UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

**RONALDO HÜSEMANN**

# SISTEMA DE VALIDAÇÃO TEMPORAL PARA REDES DE BARRAMENTOS DE CAMPO

Porto Alegre
2003

RONALDO HÜSEMANN

# SISTEMA DE VALIDAÇÃO TEMPORAL PARA REDES DE BARRAMENTOS DE CAMPO

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica (PPGEE), da Universidade Federal do Rio Grande do Sul (UFRGS), como parte dos requisitos para a obtenção do título de Mestre em Engenharia Elétrica.
Área de concentração: Automação e Instrumentação Eletro-Eletrônica

ORIENTADOR: Prof. Dr. Carlos Eduardo Pereira

Porto Alegre

2003

RONALDO HÜSEMANN

# SISTEMA DE VALIDAÇÃO TEMPORAL PARA REDES DE BARRAMENTOS DE CAMPO

Esta dissertação foi julgada adequada para a obtenção do título de Mestre em Engenharia Elétrica e aprovada em sua forma final pelo Orientador e pela Banca Examinadora.

Orientador: _____

Prof. Dr. Carlos Eduardo Pereira, UFRGS

Doutor pela Universität Stuttgart – Stuttgart, Alemanha

Banca Examinadora:

Prof. Dra. Lucia Regina Horta Rodrigues Franco,

Doutora em Engenharia Elétrica pela Universidade de São Paulo – Brasil

Prof. Dr. Dr. João César Netto,

Doutor pela Universite Catholique de Louvain – Bélgica

Prof. Dr. Altamiro Amadeu Susin,

Doutor pelo Institut National Polytechnique de Grenoble – França

Coordenador do PPGEE: _____

Prof. Dr. Carlos Eduardo Pereira

Porto Alegre, maio de 2003.

Dedico este trabalho a meu pai, Erli, a minha mãe, Nelcy, pelo incondicional apoio, e a meu irmão Rudimar por sua consideração.

# AGRADECIMENTOS

Ao meu orientador Carlos Eduardo Pereira por prover a oportunidade de trabalho em minha área de pesquisa e pela confiança depositada nesta realização.

Agradeço também aos colegas e amigos João Mano Júnior, Rafael Laufer, João Paulo Dullius e Gustavo Vier, os quais tiveram participação presente durante o desenvolvimento deste trabalho.

Agradeço adicionalmente aos professores Renato Machado de Brito, Altamiro Susin e Hamilton Klimach pelo apoio em minha decisão de fazer mestrado no PPGEE.

Também aos colegas do PPGEE e do Laboratório de Automação Industrial Rafael Pereira Zeilmann, Hermes Gonçalves Junior, Rafael Wild, Leandro Becker, Cicero Lorenzi, Marcelo Negreiros e outros que não foram citados aqui, mas que, de alguma forma, me auxiliaram nas atividades desempenhadas neste período.

Por fim agradeço à CAPES pela provisão de minha bolsa de mestrado.

# RESUMO

Aplicações recentes no setor de automação industrial utilizam barramentos de campo para prover comunicação entre dispositivos. Estas aplicações normalmente exigem que os barramentos apresentem suporte tempo real. A garantia do adequado atendimento a requisitos temporais restritos é de fundamental importância para o correto funcionamento do sistema. Este documento apresenta um sistema de validação temporal para aplicações desenvolvidas utilizando tecnologias de barramentos de campo. O sistema desenvolvido, chamado BR-Tool, permite monitoração em tempo de execução de uma rede de barramento de campo, confrontando os dados obtidos com requisitos temporais previamente definidos pelo operador. O sistema BR-Tool é composto por dois elementos: um sub-sistema de aquisição de mensagens (placa de aquisição) e um sub-sistema de validação (ferramenta computacional). A placa de aquisição foi especialmente projetada para operar com diferentes interfaces de barramentos de campo, realizando as tarefas de captura de eventos, marcação temporal e salvamento de um histórico de eventos. A ferramenta de validação, que roda no computador, realiza as tarefas de filtragem de eventos, especificação de requisitos e validação temporal, permitindo diversos modos de visualização dos resultados. A comunicação entre a placa de aquisição e a ferramenta de validação é implementada por uma interface PCI, permitindo operar com velocidades de até 12Mbps.

**Palavras-chave: Barramentos de campo. Validação temporal. Sistemas tempo real.**

# ABSTRACT

Modern industrial automation applications are increasingly using fieldbus protocols to provide communication among automation devices such as sensors, actuators, and controllers. These applications normally present real-time requirements, for which the correct system's temporal constraints fulfillment is mandatory to ensure correct and safe operation. This work introduces a temporal validation system for fieldbus based automation applications. The developed system, called BR-Tool, allows runtime monitoring of a fieldbus network, comparing acquired network's runtime data with temporal constraints previously defined. The BR-Tool system consists of two parts: a message acquisition sub-system (acquisition board) and a validation sub-system (software tool). The acquisition board was specially designed to operate with different fieldbus interfaces, performing tasks such as event capture, time stamping and event logging. The validation tool, which executes on a computer, performs the tasks of event filtering, timing requirements specification, temporal validation, and also includes a graphical user interface for depicting validation results. Communication between acquisition board and validation tool is implemented by a PCI interface, allowing operation with rates up to 12Mbps.



**Keywords: Fieldbus. Temporal validation. Real-time systems.**

# SUMÁRIO

# LISTA DE ILUSTRAÇÕES

# LISTA DE TABELAS

# LISTA DE ABREVIATURAS

ATRT - Actual Token Rotation Time

CAN - Controller Area Network

COB - Communication OBject

COTS - Commercial Off The Shelf

CSRD - Cyclic Send and Request Data with Reply

CTDMA - Concurrent Time Domain Multiple Access

DMA - Direct Memory Access

DP - Decentralized Periphery

ES - Executive Sequence

FCFS - First Come First Served

FDL - Fieldbus Data Link

FMA - Fieldbus MAnagement

FMS - Fieldbus Message Specification

FPGA - Field Programmable Gate Array

HSE - High Speed Ethernet

IEC - International Electrotechnical Commision

IP - Internet Protocol

ISA - Industrial Standard Architecture

ISO - International Stardardization Organization

IUC - Intelligent Universal Controller

KMD - Kernel Mode Driver

LAS - Link Active Scheduler

LLF - Least Laxity First

LLI - Lower Layer Interface

LOTOS - Language Of Temporary Ordening Specification

LSAP - Local Service Access Point

MAC - Medium Access Control

MAP - Manufacturing Automation Protocol

MIPS - Million of Instructions Per Second

NM - Network Manager

NRZ - Non Return to Zero

NS - Next Station

NUT - Network Update Time

OSI - Open Systems Interconnection

PA - Process Automation

PC - Personal Computer

PCI - Peripheral Component Interconnect

PDO - Process Data Object

PNMA - Performance Network Management Application

PS - Previous Station

PT - Pass Token

RAM - Random Access Memory

RISC - Reduced Instruction Set Computer

RTL - Real Time Logic

SAP - Service Access Point

SDA - Send Data with Acknowledge

SDN - Send Data with No acknowledge

SoC - System on Chip

SRAM - Static Random Access Memory

SRD - Send and Request Data with Reply

TDMA - Time Division Multiple Access

THT - Token Holding Time

TMP - Test and Measurement Processor

TRT - Token Rotation Time

TS - This Station

TTRT - Total Token Rotation Time

USB - Universal Serial Bus

VHDL - VHSIC Hardware Description Language

VLSI - Very Large Scale of Integration

WDM - Windows Driver Model

# 1 INTRODUÇÃO

Sistemas de automação industrial têm se tornado uma realidade cada vez mais presente em empresas de manufatura e processos, visando melhoria na qualidade de produtos, maior controle na produção e menores custos. Antigos sistemas de automação se baseavam no controle centralizado, normalmente empregando um computador de grande capacidade para realizar o processamento, através do uso de placas de entradas e saídas conectadas ao mesmo, de inúmeros nós (pontos de entrada e/ou saída) espalhados pela planta industrial. Esta abordagem entretanto apresentava algumas desvantagens como baixa flexibilidade, grande dependência da estrutura do sistema com o computador principal e elevados custos de instalação e manutenção.

Com os avanços mais recentes alcançados nas áreas de microeletrônica e informática passou-se também a utilizar microprocessadores embarcados nos diversos elementos da planta industrial, trazendo como vantagem direta uma maior autonomia dos dispositivos (ULLOA et. al., 1992; TOVAR et. al., 1999). Os dispositivos (sensores e atuadores) passaram a utilizar esta capacidade de processamento para prover interfaces de comunicação que permitissem a troca de dados não só com a estação central de controle como também entre elas mesmas, através de um barramento compartilhado.

Diversos protocolos de comunicação, alguns proprietários de empresas desenvolvedoras, outros mantidos por organizações regionais ou internacionais na forma de padrões abertos, foram desenvolvidos, nos últimos anos, visando a interligação de equipamentos distintos em uma mesma rede (SCHICKHUBER; MCCARTHY, 1997). Tendo sido inicialmente aplicado no nível de campo da pirâmide de automação (atualmente apresentam-se soluções que se estendem a todos os níveis), este tipo de tecnologia recebeu o nome de **barramento de campo**.

Devido às exigências severas de tempo de atuação e resposta de grande parte das aplicações de automação industrial, os barramentos de campo não podem se limitar a somente prover comunicação entre diferentes equipamentos, como também devem apresentar suporte a aplicações tempo real (PEREIRA; WILD, 1999). Não raramente ocorrem situações na operação de células de manufaturas ou plantas industriais, onde o atraso na execução de uma ação de controle pode causar graves prejuízos ou até mesmo danos a vidas humanas.

A tecnologia de barramento de campo deve assegurar o atendimento de requisitos temporais de aplicações implementadas, o que é uma característica comum de sistemas tempo real distribuídos. Assim sendo se torna adequado o estudo de adaptabilidade de sistemas de avaliação temporal, desenvolvidos para sistemas tempo real distribuídos, também para emprego com redes de barramentos de campo.

Técnicas de verificação formal, as quais visam garantir antecipadamente o atendimento temporal a requisitos da aplicação, têm, em geral, pouca aplicabilidade para avaliação de barramentos de campo, devido à alta complexidade de modelos para redes industriais, as quais são geralmente compostas por diversos elementos com tecnologias e características de implementações distintas. Além disso, a ocorrência de eventos aleatórios ou espúrios na rede, imprevisíveis durante a etapa de projeto, podem levar a situações inesperadas que venham a comprometer os resultados previamente calculados (TSAI; YANG, 1995).

Para sistemas tempo real complexos muitas vezes se utiliza a estratégia de validação temporal, onde o comportamento do sistema, observado em tempo de execução por monitoração de seu funcionamento durante uma operação real, é confrontado com requisitos temporais previamente definidos. Os resultados obtidos após a realização da análise do comportamento temporal do sistema monitorado podem servir de auxílio aos projetistas das

aplicações envolvidas, ajudando-os a identificar possíveis falhas de programação e/ou pontos críticos no projeto.

Diversos sistemas de validação temporal têm sido desenvolvidos, nos últimos anos, para análise do atendimento de requisitos de aplicações em sistemas tempo real distribuídos, principalmente visando utilização em redes de computadores (CHODROW; JAHANIAN; DONNER, 1991; JAHANIAN; RAJKUMAR; RAKU, 1994; LIU; HÁ, 1997).

Em especial se destacam os sistemas de monitoração, validação e visualização integrados, onde o usuário, conhecedor do sistema sob análise, pode especificar as restrições temporais próprias de sua aplicação, as quais serão consideradas para validação do sistema (TOKURA; KOTERA; MERCER, 1988; MINK et. al., 1998; SHOBAKI; LINDH, 2001).

## 1.1 MOTIVAÇÃO

Conforme mencionado anteriormente, sistemas de automação industrial normalmente apresentam requisitos temporais cujo atendimento é imprescindível para o correto funcionamento do sistema, sendo que violações destes requisitos podem levar a situações de alto risco. Nos sistemas de automação industrial desenvolvidos com tecnologia de barramentos de campo, o comportamento dos sistemas é fortemente dependente dos protocolos de comunicação adotados.

Analisando as tendências do mercado de automação industrial mundial, conforme pode ser evidenciado pela norma internacional IEC61158 (IEC, 2001), observa-se que este deve continuar sendo composto por distintas soluções de barramento de campo. O determinismo temporal das aplicações desenvolvidas neste mercado depende portanto da tecnologia utilizada e dos diferentes mecanismos de escalonamento de mensagens: passagem

de *token*, reserva estática de tempo para cada estação, *token* virtual, escalonador central (árbitro) de barramento, entre outros.

Metodologias e ferramentas para análise do comportamento temporal para diferentes protocolos têm sido alvo de pesquisas em várias universidades e centros de pesquisa (TINDEL; BURNS, 1994; MARINO et. al., 1998; TOVAR; VASQUES; BURNS, 1999; TOVAR et. al., 2002). Todas estas propostas, entretanto, enfocam apenas um determinado protocolo por vez. Um sistema genérico intercambiável entre diferentes tecnologias é ainda uma necessidade emergente.

A proposta da presente dissertação é desenvolver um ambiente genérico de validação temporal aplicável a diferentes tecnologias de barramentos de campo, dando ênfase especial para os protocolos definidos na norma IEC61158. Os principais objetivos do ambiente proposto são os seguintes:

a) validar o comportamento temporal de sistemas de automação industrial baseados em barramentos de campo, com suporte a diferentes protocolos de comunicação, identificando se os sistemas implementados atendem aos requisitos temporais das aplicações;
b) comparar o desempenho de diferentes protocolos, buscando-se fornecer informações que permitam se determinar qual o melhor protocolo para uma determinada aplicação.

Esta dissertação é dividida da seguinte forma: o capítulo 2 apresenta um estudo sobre estado da arte de sistemas de avaliação temporal; o capítulo 3 traz um resumo sobre a tecnologia de barramentos de campo com especial atenção voltada para os barramentos presentes na especificação IEC 61158; o capítulo 4 apresenta a proposta do sistema de validação desenvolvido nesta dissertação; o capítulo 5 descreve os recursos de *hardware* do trabalho enquanto que o capítulo 6 apresenta as implementações de *software* da ferramenta; o

capítulo 7 traz a verificação formal das capacidades do sistema; o capítulo 8 descreve os resultados obtidos com alguns estudos de caso e o capítulo 9 apresenta as conclusões da dissertação e propostas de trabalhos futuros.

# 2 ANÁLISE DO ESTADO DA ARTE EM SISTEMAS DE AVALIAÇÃO TEMPORAL

A definição do estado da arte de sistemas relacionados com esta dissertação passa pela análise de trabalhos desenvolvidos pela comunidade acadêmica nos campos de validação temporal de sistemas, bem como de trabalhos que estudem o determinismo temporal de barramentos de campo.

## 2.1 DEFINIÇÃO E CLASSIFICAÇÃO DE SISTEMAS DE VALIDAÇÃO TEMPORAL

Sistemas tempo real exigem não somente a adequada funcionalidade de suas aplicações como também, e principalmente, que a execução de suas tarefas ocorra dentro de prazos pré-determinados. O conjunto das especificações de tempo para um sistema ou aplicação (tais como tempo máximo para realização de uma tarefa, periodicidade de funções cíclicas, máximo tempo de resposta a uma requisição, etc) compõe seus requisitos temporais. Aplicações tempo real rígidas (*hard real time*) não aceitam falhas no atendimento a estes requisitos sob risco de graves prejuízos ao sistema ou ao ambiente onde estiver operando. Para garantir o seu funcionamento adequado é portanto necessário garantir-se o correto atendimento de seus requisitos temporais. Existem basicamente duas formas de avaliar o atendimento destes requisitos: verificação e validação temporal (TSAI; YANG, 1995).

A verificação temporal (também chamada de análise estática) é o procedimento de demonstração teórica, durante a fase de desenvolvimento, que tem por objetivo comprovar se um sistema tempo real atende a seus requisitos temporais (IEEE, 1990). Para tanto faz-se uso do modelo matemático do sistema, que uma vez equacionado pode ser verificado por cálculos ou pelo uso de ferramentas matemáticas. Este método apesar de trabalhoso é fortemente indicado para aplicações com requisitos tempo real rígidos, pois visam garantir *a priori*, isto é, antes mesmo da implementação física do sistema, que os requisitos temporais serão atendidos.

A validação temporal (ou depuração dinâmica) é o processo de avaliação de características temporais de um sistema confrontando requisitos especificados com informações práticas obtidas a partir de monitoração de seu funcionamento (STEWART; GENTLEMAN, 1997). Os dados utilizados pelo sistema de validação podem ser adquiridos em tempo de execução ou extraídos de um arquivo de histórico previamente salvo. Sistemas de validação são normalmente mais simples de se implementar quando comparados com o procedimento de verificação, uma vez que podem ser utilizados de forma genérica sem conhecimento aprofundado da implementação da aplicação. Apresentam porém a desvantagem de não poderem fornecer uma real garantia do correto funcionamento do sistema alvo, mesmo que nenhuma falha tenha sido registrada durante seu período de análise (somente se pode garantir a ausência de erros nos casos simulados/testados, mas não em qualquer condição de operação).

Na prática, os modernos sistemas tempo real distribuídos exigem modelos matemáticos extremamente complexos, uma vez que normalmente são compostos por elementos distintos com características e parâmetros individuais diferenciados, algumas vezes, até imprevisíveis. Isto torna a tarefa de verificação formal de muitos sistemas tempo real impraticável (HABAN; WYBRANIETZ, 1990; LIU; MOK, 1997). O uso de sistemas de validação têm tido, por outro lado, uma grande utilização na avaliação de sistemas temporais distribuídos, principalmente como suporte durante a fase de projeto.

Um sistema de validação completo pode ser basicamente subdividido em quatro módulos: Módulo de captura/registro de eventos, módulo de validação, módulo de especificação de requisitos temporais e módulo de visualização, conforme apresentado na Figura 1.

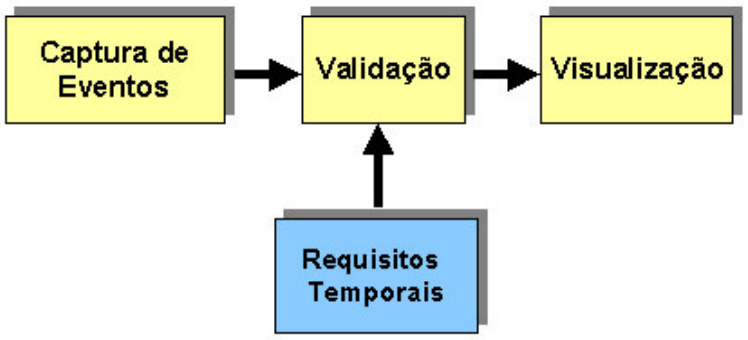



**Figura 1 - Diagrama de blocos de uma ferramenta de validação**

O módulo de captura de eventos é responsável pela tarefa de deteção, coleta e registro de tempo de ocorrência de eventos significativos, tais como a mudança de estado de um dos componentes do sistema de automação, o envio/recebimento de uma mensagem específica, entre outros. O módulo de validação é responsável pela avaliação se os eventos detectados se adequam aos requisitos temporais da aplicação. O módulo de especificação de requisitos temporais deve comportar formalmente, com clareza e sem ambiguidades, a especificação das restrições temporais para a aplicação alvo. Por fim, o módulo de visualização é responsável pela exibição dos resultados processados pela ferramenta.

Parte importante de um sistema de validação é a unidade de monitoração (responsável pela captura de eventos), uma vez que a confiabilidade do sistema está diretamente vinculada à adequada consistência das informações temporais detectadas (THANE, 2000). Tendo-se em vista esta unidade pode-se subdividir os sistemas de validação em três tipos:

a) monitoração por *software*: onde toda monitoração é implementada dentro do código do programa da aplicação pela inserção de pontos de medida em trechos significativos a serem monitorados;
b) monitoração por *hardware*: onde a monitoração é feita externamente ao sistema alvo pela amostragem de sinais de barramento ou de memória compartilhada, através de módulos eletroeletrônicos conectados ao sistema;

c) monitoração híbrida: quando se utilizam ambos os métodos em conjunto para obter as informações de monitoração.

O primeiro método tem acesso direto aos pontos de inicialização e término das funções, bem como das variáveis internas do sistema, podendo detectar qualquer variação no preciso instante em que ocorre (TOKURA; KOTERA; MERCER, 1988; JAHANIAN; MOK, 1994; CATANIA et. al., 1996). Traz entretanto a desvantagem de ser um método invasivo ou interferente, pois normalmente compartilha recursos de processador e memória, o que inevitavelmente causa interferência no comportamento temporal do sistema alvo quando utilizado. Este fenômeno não é desejável, mas muitas vezes é aceito se a interferência for considerada pouco significativa.

Um sistema de monitoração por *hardware* é totalmente não invasivo, uma vez que apenas lê sinais externos ao sistema, sejam os barramentos do processador, suas memórias ou as interfaces seriais e paralelas, sem influir nos valores (HABAN; WYBRANIETZ, 1990; JUNDI; MOON, 1992). Uma desvantagem é a necessidade de se desenvolver circuitos de monitoração e interpretação destes sinais especialmente dedicados para a arquitetura do sistema alvo, o que pode se tornar complexo para os desenvolvedores da aplicação. Outra desvantagem é o fato de não ter acesso a informações internas do sistema, que estiver sendo analisado, como por exemplo a alteração de registradores internos do processador principal.

A terceira abordagem alia ambos os métodos para obter um quadro mais completo do sistema, permitindo a obtenção de informações intradispositivos, a partir da monitoração por *software*, e interdispositivos, obtida com monitoração externa por *hardware* (TSAI; FANG; CHEN, 1990; AL-ANBUKI; DANIAL, 1993, MIK et. al., 1998; SHOBAKI; LINDH, 2001). Este monitor apresenta as desvantagens de compor um sistema relativamente complexo de se programar, a fim de gerenciar todas as diferentes informações obtidas, e de ainda causar interferência no sistema alvo, devido a sua parte implementada em *software*.

Estes conceitos são válidos para quaisquer sistemas de avaliação temporal sendo atualmente aplicáveis em diversas áreas como redes de telefonia, controle de processos, monitoradores clínicos de pacientes, controle de robôs, etc.

Preocupações constantes com o comportamento temporal de aplicações de automação têm levado as comunidades científica e industrial ao desenvolvimento de diversos trabalhos visando avaliar o desempenho temporal de barramentos de campo. Os trabalhos encontrados vão desde estudos sobre o equacionamento teórico dos mecanismos de escalonamento de cada protocolo, responsável em primeira instância pelo determinismo temporal do sistema, como também para o desenvolvimento e uso de ferramentas de avaliação de performance temporal, que analisam os resultados obtidos a partir de monitoração da atividade no sistema alvo.

A seguir são apresentados diversos trabalhos publicados que caracterizam o estado da arte na área de validação temporal. A fim de facilitar a apresentação, os trabalhos foram basicamente separados em três grupos. Inicialmente apresentam-se trabalhos relacionados ao desenvolvimento de sistemas genéricos de validação. Posteriormente apresentam-se diversos trabalhos relevantes na avaliação temporal de barramentos de campo. Por fim discutem-se soluções comerciais atualmente disponíveis para a avaliação do comportamento de redes de barramentos de campo.

## 2.2 TRABALHOS RELACIONADOS À VALIDAÇÃO TEMPORAL

Com o avanço da eletrônica digital e principalmente dos dispositivos embarcados, a partir dos anos 80, o desenvolvimento de sistemas tempo real distribuídos aumentou de importância. O avanço tecnológico crescente tem levado também ao desenvolvimento de sistemas tempo real cada vez mais complexos.

Visando auxiliar o desenvolvimento destes sistemas, vários estudos sobre arquiteturas de validação de sistemas tempo real têm sido desenvolvidos pelas comunidades acadêmica e científica.

No trabalho (TOKURA; KOTERA; MERCER, 1988) apresenta-se um sistema de monitoração baseado em *software* para avaliação temporal de sistemas distribuídos. O sistema é entretanto invasivo, pois exige que a aplicação alvo seja desenvolvida sobre um sistema operacional ARTS, projetado pelos autores.

O trabalho de Tsai, Fang e Chen (1990) descreve um sistema monitorador tempo real desenvolvido em *hardware*, que se baseia na coleta de informações diretamente dos barramentos internos do sistema, suportando diferentes níveis de abstração, como atividades a nível de processo (comunicação interna e externa ao dispositivo alvo), nível de função (chamada de procedimentos) e nível de instrução (por exemplo execução passo a passo).

O sistema implementado por Haban e Wybranietz (1990) combina as abordagens de monitoração por *hardware* e *software* para medida de performance e observação do comportamento de sistemas distribuídos durante execução. Para suporte a *hardware* foi especialmente desenvolvido um dispositivo de medição chamado *Test and Measurement Processor* (TMP) para ser parte integrante de cada estação em um sistema de múltiplos computadores. Diversos TMP são conectados via uma rede independente com uma estação de

monitoração central. A geração de eventos no sistema é feita a partir da inserção de instruções específicas no código fonte em pontos escolhidos para monitoração.

O artigo (JUNDI; MOON, 1992) propõe o desenvolvimento de uma estratégia de *hardware* para monitoração da execução de códigos em projetos baseados em microprocessadores de tecnologia *Reduced Instruction Set Computer* (RISC). Nesta solução registradores de propósito geral são usados para manipulação de dados e resultados. O sistema procura também monitorar o estado de registradores internos do microprocessador, mantendo uma imagem do processador principal implementada em lógicas programáveis Xilinx, que opera em paralelo com o mesmo.

Em (AL-ANBUKI; DANIAL, 1993) apresenta-se um sistema de monitoração que utiliza um computador padrão IBM-PC como elemento supervisor, gerenciando dados extraídos a partir de uma placa externa de aquisição e controle. A comunicação entre a placa de aquisição e o computador é feita por uma memória dupla porta. Os dados são adquiridos e carregados para a memória dupla porta, armazenando a diferença de tempos entre processos (por exemplo, entre eventos de produção e consumo de dados).

É descrito no artigo de Jahanian, Rajkumar e Raju (1994) um ambiente de validação distribuída. A monitoração é implementada puramente por *software*, através de um conjunto de processos de monitoração cooperativos, um processo rodando em cada processador. Cada processo monitor mantém um conjunto de restrições temporais referentes ao seu contexto, que são validadas a partir de um histórico de eventos de tarefas da aplicação local. Eventos de interesse global podem ser enviados para outros monitores.

