XVIII Seminário Nacional de Distribuição de Energia Elétrica

SENDI 2008 - 06 a 10 de outubro

Olinda - Pernambuco - Brasil

Desenvolvimento De Unidade Terminal Remota De Baixo Custo Para Reset Remoto De Equipamentos De Comunicação

Marcio E. C. Brito		Elaine M. Silva
CELPE		CELPE
Autor 1 – marcioecb@celpe.com.br		Autor 2– Elaine.mendes@celpe.com.br

Palavras-chave

Atuação da manutenção
Baixo custo
Disponibilidade
Sistema de Automação
Unidade terminal remota

Resumo

Atualmente a CELPE possui um centro de operações integrado (COI), onde está concentrada a operação de 122 subestações. Estas subestações são operadas remotamente através do sistema de automação, o que possibilita redução de custos e melhor desempenho do sistema. Entretanto, é indispensável uma elevada disponibilidade do sistema de automação, principalmente dos equipamentos que concentram a comunicação de várias subestações. Estes equipamentos são conhecidos como processadores de comunicação (PCOM), que possuem 16 subestações associadas a cada um. Apesar da elevada disponibilidade apresentada por estes equipamentos, eventualmente ocorrem falhas, que em mais de 98% dos casos são sanadas através de um simples reset. Entretanto estes PCOM's encontram-se em locais onde não há a presença de pessoal técnico, o que resulta em elevados tempos de recomposição e conseqüente indisponibilidade das SE's associadas a eles. Isto ocasiona a necessidade de deslocamento de equipes de operadores para as SE's a fim de realizar as manobras necessárias, durante o período de falha do PCOM. A solução mais lógica séria realizar este reset remotamente, através da instalação de uma unidade terminal remota (UTR) com esta finalidade. Entretanto, esta implementação não é tão simples uma vez que para utilização de uma UTR convencional é necessária a disponibilidade do próprio PCOM, que está em falha. Nesta situação, utilizaríamos um meio de comunicação e infra-estrutura alternativos, o que encarece a solução. Em função deste panorama optou-se pelo desenvolvimento de uma UTR que utiliza o meio de comunicação já existente. O desenvolvimento desta UTR é o objeto deste trabalho. Utilizando como base um micro-controlador da Microchip de baixo custo e o desenvolvimento de um protocolo de comunicação para garantir uma operação segura e elevada confiabilidade. Esta solução foi utilizada em uma área piloto e apresentou excelentes resultados.

1. Introdução

Nos últimos anos o setor elétrico brasileiro vem passando por grandes transformações, que foram intensificadas a partir de 1995 com o processo de privatização das empresas estatais de energia elétrica. Nesse contexto, a Celpe – Companhia Energética de Pernambuco – é a empresa detentora da concessão dos serviços públicos de distribuição de energia elétrica para o Estado de Pernambuco. A sua abrangência está sumariamente representada pelos dados do quadro abaixo:

Área de concessão	102.745 km²
Quantidade de municípios	186
População Total do Estado	7.918.344 habitantes
Domicílios Particulares do Estado	1.968.761

Quadro 1 - Dados de mercado

Para atender a esse mercado, a Celpe conta atualmente com 125 subestações de distribuição nas tensões de 13.8, 69 e 138kV, todas estão automatizadas. O que representa mais de 50.000 pontos monitorados.
Abaixo temos uma representação sucinta do sistema de automação da CELPE.

