulo, conforme figura 5-5. Tempo mínimo 500ns.