O trabalho de Catania et. al. (1996) descreve uma plataforma de desenvolvimento que possibilita a monitoração tempo real da performance de objetos distribuídos, a partir de aplicações especiais chamadas *Performance Network Management Application* (PNMA). Os processos desenvolvidos apresentam recursos embutidos de administração de eventos

(notificação e armazenamento), gerência de dados e manutenção de recursos de sincronização. Para aumento da flexibilidade em redes de computadores, processos roteadores são disponibilizados permitindo-se trabalhar com taxa média de pacotes, tempo de atraso médio nas filas de mensagens e notificação de eventos no roteador.

É apresentada no artigo de Liu e Mok (1997) uma abordagem para a especificação de eventos e restrições temporais. A base da especificação de eventos adotada segue a notação RTL e gráficos de restrição (JAHANIAN; RAJKUMAR; RAKU, 1994). O artigo, por fim, apresenta um estudo de caso de um monitorador em tempo de execução implementado para verificar restrições temporais em aplicações Java. O monitorador proposto roda em diferentes processadores coletando informações de eventos enviados a partir de mensagens pelas tarefas analisadas. Quando violações de especificações forem detectadas as aplicações do usuário podem ser notificadas por mensagens de exceção.

O trabalho de Mink et. al. (1998) apresenta uma proposta de sistema de monitoração baseada no uso de uma placa de aquisição padrão *Peripheral Component Interconnect* (PCI), que possui diversos temporizadores de alta resolução. A placa se baseia em componentes de tecnologia *Very Large Scale of Integration* (VLSI), chamados MultiKron, especialmente projetados para registro temporal de eventos com um processador externo através de mapeamento em memória. A placa usa um dispositivo programável que implementa as funções de um dispositivo alvo PCI 32 bits a 33MHz. A placa de aquisição é integrada a uma ferramenta de medida de performance de programas, chamada Paradyn. A ferramenta utiliza processos de monitoração chamados Paradyn *daemon*, que amostram periodicamente as estruturas de dados disponíveis para processar e exibir na tela os resultados obtidos.

No artigo (SHOBAKI; LINDH, 2001) apresenta-se um sistema de monitoração chamado MAMon que utiliza um pequeno componente de *hardware* desenvolvido como *System on Chip* (SoC), que pode ser incorporado ao processador principal do equipamento a

ser analisado. A unidade de medição compara sinais do barramento com condições internamente estabelecidas para a deteção de eventos. Quando um evento é detectado, este é armazenado junto com a informação temporal do instante de ocorrência e com uma estrutura de parâmetros adicionais que pode armazenar informações relevantes do evento. A ferramenta de operação implementa a interface com o usuário, manipulando e exibindo os dados obtidos pela unidade de medição, de forma a facilitar as tarefas de identificação de padrões errôneos na execução do sistema. Estatísticas são fornecidas pelo sistema além de diagramas e histogramas.

## 2.3 TRABALHOS LIGADOS A BARRAMENTOS DE CAMPO

Diversos trabalhos relacionados com a análise de comportamento temporal de redes de barramento industrial têm sido realizados nos últimos anos. A análise temporal destas redes deve levar em conta as características próprias de cada solução.

No trabalho (SHIN; CHOU, 1992) é proposta uma solução para suporte de comunicação tempo real em redes Foundation Fieldbus. Para uma solução completa sugere-se a divisão do problema de gerência temporal da comunicação na rede em duas partes: comunicação intra-rede e comunicação inter-rede. A comunicação intra-rede se baseia na comunicação entre elementos situados no mesmo barramento local, cuja gerência de acesso ao meio é feita por um escalonador ativo de conexão, *Link Active Scheduler* (LAS), através de mensagens de *token*. A comunicação inter-rede representa a troca de dados entre estações localizadas em redes distintas, onde um gerente de rede, *Network Manager* (NM), centraliza informações geradas pelas estações, estabelecendo rotas de mensagens e controle de serviços orientados à conexão. Assim para a comunicação entre estações em uma mesma rede local, o elemento LAS se responsabiliza por garantir o tempo necessário a cada estação, reservando ainda tempo para que o NM possa agendar as mensagens inter-rede requisitadas.

O artigo de Tindel e Burns (1994) apresenta a análise do comportamento temporal de uma rede CAN, principalmente considerando a garantia de latências de mensagens. Para implementação do modelo proposto é usado como elemento base o controlador CAN 82527. O uso do mecanismo de bit dominante do protocolo traz intrínseco o conceito de prioridade entre mensagens. O pior caso de tempo de resposta é portanto dependente do valor de prioridade de cada mensagem. É apresentado neste artigo o equacionamento do tempo de resposta relacionado a diferentes valores de prioridade. Além disso consideram-se parâmetros como a variação temporal no atendimento da fila de mensagens do controlador 82527 (15 mensagens ao todo), carga da rede por cabeçalhos das mensagens e uso de *bit stuffing*.

O artigo de Hong e Ko (1998) analisa as características temporais do barramento de campo Foundation Fieldbus. Após finalizar a passagem por todas estações, o árbitro confere o tempo de rotação do *token* atual (*Actual Token Rotation Time* ou ATRT) para determinar a prioridade do novo ciclo de *token*, o qual é calculado comparando-se ATRT com o tempo de rotação total do *token* (*Total Token Rotation Time* ou TTRT). A simulação consiste de um submodelo de geração de mensagens, que determina as mensagens a serem transmitidas e que são colocadas em uma pilha de transmissão, e um submodelo de protocolo, que transmite a mensagem de acordo com o protocolo da camada de enlace de dados. Foram montados três cenários distintos: a) LAS usando somente serviços de passagem de *token* (*Pass Token* - PT), que suporta a troca de dados cíclicos de acordo com a lista de circulação de *token* previamente gravada, b) LAS usando somente serviços de execução (*Executive Sequence* - ES), que possibilitam o escalonamento dinâmico de acesso às estações ao barramento e c) LAS usando serviços PT e ES. Neste último caso, uma parcela do tempo total de rotação do *token* TTRT é reservada para mensagens cíclicas com serviço PT, enquanto que o tempo restante é administrado por escalonamento dinâmico usando serviços ES.

No artigo (MARINO et. al., 1998) descreve-se uma especificação formal para simulação e avaliação dos níveis de camada de enlace de dados e gerenciamento de uma rede PROFIBUS através da linguagem *Language Of Temporary Ordering Specification* - LOTOS, implementando as diversas camadas do protocolo. O primeiro processo implementado, PHY, representa a camada física do protocolo, sendo responsável pela conversão de mensagens MAC em bytes a serem enviados pelo barramento. O processo FDL implementa a máquina de estados da camada de enlace de dados, que deve monitorar o anel lógico entre estações mestre da rede, montando sua lista de estações ativas (LAS) por varredura. Os parâmetros mais importantes deste tipo de rede são Ttr (tempo de rotação do *token*) e GAP (fator de atualização). São suportados até 4 serviços pelo protocolo: SDA (envio de dados com confirmação de recepção), SDN (envio de dados sem confirmação de recepção), SRD (envio de dados com requisição de resposta) e CSRD (serviço SRD repetido ciclicamente). A camada de gerência FMA 1/2 deve implementar funções como reinicialização das camadas física e de enlace de dados, modificação de estados atuais e contadores das camadas 1 e 2, notificação de eventos inesperados, erros ou mudanças de estado, além de ativação, configuração e desativação de serviços LSAP.

No artigo de Kaiser, Livani e Jia (1999) é apresentado um mecanismo de escalonamento distribuído para uma rede CAN, a fim de garantir o atendimento de requisitos temporais. São considerados nesta proposta três tipos de atividades: tempo real rígida (*hard real time*), tempo real suave (*soft real time*) e não tempo real. O artigo propõe um escalonamento híbrido que combina o determinismo do mecanismo *Time Division Multiple Access* (TDMA) com a flexibidade do escalonamento dinâmico do esquema *Least Laxity First* (LLF). O método de escalonamento híbrido divide o campo de prioridades de mensagens CAN em três faixas. A faixa de prioridades mais elevadas é reservada para as mensagens tempo real rígidas, gerenciada pelo mecanismo TDMA. A faixa intermediária usa o esquema

de prioridade dinâmica LLF. Os serviços não tempo real, por fim, são assinalados como de baixa prioridade.

No artigo de Tovar, Vasques e Burns (1999) é realizada a análise das características necessárias para garantir o determinismo temporal em redes do tipo P-NET, que utilizam o mecanismo de passagem de *token* virtual para prover o acesso dos mestres na rede compartilhada. Este mecanismo se baseia no uso de um contador de tempo de linha muda, que é inicializado a cada novo ciclo. Quando o contador atinge um valor de contagem pré-determinado o mestre selecionado assume isto como uma passagem de *token* e pode então acessar o barramento. Apresenta-se neste artigo um modelo de análise temporal que leva em conta o período de utilização de *token* a cada ciclo para diferentes mestres. Cada mestre é considerado como possuidor de uma fila de mensagens administrada pelo esquema *First Come First Served* (FCFS). O modelo proposto apresenta resultados mais realistas por não considerar sempre o pior caso de tempo de permanência do *token* para determinação de tempo de resposta. Por fim são apresentados exemplos numéricos que ilustram o modelo.

O artigo de Pereira e Wild (1999) apresenta uma ferramenta para avaliação de requisitos temporais de aplicações desenvolvidas utilizando o barramento Foundation Fieldbus. Os dados adquiridos em uma rede real são comparados com requisitos temporais previamente definidos pelo usuário e os resultados são exibidos de forma gráfica. A ferramenta funciona em conjunto com uma placa Syscon da empresa Smar, responsável pela monitoração de eventos (mensagens) no barramento e geração de arquivos de histórico que preservam as informações temporais de seus instantes de ocorrência. A representação gráfica dos resultados inclui gráficos de área que informam a distribuição de eventos no barramento. Histogramas representam a distribuição temporal de eventos em relação aos requisitos especificados, especialmente interessantes para análise de eventos periódicos. O diagrama de Gantt, por fim, ilustra a sequência de eventos detectados em um trecho temporal registrado.

O artigo (HONG, 2000) descreve a avaliação temporal de uma rede PROFIBUS FMS. O programa de interface com o barramento PROFIBUS foi implementado em um PC com um controlador modular chamado *Intelligent Universal Controller* (IUC) utilizando um ambiente de programação tempo real multitarefa. Foram avaliados principalmente os atrasos de mensagens transmitidas no experimento montado. Os resultados extraídos distinguem os atrasos presentes nas diversas camadas do protocolo: PHY (camada física), FDL (camada de enlace de dados), LLI (subcamada de aplicação que permite interface com a camada de enlace) e FMS (subcamada de aplicação que implementa interface com o usuário).

No artigo (TOVAR et. al., 2002) é apresentada uma metodologia para avaliação de aplicações em redes PROFIBUS. Avalia-se principalmente o parâmetro de pior caso de tempos de respostas para mensagens na rede, considerando-se, no modelo, o controle determinístico de mensagens cíclicas de alta e baixa prioridades, que podem ser diferentes para cada estação. O modelo apresentado avalia também o controle do mecanismo de passagem de *token* do protocolo para manutenção dos tempos de permanência e rotação de *token*. A avaliação completa considera dois tipos de redes: mono-mestre e multi-mestre. Uma rede mono-mestre é aquela onde apenas uma estação ativa encontra-se presente. Uma rede multi-mestre mantém várias estações ativas que devem compartilhar o acesso ao meio. O segundo caso apresenta-se mais complexo, pois o comportamento de um dado mestre influi diretamente no tempo de manutenção de *token* dos demais.

É proposto por Alves (2003) o desenvolvimento de uma solução que permita interligar, em uma mesma rede lógica, dispositivos PROFIBUS que utilizam meios físicos RS-485 e link de rádio (*wireless*). Um estudo detalhado sobre os tempos de comunicação é feito, principalmente levando em conta os atrasos de propagação dos dispositivos que empregam comunicação sem fio, de forma a que se possa garantir o determinismo temporal desta rede híbrida.

## 2.4 SOLUÇÕES COMERCIAIS

Os trabalhos mencionados anteriormente apresentam o estado da arte principalmente do ponto de vista acadêmico. Uma vez que a tecnologia de barramentos de campo encontra larga aplicação em sistemas industriais, não é desprezível o número de empresas com soluções comerciais para análise do comportamento de sistemas baseados em barramentos de campo. Assim sendo, nesta seção serão apresentados os mais importantes sistemas disponíveis comercialmente e que relacionam-se com o trabalho desenvolvido.

Dentre os sistemas de monitoração de redes de barramento de campo atualmente disponíveis se destacam as placas de expansão em slots ISA e PCI, com formatos para PC convencionais ou modelos compactos PC-104. Possuem normalmente processamento próprio a fim de isolar detalhes de implementação do computador, provendo comunicação externa através de memórias dupla porta. Empresas especializadas no setor de automação industrial (Hilsher, INS, Beckhoff, National Instruments, Ifak System, Siemens e Softing) disponibilizam versões diferentes de placas para interface suportando principalmente os barramentos CANOpen, ControlNet, DeviceNet, PROFIBUS, Modbus e Interbus-S.

Dentre as ferramentas comerciais atualmente voltadas à análise de barramentos de campo destacam-se três: Dscope, PROFIBUS PA Device Test Tool e CANanalyzer (SIEMENS, 1998; TMG, 2002; VECTOR, 1999).

A empresa Siemens distribui o *software* Dscope para análise de redes PROFIBUS, monitoradas através de placas de comunicação para PC CP5613 ou CP5412, também produzidas pela empresa. A ferramenta permite monitorar e salvar trechos de troca de mensagens no barramento com informações temporais. Recursos extras de gatilho e filtragem de mensagens são previstos.

A ferramenta PROFIBUS PA Device Test Tool, da empresa TMG, tem por finalidade validar o atendimento a especificações e requisitos de um dispositivo PROFIBUS-PA. A ferramenta permite o acesso direto a estes parâmetros pelo uso de um gateway FNL1.

Uma empresa européia chamada Vector desenvolveu uma ferramenta chamada CANanalizer para monitoração, análise e geração de mensagens em redes CAN. As funções da ferramenta incluem listagem de tráfego (*tracing*), exibição de segmentos de dados das mensagens, avaliação estatística de mensagens e estatísticas de carga do barramento. A interface com o usuário permite que se trabalhe em nível de bytes (com visualização de dados conforme coletados diretamente do barramento) ou em nível de aplicação (com representação de mensagens por seus conteúdos).

A diversidade de soluções encontradas para o setor de automação industrial parece seguir a tendência de suporte aos barramento de campo especificados pela norma IEC 61158, os quais são atualmente definidos como padrões internacionais. Os produtos encontrados entretanto representam soluções isoladas, cada qual voltada para uma tecnologia específica. Não existe, até o presente momento, uma solução global que atenda a uma gama de protocolos, tanto do ponto de vista de sistemas de placas de monitoração como de ferramentas de avaliação.

# 3 CENÁRIO DA AUTOMAÇÃO INDUSTRIAL

O cenário atual de automação industrial encontra-se dividido entre soluções utilizando diferentes tecnologias de controladores, atuadores e sensores. A proposta da unificação de diversas tecnologias de barramento de campo tem sido buscada nos últimos anos, resultando na norma IEC61158.

## 3.1 BARRAMENTOS DE CAMPO

Aplicações de automação industrial têm características intrínsecas de operação bastante exigentes, tais como necessidade de interligação de elevado número de dispositivos em uma mesma planta (podendo chegar a milhares de unidades), grande distância geográfica entre pontos de medição/atuação (até alguns quilômetros), ambiente agressivo sujeito a presença de ruído elétrico (muitos ambientes são sujeitos a elevados índices de calor e umidade e, em alguns casos, sujeitos até a riscos de explosão), monitoração constante com possibilidade de reconfiguração sem parada (o desligamento de certos processos pode acarretar perdas de todo um lote de produção) e baixo tempo de resposta (na ordem de alguns milissegundos). Estes fatores somados exigem que sistemas de automação industrial sejam bastante robustos tanto do ponto de vista físico quanto lógico.

Tecnologias tradicionais de automação industrial baseados em sistemas centralizados, apesar de fisicamente robustas, apresentam desvantagens como baixa flexibilidade, tempo de processamento limitado e elevados custos de instalação e manutenção.

Do ponto de vista lógico, as aplicações de automação industrial têm tido uma demanda cada vez maior por operação distribuída com suporte tempo real. Visando atender esta demanda surgiu a tecnologia de barramentos de campo, composta de uma rede de dispositivos inteligentes interligados por um meio físico compartilhado. A tecnologia de barramentos de

campo tem possibilitado uma real distribuição de inteligência pelos elementos instalados na planta industrial, propiciando o desenvolvimento de malhas de controle mais complexas e confiáveis. Para a troca de dados entre diversas estações, protocolos de comunicação padronizados têm sido difundidos.

A tecnologia de barramentos de campo segue o modelo de referência para sistemas de comunicação *Open Systems Interconnection* (OSI) criado em 1979 pela entidade *International Organization for Standardization* (ISO). O modelo OSI divide a especificação de um sistema de comunicação em sete camadas distintas, cada qual responsável pela definição e implementação de serviços afins (TANENBAUM, 1989). As sete camadas do modelo OSI são: camada física, enlace de dados, rede, transporte, sessão, apresentação e aplicação.

Esta divisão em camadas traz como vantagens uma maior facilidade de programação e portabilidade de módulos. Pode-se assim, por exemplo, propiciar que um barramento utilize diferentes meios físicos sem necessitar a reprogramação de todo o sistema de comunicação (apenas o módulo da camada física precisa ser adaptado). A utilização deste modelo de camadas para sistemas de controle de processos industriais foi difundida inicialmente pela Bosh e outras empresas européias como padrão *Manufacturing Automation Protocol* (MAP). Com o tempo, entretanto, percebeu-se que o atraso gerado pelo processamento das diversas camadas do padrão MAP tornava-se muito dispendioso para a comunicação de processos que exigem pequeno tempo de resposta, como é o caso de grande parte das aplicações de automação industrial. Visando resolver este problema foi proposto, no final dos anos 80, um modelo simplificado, chamado MiniMAP, o qual implementa apenas três das sete camadas originais: camada física, enlace de dados e de aplicação. Apesar da redução dos recursos suportados pelo novo modelo, causada pela diminuição do número de camadas, o ganho de performance da rede em termos de velocidade mostrou-o viável para a utilização em processos críticos no tempo (BEUS-DUKIC; WELLINGS, 1991).

## 3.2 NORMA IEC61158

Em 1988 um comitê formado por representantes das entidades *International Electrotechnical Commission* (IEC) e *Instrumentation, Systems and Automation Society* (ISA) iniciaram esforços para a definição de um padrão internacional de barramento de campo para o setor industrial, a especificação IEC61158. Após muitas discussões, acabou-se por definir não um, mas sim um conjunto de oito diferentes barramentos de campo (IEC, 2001).

A norma IEC61158 é dividida em vários módulos, cada qual responsável pela especificação de uma das camadas que compõem os barramentos de campo. Conforme a tendência do padrão MiniMAP, também a IEC61158 especifica apenas as camadas física, enlace de dados e aplicação do modelo OSI/ISO. O primeiro módulo, IEC61158-1, é uma introdução à norma, apresentando uma visão geral e descrevendo as diversas partes que a constituem. Os demais módulos da norma são associados com diferentes camadas do modelo de referência OSI, conforme representado na Figura 2.

O módulo IEC61158-2 define a camada física dos protocolos de comunicação, sendo responsável pela descrição das funções de sinalização do meio, especificação de níveis de tensão e corrente, geração de sincronismo e demais características dependentes do meio físico.

As especificações IEC61158-3 e 4 descrevem a camada de enlace de dados, responsável por funções de montagem de mensagens, controle de acesso ao meio e garantia de consistência dos dados transferidos e recebidos. A especificação IEC61158-3 trata dos serviços da camada de enlace de dados e a especificação IEC61158-4 dos protocolos desta camada.

As especificações IEC61158-5 e 6 descrevem a camada de aplicação que contém as funções de interação de aplicações do usuário com a funcionalidade dos dispositivos de

campo: a especificação IEC61158-5 trata da descrição de serviços e a IEC61158-6 dos protocolos da camada de aplicação.

A especificação IEC61158-7 é responsável por funções globais de serviço e controle que, entre outras finalidades, trata da gerência do funcionamento do sistema como um todo.

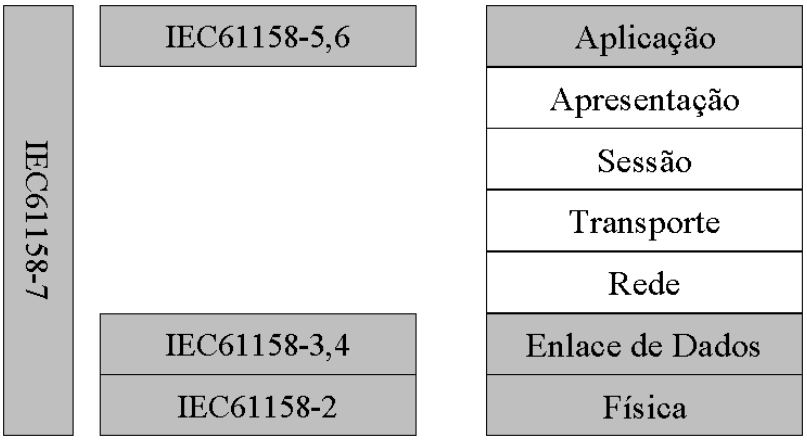



**Figura 2 - Relação entre a norma IEC61158 e o modelo OSI/ISO**

### 3.2.1 Tipos de barramentos de campo presentes na norma IEC61158

Conforme já mencionado, a norma IEC61158 especifica 8 tipos de barramentos de campo distintos, os quais são resumidamente apresentados a seguir:

#### *3.2.1.1 Tipo 1 - Foundation Fieldbus*

A especificação apresentada pelo IEC61158 Tipo 1 trata do barramento Foundation Fieldbus, que visa basicamente o controle de processos contínuos. O meio de comunicação mais comumente utilizado pelo barramento é denominado modo tensão sobre fio com

transmissão síncrona, padrão anteriormente definido como IEC1158-2 (para maior simplicidade do texto esta nomenclatura será adotada no presente trabalho sempre que se referenciar este meio físico).

Neste modo de comunicação, os dados são transmitidos sobre a linha de alimentação em uma taxa de operação de 31,25 Kbps. Este meio físico é indicado para emprego em áreas classificadas (intrinsecamente seguras) e até por isso possui limitações bastante severas quanto aos níveis de sinal na operação do barramento. Utiliza, para representação de mensagens no barramento, a codificação Manchester, que transmite dados, sinais de sincronismo e relógio. Uma comparação entre a codificação Manchester e a NRZ (também bastante difundida para barramentos de campo) é apresentada na Figura 3

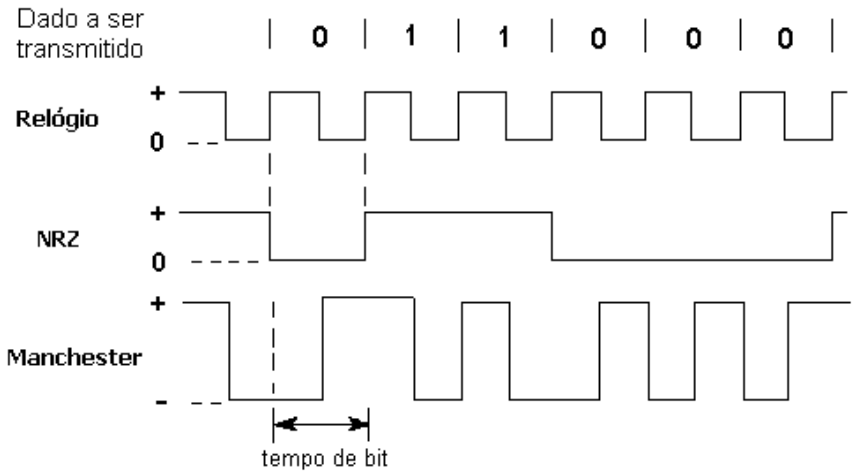



**Figura 3 - Comparação entre as codificações NRZ e Manchester**

Como mecanismo de acesso ao meio utiliza passagem de *token* com controle centralizado em um dispositivo especial chamado *Link Active Scheduler* (LAS). O LAS envia uma mensagem especial de *token* para a estação que estiver habilitada a fazer uso do barramento. Assim que o tempo de permanência do *token* expira ou que a estação retorna o *token*, indicando que não tem mais mensagens a tratar, o LAS deve passar o *token* para a próxima estação em sua lista. O escalonamento da lista de dispositivos habilitados para

utilizar o barramento, bem como dos respectivos tempos de permanência de *token* é administrado dinamicamente pelo LAS. O mecanismo adotado permite a troca de dados cíclica e acíclica. Na troca cíclica de dados, as estações que trocam dados periodicamente são colocadas na lista do LAS, ocupando uma parcela fixa (mas não total) do tempo de circulação do *token*. Durante o tempo restante do ciclo, mensagens acíclicas podem ser agendadas.

Uma característica diferenciada do barramento Foundation Fieldbus é a definição de blocos de funções básicas e/ou avançadas, denominados blocos funcionais, que podem ser alocados em diferentes dispositivos da rede.

#### *3.2.1.2 Tipo 2 - ControlNet*

O barramento ControlNet, desenvolvido para aplicações de controle discreto distribuído, utiliza como meio físico um cabo coaxial com isolação indutiva (normalmente por transformador) ou fibra ótica. Para codificação dos dados, usa também codificação Manchester operando até a 5Mbps.

Como estratégia de acesso ao meio usa o esquema *Concurrent Time Domain Multiple Access* (CTDMA), que se baseia na divisão do tempo em repetidos *Network Update Time* (NUT), dentro dos quais os dados são alocados utilizando-se um algoritmo de rotação de *token* implícita. ControlNet reserva tempo para transmissão de dados com característica tempo real. Parte do tempo de ciclo é usado para transmissão de dados não críticos no tempo, com prioridade de acesso rotativa, o que permite que todos os dispositivos da rede tenham igualitariamente acesso à escrita no barramento (a prioridade de uso rotaciona entre os dispositivos a cada novo ciclo do barramento). Entre cada NUT existe um período em que uma estação chamada moderador transmite uma mensagem de sincronismo

ControlNet implementa o modelo produtor/consumidor. As mensagens ao invés de utilizarem endereços de origem e destino são identificadas individualmente por um identificador *Connection ID* (CID), deixando a encargo das estações receptoras a decisão de filtragem ou não destes dados.