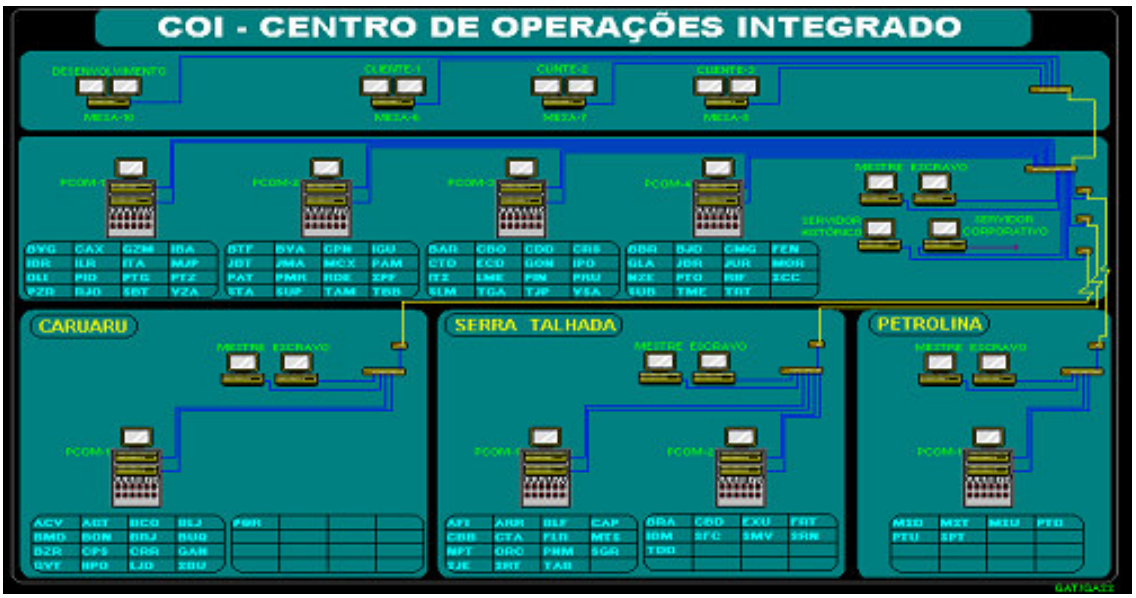

Fig. 1 – Arquitetura do sistema de automação

Os principais componentes desta arquitetura são:

UCC's – Unidades centrais de controle são unidades terminais remotas (UTR) responsáveis pela concentração e distribuição dos dados gerados dentro das subestações. Representadas acima pela sigla dentro de cada quadriculo.
PCOM's – Processadores de comunicação são unidades terminais remotas (UTR) responsáveis pela concentração e distribuição dos dados gerados por 16 subestações. Atualmente existem 6 na nossa sede em Recife, 3 na regional de Caruaru (130km do Recife), 3 na regional Serra Talhada (420km do Recife) e na regional de Petrolina (730km do Recife) num total de 14 em operação, na configuração 1+1 com reserva fria ou seja desligado.

Mestre de Comunicação – São servidores (SISC-X86) que concentram, processam e filtram todos os dados provenientes do sistema. Comunicam-se com os PCOM's e também estão na configuração de

1+1. Note-se que existem mestres regionais (Caruaru, Serra Talhada e Petrolina) que se comunicam com um único mestre geral, localizado em Recife.
Clientes – São servidores (SISC-X86) onde roda o software de controle e monitoramento (software SCADA), é onde são geradas as ações de operação e o monitoramento do sistema.
Nosso trabalho tem o foco nos processadores de comunicação (PCOM's) que se destacaram como a mais representativa origem de falha no sistema, pois sua falha ocasiona a perda de funcionalidade em 16 subestações simultaneamente. Nas figuras abaixo vemos o aspecto físico e sua estrutura interna.

fig. 3 – Aspecto físico do PCOM

Fig. 4 – Detalhe dos componentes internos

Note-se que o PCOM é essencialmente um micro industrial, com um software dedicado, desenvolvido pela empresa que forneceu o sistema de automação. Este equipamento possui um hardware robusto e com baixíssima taxa de falha. Entretanto os PCOM's vêem apresentando uma taxa de falha bastante significativa, devido ao seu componente de software. Os defeitos apresentados pelo software são intermitentes e como tal de difícil identificação e solução, uma vez que pode se originar nos próprios componentes do software, na interação entre eles ou na interação com o hardware, além disso, a intermitência é bastante esparsa o que dificulta ainda mais a identificação. Por outro lado a medida mais eficaz para restaura ao funcionamento normal do sistema é um simples "RESET", ou seja apenas reiniciar o PCOM, e foi esta a motivação para desenvolver o dispositivo objeto deste trabalho. Na figura baixo temos um gráfico com o numero de falhas dos PCOM's nos últimos 2 anos.

O software que "roda" no PCOM está longe de ser trivial, é montado em varias camadas e executa varias atividades simultaneamente. Na primeira camada temos o firmware da BIOS do cartão CPU, depois DOS 6.23 da Microsoft e sobre este o Rt-target que proporciona um ambiente multitarefa sobre

o quais são criadas várias instâncias, sobre as quais são executados os vários componentes do sofware, desenvolvidos em linguagem “C”.

2. Desenvolvimento

O dispositivo objeto deste trabalho denominado “reset de PCOM”, deve ser capaz de receber comandos de ligar e desligar individualmente até 4 PCOM’s, retornar o estado dos PCOM’s, utilizar canais de comunicação já disponíveis e utilizar-se de um protocolo de comunicação afim de garantir a confiabilidade dos comandos. Alem disso devera ser constituído pela menor quantidade de componentes possível, com elevada confiabilidade para garantir que a confiabilidade final do conjunto não seja menor que a do PCOM individualmente, o software deve ser o mais curto, rápido e confiável possível e o custo extremamente baixo. Após a definição das características pretendidas iniciou-se o projeto que seguiu as etapas descritas a seguir.

2.1 – Arquitetura do hardware

A primeira etapa do projeto é a definição da arquitetura do hardware, que no nosso caso optou-se por utilizar um microcontrolador, pois é necessário a utilização de protocolo de comunicação o que não é recomendável com o uso de componentes discretos devido ao elevado numero necessário e a falta de flexibilidade. Neste caso também é possível a utilização de um microprocessador, mas este necessitar de vários componentes auxiliares como memórias, buffers de entrada de saída, dentre outros, já o microcontrolador não necessita de componentes externos para seu funcionamento e o coloca como solução mais adequada. Na figura abaixo vemos um esquema de seus componentes básicos.

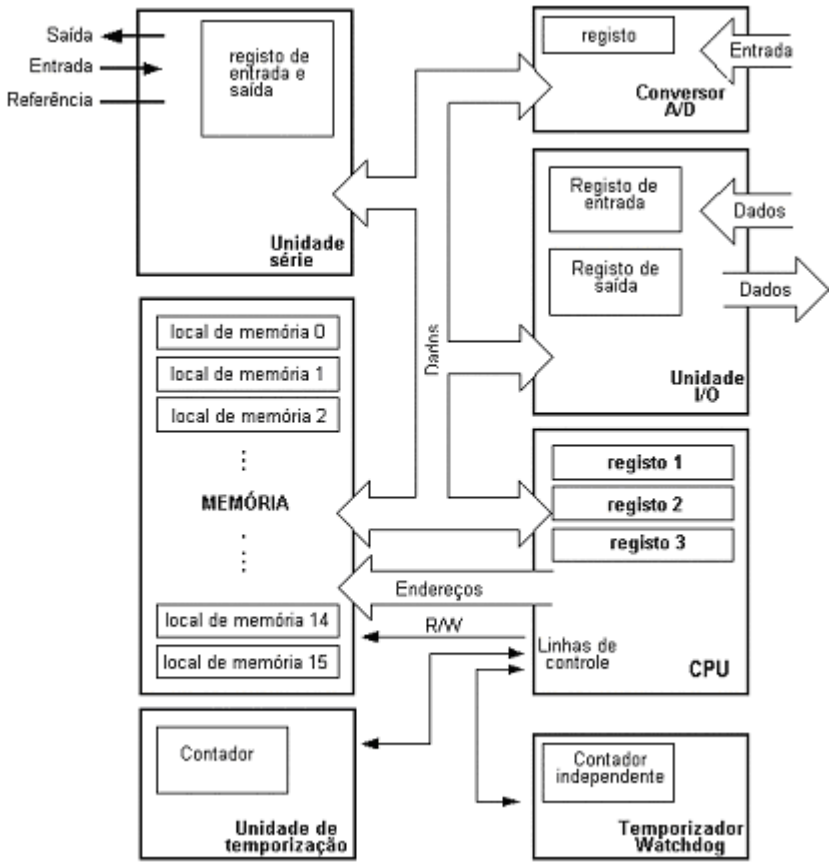

Fig.5 – Arquitetura interna de um microcontrolador

Uma vez feita a escolha por microcontrolador é necessário ainda definir-se quais das arquiteturas de instruções é a mais adequada, CISC ou RISC. Optamos pela arquitetura RISC, na implementação de Havard, por ser mais rápida que a de von-Neumann uma vez que as memórias de programa e dados

são separadas e que a maioria das instruções são executadas em um único ciclo de relógio. Além disso o conjunto de instruções é bem mais simples o que permite um controle preciso dos tempos de execução de cada rotina implementada. Foi selecionada a linha de microcontroladores da Microchip que atende perfeitamente todos os requisitos. Dentro da vasta linha de produtos da Microchip selecionamos o 12F629 que possui 6 I/O, oscilador interno com precisão de 1%, o dispensa a necessidade de cristal oscilador nesta aplicação. No quadro abaixo sua principais características.