#### *3.2.1.3 Tipo 3 - PROFIBUS*

Barramento de campo utilizado para aplicações de controle de processos contínuos e discretos. Para tanto possui respectivamente duas versões normatizadas na IEC61158, PROFIBUS-DP (*Decentralized Periphery*) e PROFIBUS-PA (*Process Automation*). A versão DP utiliza meio físico RS485, operando em até 12Mbps. A versão PA utiliza meio físico IEC1158-2 operando até 31,25kbps.

A topologia da rede PROFIBUS utiliza um mecanismo híbrido de mestre/escravo e passagem de *token*. As estações mestres podem assumir controle do barramento enviando ou requisitando dados. As estações escravas se limitam a responder requisições feitas pelo mestre. A gerência de acesso ao meio é feita por passagem de *token*, que circula entre os mestres, formando um anel virtual.

#### *3.2.1.4 Tipo 4 - P-NET*

O barramento P-NET também utiliza o conceito hierárquico da topologia mestre/escravo em uma rede RS485 com isolação galvânica, empregando codificação NRZ, com taxas de até 76800bps. Como controle de acesso ao meio utiliza o mecanismo de passagem de *token* virtual (*virtual token passing*), onde um contador de tempo de linha muda

é utilizado para determinar qual mestre deve assumir o barramento. Cada mestre possui um contador de acesso próprio que é incrementado quando o contador de tempo de linha muda atinge os valores 40, 50, 60, etc. Quando o contador de acesso atinge o valor relacionado ao endereço da estação mestre este assume como tendo recebido uma passagem de *token* e pode assumir o barramento.

P-NET possui ainda a opção de montagem de uma topologia multirede permitindo a ligação entre diferentes redes locais através do uso de estações chamadas de mestres multi portas. Para administrar esta topologia multirede, o protocolo P-NET precisou especificar as camadas de rede e transporte do modelo OSI/ISO.

#### *3.2.1.5 Tipo 5 - High Speed Ethernet (HSE)*

A especificação *High Speed Ethernet* (HSE) visa possibilitar o uso do meio físico Ethernet para dispositivos de campo, principalmente voltado a operar com redes que utilizem o barramento Foundation Fieldbus.

A fim de prover a interconexão com redes de computadores atuais, o padrão HSE abrange pelo menos quatro camadas do modelo OSI: camadas física, enlace de dados, rede e transporte. São disponíveis especificações próprias para operar com protocolos como TCP-IP, DHCP, SNMP, entre outros. Dispositivos especiais chamados *Commercial Off The Shelf* (COTS) são definidos para ligação entre redes operando em até 100Mbps.

São permitidos três tipos de sessões de comunicação distintas:

a) cliente/servidor: onde uma porta de servidor espera pedido de clientes;

b) publicador/assinante: que utiliza IP global para envio a grupo;

c) distribuição de informação: utilizando mensagens de IP de difusão global.

#### *3.2.1.6 Tipo 6 - SwiftNet*

Barramento de campo que opera em uma rede RS485 até 5Mbps utilizando codificação Manchester. Utiliza o modelo produtor/consumidor para troca de dados.

Como estratégia de acesso ao meio utiliza o esquema *Time Division Multiple Access* (TDMA), que elimina campos de endereçamento e controle de mensagens, para garantir melhor aproveitamento do canal de comunicação. Neste esquema o tempo de uso do barramento é previamente dividido em slots de tamanhos iguais, que são distribuídos conforme carga de dados entre os dispositivos que operam em modo cíclico.

Uma estação especial do barramento, chamada condutor, é responsável por sincronizar os demais dispositivos através de mensagens de sincronização.

#### *3.2.1.7 Tipo 7 - WorldFIP*

Este barramento utiliza, entre outros, o meio físico IEC1158-2 a 31.25kbps. Utiliza o modelo produtor/consumidor para comunicação entre dispositivos. Um identificador chamado ID (representando um tipo específico de informação a ser publicada) é atribuído a cada transação que possa ocorrer. Um dispositivo, chamado árbitro ou escalonador, administra o uso do barramento. O árbitro coloca mensagens de requisição de informações no barramento. O dispositivo produtor desta informação disponibiliza no barramento os dados requisitados através de uma mensagem de difusão global (*broadcast*). Todos os consumidores desta informação podem então ler e registrá-la. Para garantir determinismo temporal na rede, o protocolo provê duas formas de comunicação:

a) configurada: que trata da circulação de dados cíclicos;

b) sob demanda: opera com informações baseadas em eventos assíncronos.

#### *3.2.1.8 Tipo 8 - Interbus-S*

Utiliza uma topologia de anel para ligação dos dispositivos. Como meio físico utiliza várias ligações RS485 ponto-a-ponto entre dispositivos, possibilitando assim comunicação *full duplex* (dados podem estar concorrentemente sendo enviados pelo dispositivo durante uma recepção).

A forma de tratamento dos dados é bastante simples, utilizando registradores de deslocamento para armazenar mensagens transferidas de um dispositivo para outro. As mensagens circulam em anel entre as estações através de várias conexões ponto-a-ponto. Cada estação tem um campo de dados (*slot*) próprio, reservado na mensagem circulante, que pode ser usado para inserção de dados a serem transmitidos. A estação destino do campo de dados preenchido, após tê-lo recebido corretamente, retira o mesmo da mensagem total, evitando assim a transmissão de dados já processados na rede. Desta forma elimina-se a necessidade do uso de complexos mecanismos de gerência de acesso ao meio que utilizem passagem de mensagens de controle.

O protocolo Interbus-S utiliza também o conceito de hierarquia por mestre e escravo. Cada dispositivo possui um intervalo de tempo físico, determinado pelo tamanho de seu campo de dados.

Dois mecanismos são implementados: transmissão síncrona, usando o modelo publicador/assinante (*publisher/subscriber*), onde os consumidores de uma dada informação precisam se registrar antecipadamente para poder recebê-la, e transmissão assíncrona, que usa o modelo cliente/servidor.

Um resumo dos diversos protocolos que compõem a norma IEC61158 é apresentado na Tabela 1.

**Tabela 1 - Resumo dos tipos de barramentos da norma IEC61158**

| | Descrição | Meio Físico | Velocidade máxima | Acesso ao meio |
|---|---|---|---|---|
| Tipo 1 | Foundation Fieldbus | IEC1158-2 | 31.25 Kbps | Árbitro no barramento |
| Tipo 2 | ControlNet | Coaxial c/ isolação por transformador | 5 Mbps | CTDMA |
| Tipo 3 | PROFIBUS-DP/ PROFIBUS-PA | RS485 / IEC1158-2 | 12Mbps / 31.25kbps | Passagem de *token* |
| Tipo 4 | P-Net | RS485 | 76.8 Kbps | *Token* virtual |
| Tipo 5 | HSE | Ethernet | 100 Mbps | CSMA/CD |
| Tipo 6 | SwiftNet | RS485 | 5 Mbps | TDMA |
| Tipo 7 | WorldFIP | IEC1158-2 | 31.25 Kbps | Árbitro no barramento |
| Tipo 8 | Interbus-S | RS485 | 500 Kbps | Divisão mensagens no tempo (*message slot*) |

# 4 PROPOSTA DO SISTEMA BR-TOOL: BUS REAL-TIME VALIDATION AND MONITORING TOOL

O sistema *Bus Real-Time Validation and Monitoring Tool* (BR-Tool) foi proposto para permitir monitoração e validação temporal de aplicações desenvolvidas utilizando a tecnologia de barramentos de campo. A seguir apresentam-se as decisões de projeto feitas para a implementação deste sistema.

## 4.1 DETALHAMENTO DOS PRINCIPAIS MÓDULOS

A validação de um sistema tempo real consiste da avaliação do seu comportamento temporal, a partir de dados extraídos de sua operação em tempo de execução, os quais são comparados com restrições temporais especificadas. Os dados obtidos do sistema alvo devem portanto conter informações relevantes para a análise à qual serão submetidos. De uma forma geral o levantamento das atividades de um sistema pode ser determinado de acordo com a ocorrência de eventos. Eventos caracterizam mudanças de estado válidas no sistema, podendo representar a alteração de uma variável, a chamada de uma função, um sinal de interrupção externa, a transmissão de uma mensagem, entre outros.

Um sistema de validação pode ser dividido em diversos módulos funcionais. Refinando-se o modelo apresentado no capítulo 2, pode-se chegar ao modelo mais completo apresentado na Figura 4.

Sistemas de captura de eventos podem apresentar basicamente duas estratégias de funcionamento: baseado no tempo ou em eventos (KOPETZ, 1997). Nos sistemas baseados no tempo a amostragem de dados ocorre em instantes de tempos pré-definidos, sendo portanto indicados para sistemas que operam com processos síncronos. Para sistemas baseados em eventos, a amostragem de dados está vinculada à deteção da ocorrência de

eventos no sistema alvo, podendo neste caso ser utilizado em sistemas com processos síncronos ou assíncronos. Para sistemas tempo real distribuídos, compostos tanto por processos síncronos como assíncronos, normalmente se utilizam sistemas de monitoração gatilhados por eventos (TSAI; YANG, 1995; LIU; MOK, 1997).

Um evento detectado por um monitorador baseado em eventos deve ser marcado temporalmente, a fim de que se possa levantar o comportamento temporal do sistema (não é suficiente saber o fluxo de operação do sistema, precisa-se saber quando as transições ocorrem). Esta etapa de marcação temporal é fundamental para garantir a consistência dos dados a serem analisados pelo sistema de validação (MINK et. al., 1998).

A informação temporal, que normalmente caracteriza o instante de ocorrência do evento, deve ser acoplada à informação de tipo de evento detectado, formando assim o que se chama uma estrutura evento-tempo (SHOBAKI; LINDH, 2001). As diversas estruturas evento-tempo são normalmente salvas em um buffer de ocorrência para posterior análise. A tarefa de salvar os dados provenientes do sistema alvo é feita pelo módulo de registro.

Um módulo de filtragem pode ser utilizado para selecionar dentre os eventos salvos no buffer de ocorrências quais são particularmente relevantes e quais podem ser desprezados, reduzindo assim a quantidades de dados a serem processados. A etapa de filtragem em alguns casos pode ser realizada antes do registro, a fim de reduzir também o tamanho do buffer de ocorrência.

Os requisitos temporais de um sistema tempo real normalmente compreendem a definição de intervalos de tempo válidos para ocorrência de eventos (JAHANIAN; RAJKUMAR; RAJU, 1994). Entre os requisitos possíveis de um evento pode-se destacar valores de periodicidade, duração (intervalo entre o início do tratamento de um evento e sua conclusão), tempo de ativação (intervalo entre a chamada e o início do tratamento do pedido) e tempo de resposta (intervalo entre a chamada e o retorno após a execução do evento).

As informações lidas pelo módulo de validação devem ser capazes de fornecer dados que possibilitem o acompanhamento da performance temporal do sistema tempo real. O módulo de validação é responsável pela avaliação se os eventos detectados estão dentro das especificações de requisitos temporais previamente definidas (LIU; MOK, 1997).

O módulo de visualização é responsável pela exibição dos resultados processados pela ferramenta, fornecendo ao usuário informações relevantes que permitam identificar possíveis erros de implementação ou configuração. Os modos de visualização podem consistir de dados estatísticos, histogramas de eventos, gráficos de varredura (*tracing*) ou diagramas de Gantt.

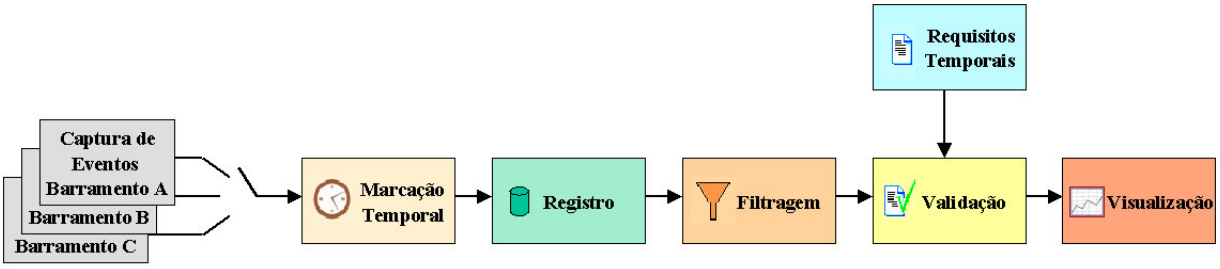



**Figura 4 - Refinamento do diagrama em blocos de um sistema de validação**

## 4.2 ESPECIFICIDADES DE UM SISTEMA DE VALIDAÇÃO TEMPORAL PARA BARRRAMENTOS DE CAMPO

Conforme já comentado, eventos representam mudanças de estado do sistema. Particularmente para a tecnologia de barramento de campo, composta por uma rede de dispositivos que se interligam por um barramento compartilhado, eventos importantes para o sistema podem ser caracterizados pelas mensagens circulantes, as quais efetivamente representam pedidos de ativação de recursos, envio de dados para atuação, divulgação de valores de váriáveis ou informação de mudança de estados na rede, como a entrada ou saída de uma dada estação.

Segundo a norma IEC61158, uma rede de barramento de campo é constituída por um barramento de dados digital, serial, de múltiplo acesso para comunicação com dispositivos de instrumentação e controle digital. O comportamento serial das redes de barramentos de campo traz uma vantagem significativa em relação a outros sistemas tempo real distribuídos, pois neste caso um único sistema de monitoração é necessário para capturar todas as mensagens circulantes no meio de comunicação compartilhado. Desta forma o sistema de validação apresentado neste trabalho traz as vantagens de ter uma implementação simples (apenas um elemento monitorador para todo o sistema), não invasiva (tratando-se de uma monitoração por *hardware*) e flexível (dispositivos podem ser inseridos e removidos do barramento sem necessidade de reconfiguração do sistema de monitoração).

O elemento de monitoração do sistema de validação será tratado com atenção, pois caracteriza uma das maiores contribuições deste trabalho.

A pesquisa por soluções de placas de interface no mercado apontam a ausência de um equipamento genérico que permitisse suporte a diferentes barramentos. As soluções normalmente encontradas no mercado caracterizam-se pela utilização de placas de interface específicas para cada barramento. Mesmo empresas que produzem soluções para diversos protocolos industriais apresentam placas distintas, cada qual projetada para um determinado barramento.

Distingue-se, entre as soluções atualmente disponíveis no mercado, módulos embarcados para comunicação com diferentes protocolos industriais, como os produzidos pela empresa Anybus (HMS, 2002). Estes módulos apresentam uma interface externa genérica implementada por uma memória dupla porta. Por apresentar a mesma interface para as versões de protocolo é indicada para dispositivos que necessitem prover interfaces para diferentes protocolos sem o custo de reprojeto de esquemático de *hardware*. Possuem interfaces para PROFIBUS, DeviceNet, LonWorks, Modbus, Ethernet, Interbus-S,

CANOpen, ControlNet e ASi. Esta solução entretanto não pode ser adotada no presente projeto, pois não permite a monitoração transparente de mensagens no barramento. Os módulos filtram mensagens no barramento, aceitando apenas mensagens endereçadas aos mesmos. Além disso fornecem aos usuários apenas os campos de dados das mensagens, retirando os cabeçalhos e demais campos de controle, os quais são importantes para identificação de tipo de evento.

Uma abordagem alternativa de utilizar diversas placas de interface, cada qual para um dado protocolo, também se mostra inviável, pois grande parte das placas comerciais não permitem a monitoração de atividade no barramento, isolando mensagens não diretamente relacionadas com a placa, disponibilizando apenas os campos de dados das mesmas mensagens. Algumas placas que permitem monitoração de mensagens completas no barramento, não fornecem entretanto documentações ou informações adequadas que permitam o desenvolvimento de *drivers* para leitura e processamento de mensagens.

Devido às dificuldades apresentadas, a solução adotada para o projeto foi a construção de uma placa de aquisição que suporte diferentes barramentos, fornecendo informações relevantes para a avaliação de comportamento temporal do barramento.

O trabalho apresentado segue a linha de pesquisa adotada por Wild (2000), que desenvolveu uma ferramenta para análise de comportamento temporal de redes Foundation Fieldbus. A estrutura genérica da ferramenta torna-a adequada para utilização como base para o presente trabalho. Para tanto algumas alterações sobre a ferramenta tiveram de ser processadas visando a operação com diferentes protocolos industriais, mas sobretudo procurando-se manter a estrutura original. Esta estratégia de uso de uma ferramenta para PC que realize as tarefas de validação, especificação de requisitos e visualização, deixa a cargo de um dispositivo externo a implementação das demais tarefas (captura de eventos, marcação temporal e registro), que são apresentadas a seguir.

### 4.2.1 Captura de eventos

O suporte a diferentes formatos de codificação de dados no barramento como Manchester e NRZ dificultam o uso de lógicas digitais discretas para este fim devido a sua complexidade. A utilização de um processador na placa de aquisição para realizar este tipo de codificação representa uma solução mais viável, limitada entretanto às velocidades envolvidas. Mesmo para codificação NRZ que já se apresenta disponível para muitos microcontroladores, não se encontram componentes de baixo custo capazes de suportar diretamente velocidades da ordem de 12Mbps. Assim sendo as interfaces seriais, tanto para codificação NRZ como Manchester, teriam que ser implementadas em pinos de entrada e saída do microprocessador levando a um custo razoável de carga de processamento, somente para gerar/tratar os sinais do barramento em taxas de comunicação mais elevadas.

A utilização de circuitos integrados dedicados, ASICs, que realizam a interpretação de mensagens nos barramentos, é uma solução que desonera a carga do microprocessador da placa. O microprocessador se encarrega, nesta abordagem, apenas de consultar os ASICs para fazer a leitura dos dados capturados. O aumento dos custos desta implementação e a dificuldade de se obter componentes que permitam a cópia transparente de dados do barramento apresentam-se como desvantagens desta solução.

Optou-se assim pela utilização de lógicas programáveis, gravadas com módulos de interface capazes de realizar a interpretação e aquisição de dados do barramento. A arquitetura interna das lógicas programáveis, compostas por diversos blocos operacionais capazes de desempenhar funções lógicas básicas traz algumas vantagens sobre arquiteturas convencionais. A flexibilidade de interligação lógica entre estes blocos permite o desenvolvimento de uma gama enorme de sistemas digitais, capazes de desempenhar funções anteriormente só implementadas por circuitos microprocessados. Devido ao fato destas

implementações estarem sendo feitas em *hardware* pode-se atingir taxas de processamento bastante elevadas. Além disso a estrutura interna destes dispositivos permite o desenvolvimento de inúmeros sub-sistemas processando em paralelo.

### 4.2.2 Marcação temporal

A tarefa de marcação temporal de eventos no barramento pode ser implementada pelo uso de um microcontrolador da placa, que possua contadores internos de alta resolução (32 bits). A fim de reduzir os atrasos na aquisição, a interface entre a lógica programável e o microcontrolador seria feita através de interrupção. Durante o atendimento à interrupção solicitada pela lógica programável, o valor corrente do temporizador interno do microcontrolador é gravado e anexado ao valor de evento detectado pela FPGA para a geração da informação evento-tempo. Uma desvantagem desta implementação é que o atraso referente ao atendimento da interrupção, apesar de pequeno, não é previsível, pois depende do fluxo corrente do programa no microcontrolador, quando da ocorrência de um novo pedido de interrupção. Este intervalo somado ao tempo gasto com as etapas de leitura da FPGA, montagem da informação evento-tempo e posterior salvamento em um histórico de eventos pode acabar gerando um atraso muito grande para o sistema de aquisição.

Em muitos casos onde o *software* desenvolvido para um sistema embarcado se torna crítico para a performance temporal desejada, pode-se recorrer ao uso de implementações em *hardware* (THOMAS et. al., 2000). Desta forma uma segunda abordagem seria transferir estas tarefas de marcação temporal e salvamento de informações em histórico também para dentro da lógica programável. Devido à característica inerente de execução de operações em paralelo destes componentes os tempos de atendimento à leitura de eventos detectados e busca da informação de temporizadores internos se reduzem praticamente aos atrasos de

propagação das células lógicas, os quais são bastante pequenos (da ordem de alguns nanosegundos). A única desvantagem é o aumento do custo de processamento dentro da lógica programável, mas este não chega a ser um grave problema atualmente devido ao alto grau de integração dos componentes atuais.

### 4.2.3 Registro

O módulo de registro é responsável pela manutenção dos dados adquiridos no barramento de campo, juntamente com a informação temporal de momento de ocorrência. Este é normalmente implementado através de um buffer de memória do tipo FIFO circular (TSAI; FANG; CHEN, 1995), permitindo aos demais módulos recuperar e analisar o histórico da rede monitorada. O tempo de acesso e tamanho do buffer de registro são parâmetros que devem ser considerados principalmente quando se pretende realizar amostragem e processamento em tempo de execução.

A implementação de buffers de registro dentro da mesma lógica programável que realiza as demais etapas de baixo nível do sistema (captura de eventos e marcação temporal) traz como vantagem a redução de atrasos de escrita e leitura de dados além de contribuir para uma maior compactação do *hardware* do sistema. Esta abordagem traz entretanto a desvantagem de se tornar onerosa caso quando se pretende trabalhar com buffers de registro grandes. Uma solução mais economicamente viável é a utilização de memórias externas de acesso rápido para implementar os buffers de registro do sistema.

### 4.2.4 Filtragem

Outra abordagem considerada para reduzir a quantidade de dados gerados pela monitoração é a realização de filtragens de informações não relevantes ao ensaio pretendido. Dependendo do nível de abstração pretendido, diversas formas de filtragens podem ser implementadas. Para redes de barramento de campo especificamente, onde os eventos são caracterizados pelas mensagens circulantes, a filtragem de eventos deve ser implementada por conferência dos campos de controle das mensagens. Pode-se, por exemplo, filtrar todas as mensagens de controle enviadas para um determinado nó verificando o campo de endereço de destino. A identificação de mensagens de alarme podem ser feita por monitoração do tipo de serviço utilizado ou prioridade associada.

De fato, as tarefas desempenhadas por este módulo são fortemente dependentes do protocolo que estiver sendo analisado, pois cada protocolo define um formato distinto de campo de controle. Além disso as diferentes estratégias de acesso de cada protocolo provêm diferentes formas de tratamento das mensagens. Para barramentos que utilizam um modelo produtor/consumidor, por exemplo, o endereço de destino da mensagem não existe necessariamente.

Pode-se imaginar a implementação deste recurso de filtragem em pontos distintos do projeto, desde a placa de aquisição até a ferramenta de validação desenvolvida no PC.

De uma forma geral a disponibilidade deste recurso na placa torna vantajosa a redução da quantidade de dados armazenados e transferidos, uma vez que parte dos dados são descartados durante a monitoração, além de reduzir a carga de processamento no PC. O maior problema desta solução é a quantidade de memória necessária para a implementação deste recurso. Para permitir funcionamento em tempo de execução são necessárias memórias rápidas que evitem a geração de atrasos na leitura dos barramentos de campo. Torna-se

recomendável a utilização de memórias adicionais, onde os dados sejam antecipadamente salvos, para fins de análise pelo módulo de filtragem. Os dados considerados válidos seriam enviados para a memória de saída e os demais seriam descartados. Estes recursos de gerência de memória adicional acrescido à programação de módulos de interpretação de mensagens diversificados, os quais são dependentes das camadas de enlace de cada protocolo (um módulo de identificação e tratamento de mensagens específico para cada protocolo deve ser desenvolvido), geram um aumento significativo no processamento a ser implementado dentro da FPGA.

A segunda solução (análise e filtragem realizada no PC) traz a vantagem de apresentar uma maior flexibilidade no tratamento de dados, possibilitando ao usuário no PC trabalhar com a forma de identificação de eventos, observando diferentes resultados para um mesmo conjunto de dados, o que não seria possível caso os dados fossem filtrados na aquisição. Os custos de processamento no PC não são elevados considerando-se a alta performance do microprocessadores presentes nos computadores atuais. O aumento da carga de dados transferida se torna portanto a preocupação mais relevante, porém com as altas velocidades de barramentos locais (centenas de megabytes por segundo) esta solução se mostra ainda viável.

### 4.2.5 Especificação de restrições temporais

As especificações temporais de um sistema temporal deve referenciar eventos com instantes válidos de ocorrência. Particularmente para o caso de protocolos de comunicação, onde eventos são caracterizados por mensagens, torna-se bastante prático a especificação de restrições temporais a partir da relação entre eventos (por exemplo entre uma mensagem de pedido de dados e a sua efetiva resposta). Para estas restrições se seguirá a notação sugerida por (PEREIRA, 1995), que permite a definição de relações entre eventos e intervalos de validade a partir de um formato baseado na notação lógica tempo real (RTL) apresentada por (JAHANIAN, 1986).

A notação adotada utiliza uma série de predicatos, conforme listados na Tabela 2, que permitem a definição das relações mais comuns entre eventos. Após a relação segue o intervalo de validade previsto para a dada relação, com valor relativo ao instante de ocorrência do evento anterior.

**Tabela 2 - Notação adotada para especificação de requisitos temporais**

| **Predicado** | **Notação** | **Exemplo** |
|---|---|---|
| AFTER | (evento1 AFTER evento2) $[t_{min}, t_{max}]$ | (ev1 AFTER ev3) [1000, 3000] |
| BEFORE | (evento1 BEFORE evento2) $[t_{min}, t_{max}]$ | (ev2 BEFORE ev0) [500, 2000] |
| CYCLIC | (CYCLIC evento1, $t_{max}$ ) | (CYCLIC ev2, 3000) |

O predicato AFTER indica que o primeiro evento, "evento1", deve ocorrer depois do segundo, "evento2", dentro dos instantes definidos pelo intervalo de tempo ($t_{min}$ representa o tempo mínimo e $t_{max}$ o tempo máximo de ocorrência). Analogamente o predicato BEFORE representa que o "evento1" deve ocorrer antes do "evento2", dentro do intervalo especificado. O último predicato CYCLIC indica que o evento definido, "evento1" deve

ocorre ciclicamente no intervalo que segue (maior período de ocorrência dado por $t_{max}$). A relação de ciclicidade pode ainda ser definida pelos predicados AFTER e BEFORE quando os parâmetros "evento1" e "evento2" são preenchidos com o mesmo evento acrescido da informação relativa de instante de ocorrência deste no histórico de eventos. Por exemplo, a restrição "(ev1@-1 AFTER ev1@-2) [1000, 2000]" indica que o o último evento "ev1" detectado deveria ter ocorrido após o penúltimo evento de mesma natureza no intervalo de 1000 a 2000 unidades de tempo.