High Performance RISC CPU:
• Only 35 instructions to learn
- All single cycle instructions except branches
• Operating speed:
- DC - 20 MHz oscillator/clock input
- DC - 200 ns instruction cycle
• Interrupt capability
• 8-level deep hardware stack
• Direct, Indirect, and Relative Addressing modes

Special Microcontroller Features:
• Internal and external oscillator options
- Precision Internal 4 MHz oscillator factory calibrated to ±1%
- 5 µs wake-up from SLEEP, 3.0V, typical
• Power saving SLEEP mode
• Wide operating voltage range - 2.0V to 5.5V
• Industrial and Extended temperature range
• Low power Power-on Reset (POR)
• Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)
• Brown-out Detect (BOD)
• Watchdog Timer (WDT) with independent oscillator for reliable operation
• Multiplexed MCLR/Input-pin
• Interrupt-on-pin change
• Individual programmable weak pull-ups
• Programmable code protection
• High Endurance FLASH/EEPROM Cell
- 100,000 write FLASH endurance
- 1,000,000 write EEPROM endurance
- FLASH/Data EEPROM Retention: > 40 years

Peripheral Features:
• 6 I/O pins with individual direction control
• High current sink/source for direct LED drive
• Analog comparator module with:
- One analog comparator
- Programmable on-chip comparator voltage reference (CV_{REF}) module
- Programmable input multiplexing from device inputs
- Comparator output is externally accessible
- Programmable 4-channel input
- Voltage reference input
• Timer0: 8-bit timer/counter with 8-bit programmable prescaler
• Enhanced Timer1:
- 16-bit timer/counter with prescaler
- External Gate Input mode
- Option to use OSC1 and OSC2 in LP mode as Timer1 oscillator, if INTOSC mode selected
• In-Circuit Serial Programming™ (ICSP™) via two pins

Quadro 2 – Pricipais características do PIC 12F629

Este microcontrolador possui ainda 1k (words) de memória de programa, 64 bytes de memória RAM e 128 bytes de EEPROM.