### 4.2.6 Validação temporal

Conforme já comentado, os módulos de validação temporal e visualização utilizados neste trabalho baseiam-se na ferramenta VCAT (WILD, 2000).

O mecanismo de validação adotado inicialmente avalia os predicatos definidos pelo usuário no módulo de especificação de restrições temporais, marcando-o como verdadeiro ou falso dependendo da última relação detectada. Em seguida os intervalos de validade, representados por intervalos relativos são transformados em intervalos absolutos, a partir do instante de ocorrência do evento anterior ("evento2" para o predicado AFTER ou "evento1" para os predicatos BEFORE e CYCLIC).

O intervalo calculado representa os instantes de ocorrência máximo e mínimo para o último evento. Se o instante detectado para o mesmo fica dentro do intervalo de validade calculado pode-se dizer que o evento atende aos seus requisitos temporais, caso contrário não atende. Os resultados obtidos por esta avaliação são salvos como uma série de estruturas evento-requisito que são passadas ao módulo de visualização.

### 4.2.7 Visualização

Diferentes formas de visualização dos resultados estão disponíveis pela ferramenta: diagramas tipo área (*chart*), histogramas e diagramas tipo Gantt. O diagrama tipo área apresenta de maneira gráfica a participação relativa dos diversos eventos na comunicação, seja em termos do número de mensagens transmitidas ou em termos da ocupação do barramento. Já o histograma de evento e requisito, expõe a ocorrência de eventos de um determinado tipo, permitindo identificar a distribuição destes no tempo, durante o período de monitoração. O histograma de eventos pode ser confrontado visualmente com a restrição temporal especificada.

O diagrama de Gantt é uma representação temporal tradicional e bastante expressiva para compreensão de ordenação de eventos, pela obtenção visual da sequência de eventos detectada. Os intervalos de validade são também apresentados neste diagrama, acompanhado cada evento representado juntamente com a informação de atendimento ou não dos requisitos temporais.

### 4.2.8 Mapeamento de funções no sistema proposto

Com base nas considerações feitas neste capítulo chega-se então ao modelo proposto na Figura 5, onde se pode distinguir os dois elementos constituintes do sistema: placa de aquisição (chamada BUSMON) e *software* de validação no PC.

A placa BUSMON é responsável por desempenhar as tarefas de captura de eventos, marcação temporal e registro. O *software* de validação no PC, por sua vez, implementa as funcionalidades de filtragem, especificação de requisitos, validação temporal e visualização de resultados.

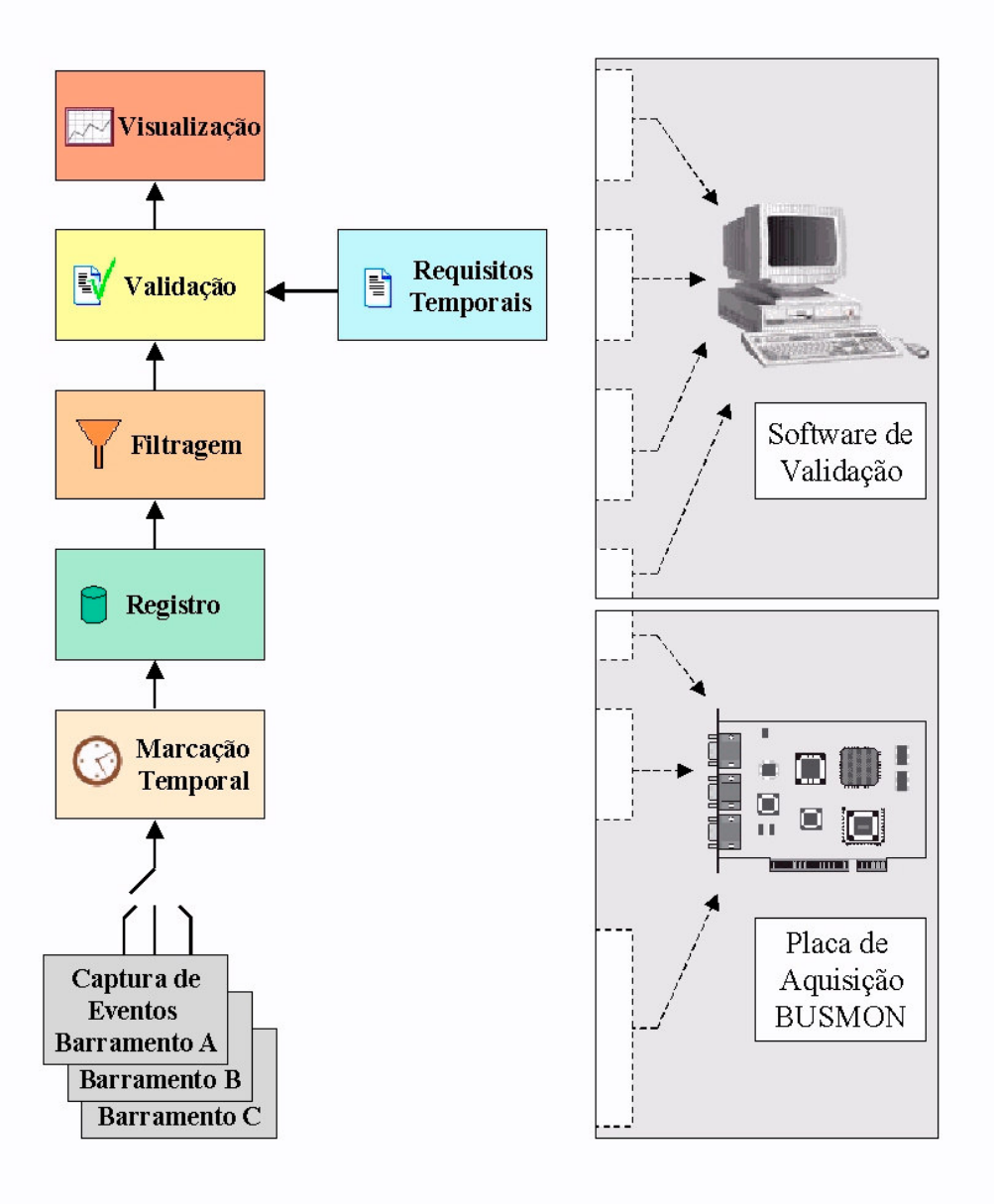



**Figura 5 - Diagrama de blocos do sistema de validação desenvolvido**

### 4.2.9 Interface de comunicação com PC

Em função da decisão de projeto de dividir-se o sistema BR-Tool em dois sub-sistemas, uma placa de aquisição de mensagens e um *software* de validação rodando em um computador, a transferência de informação entre os dois módulos torna-se de grande relevância.

Para a escolha da interface de comunicação a ser implementada na placa de aquisição levou-se em consideração a taxa de comunicação mais alta a ser suportada. Pode-se observar por análise da Tabela 1 que esta corresponde à velocidade de 12 Mbps, prevista para o barramento PROFIBUS-DP. Como cada byte no barramento PROFIBUS é codificado com 11 bits (um bit de início, um bit de paridade, um bit de parada e oito bits de dados) tem-se uma taxa de dados máxima efetiva de 1,1 MB/s. Considerando que a informação temporal contida em cada estrutura evento-tempo seja dada por um temporizador de 32 bits, chega-se à especificação de taxa máxima efetiva de 5,1 MB/s, supondo funcionamento contínuo de aquisição e transferência de dados em tempo real. Foram abordados as principais interfaces com computador na especificação do projeto:

a) interface paralela EPP: inicialmente projetada para trabalhar com impressoras, teve sua funcionalidade extendida através das especificações ECP e EPP (MENDONÇA; ZELENOVSKY, 2002), visando aumentar a taxa de transferência de dados. Mesmo na mais alta taxa suportada com recurso de acesso direto a memória (recurso que permite a mais rápida forma de transferência de dados entre memórias no computador, uma vez que remove a participação do microprocessador desta tarefa), a velocidade não atinge o valor de 2 MB/s, tornando-se portanto inviável para atender à demanda exigida.

b) Barramento *Industrial Standard Architecture* (ISA) é uma das interface em barramento paralelo mais antigas disponíveis pelo computador padrão IBM-PC. Em sua última versão conta-se um barramento de dados de 16 bits, com relógio de barramento de 8,33MHz (MENDONÇA; ZELENOVSKY, 2002). Os acessos externos não podem entretanto ser realizados em um único ciclo, sendo necessários ao menos dois ciclos de

relógio a cada acesso. Assim sendo a máxima taxa teórica prevista para este protocolo é de 8,33 MB/s. Na prática entretanto esta taxa não pode ser atingida devido ao compartilhamento deste barramento com diversos periféricos da placa mãe, com prioridades de atendimento iguais ou superiores às disponíveis no barramento. Mesmo que o recurso de acesso direto a memória, DMA, fosse utilizado, a taxa máxima permitida seria de 3,33 MB/s (o atraso relacionado ao gerenciamento dos sinais de controle de DMA é de pelo menos cinco ciclos de relógio por acesso), também inferior ao exigido pelo projeto.

c) barramento serial *Universal Serial Bus* (USB) foi projetado para possibilitar a interligação de periféricos ao computador sem a necessidade de abrir a máquina para conexão física de uma placa (USB, 1998). Seu funcionamento propicia recursos modernos de *plug and play*, com detecção automática de dispositivos através de um protocolo que utiliza uma hierarquia do tipo mestre e escravo (denominados no barramento USB como *host* e *function* respectivamente). A especificação 1.0 do barramento USB opera a uma taxa de 12 Mbps com uma velocidade máxima teórica de 1,1 MB/s, que é inferior ao valor requerido. A especificação 2.0 extende o barramento à velocidade de 480Mbps, ou 43,5MB/s, podendo assim atender às especificações do projeto (USB, 2000). Entretanto a dificuldade de se encontrar, durante o período de definição da arquitetura do projeto, componentes comerciais ou placas-mães que suportassem esta interface impediram a utilização desta estratégia.

d) barramento *Peripheral Component Interconnect* (PCI) é um barramento de alta velocidade definido para interface de periféricos que exigem grandes

> taxas de transferências de dados (PCI, 2002). Possui três categorias de dispositivos: inicializador, dispositivo que inicia uma transação, alvo, que responde ao inicializador enviando ou recebendo dados, e ponte PCI-PCI, que permite a troca de informação entre dois barramentos PCI. O barramento PCI opera sempre em modo rajada, onde uma única fase de endereçamento é seguida por diversas fases de leitura ou escrita. O padrão PCI permite duas larguras de barramentos de dados (32 ou 64 bits) e duas velocidades de operação (33 ou 66MHz). Para um barramento de 32 bits a 33MHz a taxa de transferência de dados chega a 132MB/s. Para um barramento de 32 bits a 66MHz chega-se a 264MB/s e para um barramento de 64 bits a 66MHz atinge-se até 528MB/s. Devido ao atendimento da demanda do projeto e à grande difusão deste barramento no mercado de computadores, esta foi portanto a solução de interface com PC adotada.

A implementação da interface PCI pode basicamente ser implementada com a utilização de um chipset PCI externo ou pela implementação de seu código em uma lógica programável (SHANLEY; ANDERSON, 1999).

A implementação do código em uma lógica programável permite maior flexibilidade do projeto (juntamente com a interface de comunicação PCI outras tarefas próprias da placa poderiam ser realizadas dentro da mesma lógica programável). Módulos de programação PCI comerciais, também chamados *cores* PCI, são entretanto extremamente onerosos, pois visam um mercado de produção em escala  O custo da aquisição em um *core* PCI para lotes de fabricação de milhares de placas por ano facilmente se dilui, tornando-se porém inviável para um projeto que visa a construção de poucas unidades. A utilização de *cores* PCI gratuítos, disponíveis por universidades e centros de pesquisas, são soluções que apresentam-se ainda incompletas para utilização (KUUSILINNA; HAMALAINEN; SAARINEN, 1999).

A utilização de chipsets PCI comerciais se apresenta como uma solução economicamente viável. Além disso normalmente estas soluções possuem um suporte adequado para o desenvolvimento de *drivers* em PC, o que reduz consideravelmente o tempo de desenvolvimento do projeto.

## 4.3 VISÃO GERAL DO SISTEMA PROPOSTO

O sistema proposto, que visa a monitoração de aplicações baseadas em barramentos industriais, utiliza uma placa de aquisição, BUSMON, com suporte para aplicações tempo-real, a qual é instalada no barramento local PCI de um computador hospedeiro. No computador roda uma ferramenta computacional responsável pela leitura e interpretação de dados provenientes da placa. Por utilizar uma placa especialmente projetada para a finalidade de leitura e um computador externo, este sistema representa um monitorador não invasivo, pois não interfere na operação do sistema sob teste, ao mesmo tempo em que permite a flexibilidade de operar em diferentes barramentos.

Na Figura 6 é apresentada uma representação do sistema de monitoração proposto neste trabalho, identificando seus principais elementos componentes: placa de aquisição e *software* de visualização/análise.

A placa de aquisição possui processamento autônomo adquirindo continuamente dados do barramento industrial em que foi interligada e armazenando-os para posterior leitura pelo computador via barramento PCI. Conforme já mencionado, a placa de aquisição é responsável pelas etapas de coleta, armazenamento e transferência de dados adquiridos do barramento de campo escolhido, correspondendo ao computador as tarefas de processamento e análise das informações coletadas, de acordo com os requisitos especificados para a aplicação.

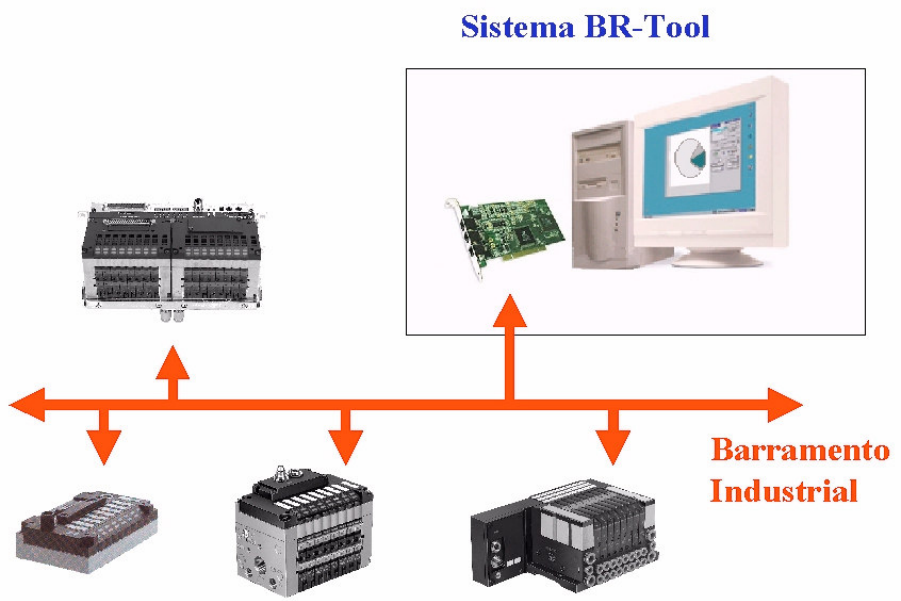



**Figura 6 - Representação do sistema BR-Tool**

Para garantir a adequada análise dos dados, a placa de aquisição salva cada evento detectado no barramento (pode ser um byte ou sequências de bits, dependendo da configuração) juntamente com uma informação de tempo de ocorrência. A informação salva é de tempo relativo, ou seja é o intervalo de tempo da ocorrência do evento corrente em relação ao último evento detectado.

Um programa no PC periodicamente busca e trata os dados adquiridos pela placa, de forma a identificar eventos relevantes. A ferramenta computacional valida os eventos identificados e, baseando-se nas especificações temporais definidas para a aplicação, determina o atendimento ou não de seus requisitos temporais. Uma visualização gráfica dos resultados obtidos pela ferramenta permite fornecer ao usuário informações adicionais sobre o comportamento temporal do sistema.

# 5 PROJETO E IMPLEMENTAÇÃO DO HARDWARE DA PLACA DE AQUISIÇÃO (BUSMON)

A placa de aquisição desenvolvida visa constituir um sistema de aquisição de dados genérico a ser empregado em diferentes redes de barramento de campo, com capacidade de fornecer em tempo real informações de eventos detectados no barramento.

A principal preocupação quando do desenvolvimento da placa de aquisição foi relativa à velocidade de operação. Os componentes escolhidos visam o atendimento às maiores velocidades possíveis dentre os diversos protocolos suportados, permitindo a realização das tarefas de identificação de eventos, marcação temporal e transferência de dados em tempo de execução.

Um diagrama de blocos da placa de aquisição, representando seus componentes principais encontra-se na Figura 7.

O módulo de captura de eventos é implementado na placa pela lógica programável (FPGA) e por diferentes interfaces elétricas. O módulo de marcação temporal é também implementado dentro da FPGA. O registro de eventos é salvo em uma memória compartilhada entre a lógica prgramável e o chipset PCI, possibilitando a leitura destes dados pelo PC.

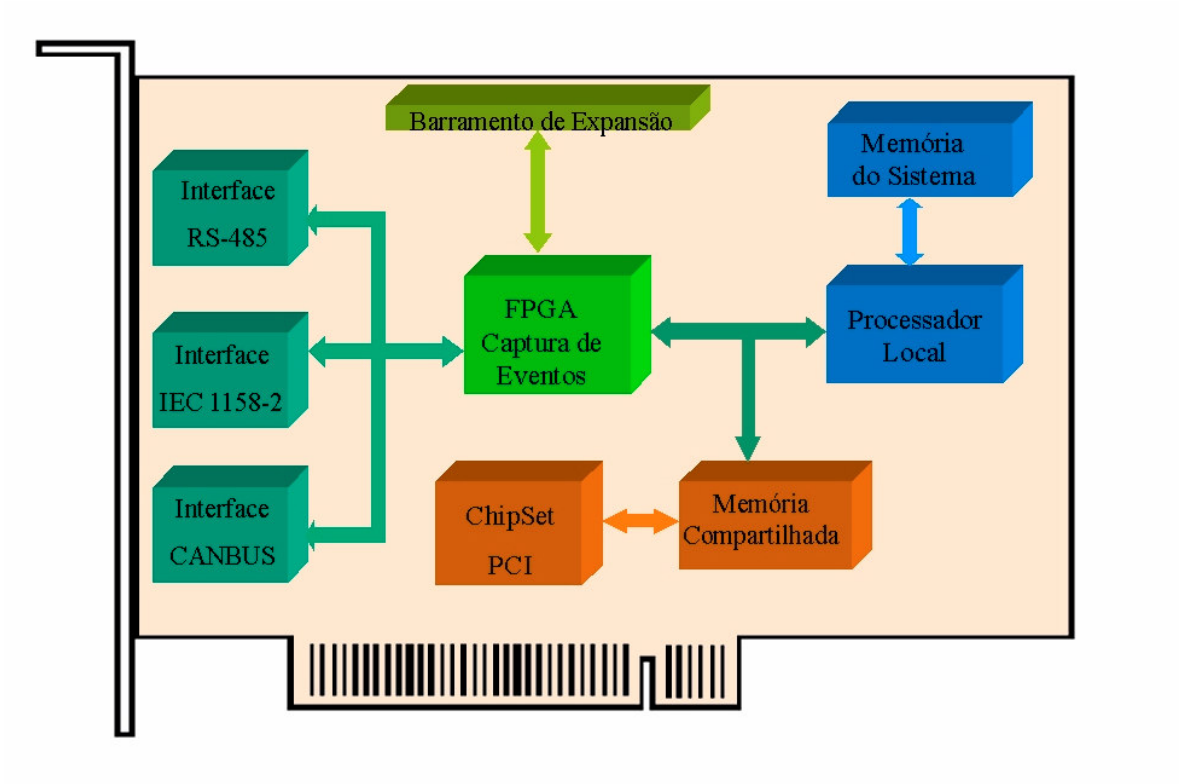



**Figura 7 - Diagrama de blocos da placa BUSMON**

## 5.1 ESCOLHA DOS COMPONENTES DE HARDWARE

Alguns fatores importantes para a definição do projeto foram as escolhas dos componentes eletrônicos, diretamente relacionados com as estratégias de leitura, armazenamento e transferência de dados.

### 5.1.1 Lógica programável

O componente principal da placa de aquisição é a lógica programável FPGA, que mantém as funções de interpretação de sinais dos barramentos, identificação de eventos válidos, marcação temporal e salvamento de informações relevantes. A estratégia de utilizar

uma lógica programável para estas tarefas visa a redução de atrasos no processo de aquisição. A estrutura interna de uma lógica programável permite o desenvolvimento diversas funções lógicas em paralelo, apresentando uma grande vantagem de ganho de performance quando comparado com uma arquitetura tradicional baseada em um microprocessador de propósito geral, que possui funcionamento eminentemente sequencial. Outra vantagem da utilização de uma lógica programável é a de que esta permite o mapeamento de sinais externos de entrada e saída em diferentes pinos do componente, facilitando bastante o layout da placa. A lógica programável selecionada é um XCS30 da família Spartan, que possui cerca de 30000 portas, com a possibilidade de múltiplas gravações através de uma interface de programação serial (XILINX, 2000).

### 5.1.2 Interface com computador

A interface de comunicação com o computador foi implementada através do barramento PCI, devido às velocidades envolvidas (ver seção 4.2.9). Na placa de aquisição projetada a interface PCI é implementada por um chipset comercial (PLX9052) que possui a máquina de estados de um dispositivo alvo PCI 32 bits a 33 MHz.

A interface entre o chipset PCI e a lógica programável é feita por memórias dupla porta, o que permite que a lógica programável atualize os dados adquiridos na memória, mesmo enquanto o chipset PCI estiver fazendo leituras na mesma. A memória dupla porta selecionada para uso é a 707288, com capacidade de 64k x 16bits com tempo de acesso de 20ns.

### 5.1.3 Buffer de registro

A memória utilizada é administrada sob a forma de um buffer circular de dados. Apesar da utilização de uma memória dupla porta evitar os conflitos nos acessos (existem sinais de controle independentes para dois barramentos de dados) é necessário se tomar alguns cuidados para garantir que dados sejam lidos e interpretados corretamente. Para garantir a consistência das leituras de dados salvos na memória foi implementado um mecanismo de sinalização de validade de endereços bastante simples. Neste mecanismo, o primeiro endereço da memória contém o último endereço do buffer circular que contém dados atualizados. Cada vez que o FPGA salva novos dados na memória dupla-porta, atualizando novos endereços da memória, o valor do último endereço acessado deve também ser alterado. O valor do último endereço acessado do buffer circular é informado somente após os dados terem sido escritos na memória. Assim tem-se a garantia que os dados lidos somente serão interpretados após terem sido de fato atualizados.

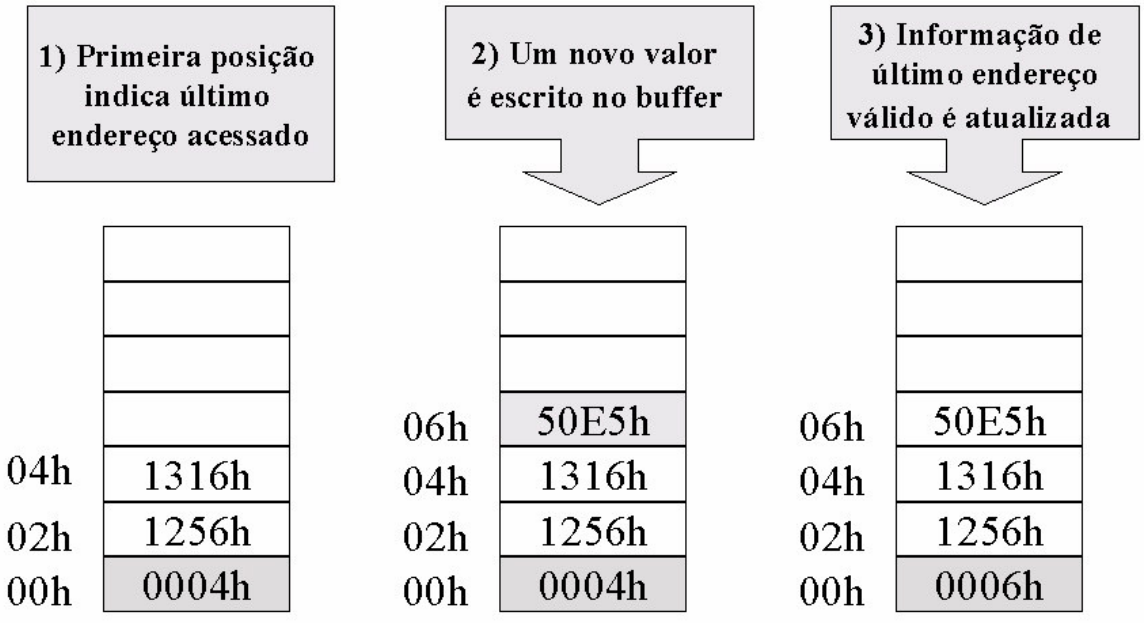



**Figura 8 - Mecanismo de sinalização de escrita no buffer**

Os dados escritos na memória dupla porta devem ser lidos pelo PC em tempos pré-determinados. Se o PC demorar tempo demais para ler estes valores e a lógica programável continuar atualizando esta memória pode ocorre uma sobrescrita de dados válidos (estouro do buffer). A verificação temporal feita para esta ferramenta (demonstração no capítulo 7) garante que os dados serão lidos do PC sem atrasos que possam levar ao estouro do buffer circular. De qualquer forma para garantir que dados adquiridos não sejam perdidos, existe uma proteção que leva à geração de um pedido de interrupção, através do barramento PCI do computador, se mais de 32000 palavras de 16 bits forem escritas no buffer sem leituras pelo *software* de validação.