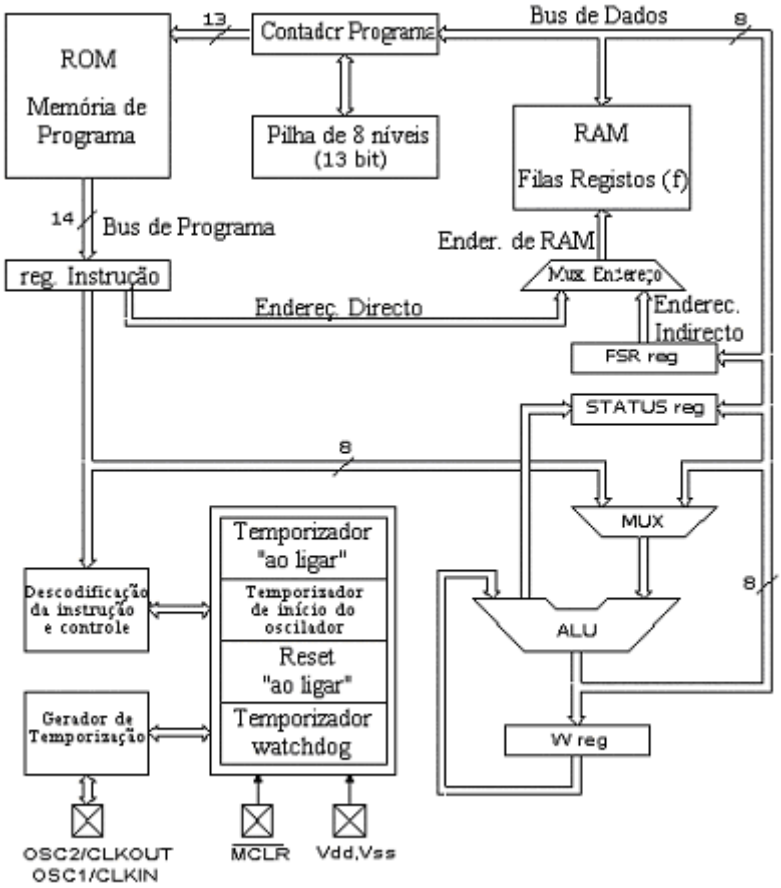

Fig.6 – PIC12F629 diagrama de blocos simplificado

2.2 – Projeto

Uma vez definido o microcontrolador, foi projetado o circuito eletrônico de nosso dispositivo, utilizando a menor quantidade de componentes possível, neste caso 20 componentes, além dos necessários para interfacear com o meio de comunicação (5 no caso RS232 ou 1 modulo de rádio de 433,92 MHz). Abaixo o diagrama esquemático do circuito eletrônico do reset de PCOM.

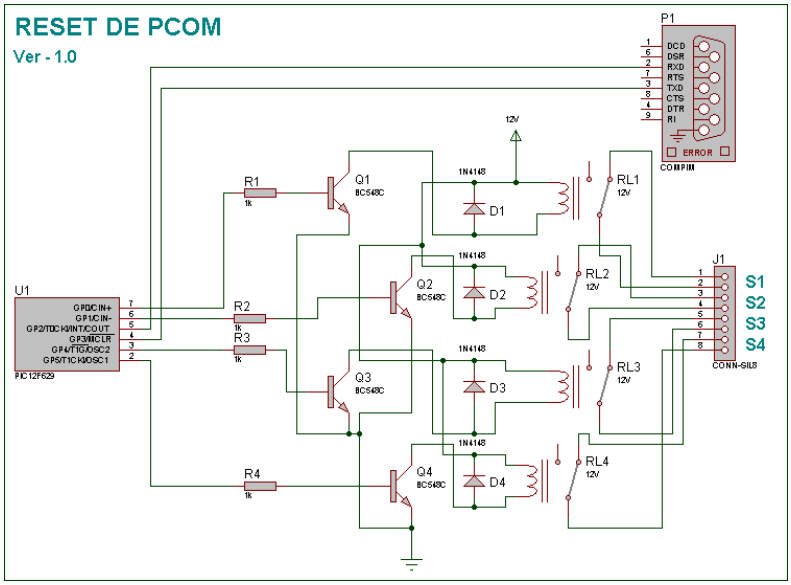

Fig. 7 – Diagrama esquemático

Como podemos ver no diagrama esquemático o circuito eletrônico é bastante simples, toda a lógica de funcionamento do dispositivo está implementada no software que será tratado mais adiante. Abaixo

vemos o layout da placa de circuito impresso que serve de suporte e interliga todos os componentes do dispositivo.

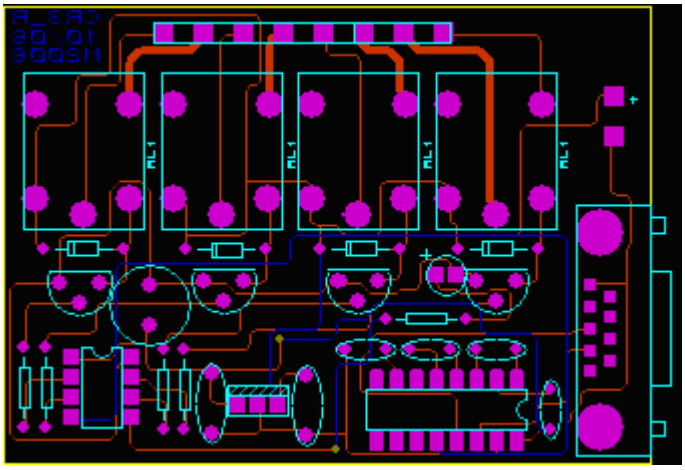

Fig. 8 – Layout da placa de circuito impresso

2.3 – Protótipo

Uma vez que o projeto foi concebido, chegou a hora de construir um protótipo, para realização dos ensaios necessários e testes de validação da solução. Posteriormente este protótipo foi instalado em uma área piloto. Nas figuras abaixo, vemos os vários componentes do protótipo.