### 5.1.4 Microcontrolador e memórias

Um microcontrolador é utilizado na placa, principalmente para fins de manutenção de dados de configuração úteis para a placa (como por exemplo tipo de protocolo utilizado e velocidade de operação da rede) e para programação do código da FPGA. O microprocessador é também conectado à memória dupla porta para receber dados do computador através da interface PCI. Normalmente o microcontrolador não atua no tratamento dos dados na memória dupla porta, mas passa a fazê-lo a partir da ativação de pinos de controle disponíveis pelo chipset PCI (o valor destes pinos podem ser alterados através do barramento PCI pela ferramenta que roda no PC). Quando detecta a sinalização destes pinos de controle o microcontrolador desabilita o funcionamento da FPGA para tratar diretamente os dados provenientes do computador, evitando assim que a FPGA interfira nos dados da memória dupla porta. Com estes recursos é possível a reprogramação das funcionalidades da placa durante seu uso sem a necessidade de desconectá-la do computador. Assim a mesma placa passa a ser utilizada para diferentes protocolos industriais. A

configuração da placa para troca de protocolo pode ser feita carregando-se uma nova programação pelo barramento PCI e o microprocessador da placa se encarrega de reprogramar a FPGA. Devido ao fato de os arquivos de programação serem maiores que o buffer da placa, a transferência de dados deve ser feita em diversas etapas sucessivas. Uma representação do método de programação da placa através do barramento PCI é apresentado na Figura 9.

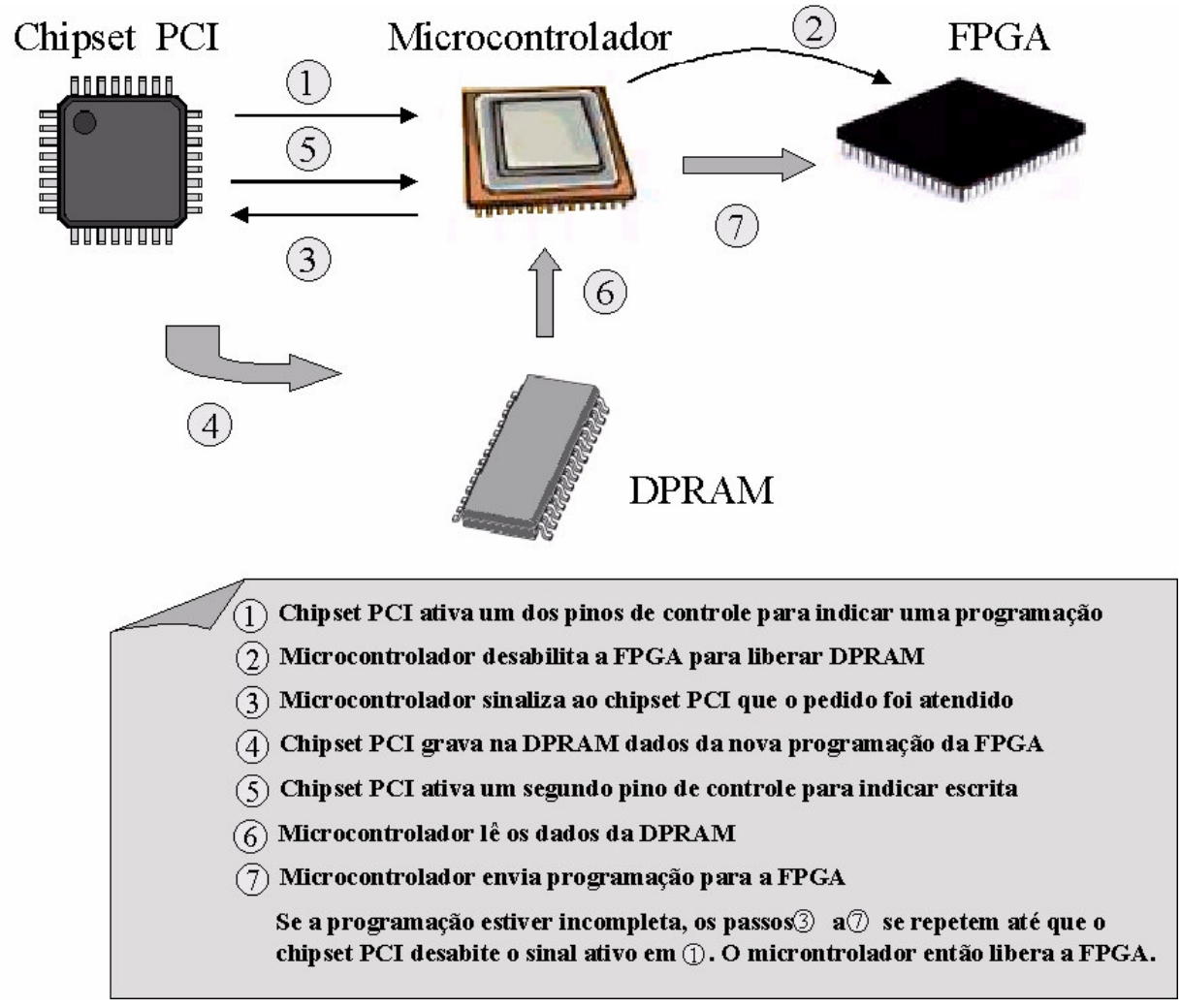



**Figura 9 - Interação entre os componentes da placa durante a reprogramação**

O microprocessador utilizado foi o MCF5206e, da família Coldfire de Motorola. Trata-se de um microprocessador RISC de 32 bits com memória SRAM interna de 8KB e cache de 4KB permitindo realizar as etapas de transferência e processamento dos dados da

placa em alta velocidade. Especificamente para um relógio de 40MHz, que foi utilizado no projeto, o microprocessador apresenta uma performance de 33MIPS (milhões de instruções por segundo).

Dois módulos de memória RAM estática são disponíveis na placa para uso com processamento interno. A escolha de memórias estáticas em detrimento às memórias dinâmicas, normalmente mais baratas, se dá pela necessidade de reduzido tempo de acesso quando da transferência de dados em tempo real para o computador. Enquanto memórias dinâmicas possuem tempo de acesso em torno de 30 a 70ns, memória estáticas operam com tempos de 5 a 30ns. As memórias utilizadas na placa são 71016LS15 com tempo de acesso de 15ns. Cada módulo possui capacidade de 64k x 16 bits, perfazendo um total de 256 Kbytes de memória para a placa. A previsão destes módulos de memória tem dois objetivos. O primeiro módulo de memória é utilizado para a gravação de dados que são utilizados pelo microcontrolador para a programação da FPGA. O segundo módulo foi disponibilizado para a FPGA para implementação de recursos adicionais como buffer secundário de dados ou repositório temporário para implementação de mecanismos de filtragem, caso venha a ser necessário.

### 5.1.5 Interface com camadas física de barramentos industriais

O interfaceamento com os diferentes barramentos suportados é feito por circuitos externos que utilizam transceivers próprios para cada meio físico suportado. São três as interfaces previstas: RS-485, IEC1158-2 e CAN.

#### *5.1.5.1 Interface RS485*

A interface RS485 da placa utiliza um transceiver 75LS176 que adapta os sinais provenientes do barramento em sinais de nível TTL. Este transceiver é também compatível com o padrão RS422, suportado pelo barramento Interbus-S. A fim de garantir maior imunidade à placa foi projetada uma interface de isolamento ótico entre o transceiver RS485 e a lógica programável. Devido às velocidades envolvidas com esta interface (até 12 Mbps) foi necessário selecionar opto isoladores de alta velocidade (HCPL0710).

#### *5.1.5.2 Interface IEC1158-2*

A interface IEC1158-2 foi implementada a partir de um componente transceiver específico (SIM1), projetado para operar em áreas intrinsecamente seguras. Os sinais provenientes do SIM1 são isolados opticamente da lógica programável. Como este barramento normalmente funciona a baixas velocidades (31,25 Kbps) foram utilizados opto isoladores de uso mais comum (HCPL061).

#### *5.1.5.3 Interface CAN*

A interface CAN da placa de aquisição foi, por sua vez, implementada utilizando-se dois componentes distintos. Do lado do barramento empregou-se o transceiver 82C250, com a finalidade de interpretação dos sinais do barramento. Acoplado a este transceiver está um controlador CAN, SJA1000, responsável pela identificação de mensagens e recuperação das informações da rede. A comunicação entre o transceiver e o controlador CAN é isolada opticamente através de opto isoladores de alta velocidade (HCPL0710).

#### *5.1.5.4 Outras interfaces*

Um conector adicional de expansão, contendo os principais sinais de controle da placa, foi disponibilizado durante o projeto da placa BUSMON, a fim de possibilitar a implementação de alguma outra interface externa necessária.

Uma opção interessante seria utilizar esta expansão para o projeto de uma interface Ethernet de alta velocidade (100 Mbps), buscando assim atender à especificação tipo 5 (HSE) da norma IEC61158.

# 6 PROJETO E IMPLEMENTAÇÃO DO SOFTWARE

Este capítulo será dividido em três partes, a fim de melhor descrever os diferentes sub-sistemas em *software* que compõem o sistema BR-Tool: programação da FPGA da placa, programação do *driver* de comunicação para PC e a ferramenta de validação propriamente dita que executa no PC.

## 6.1 PROGRAMAÇÃO DA FPGA

A programação da FPGA utilizada na placa de aquisição foi desenvolvida na linguagem de descrição de *hardware VHSIC Hardware Description Language* (VHDL), que visa o desenvolvimento de descrições modulares. A utilização de uma programação modular tem especial importância para este tipo de componentes, de forma a permitir o desenvolvimento de processos em paralelo. A programação desenvolvida para o projeto pode ser basicamente dividida em três módulos: Serial, Timer e RAM.

O módulo Serial é responsável pela identificação e montagem de informações provenientes do barramento. O módulo Timer é responsável pela manutenção de temporizadores do sistema, fazendo a marcação temporal de instante de ocorrência de eventos detectados. O módulo RAM é responsável pelo interfaceamento da lógica programável com memórias externas.

### 6.1.1 Módulo Serial

O módulo Serial, desenvolvido em VHDL, é responsável pelo tratamento de informações circulantes nos diversos meios físicos suportados, a fim de identificar os dados nos barramentos de campo. Eventos de barramentos podem ser caracterizados por bytes ou

sequências de bits. O módulo Serial não processa as mensagens do barramento, apenas realiza a identificação de bytes ou sequências de bits, registrando e sinalizando como um novo dado.

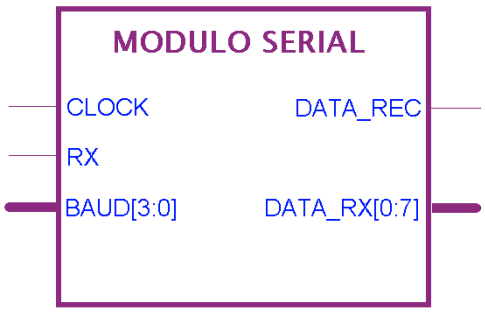



**Figura 10 - Representação do módulo Serial**

A fim de garantir a correta identificação de cada um dos bits que compõem as mensagens do barramento selecionou-se uma taxa de amostragem 4 vezes superior à taxa de comunicação (o teorema de Nyquist exige que a frequência de amostragem seja pelo menos o dobro da amostrada). Considerando-se a mais alta velocidade de barramentos da IEC61158, 12Mbps para PROFIBUS-DP, a velocidade da lógica programável foi selecionada como 48MHz. Para se definir qual a velocidade de comunicação corrente no barramento de campo amostrado, o sinal BAUD deve ser ajustado conforme a Tabela 3:

**Tabela 3 - Definição de velocidade do módulo Serial**

| Valor do sinal BAUD | Velocidade |
|---|---|
| 0 | 9600 bps |
| 1 | 19200 bps |
| 2 | 31.25 kbps |
| 3 | 76.8 kbps |
| 4 | 93.75 kbps |
| 5 | 187.5 kbps |
| 6 | 500 kbps |
| 7 | 1.5 Mbps |
| 8 | 12 Mbps |

A fim de possibilitar a interface com diversos barramentos de campo foram implementadas duas versões de módulo Serial: Serial_NRZ e Manchester.

O primeiro módulo realiza o tratamento de dados codificados no padrão *Non Return to Zero* (NRZ), utilizando delimitadores de bit de início (*start bit*), paridade e bit de parada (*stop bit*).

Os sinais de entrada do módulo são CLOCK, que recebe o sinal de relógio do sistema, RX, proveniente do barramento de dados, e BAUD, informação de velocidade de comunicação corrente. As saídas são DATA_REC, que indica a recepção de um novo dado e DATA_RX, que informa o dado recebido. Na Figura 11 apresenta-se o comportamento do módulo Serial na identificação de dois bytes de dados (E5h e BCh respectivamente).

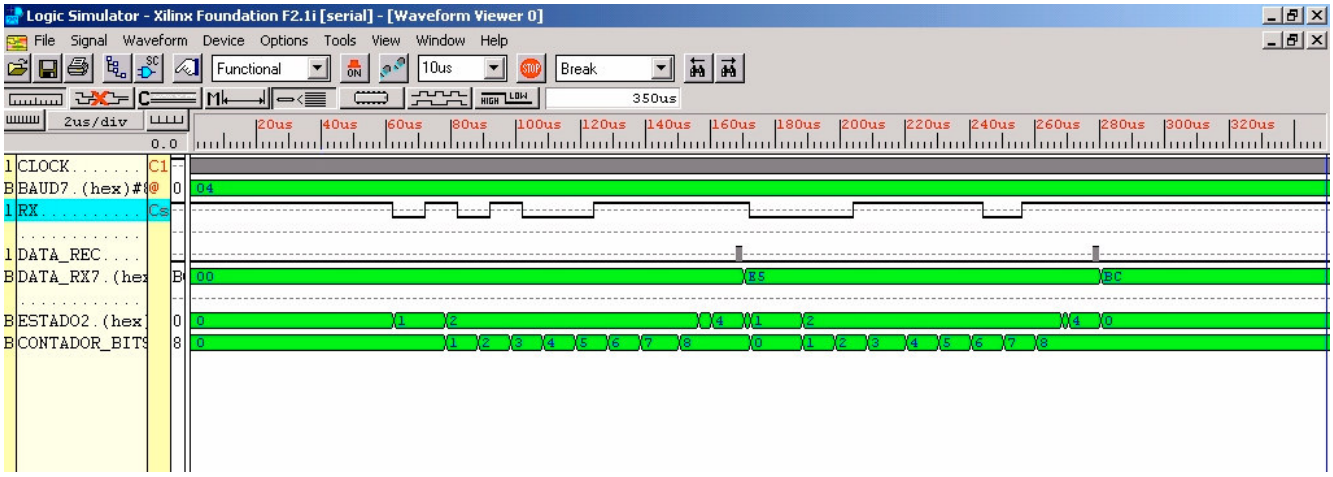



**Figura 11 - Simulação do módulo Serial_NRZ**

O módulo Serial_Manchester foi desenvolvido para tratar com sinais que usam o formato de codificação Manchester, o qual transfere informação de relógio junto ao sinal de dados. São previstos diferentes formatos e valores de preâmbulo, que dependem do tipo de barramento.

Na Figura 12 apresenta-se o comportamento do módulo Serial_Manchester ao receber um byte (55h) transmitido, segundo os padrões de preâmbulo e delimitador de início de mensagem definidos pelo protocolo Foundation Fieldbus (IEC, 2001).

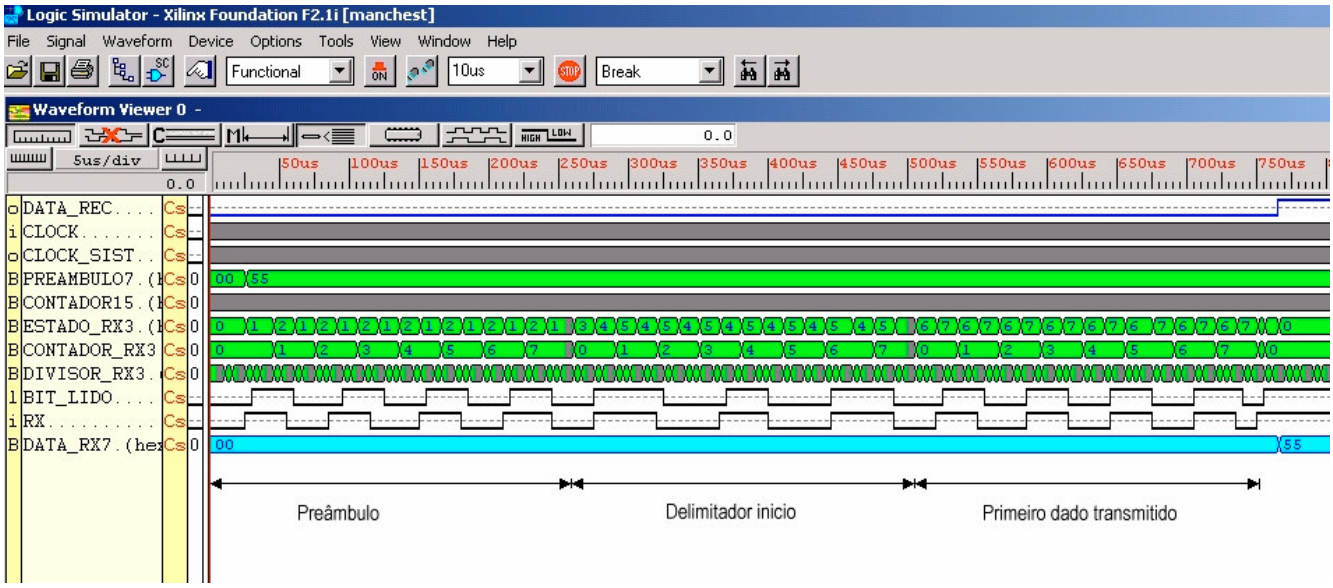



**Figura 12 - Simulação do módulo Serial_Manchester**

### 6.1.2 Módulo Timer

O módulo Timer é responsável por manter a referência de tempo da placa de aquisição e realizar a marcação temporal (*time stamping*) sobre eventos detectados no barramento. Possui um temporizador de alta resolução (37 bits) usado para registrar o tempo relativo entre eventos, ou seja grava-se a informação da diferença de tempo entre a ocorrência de dois eventos consecutivos.


**Figura 13 - Representação do módulo Timer**

A fim de se reduzir a quantidade de memória utilizada no armazenamento das mensagens adquiridas no barramento, utiliza-se uma estrutura evento-tempo (contendo tipo de evento detectado e informação temporal de ocorrência do mesmo) com tamanho variável. Dependendo da quantidade de tempo registrada para um dado evento de barramento, a estrutura pode ocupar 16, 32 ou 48 bits.

Se o valor de tempo entre dois eventos consecutivos for menor que 128 (7 bits de resolução), apenas uma palavra de 16 bits é utilizada. O byte inferior da palavra contém o tipo de evento detectado e o byte superior a informação de tempo relativo.

Se o valor de tempo registrado for maior que 127, porém menor que 4194304 (22 bits), são utilizados 32 bits para representar a estrutura evento-tempo. O byte inferior da primeira palavra de 16 bits contém a informação de evento detectado, enquanto que o byte superior da mesma palavra mantém os 7 bits menos significativos da informação temporal. Para indicar que a informação temporal não está completa, o bit mais significativo da palavra inferior é ativado e a palavra superior recebe então a parte restante da informação temporal.

Se a informação temporal registrada ultrapassar 4194303, são utilizados 48 bits para representar a estrutura evento-tempo. A representação da palavra de 16 bits inferior segue a mesma codificação do caso anterior. A segunda palavra da estrutura mantém os 15 bits imediatamente superiores do valor de informação temporal. Para indicar a utilização de uma terceira palavra, o bit mais significativo da segunda palavra deve também ser ativado. A

última palavra da estrutura recebe então a parte mais significativa restante da informação temporal registrada para o evento.

Uma representação desta codificação é feita na Figura 14, onde se apresentam os três possíveis formatos de estruturas evento-tempo. A primeira estrutura (mais à esquerda) possui 16 bits de tamanho (o byte menos significativo representa o evento, D8h, e o mais significativo a informação temporal, 2Eh). A estrutura central possui representação de 32 bits, sendo o evento, 53h, (byte menos significativo da primeira palavra). Os bits restantes compõem a informação temporal 104C26h (deve-se desprezar o bit mais significativo da primeira palavra, que serve apenas para informar o uso de mais uma palavra para compor a informação temporal). A estrutura à direita da figura ocupa 48 bits e segue o mesmo princípio de interpretação. Neste caso o valor do evento é 95h e da informação temporal 9105C6CDh (também aqui os bits mais significativos das palavras inferiores são desprezados).

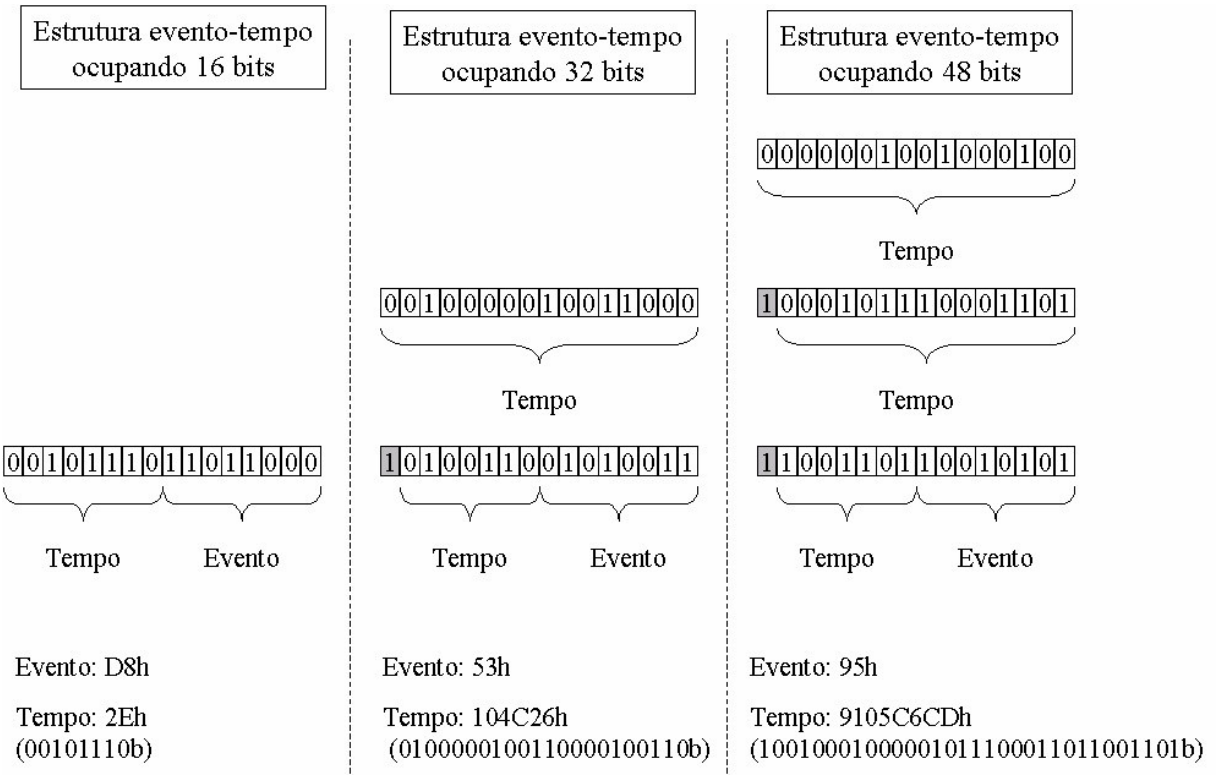



**Figura 14 - Codificação das informações evento-tempo**

Os sinais de entrada do módulo Timer são CLOCK, que recebe o sinal de relógio do sistema, DATA_REC, que informa a presença de um dado novo proveniente do módulo de recepção serial, DATA_RX, que informa o dado detectado e ESCREVENDO, que serve para controlar os resultados disponíveis em suas saídas evitando a mudança de valor enquanto os dados ainda não tiverem sido totalmente processados pelo módulo de escrita em memória. As saídas do módulo são DADO_PRONTO, que indica a disponibilidade de novos dados nas suas saídas e REGISTRADOR_A, REGISTRADOR_B e REGISTRADOR_C, que contém as três possíveis palavras de 16 bits com a informação evento-tempo calculada.

A granularidade do temporizador leva em consideração a mais elevada taxa de comunicação prevista pelos barramentos de campo suportados, que no caso é 12 Mbps para o protocolo PROFIBUS-DP (ver Tabela 1). Nesta velocidade o menor intervalo entre bytes (considerando os bits de início, parada e paridade) é de 909 ns (PROFIBUS, 1991). Afim de garantir a consistência temporal para este caso extremo, a resolução do temporizador do sistema foi ajustada para 500 ns.

A Figura 15 apresenta o comportamento do módulo Timer na detecção de três eventos novos. A primeira estrutura evento-tempo calculada tem tamanho de 16 bits (10E5h), enquanto que as demais são representadas em 32 bits (1FBC6h e 2C2E5h respectivamente). A interpretação do primeira estrutura representa um evento de valor E5h (byte inferior) e 10h de informação temporal, o que corresponde a 8µs (10h x 500ns = 16 x 500ns = 8µs). A segunda estrutura representa o evento C6h e o tempo F3h (desprezando bit de sinalização), o que corresponde a 121µs (F3h x 500ns = 243 x 500ns = 121,5µs). Similarmente a última estrutura representa o evento E5h com informação temporal de 142h (desprezando bit de sinalização), o que corresponde a 161µs (142h x 500ns = 322 x 500ns = 161µs).

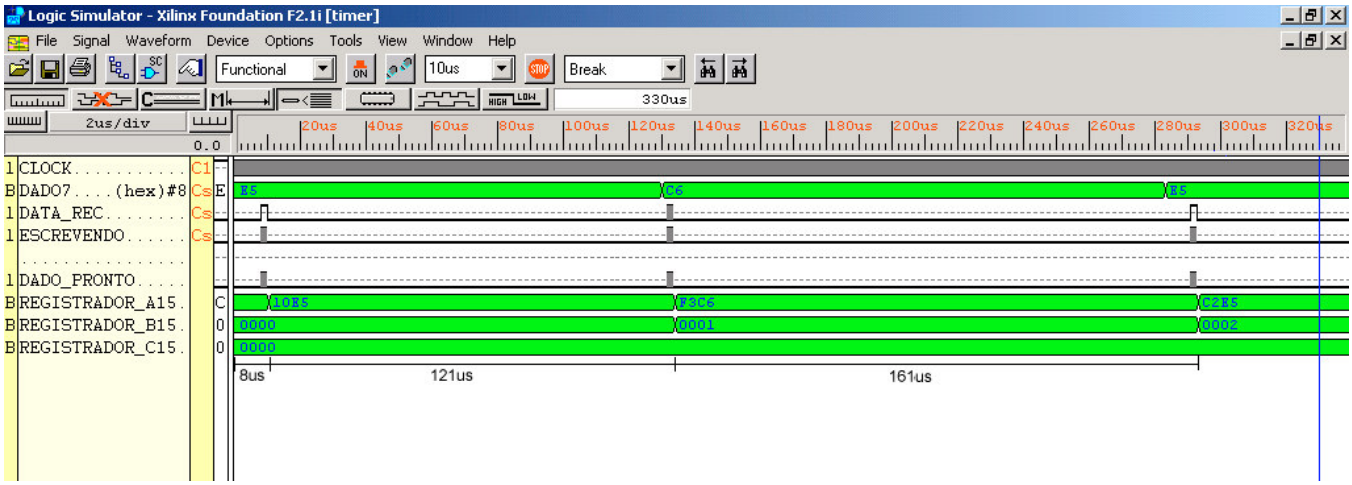



**Figura 15 - Simulação do módulo Timer**

### 6.1.3 Módulo RAM

O módulo RAM, desenvolvido em VHDL, é responsável pelo interfaceamento com memórias externas, funcionando em conjunto com o módulo Timer. A fim de garantir a operação com memórias *Static RAM* (SRAM) comerciais, sinais de controle foram gerados.

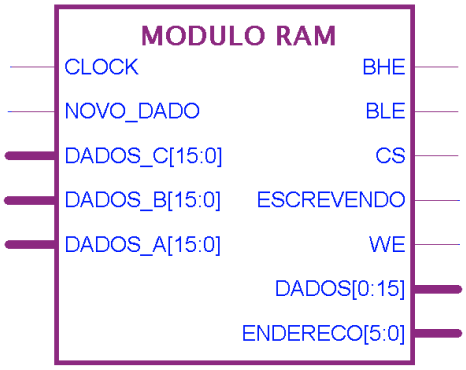



**Figura 16 - Representação do módulo RAM**

O módulo RAM lê três palavras de 16 bits ao perceber o sinal NOVO_DADO e interpreta os valores recebidos para identificar o tamanho da informação fornecida. Conforme apresentado anteriormente, esta informação pode ocupar até três palavras de 16 bits. Como as memórias utilizadas na placa são de 16 bits, caso a informação evento-tempo ocupe mais que uma palavra, o valor deve ser escrito na memória por uma sequência de acessos.

Os sinais de entrada do módulo RAM são CLOCK, que recebe o sinal de relógio do sistema, NOVO_DADO, que indica a presença de novos dados nas entradas e DADOS_A, DADOS_B e DADOS_C, que contém as três palavras de 16 bits provenientes do módulo temporizador (nova informação evento-tempo). As saídas do módulo são *Bus High Enable* (BHE) e *Bus Low Enable* (BLE), que indicam a validade de dados presentes nos barramentos de dados alto e baixo respectivamente, *Chip Select* (CS), que habilita a operação da memória, ESCREVENDO, que informa que dados estão sendo escritos na memória no presente instante, *Write Enable Write Enable* (WE), que ativa a memória para escrita, DADOS, que contém os dados a serem escritos e ENDERECOS, que informa o endereço de escrita.

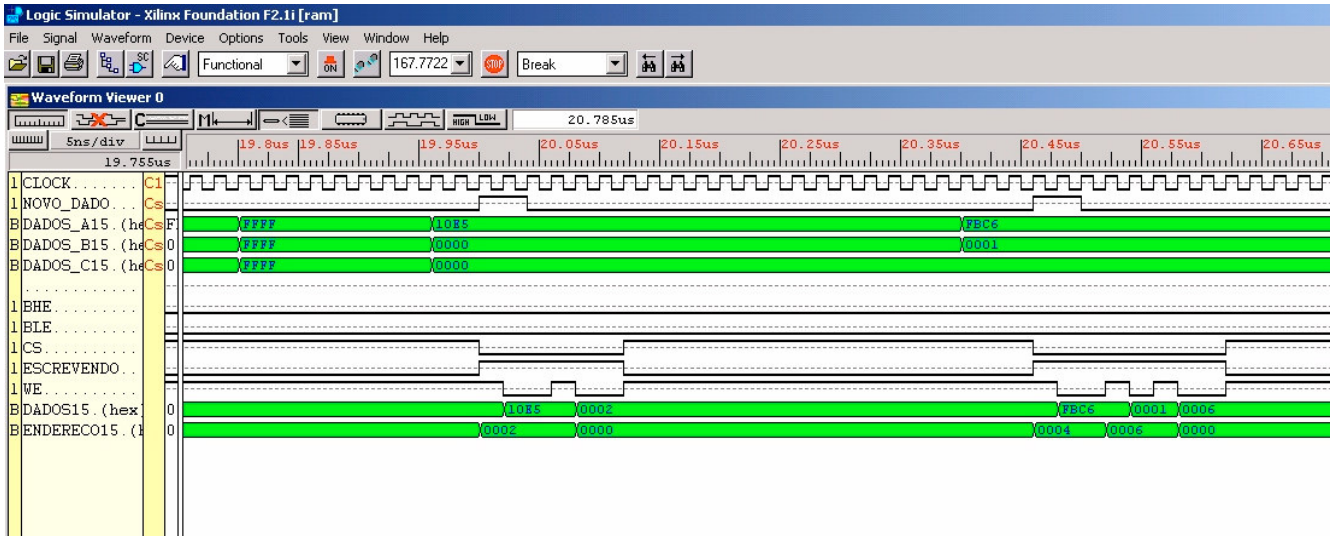



**Figura 17 - Simulação do módulo RAM**

A Figura 17 ilustra o funcionamento do módulo RAM ao se escrever duas estruturas evento-tempo. A primeira ocupando 16 bits gera dois pulsos de escrita em memória externa, o primeiro para gravação da estrutura propriamente dita e o segundo para informar o último endereço atualizado do buffer circular. Similarmente a segunda estrutura evento-tempo de 32 bits gera três acessos de escrita em memória (dois para gravação da estrutura em uma memória de 16 bits e um para informar o novo endereço atualizado no buffer circular).

## 6.2 DRIVER DE COMUNICAÇÃO

Um *driver* para PC é uma rotina ou conjunto de rotinas que implementam operações de I/O relacionadas a um dispositivo de *hardware*. Para acesso aos dados da placa BUSMON um *driver* de comunicação próprio foi desenvolvido. A plataforma de trabalho foi Windows 2000, para manter compatibilidade com as ferramentas já previamente desenvolvidas por (WILD, 2000) e também por ser um sistema operacional largamente utilizado em empresas de automação industrial.

A partir de 1994, com a plataforma Windows NT, a empresa Microsoft passou a utilizar uma estratégia de proteção de acesso a recursos do sistema, principalmente visando um aumento de confiabilidade (assim como é feito por outros sistemas operacionais modernos). Uma aplicação rodando dentro de Windows (incluindo Windows NT, 2000, Me e XP) não pode acessar diretamente recursos de *hardware*, ficando limitada à utilização de uma camada de abstração de *hardware*, chamada *Hardware Abstration Layer* (HAL), que isola as aplicações desenvolvidas do *hardware* do sistema. O gerenciamento implementado para uso do HAL evita o acesso indevido a recursos compartilhados, o que representaria uma grande fonte de falhas de funcionamento do sistema operacional.

A arquitetura de microprocessadores Intel implementa quatro níveis de privilégios: Ring 0, 1, 2 e 3. Ring 0 é nível mais privilegiado onde todos os recursos internos do processador são disponibilizados. Consequentemente o nível Ring 3 é o mais restrito, inibindo o acesso a quaisquer recursos críticos do processador. O Windows 2000 utiliza apenas dois destes níveis de privilégios em sua estrutura de programação, provendo dois modos de operação: modo kernel (ou modo administrador), que roda em Ring 0, e modo usuário, que roda em Ring 3.

Assim sendo a filosofia para desenvolvimento de uma aplicação que necessite fazer acesso a algum recurso de *hardware* do sistema exige que a mesma seja dividida em dois módulos: um *driver*, rodando em modo *kernel*, que pode fazer acesso ao HAL, e um programa de interface (incluído dentro da aplicação propriamente dita), rodando em modo usuário, que acessa o *driver* através funções disponibilizadas (BAKER, 1996).

São dois os principais tipos de *drivers* disponíveis para ambiente Windows 2000: KDM e WDM:

a) *driver Kernel Mode Driver* (KMD) permite acesso de *hardware* a toda arquitetura do sistema previamente definida durante sua operação. Sua definição entretanto por datar da época de criação do Windows NT, possui limitação de acesso a alguns recursos atuais com operação dinâmica;
b) modelo *Windows Driver Model* (WDM) foi desenvolvido a partir da criação do Windows 2000, apresentado as mesmas capacidades de um *driver* KMD e permitindo ainda recursos extras de *plug and play* e administração de energia.

A fim de atender aos recursos mais atuais, um *driver* WDM foi implementado para o projeto. Para reduzir as etapas de desenvolvimento de código, foi adquirida uma *Application Programming Interface* (API) fornecida pela empresa PLX, desenvolvedora do chipset PCI utilizado no projeto da placa BUSMON. A API adquirida constitui-se de uma biblioteca de funções de programação, leitura e escrita e atendimento a interrupções, que são executadas pelo *driver* WDM.

A estrutura de comunicação com a placa, indicando os vários módulos constituintes: aplicação (representando a ferramenta de validação desenvolvida), biblioteca API (adquirida da empresa PLX) e *driver* WDM BUSMON (instalado em modo administrador) é apresentada na Figura 18.

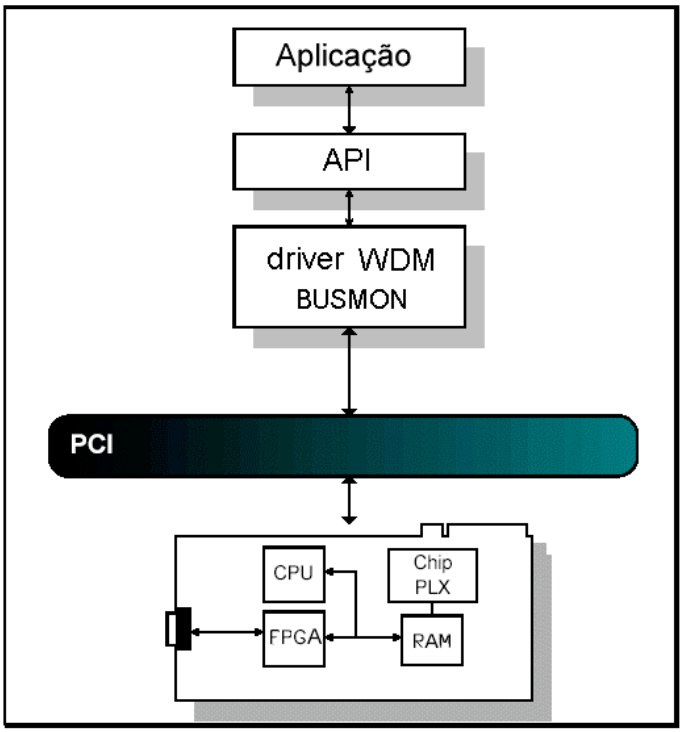



**Figura 18 - Estrutura do driver de comunicação com a placa BUSMON**

## 6.3 FERRAMENTA DE VALIDAÇÃO

Conforme já comentado, o *software* VCAT, desenvolvido inicialmente no âmbito da dissertação de Rafael Wild (WILD, 2000), foi utilizado como base para o desenvolvimento de uma ferramenta de validação temporal para diferentes protocolos de comunicação usados em sistemas de automação industrial.

A fim de tornar a ferramenta VCAT genérica para a operação com diferentes barramentos de campo, algumas atualizações se fizeram necessárias. Para facilitar a identificação das alterações implementadas, as mesmas são apresentadas seguindo o modelo hierárquico apresentado na Figura 4.

### 6.3.1 Registro de eventos

A primeira alteração implementada refere-se ao reconhecimento dos dados provenientes da placa BUSMON. Na primeira versão do *software* VCAT, a entrada de dados era feita por arquivos previamente salvos, segundo o formato definido pela ferramenta de monitoração FBView da empresa SMAR.

Nesta nova versão da ferramenta permite-se duas formas de leitura de eventos:

a) mensagens são lidas diretamente da placa de aquisição, de maneira a permitir uma validação *on-line* do sistema alvo;

b) mensagens são lidas de arquivos, após terem sido previamente salvas.

Para permitir a leitura *on-line* de eventos, o *software* VCAT passou a interagir diretamente com o *driver* de comunicação da placa BUSMON (ver seção 6.2), a fim de buscar, em tempo de execução, dados do barramento. Os dados são adquiridos via um buffer de dados com tamanho dependente da quantidade de mensagens detectadas no intervalo de

monitoração. O formato do buffer de dados recebido da placa BUSMON segue a codificação apresentada na Figura 14, que permite a representação de estruturas evento-tempo de tamanho variável. A interface de entrada de dados da ferramenta foi então adaptada para realizar a interpretação de dados segundo o formato adotado. A fim de facilitar o tratamento destes dados pelos demais módulos da ferramenta, as estruturas de tamanho variável são convertidas, tão logo sejam lidas, para um buffer local composto por estruturas de tamanho fixo. O uso de estruturas de tamanho fixo, apesar de levarem a um maior consumo de memória, facilitam bastante o processamento das informações. Dado que o sistema VCAT roda em ambiente PC com recursos de memória muito superiores aos da placa BUSMON, o maior consumo de memória é aceitável e representa uma solução interessante em função da maior agilidade de processamento.

Para a realização de processamento *off-line*, existe a possibilidade de se trabalhar com dados provenientes de arquivos previamente salvos. São previstos dois formatos de arquivos: binários e processados. Os arquivos binários contém dados montados segundo a mesma estrutura de buffers usada internamente pela ferramenta. Os arquivos processados são arquivos que contém os dados resultantes de uma análise previamente realizada pela ferramenta. Os arquivos processados são significativamente menores que os arquivos binários por conterem apenas as informações utilizadas pelo módulo de visualização (após passarem por processamento dos módulos de filtragem e validação). Por sua vez os arquivos binários permitem maior flexibilidade ao usuário. Uma vez que estes contêm os dados originais obtidos da monitoração, permite-se a realização de diferentes formas de análise para o mesmo conjunto de dados, se necessário, alterando-se os parâmetros de filtragem e validação. Ambos os formatos podem ser salvos pela ferramenta.

### 6.3.2 Filtragem de eventos

A ferramenta precisa interpretar os dados obtidos durante a monitoração (bytes ou sequências de bits detectadas no barramento) a fim de montar as mensagens correspondentes. A interpretação destes dados leva em consideração as especificações definidas pela camada 2 (camada de enlace de dados) do modelo de referência OSI de cada um dos protocolos adotados (TANENBAUM, 1989).

Por ser dependente do protocolo que está sendo tratado, o módulo de filtragem da ferramenta, precisou ser alterado para possibilitar o tratamento de diferentes tecnologias de barramentos de campo. Para a correta operação da ferramenta de validação, o barramento de campo que irá ser analisado deve ser informado pelo usuário (através do menu de configurações da ferramenta).

Para prover uma filtragem bastante seletiva de diversos tipos de eventos (eventos estes relacionados às mensagens detectadas no barramento) foi criada uma janela para especificação de eventos, que possibilita a definição de diversos parâmetros de filtragem. Os parâmetros disponíveis permitem tratamento de informações que distingam mensagens no barramento, tais como endereçamento, campo de controle e valores de dados transmitidos, sendo entretanto estes parâmetros fortemente relacionados às características específicas de cada protocolo.

Dependendo do tipo de barramento selecionado pelo usuário, uma janela de especificação de eventos diferente será habilitada. Na Figura 19 tem-se um exemplo da janela correspondente ao protocolo PROFIBUS-DP.

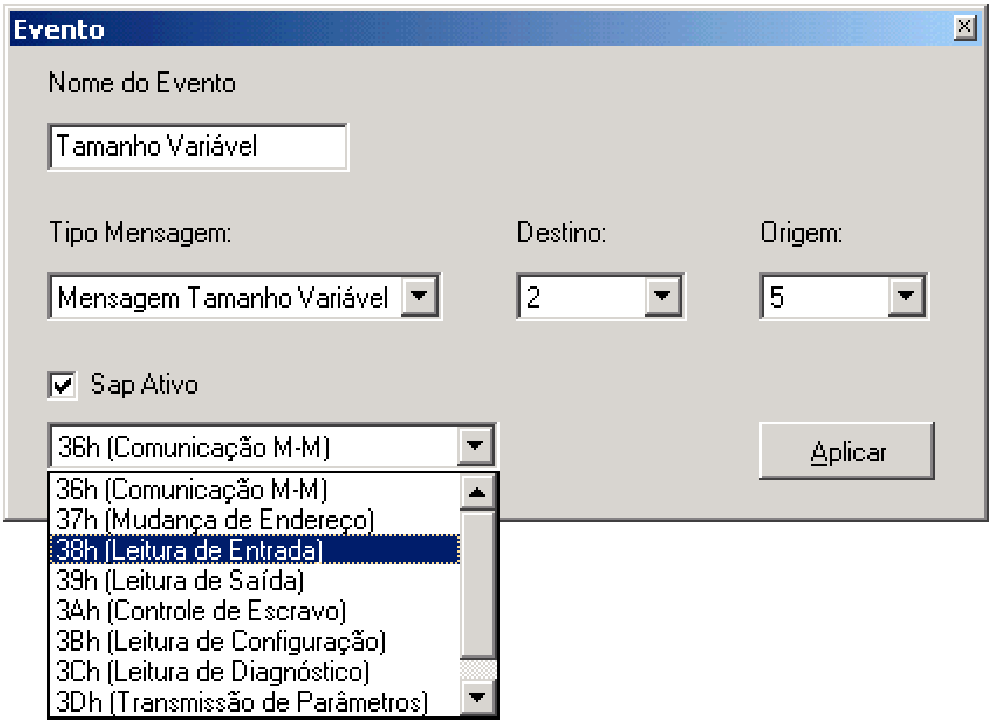



**Figura 19 - Janela para especificação de eventos em uma rede PROFIBUS-DP**

A fim de facilitar a compreensão dos usuários na interpretação dos eventos, na atual versão da ferramenta de validação, os eventos trabalhados são identificados por nomes definidos pelo usuário e podem ser referenciados por estes nomes durante toda a análise. A primeira versão do software VCAT identificava os eventos válidos por números fixos internamente definidos pela ferramenta, o que muitas vezes dificultava a análise dos resultados obtidos.

Eventos que não atendam às especificações de qualquer um dos eventos definidos pelo usuário são caracterizados como “Outros”, e sua influência na rede é considerada durante a análise temporal.

### 6.3.3 Especificação de requisitos temporais

A ferramenta VCAT se utiliza de uma notação proposta por (PEREIRA, 1995), a qual se baseou na lógica RTL (JAHANIAN; MOK, 1986) para especificação de requisitos temporais. A especificação pode ser feita em um arquivo texto e lida pela ferramenta, possibilitando salvamento de diferentes arquivos de restrição temporal.

Esta estratégia foi mantida na atual versão, incluindo-se porém uma janela própria para a especificação de requisitos, o que facilita a interação com o operador. Os eventos são tratados pelos próprios nomes definidos pelo usuário durante a etapa de filtragem. A partir da janela de especificação de requisitos temporais pode-se definir diferentes relações entre eventos suportadas pela ferramenta. São possíveis definições de relações do tipo "antes de" (BEFORE), "após" (AFTER) ou "com ciclicidade" (CYCLIC), com especificação de intervalo temporal válido, em resolução de microssegundos.

Um exemplo desta janela de especificação de restrições temporais é apresentada na Figura 20.


**Figura 20 - Janela para especificação de requisitos temporais**

### 6.3.4 Análise de requisitos temporais

A análise de requisitos temporais teve também que sofrer algumas pequenas alterações para esta nova versão. Para que a análise temporal dos diferentes barramentos de campo se processe de maneira correta, a velocidade de transmissão utilizada pela rede alvo deve ser informada pelo usuário (através do menu de configurações da ferramenta).

Uma limitação detectada com a experiência de uso do sistema VCAT foi a falta de uma informação quantitativa que descrevesse o atendimento temporal dos requisitos especificados. Muitas vezes torna-se difícil para o usuário determinar a quantidade total de eventos que encontram-se fora de seus intervalos de validade. Assim sendo esta informação passou a ser gerada pela ferramenta, podendo ser visualizada pelo usuário durante as análises de histograma (Figura 21).

### 6.3.5 Visualização de resultados

Alguns recursos adicionais foram disponibilizados para aperfeiçoar as interfaces de visualização de resultados da ferramenta VCAT. As primeiras alterações se encarregaram de adaptar as novas estruturas de eventos, criadas durante a etapa de filtragem, para exibição nos gráficos produzidos. Todos os eventos passaram a ser identificados pelos nomes definidos pelo usuário, facilitando a interpretação de resultados.

O gráfico de histograma foi adaptado para apresentar também a informação de porcentagem de eventos que não atendem seus requisitos temporais, conforme pode ser visto na Figura 21 (parte superior da janela).

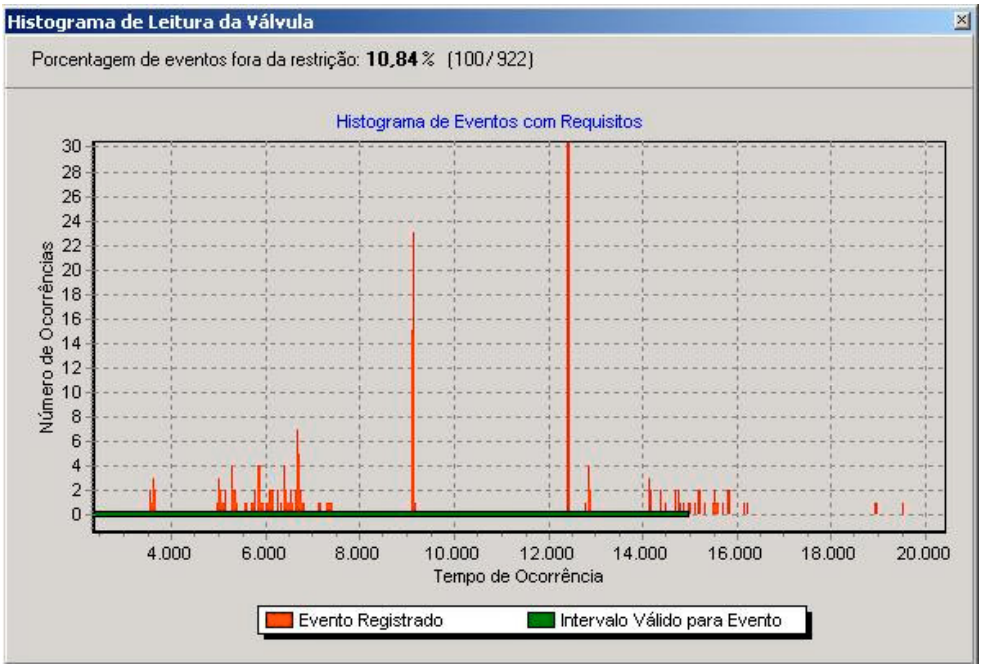



**Figura 21 - Histograma com informação quantitativa de atendimento a requisitos**

O gráfico de Gantt foi também alterado para possibilitar uma análise mais seletiva das restrições de cada evento. Se o usuário desejar realizar a análise de apenas um ou de parte dos eventos registrados, este pode selecionar isto por uma janela de seleção de eventos a serem avaliados. Um exemplo de uso deste recurso de tratamento seletivo é apresentado na Figura 22.

Algumas outras alterações gráficas foram também implementadas permitindo uma melhor interpretação visual dos intervalos de validade de cada evento e da identificação da condição de seu estado corrente (se o evento atende aos requisitos temporais, não atende ou simplesmente não foi avaliado).

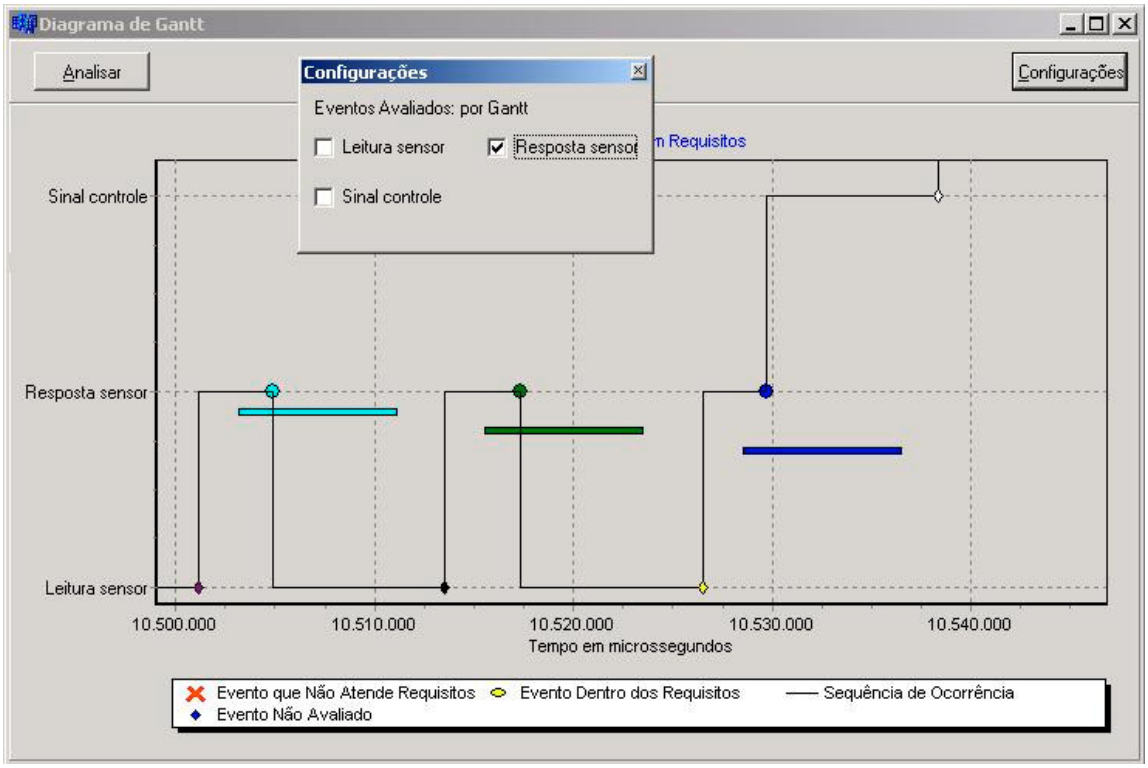



**Figura 22 - Diagrama de Gantt com apenas um evento sendo avaliado (Resposta sensor)**

# 7 VERIFICAÇÃO TEMPORAL E ELÉTRICA DO BR-TOOL

Para facilitar a verificação de funcionalidade do sistema BR-Tool a mesma foi dividida em duas partes: verificação temporal e verificação elétrica .