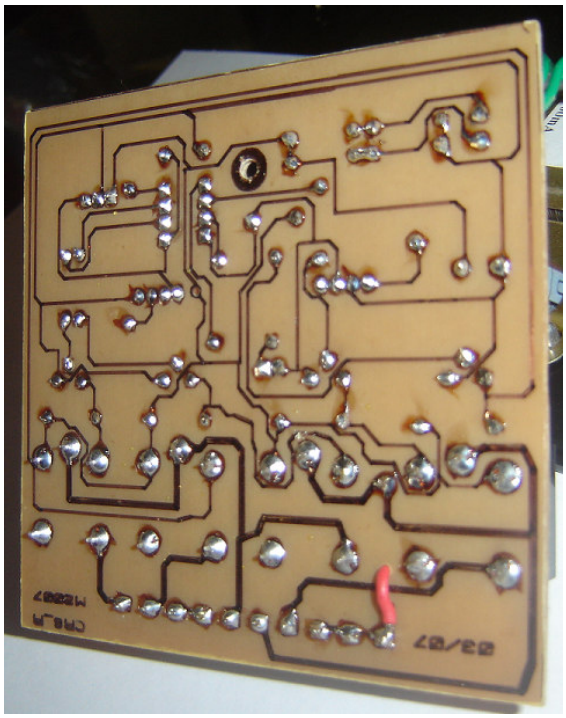

Fig. 9 – Placa de circuito impresso do protótipo

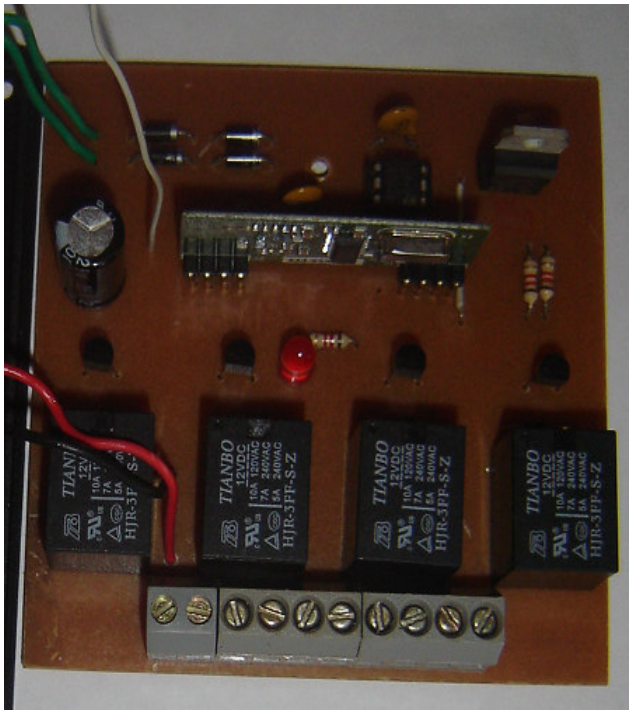

Fig.10 – Vista superior do protótipo (interface RF)

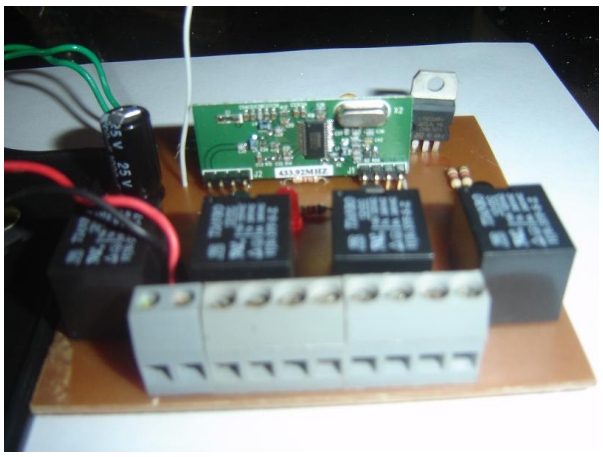

Fig. 11 – Vista frontal

Fig. 12 – Caixa para trilho DIN

2.3.1 – Softwares

Para este dispositivo foram desenvolvidos dois softwares, um em linguagem assembly, que constitui o firmware do dispositivo, este atua como um escravo e outro em DEPHI, que controla o dispositivo, por conseguinte funciona como mestre, faz também a função de IHM (interface homem máquina) e se comunica com o dispositivo através de um link RS232, com a utilização de um cabo cruzado ou atravez de rádio na freqüência de 433,92MHz. Nos tópicos seguintes vamos aborda-los mais detalhadamente.