## 7.1 VERIFICAÇÃO TEMPORAL

A verificação temporal da ferramenta, apresentada nesta seção, irá considerar inicialmente os módulos principais da placa BUSMON: captura de eventos, marcação temporal e registro, passando posteriormente para a verificação do sistema de comunicação entre a placa e o PC.

### 7.1.1 Módulo de captura de eventos

A verificação temporal efetuada sobre o módulo de captura de eventos do sistema considera o tempo de reação deste para as diversas taxas de bit suportadas. A fim de facilitar a apresentação, as considerações e cálculos foram organizados separadamente abrangendo os diversos barramentos de campo suportados pela placa BUSMON (6 tipos definidos pela norma IEC61158 mais o barramento CAN).

#### *7.1.1.1 Tipo 1 da norma IEC61158 – Foundation Fieldbus*

Segundo a norma 61158, o protocolo Foundation Fieldbus quando utilizando o meio físico IEC1158-2 deve operar a uma velocidade de comunicação de 31,25 Kbps com um desvio máximo de 0,2%.

Para a verificação da deteção correta destes sinais pelo módulo de captura de eventos, deve-se considerar os atrasos referentes de todos componentes utilizados na interface com o

meio físico. Para o caso em questão, a interface IEC1158-2, foi implementada utilizando o transceiver SIM1 da Siemens e optoisoladores HCPL061N.

As características temporais, disponíveis na documentação do transceiver SIM1, importantes para a verificação da interface são os parâmetros de tempo de subida, *rise time* (*tr*), e tempo de descida, *fall time* (*tf*). O tempo *tr* representa o atraso que o dispositivo leva para transitar em sua saída do valor de 10% até 90% da tensão de alimentação. Similarmente o tempo *tf* representa o atraso para transição de 90% para 10% de sua excursão de saída.

O protocolo Foundation Fieldbus utiliza codificação Manchester para transmissão de dados, onde cada bit no barramento pode ser representado por até dois níveis lógicos. Assim sendo o menor tempo de bit (*tbit*) no barramento deve ser maior que a soma dos tempos *tr* e *tf*, pois caso esta condição não seja respeitada, o sinal na entrada do componente estará variando a uma taxa mais rápida do que a saída consegue acompanhar.

Como a interface com o SIM1 é isolada oticamente do restante da placa por intermédio de optoacopladores HCPL061N, os atrasos destes componentes devem também ser considerados. Os parâmetros a serem considerados para este componente são os tempos *Time of Propagation Low to High* ($t_{PLH}$) e *Time of Propagation High to Low* ($t_{PHL}$), que representam os atrasos de propagação após a deteção de um sinal na entrada até a transição de sua saída para nível alto e baixo respectivamente, conforme ilustrado na Figura 23.

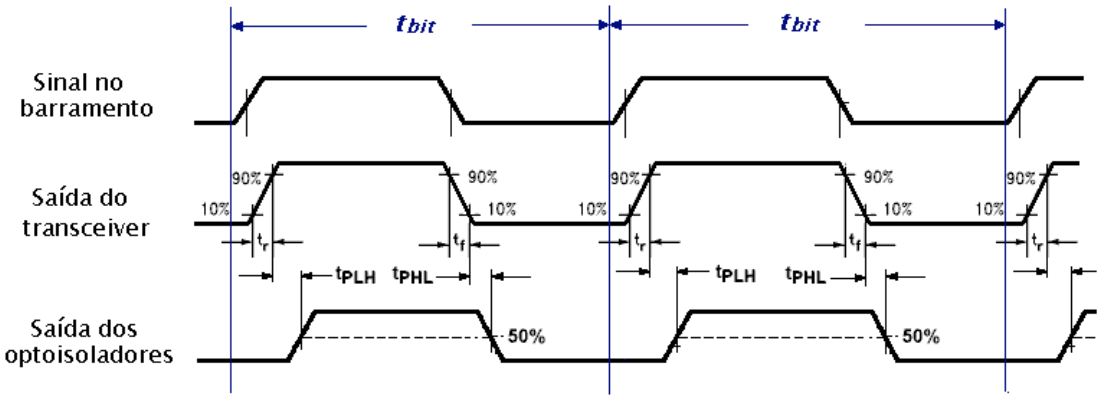



**Figura 23 - Representação de diagramas de tempo da interface IEC1158-2**

Para o caso analisado, os tempos máximos de *tr* e *tf* do transceiver SIM1 são ambos 500ns e os tempos $t_{PLH}$ e $t_{PHL}$ do HCPL061N são ambos 100ns. Para uma velocidade de 31,25 Kbps ± 0,2%, tem-se um tempo de bit de 32µs ±0,9 µs. O pior caso é dado pelo menor tempo de bit suportado.

Verificação:

$$Tbit_{min} \geq tr_{(transc.)} + tf_{(transc.)} + t_{PLH(opto)} + t_{PHL(opto.)}$$

$$32 - 0{,}9 \geq 0{,}5 + 0{,}5 + 0{,}1 + 0{,}1 \qquad \text{(em microssegundos)}$$

$$31{,}1 \geq 1{,}2$$

Devido às grandezas envolvidas no tratamento deste meio físico, atrasos na aquisição pela lógica programável foram desprezados (para a frequência de 48MHz o atraso de um ciclo de relógio da FPGA fica em torno de 0,02µs).

#### *7.1.1.2 Tipo 3 da norma IEC61158 – PROFIBUS*

O protocolo PROFIBUS possui duas versões na norma IEC61158: PROFIBUS-PA, que utiliza o meio físico IEC1158-2, e PROFIBUS-DP, que usa o meio físico RS 485.

As especificações para o Profibus PA são as mesmas definidas para o barramento Foundation Fieldbus utilizando o meio físico IEC1158-2 (31,25Kbps ± 0,2%). Como a interface da placa é a mesma, todas as considerações anteriores são também válidas, o que leva à mesma condição:

Verificação:

$$Tbit_{min} \geq tr_{(transc.)} + tf_{(transc.)} + t_{PLH(opto)} + t_{PHL(opto.)}$$

$$32 - 0{,}9 \geq 0{,}5 + 0{,}5 + 0{,}1 + 0{,}1 \qquad \text{(em microssegundos)}$$

$$31{,}1 \geq 1{,}2$$

Para o protocolo PROFIBUS-DP, que utiliza o meio físico RS 485, a norma define diversas velocidades de comunicação, de 9600 Kbps a 12 Mbps. Não há entretanto especificação quanto ao intervalo de variação destas velocidades na IEC 61158. Será considerada para esta verificação temporal o atendimento à mais alta taxa suportada pelo PROFIBUS-DP, 12 Mbps.

Conforme já apresentado, a captura de eventos para a interface RS485 é implementada pelo transceiver 75LS176 com isolação provida pelos optoisoladores HCPL0710. Assim para os cálculos da verificação tomam-se os tempos $t_{PLH}$ e $t_{PHL}$ de ambos os componentes. Na codificação NRZ adotada pelo barramento PROFIBUS-DP cada tempo de bit pode-se ter uma transição de nível lógico (representação dos atrasos gerados nesta interface está na Figura 24).

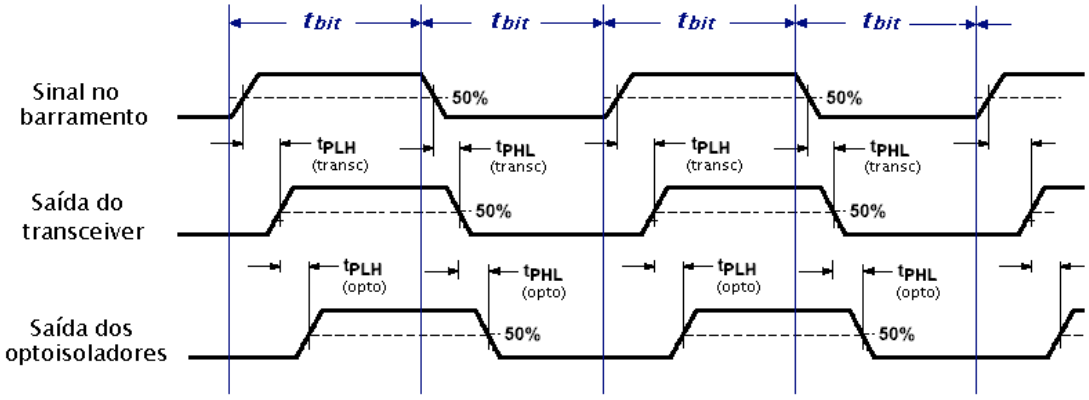



**Figura 24 - Representação de diagramas de tempo da interface RS485**

O pior caso de tempo de reação ocorre então para o maior valor de $t_{PLH}$ ou $t_{PHL}$ dos componentes operando na taxa mais alta do barramento. Os valores máximos de $t_{PLH}$ e $t_{PHL}$ para o transceiver são ambos 22 ns enquanto que para os optoisoladores são 40 ns. Assim:

Verificação:

$$Tbit_{min} \geq \max((t_{PLH(\text{transc.})} + t_{PLH(\text{opto})}), (t_{PHL(\text{transc.})} + t_{PHL(\text{opto.})}))$$

$$83{,}33 \geq \max((22 + 40), (22 + 40)) \qquad \text{(em nanossegundos)}$$

$$83{,}33 \geq 62$$

Pode-se considerar ainda um possível atraso na leitura do sinal resultante do circuito optoisolador, devido à amostragem da lógica programável. O máximo erro de leitura do arranjo lógico programável seria de um período de amostragem, que para a velocidade de 12Mbps é de 20,83ns (FPGA ajustada para uma frequência de 48MHz). Desta forma o cálculo para o pior caso passa a ser determinado por:

Verificação:

$$Tbit_{min} \geq \max((t_{PLH(\text{transc.})} + t_{PLH(\text{opto})}), (t_{PHL(\text{transc.})} + t_{PHL(\text{opto.})})) + t_{amostr(\text{FPGA})}$$

$$83{,}33 \geq \max((22 + 40), (22 + 40)) + 20{,}83 \qquad \text{(em nanossegundos)}$$

$$83{,}33 \geq 82{,}83$$

### *7.1.1.3 Tipo 4 da norma IEC61158 – P-NET*

O protocolo P-NET opera com o meio físico RS485 em taxas que vão de 9600 a 76800 bps. Não existe especificação na norma IEC61158 que defina faixas de variação das velocidades do barramento, assim considera-se a mais elevada taxa como 76800 bps. Como a interface é a mesma utilizada pelo protocolo PROFIBUS-DP pode-se utilizar a mesma análise anteriormente apresentada. Levando-se em conta que a taxa de amostragem da lógica programável seja configurada para o equivalente a quatro vezes a velocidade do barramento chega-se a:

Verificação:

$$Tbit_{min} \geq \max((t_{PLH(\text{transc.})} + t_{PLH(\text{opto})}), (t_{PHL(\text{transc.})} + t_{PHL(\text{opto.})})) + t_{amostr(\text{FPGA})}$$

$$13{,}02 \geq \max((0{,}022 + 0{,}04), (0{,}022 + 0{,}04)) + 3{,}25 \quad \text{(em microssegundos)}$$

$$13{,}02 \geq 3{,}312$$

### *7.1.1.4 Tipo 6 da norma IEC61158 – SwiftNet*

Também o protocolo SwiftNet opera em um barramento RS485 em taxas que vão de 78125bps a 5 Mbps. A norma IEC61158 também não especifica faixas de variação das velocidades do barramento. Devido à dificuldade de se obter valores divisíveis inteiros a partir da frequência de 48MHz, a placa BUSMON foi ajustada para operar com uma taxa máxima de 625 KHz neste protocolo (de fato o recomendado seria a utilização de um oscilador de 50MHz para operar com as taxas mais elevadas deste protocolo). Fazendo as mesmas considerações dos demais casos de interface RS 485, chega-se a:

Verificação:

$$Tbit_{min} \geq \max((t_{PLH(\text{transc.})} + t_{PLH(\text{opto})}), (t_{PHL(\text{transc.})} + t_{PHL(\text{opto.})})) + t_{amostr(\text{FPGA})}$$

$$1600 \geq \max((22 + 40), (22 + 40)) + 396 \qquad \text{(em nanossegundos)}$$

$$1600 \geq 458$$

### *7.1.1.5 Tipo 7 da norma IEC61158 – WorldFIP*

O protocolo WorldFIP, similarmente ao protocolo Foundation Fieldbus, opera com o meio físico IEC1158-2 na velocidade de comunicação de 31,25 Kbps ±0,2%. Levando-se em consideração os parâmetros já comentados para o tipo 1, chega-se a:

Verificação:

$$Tbit_{min} \geq tr_{(\text{transc.})} + tf_{(\text{transc.})} + t_{PLH(\text{opto})} + t_{PHL(\text{opto.})}$$

$$32 - 0{,}9 \geq 0{,}5 + 0{,}5 + 0{,}1 + 0{,}1 \qquad \text{(em microssegundos)}$$

$$31{,}1 \geq 1{,}2$$

### *7.1.1.6 Tipo 8 da norma IEC61158 – Interbus-S*

O protocolo Interbus-S opera com o meio físico RS 422/ RS 485 em taxas consideradas que vão até 500Kbps. A interface utilizada para este protocolo é a mesma interface utilizada para o meio físico RS 485 já comentado, que emprega o transceiver 75LS176. Assim, tomando-se as considerações apresentadas anteriormente, chega-se a:

Verificação:

$$Tbit_{min} \geq \max((t_{PLH(\text{transc.})} + t_{PLH(\text{opto})}), (t_{PHL(\text{transc.})} + t_{PHL(\text{opto.})})) + t_{amostr(\text{FPGA})}$$

$$2000 \geq \max((22 + 40), (22 + 40)) + 500 \qquad \text{(em nanossegundos)}$$

$$2000 \geq 562$$

### *7.1.1.7 Barramento CAN*

O barramento CAN por utilizar um mecanismo de arbitração não destrutível exige uma estrutura especial de montagem de cada bit escrito no barramento, de forma a garantir que todas estações acessem a rede no momento certo. O tempo de bit definido para comunicação, chamado de bit nominal ($t_{Bit}$), em realidade é composto de três segmentos: SYNC_SEG, TSEG1 e TSEG2. Os segmentos devem ser portanto amostrados por um período de relógio, $t_{SCL}$ (*Timer of the CAN System Clock*), várias vezes menor que o tempo de bit nominal da rede (JOHNK; DIETMAYER, 1997), conforme pode ser visto na Figura 25.

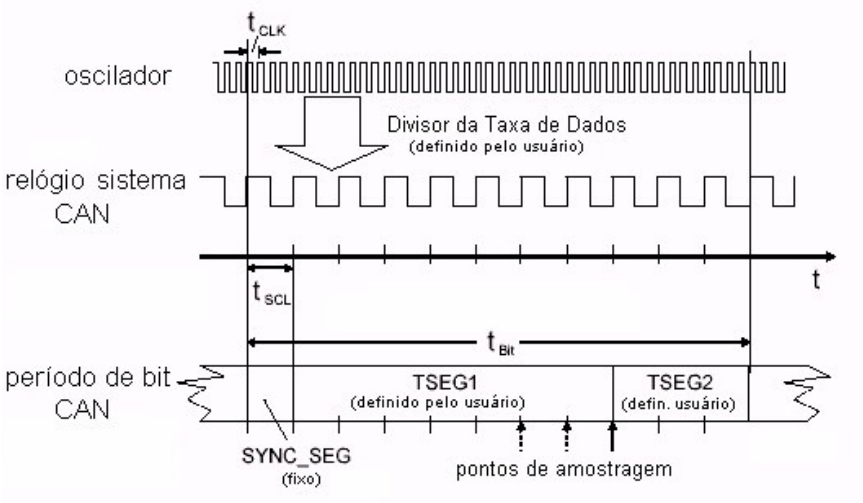



**Figura 25 - Representação da estrutura de um bit nominal CAN**

A verificação da interface CAN deve portanto considerar o tempo $t_{SCL}$ , ou também chamado de tempo de quantum ($t_q$), como requisito de atendimento. O pior caso para uma rede CAN ocorre ao utilizar a velocidade de 1Mbps (máxima taxa suportada pelo protocolo), que deve ser composto por 8 $t_q$. Assim o $t_q$ mínimo é 125ns.

O cálculo dos atrasos provenientes da interface CAN da placa BUSMON devem incluir os atrasos de propagação do transceiver, optoisolador e interface de entrada do microcontrolador CAN.

O atraso de propagação máximo documentado para o transceiver utilizado (82250) é o $t_{pd}$ (*time of propagation delay*) de 50ns. A leitura dos sinais do barramento é feita por um controlador CAN (SJA1000) que fornece como máximo atraso de entrada, $t_{SD}$ (*sum of delays*), o valor de 40ns. A isolação entre o transceiver e o controlador é provida pelos optoisoladores HCPL0710, que apresentam as mesmas características consideradas para a interface RS 485. Assim chega-se a:

Verificação:

$$T_{q\,min} \geq t_{pd} + t_{SD} + \max(t_{PLH(\text{opto})}, t_{PHL(\text{opto.})})$$

$$125 \geq 50 + 40 + \max(22, 22) \qquad \text{(em nanossegundos)}$$

$$125 \geq 112$$

### 7.1.2 Módulo de marcação temporal

A determinação de resolução do contador foi determinada visando-se a amostragem de eventos sem perdas (o período de atualização do contador deve ser inferior ao intervalo de ocorrência de eventos no barramento). Ajustou-se para tanto o período de 500ns (frequência interna de 2MHz) como para atualização do temporizador responsável pela marcação temporal de eventos na placa. Como já comentado o dado mais rapidamente atualizado no barramento ocorre com PROFIBUS-DP operando na taxa de 12Mbps. Como cada byte transmitido neste protocolo ocupa 11 bits, a taxa máxima de atualização é 1,1MB/s (período de 900ns) e a demonstração de funcionamento pode ser dada por:

Verificação:

$$T_{evento\_12MHz} \geq T_{temporiz}$$

$$900 \geq 500 \qquad \text{(em nanossegundos)}$$

O comportamento temporal interno ao módulo de marcação temporal não pode ser precisamente calculado, uma vez que os atrasos referentes ao seu processamento dependem da forma como sua programação é implementada, tipo de tecnologia adotada no projeto, estratégias de otimização de roteamento entre outros. Junto à ferramenta de desenvolvimento da Xilinx existe um módulo de verificação que pode informar características temporais da implementação. Para a lógica programável utilizada, Spartan XCS30PQ208–4 (o último número representa o atraso de propagação em ns) registra-se um atraso médio de 30ns, porém este valor pode variar de acordo com a instante temporal de ocorrência de sinais de entrada.

Para se determinar o pior caso de atraso do projeto, deve-se analisar a programação implementada. Na descrição implementada em VHDL o processo de deteção de um novo evento válido se baseia no sinal gerado pelo módulo de monitoração serial. Após detectar a

ocorrência deste sinal, ele disponibiliza em suas saídas o valor de tempo relativo obtido de seu temporizador interno. Assim para o tempo de pior caso considera-se que o sinal de novo evento válido levou um ciclo de relógio para ser detectado e que a disponibilização do valor temporal teve o atraso de um ciclo também. Chega-se ao valor de dois ciclos de relógio de atraso para o processo (este valor engloba os atrasos de propagação internos da lógica programável, por ser de uma ordem de grandeza pelo menos cinco vezes superior). Considerando-se ainda o atraso máximo de um ciclo de relógio para que o módulo de interpretação serial coloque na saída o dado detectado, chega-se ao total de três ciclos de relógio como o tempo de pior caso para a disponibilização de uma informação na saída do módulo de marcação temporal.

Este valor tem que ser inferior à resolução do temporizador, que opera concorrentemente aos demais processos da lógica programável, ou estaria-se comprometendo a confiabilidade do módulo de marcação temporal. Assim:

Verificação:

$$T_{temporiz.} \geq 3 \, . \, t_{relógio(FPGA)}$$

$500 \geq 3 \, . \, 20{,}83$ (em nanossegundos)

$500 \geq 62{,}5$

### 7.1.3 Módulo de registro de eventos

A verificação temporal do módulo de registro de eventos considera principalmente o tempo de salvamento dos registros de eventos detectados. Partindo-se do pressuposto que os módulos de marcação temporal e captura de eventos funcionam corretamente (conforme já

verificado), a próxima verificação a ser feita diz respeito ao tempo de salvamento das informações geradas pela lógica programável nas memórias do sistema.

A fim de atender às elevadas taxas de comunicação em redes de barramentos de campo, foram utilizadas na placa BUSMON memórias de alta velocidade com tempo de acesso de 15ns. Como os tempos de acesso envolvidos são inferiores ao período de relógio da lógica programável não são necessários ciclos de espera para escrita ou leitura de dados.

A análise do módulo de registro implementado na lógica programável mostra que cada acesso de escrita à memória é composto por dois ciclos de relógio (um para ativar o sinal de escrita, #WE, e um para desativá-lo). Conforme já apresentado, as estruturas evento-tempo tem tamanho variável, sendo dependentes do valor do temporizador. A maior estrutura ocupa 48 bits, o que exige três acessos de escrita à memória dupla porta de 16 bits da placa, totalizando seis ciclos de relógio. Logo após uma estrutura evento-tempo ter sido escrita, o indicador de posição atual do buffer de eventos (localizado no primeiro endereço da memória dupla porta) deve também ser atualizado, o que ocupa mais dois ciclos de relógio. Considerando-se adicionalmente o atraso máximo de um ciclo de relógio para que o módulo de registro inicie seu processo de escrita, chega-se ao maior atraso possível do módulo como sendo de nove ciclos de relógio.

Este valor tem que ser inferior ao menor tempo de ocorrência de um novo dado no barramento, sem acarretar em perda de leituras válidas. Como já citado a atualização mais rápida no barramento ocorre na taxa de 1,1MB/s. Assim:

Verificação:

$$T_{evento\_12MHz} \geq 9 \cdot t_{relógio(FPGA)}$$

$$900 \geq 9 \cdot 20{,}83 \qquad \text{(em nanossegundos)}$$

$$900 \geq 187{,}5$$

### 7.1.4 Comunicação entre a placa BUSMON e o PC

A verificação temporal efetuada visa garantir a comunicação entre a placa BUSMON e o computador através do barramento PCI. A interface PCI (33MHz) da placa BUSMON é implementada pelo chipset PLX9052 que provê acesso a uma memória dupla porta. A garantia de que os acessos de leitura à esta memória não causam atrasos maiores que o tempo de salvamento de novos dados, segue as considerações feitas durante a verificação temporal do módulo de registro de eventos. Ajustando-se o relógio do barramento no cálculo tem-se:

Verificação:

$$T_{evento\_12MHz} \geq 9 \cdot t_{relógio(PCI)}$$

$900 \geq 9 \cdot 30$ (em nanossegundos)

$900 \geq 270$

Conforme já comentado os dados adquiridos do barramento são salvos na memória dupla porta da placa BUSMON para posterior leitura por parte do barramento PCI. Se a ferramenta rodando no computador levar muito tempo para realizar a leitura destes dados na memória dupla porta estes podem ser sobreescritos, ocasionando em perda de dados válidos.

Como a estrutura de armazenamento de dados na memória é administrada como um buffer circular, pode-se dizer que uma sobreescrita somente pode ocorrer se 65535 palavras de 16 bits (capacidade máxima da memória dupla porta) forem escritas sem leitura por parte do computador. Considerando-se o pior caso como a monitoração de um barramento de 12Mbps, chega-se a uma taxa máxima de atualização da memória dupla-porta de 1.090.909 escritas por segundo, cada uma delas ocupando uma palavra de 16 bits. Pode-se provar que velocidades altas de comunicação produzem uma estrutura evento-tempo de 16 bits, pois para

se gerar valores de tempo relativo com mais de sete bits seriam necessários intervalos maiores que 64 μs (128 x 500ns) entre bytes, ou seja taxas de comunicação inferiores à 15625 kbps.

O número de posições da memória dupla porta ($Nro\_bytes_{DPRAM}$) dividido pelo número máximo de escritas a serem feitas em um segundo ($f\_atualiz_{DPRAM}$) define o tempo de estouro do buffer.

Este tempo deve ser inferior ao tempo de leitura por parte do PC. Apesar da plataforma Windows 2000 não ser um sistema operacional determinístico é indicado o atendimento a tarefas do sistema em intervalos de 10ms (DDK, 2000). Considerando o pior caso do atraso de dois intervalos de 10 ms para atendimento à tarefa de leitura no PC, tem-se:

Verificação:

$$Nro\_bytes_{DPRAM} / f\_atualiz_{DPRAM} \geq 2 \,.\, T_{Windows\_2000}$$

$$65535 / 1.090{,}909 \geq 2.\ 10 \qquad \text{(em microssegundos)}$$

$$60 \geq 20$$

### 7.1.5 Considerações sobre o comprimento da rede

Fatores adicionais referencentes à influência do comprimento dos barramentos de campo podem interferir nas medidas temporais. Isto porque para altas velocidades os tempos de propagações entre mensagens podem vir a ser significativos. A consideração destas condições não foi acrescentada na verificação temporal, pois em fato estes fatores não comprometem a funcionalidade do módulo de aquisição, mas sim provocam erros de precisão sobre as informações temporais registradas. A realização de cálculos de compensação destes erros, considerando distâncias entre as estações e a placa BUSMON e número de repetidores, deveriam ser implementados na ferramenta de validação, em casos onde estas medidas podem se tornar críticas.