2.3.2 – Firmware

O software que é executado no microcontrolador foi desenvolvido em linguagem assembly nativa do próprio microcontrolador. Este software implementa no dispositivo um protocolo de comunicação e uma porta de comunicação bidirecional serial ou uma porta de comunicação bidirecional serial sobre codificação Manchester para uso com link de radio. A elaboração deste software representou uma fase importante, meticulosa e trabalhosa do dispositivo, tendo em vista as severas limitações de memória o que determinou a linguagem a ser utilizada. Apesar da linguagem assembly ser muito difícil, só a utilização da mesma torna o projeto exeqüível com o uso deste hardware. Abaixo vemos o mapa de memória do microcontrolador.

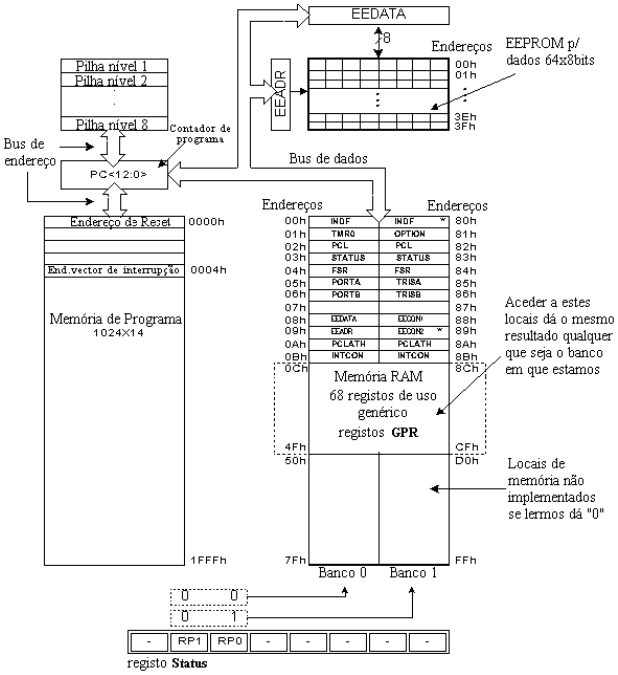

Fig. 13 – Mapa de memória do microcontrolador PIC16F629

2.3.2.1 – Protocolo de comunicação

Foi desenvolvido um protocolo de comunicação para este dispositivo com a finalidade de garantir a fidelidade dos dados traçados entre o dispositivo e a maquina mestre que executa o software de IHM. Para tanto utilizou-se um datagrama padrão com sete bytes de comprimento, dos quais 2 para definir o protocolo, 1 para o endereço do dispositivo, 1 para o comando a ser enviado, 1 para redundância em espelho e dois para o CRC (Cyclic Redundancy Check) utilizado para detectar erros na transmissão dos dados. O uso do CRC dispensa a necessidade do uso de bit de paridade na comunicação serial. O CRC utilizado é de 16 bits polinomial, e utiliza a equação $P(x) = X^{16}+X^{15}+X^{2}+1$.

2.3.2.2 – Comunicação serial bidirecional

Está funcionalidade foi implementada por software, embora a utilização de outros microcontroladores da Microchip eliminaria este desenvolvimento, uma vez que estes possuem porta serial nativa, entretanto era necessária a implementação da codificação manchester, logo optou-se por desenvolver a comunicação serial também o que gera independência em relação ao tipo de microcontrolador utilizado. A configuração da porta serial foi padroniza em:

Velocidade - 2400bps
Bits de start – 1
Bits de dados – 8
Paridade – Nenhuma
Bits de parada – 1
Controle de fluxo – Nenhum

A mesma configuração é utilizada para codificação manchester em um duticycle de 50%. Além disso também foi implementado um FIFO de 50 bytes.
Estes softwares funcionam da seguinte forma:
Um pacote de dados é enviado pelo mestre, este pacote é recebido pela rotina de comunicação e armazenado no FIFO. A rotina espera até que o fluxo de dados cesse e um tempo de espera (timeout) seja esgotado, daí então os dados são processados pela rotina do protocolo, que inicialmente calcula e verifica o CRC se for valido, verifica se o protocolo é valido, através dos dois primeiros bytes, se estiverem corretos então o protocolo confere o endereço, se estiver correto então compara o próximo byte com o inverso de seu sucessor se forem iguais então este é um comando válido e é executado. Tudo isso ocorre em cerca de 38ms, um tempo de resposta excelente. No quadro abaixo vemos um trecho do código fonte.

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                     00001 ;       Programa 12F
                     00002 ;       11/07/2005 ver
                     00003 ;       ultima alt. 12/07/2005
                     00004         PROCESSOR       12F629
                     00005         RADIX           DEC
                     00006         INCLUDE         "P12F629.INC"
                     00001         LIST
                     00002 ; P12F629.INC  Standard Header File, Version 1.04    Microchip
Technology, Inc.
                     00258         LIST
                     00007               INCLUDE               "C:\M16F84A.INC"
                     00001
                     03189         LIST
                     00008         ERRORLEVEL      -302
                     00009
2007   3F04          00010         __CONFIG        3F04H
                     00011
0000                 00012         ORG     00H
0000   2848          00013         goto    inicio
                     00014
                     00015 ;       ORG     04H
                     00016 ;       goto    inter

 inicio
0048   1683          00027         bsf             STATUS,RP0
0049   3008          00028         movlw   001000b
004A   0085          00029         movwf   TRISIO
004B   0195          00030         clrf    WPU
004C   1283          00031         bcf             STATUS,RP0
004D   0185          00032         clrf    GPIO
004E   3007          00033         movlw   7
004F   0099          00034         movwf   CMCON
0050   3006          00035         movlw   6
0051   00BF          00036         movwf   cont
0052   3002          00037         movlw   2
0053   00C1          00038         movwf   endereco
                     00039
                     00040 ;       bsf             GPIO,0
                     00041 ;       bsf             GPIO,1
                     00042
0054                   00043 loop
                     00044
                     00045 ;protocolo
                     00046
0054                   00047 enche_buffer
0054   20FC          00048         call    manchester

MPASM  4.02 Released                  12F_REC_FIFO.ASM   6-22-2007  14:06:35
```

Quadro 3 – Trecho de código do firmawe do dispositivo

2.3.3 – IHM (Interface Homem Máquina)

O software de IHM foi desenvolvido utilizado-se o DELPHI sobre a plataforma WINDOWS, e está instalado no servidor de comunicação regional que pode ser acessado remotamente. Este software funciona como um mestre para o dispositivo e se comunica com ele através de uma porta serial RS232. Em sua versão inicial trata-se de um software bastante simples, que supriu adequadamente a necessidade imediata, atualmente está em fase de finalização um software bem mais sofisticado que se comunica diretamente com o sistema SCADA e monitora constantemente o estado dos PCOM e quando detecta uma anomalia executa a operação de reset automaticamente, até um número de vezes previamente definido. Na figura abaixo vemos a tela principal desta aplicação.

5. Conclusão

Desde janeiro deste ano um protótipo está em operação na regional Caruaru e vem apresentando um desempenho extremamente satisfatório, funcionou corretamente a todas as solicitações e cumpriu seu objetivo de projeto, ou seja, reduziu o tempo de falha dos PCOM's da regional de algumas horas na maioria dos casos, para apenas alguns minutos. Com o aperfeiçoamento do software mestre este tempo deve cair para a casa dos segundos. Apesar de não ser uma solução definitiva este projeto mitiga satisfatoriamente o problema, reduzindo fortemente seu impacto no sistema. Por outro lado a empresa que desenvolveu o PCOM não existe mais, forçando a troca dos equipamentos, porem esta troca não precisa ser imediata e na maioria dos caso não será realizada até que o equipamento atinja o final de sua vida útil, pois com a implementação deste projeto as falhas dos PCOM's passaram a ser toleráveis. Isto representou uma economia da ordem de R$ 462.000,00 necessários para substitui-los.

6. Bibliografia

A. K. F. Correa, Delphi 5 com banco de dados e internet, São Paulo: Erica, 2000.
T. S. Weber, Um roteiro para exploração dos conceitos básicos de tolerância a falhas, UFRGS, 2005.
Microchip, PIC12F629/675 Data sheet DS41190C, 2003.