## 7.2 VERIFICAÇÃO ELÉTRICA

A verificação elétrica diz respeito às características de consumo da placa BUSMON. Uma especificação bastante crítica para operação desta placa diz respeito ao atendimento das características elétricas para sua interface IEC1158-2. Segundo a norma IEC61158 para permitir operação em áreas intrinsecamente seguras, nenhum dispositivo neste barramento pode consumir mais que 40mA e para áreas não intrinsecamente seguras o consumo pode chegar a 120mA. A consulta à documentação do transceiver SIM1 utilizado para a interface com este meio físico indicam um consumo máximo de até 50mA nos pinos de leitura do barramento. Sendo assim a placa BUSMON pode ser utilizada para monitoração apenas de barramentos IEC1158-2 não intrinsecamente seguros.

Para o barramento RS485 define-se uma limitação de consumo de 500 μA por dispositivo para permitir a utilização de redes com até 32 estações. Os dados do transceiver 75LS176 indicam um consumo máximo de 400 μA, o que atende aos requisitos especificados.

As especificações elétricas do barramento CAN são especialmente definidas pela norma ISO11898 (ISO11898-2, 1999). Esta norma especifica, não a corrente de consumo para os nós da rede, mas sim a resistência interna diferencial dos dispositivos, a qual deve encontrar-se entre 10 e 100kΩ. O transceiver utilizado na interface CAN 82250 possui especificada uma resistência diferencial interna mínima de 20kΩ e uma máxima de 100kΩ, o que satisfaz às especificações.

## 7.3 RESUMO DA VERIFICAÇÃO DA PLACA BUSMON

O seguinte quadro resumo foi elaborado a partir dos resultados das verificações apresentadas:

**Tabela 4 - Quadro resumo da verificação para barramentos de campo**

| | Barramento | Meio Físico | Velocidade Máxima | Atende Verificação? | Velocidade Suportada |
|---|---|---|---|---|---|
| IEC 61158 | Tipo 1: Foundation Fieldbus | IEC1158-2 | 31.25 Kbps | **SIM** | 31.25 Kbps |
| | Tipo 2: ControlNet | Coaxial c/ isolação por transformador | 5 Mbps | NÃO | - |
| | Tipo 3: PROFIBUS-DP/ PROFIBUS-PA | RS485 / IEC1158-2 | 12Mbps / 31.25kbps | **SIM/ SIM** | 12Mbps / 31.25kbps |
| | Tipo 4: P-Net | RS485 | 76.8 Kbps | **SIM** | 76.8 Kbps |
| | Tipo 5: HSE | Ethernet | 100 Mbps | NÃO | - |
| | Tipo 6: SwiftNet | RS485 | 5 Mbps | **SIM** | 625 kbps |
| | Tipo 7: WorldFIP | IEC1158-2 | 31.25 Kbps | **SIM** | 31.25 Kbps |
| | Tipo 8: Interbus-S | RS485 | 500 Kbps | **SIM** | 500 Kbps |
| | CAN : (CANOPEN) | CAN | 1 Mbps | **SIM** | 1 Mbps |

# 8 ESTUDOS DE CASO

Foram definidos três estudos de caso para validação do sistema BR-Tool: um exemplo genérico de uso de uma rede PROFIBUS-DP e duas aplicações reais, uma na área de sistemas de manufatura empregando PROFIBUS-DP e outra na área de controle de transmissão de energia elétrica usando o barramento CANOpen.

Não se julgou necessária a realização de ensaios utilizando o barramento Foundation Fieldbus, uma vez que a funcionalidade da ferramenta operando com este tipo de barramento de campo já havia sido comprovada no âmbito da dissertação de mestrado de Rafael Wild (2000).

As aplicações escolhidas visam testar a flexibilidade do sistema BR-Tool para diferentes tecnologias de barramento de campo. Procurou-se assim avaliar o desempenho do BR-Tool não somente ao trabalhar com barramentos que utilizam diferentes camadas físicas, como também ao operar com protocolos que empregam disciplinas de acesso ao meio distintas (topologia mestre/escravo e produtor/consumidor).

## 8.1 ESTUDOS DE CASO 1 E 2: BARRAMENTO PROFIBUS-DP

A fim de comprovar a funcionalidade da ferramenta BR-Tool foram feitos incialmente dois ensaios sobre redes PROFIBUS-DP reais.

### 8.1.1 Detalhamento do protocolo PROFIBUS-DP

O PROFIBUS é um barramento de campo que emprega a topologia mestre/escravo. Estações mestres são aquelas que podem iniciar uma comunicação no barramento sendo responsáveis pela monitoração do sistema, configuração e troca cíclica de dados com os escravos (sensores/atuadores inteligentes). Uma estação escrava monitora o barramento aguardando uma mensagem cujo endereço de destino seja o seu próprio, quando então deverá processar a informação, e, caso requisitado, deve enviar dados como resposta (PROFIBUS, 1991).

Para a gerência de acesso ao barramento pelos mestres utiliza-se o conceito de passagem de bastão (*token passing*), que possibilita a cada um deles assumir exclusivamente o controle do barramento por um determinado intervalo de tempo. Para a circulação do *token*, cada estação mestre deve conhecer o endereço do seu mestre predecessor, *Previous Station* (PS) e mestre sucessor, *Next Station* (NS) , além do próprio endereço, *This Station* (TS). Os endereços são determinados na inicialização, mas podem ser atualizados dinamicamente, se necessário. A passagem de *token* ocorre ordenadamente do mestre de menor endereço até o de maior endereço, retornando por fim ao primeiro. Por esta característica de passagem circular do *token*, diz-se que em uma rede PROFIBUS é formado um anel lógico entre mestres. Uma vez que o gerenciamento deste anel lógico é de fundamental importância para a operação de uma rede PROFIBUS, o protocolo inclui algoritmos de manutenção do anel lógico mesmo em caso de falhas de estações mestre.

Quando está com o direito de controle do barramento, o mestre pode transferir ou receber informações de outras estações. O tempo que um mestre mantém o *token* é dado por um intervalo pré-definido, chamado tempo de manutenção do *token*, *Token Holding Time* (Tht). Dentro do Tht é computado o tempo necessário pelo mestre para a realização de um

ciclo completo de troca de dados com todos seus escravos mais o tempo de uma mensagem acíclica. A soma do tempo de ciclo de todos os mestres caracteriza o tempo de rotação de *token*, *Token Rotation Time* (Trt).

A faixa de valores entre o endereço de PS e TS para um mestre é chamada de faixa de GAP. Além da troca cíclica de dados com seus escravos, o mestre tem a incumbência de pesquisar a entrada e saída de estações dentro de sua faixa de GAP.

A camada de enlace do PROFIBUS define três serviços acíclicos e um serviço cíclico: *Send Data with Acknowledge* (SDA), *Send and Request Data with Reply* (SRD), *Send Data with No acknowledge* (SDN) e *Cyclic Send and Request Data with Reply* (CSRD). O protocolo PROFIBUS-DP utiliza apenas dois destes serviços: SRD e SDN. O serviço SRD visa o intercâmbio de informações, permitindo ao mestre, em uma mesma mensagem, enviar e requisitar dados para/de um escravo. O serviço SDN é usado para o envio de informações sem confirmação de recebimento, possibilitando ao mestre enviar informações para um grupo de estações ou difundir mensagens globalmente.

Quatro tipos de quadros (*frames*) de mensagens são comumente usados no protocolo PROFIBUS. O quadro tipicamente utilizado pelo protocolo PROFIBUS-DP para troca de dados cíclicos é o quadro de tamanho variável (POPP, 1997). Uma mensagem de tamanho variável consiste de um delimitador de início de mensagem, identificador de tamanho da mensagem, campos com endereços de destino e de origem, byte de controle, campo de dados, byte de conferência do quadro e delimitador de fim de mensagem (ver Figura 26).

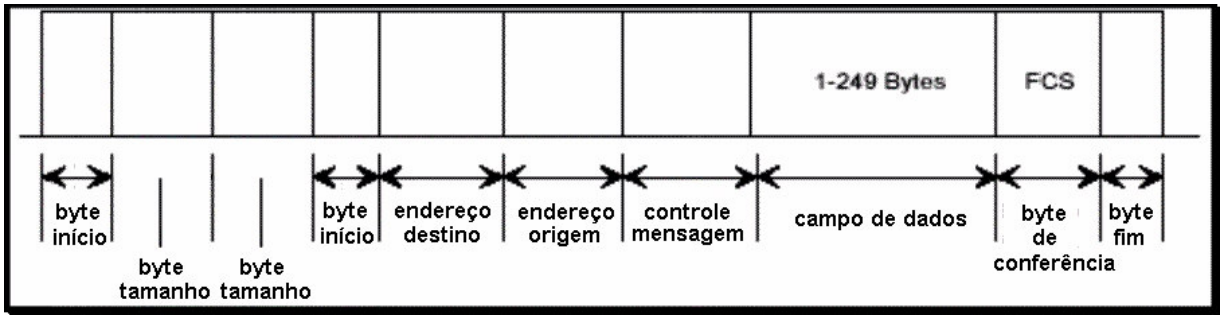



**Figura 26 - Formato de uma mensagem PROFIBUS de tamanho variável**

Uma estação PROFIBUS pode transferir ou receber até 246 bytes a cada vez com mensagens de tamanho variável. Cada estação no barramento possui um identificador próprio com endereços numerados de 0 a 126 (o endereço 127 é usado para mensagens de difusão global).

O PROFIBUS permite ainda o uso de "Pontos de Acesso a Serviços", *Service Access Points* (SAP), para comunicação entre estações. Os SAPs PROFIBUS implementam serviços especiais para programação/configuração da rede. Ao total são possíveis 127 valores de SAP (de 0 a 126). O PROFIBUS-DP padroniza a faixa de valores de SAP de 54 a 62 mais o valor 0 (*default*), permitindo serviços como configuração de escravos, pedido de diagnóstico, congelamento de entradas (modo FREEZE), sincronização na atualização de valores (modo SYNC), entre outros.

### 8.1.2 Estudo de caso 1: aplicação PROFIBUS-DP genérica

Como primeiro estudo de caso foi montada uma rede PROFIBUS-DP composta por uma estação mestre e cinco estações escravas com operação cíclica de leitura e escrita de dados. O dispositivo mestre é um controlador lógico programável da empresa Siemens,

programado em linguagem LADDER. As estações escravas utilizadas são dispositivos desenvolvidos no laboratório de automação industrial da UFRGS. Uma representação desta rede encontra-se ilustrada na Figura 27.

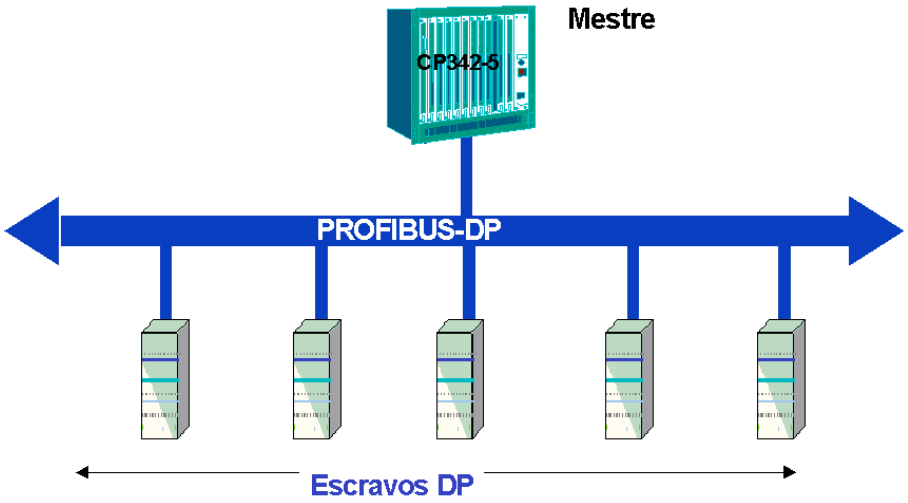



**Figura 27 - Representação da rede PROFIBUS-DP genérica**

A aplicação desenvolvida visa a troca cíclica de dados entre mestre e escravos com uma periodicidade programada de 290 ms. Todas as estações escravas recebem e retornam 10 byte de dados a cada ciclo e operam na taxa de 19200 bps. O tempo de rotação de *token* (Ttr) para a rede foi configurado pela ferramenta de programação da Siemens para 2000 tbits (tempos de bit).

Cinco eventos foram monitorados:

a) Token: mensagem de *token* passada entre mestres;

b) Pesquisa FDL: mensagem de supervisão de estado das estações na rede;

c) Inicio de ciclo: primeira mensagem de um ciclo de troca de dados;

d) Req Dados: mensagem de envio e pedido de dados às estações escravas;

e) Resp Dados: mensagem com a resposta das estações escravas .

As requisições temporais registradas visam a validação de dois parâmetros para a rede: periodicidade da troca de dados cíclica e tempo de rotação de *token*.

O diagrama de Gantt, apresentado na Figura 28 demonstra o comportamento típico de uma rede PROFIBUS-DP, onde o macro ciclo da aplicação apresenta dois intervalos distintos. No primeiro, escravos são varridos para leitura e escrita de dados periódicos. No segundo intervalo ocorrem as mensagens administrativas do protocolo.

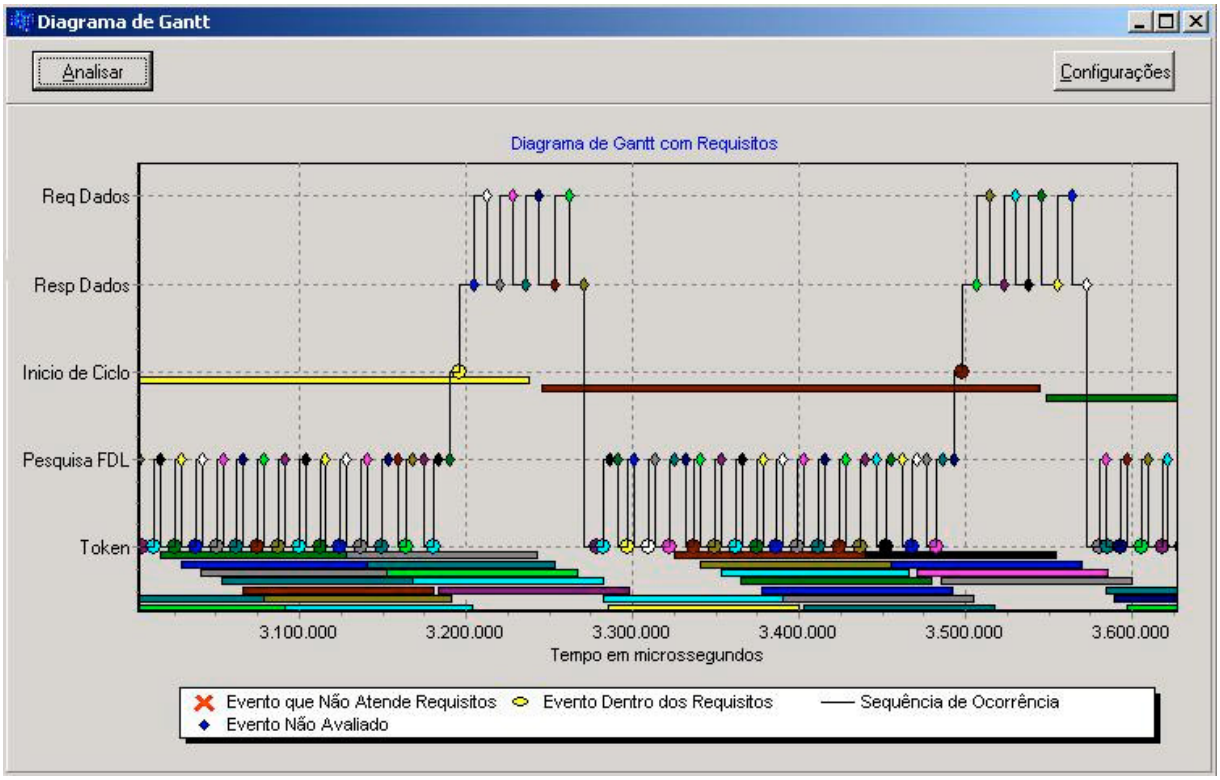



**Figura 28 - Diagrama de Gantt de um trecho da aplicação PROFIBUS-DP desenvolvida**

O período de troca de dados cíclicos entre mestres e escravos começa com o evento "Inicio de ciclo" sendo seguido pelo evento "Resp Dados". Após isto outras quatro sequências de "Req Dados" e "Resp Dados" se seguem. Ao total são 10 mensagens, duas para cada um dos escravos da rede (uma mensagem de escrita e uma de leitura de dados). Conforme especificado, o evento "Inicio de ciclo" indica o começo de um novo ciclo de troca de dados.

O período de mensagens administrativas PROFIBUS é principalmente representado pelos eventos “Token” e “Pesquisa FDL”.

Para validar Ttr foi considerado o intervalo de tempo registrado entre eventos “Token”. O valor de Ttr considerado (2000 tbits) para uma frequência de 19200bps é de 104,166 ms. A apresentação do histograma de mensagens mostra que os intervalos registrados ficam realmente em torno deste valor (Figura 29).

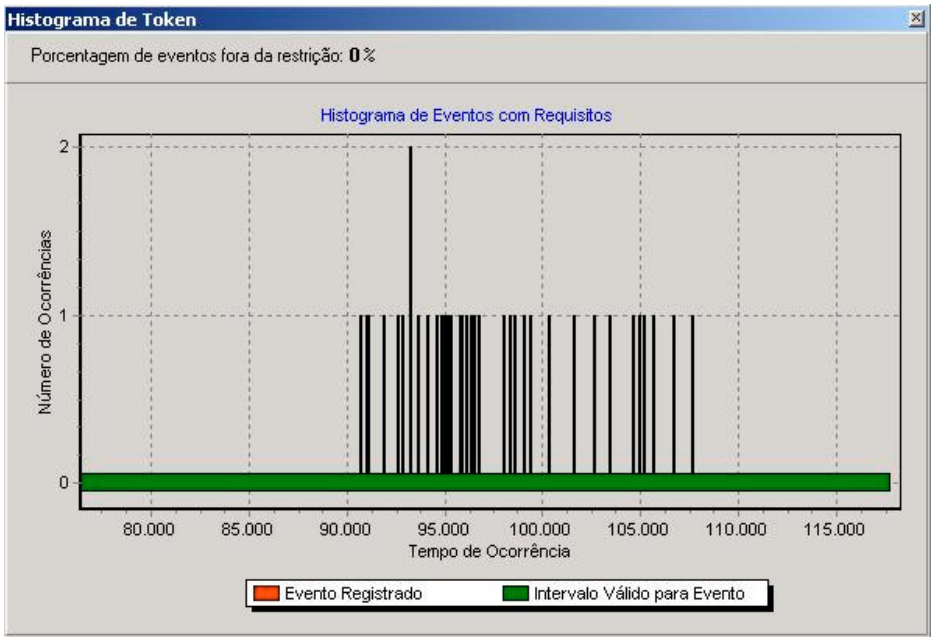



**Figura 29 - Histograma de tempo de rotação de token**

Analogamente mediu-se o tempo entre eventos “Inicio de ciclo” consecutivos para validar a periodicidade da troca de dados. Na Figura 30 observa-se o correto atendimento desta restrição. As periodicidades dos eventos “Inicio de ciclo” estão localizadas dentro do intervalo especificado, espalhados dentro da faixa de 230ms a 270ms (inferiores a 290ms).

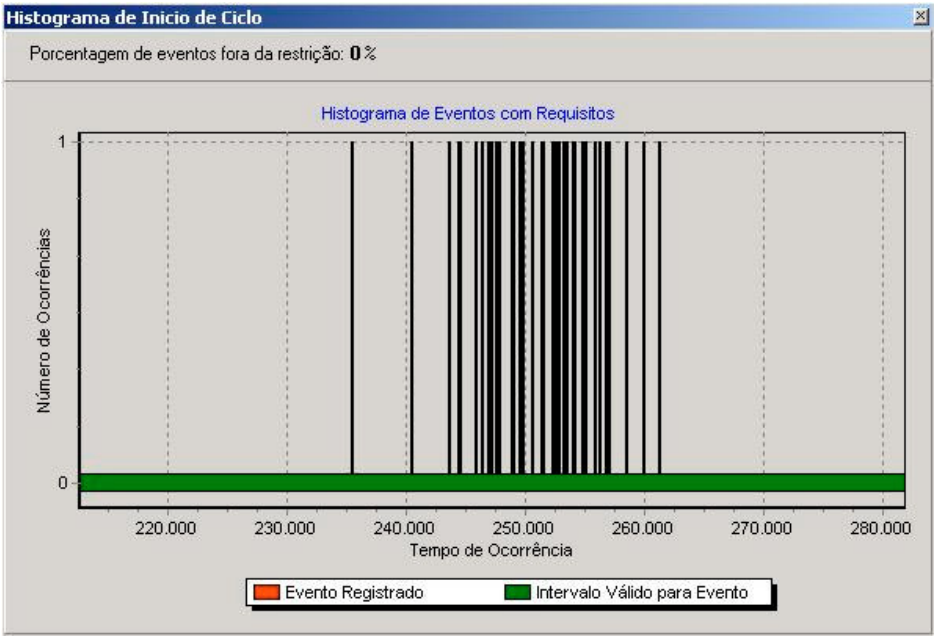



**Figura 30 - Histograma de periodicidade do ciclo de troca de dados**

### 8.1.3 Estudo de caso 2: sistema de manufatura PROFIBUS-DP

O segundo estudo de caso foi ensaiado durante funcionamento real de uma rede PROFIBUS-DP da empresa Tramontina. A aplicação da rede PROFIBUS-DP em questão, é uma linha de produção de mangueiras com diâmetros de 1/2" , 3/4" ou 5/8" de polegadas com capacidade de produção de 40 mil metros por dia. Estão em rede PROFIBUS-DP, oito inversores de freqüência (MICROMASTER4), uma interface gráfica (MP270) e um CLP (CP315-2) da Siemens.

O equipamento possui ao todo 35 metros de comprimento, e consta de duas extrusoras de plástico, dois puxadores, duas espiraladeiras (máquinas destinadas a fazer o trançado na mangueira), um marcador (imprime a marca da empresa na mangueira) e um compressor de ar. Em cada um destes equipamentos existe um inversor de freqüência com interface PROFIBUS-DP.

A interface gráfica é utilizada para monitoração de valores e escrita parâmetros do sistema, sendo possível ainda operar todo o equipamento sem a necessidade da utilização de botoeiras, servindo também como ambiente supervisório.

Para configurar e programar a rede PROFIBUS-DP foi utilizado o *software* STEP 7 da Siemens. Na aplicação, além das operações de liga e desliga nos inversores e troca de valores desejados de velocidade (*setpoints*), pode-se ler valores de tensão e corrente dos equipamentos e monitorar possíveis alarmes, utilizando uma estrutura de seis palavras (12 bytes) para a comunicação entre mestre e escravos. A rede composta por dois mestres (CLP e interface gráfica) e oito escravos opera a uma taxa de comunicação de 187,5kbps.

A Figura 31 traz a representação da rede de manufatura utilizada neste estudo de caso.

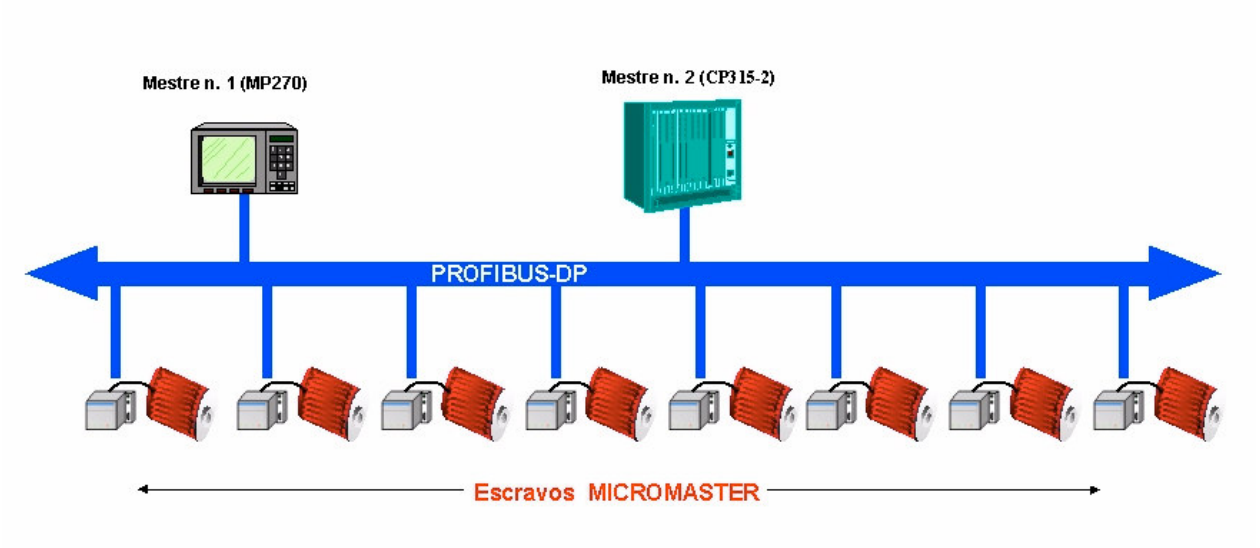



**Figura 31 - Representação da aplicação de manufatura PROFIBUS-DP**

São oito eventos possíveis para esta aplicação:

a) Token: mensagem de *token* passada entre mestres;

b) Pesquisa FDL: mensagem de supervisão de estado das estações na rede;

c) Inicio de Ciclo: primeira mensagem de um ciclo de troca de dados;

d) Req Dados: mensagem de envio e pedido de dados às estações escravas;

e) Resp Dados: mensagem com a resposta das estações escravas;

f) Parametros Mestre1: envio de parâmetros de configuração para o mestre 1;

g) Parametros Mestre2: envio de parâmetros de configuração para o mestre 2;

h) Fim Parametros: finalização do envio de parâmetros para mestres.

As requisições temporais analisadas são referentes aos mesmos parâmetros da aplicação anterior: periodicidade da troca de dados cíclica (especificada como 40 ms) e tempo de rotação de *token* (35 ms).

O diagrama de Gantt, apresentado na Figura 32 mostra também os dois períodos distintos da rede: período de escrita/leitura de escravos e período de mensagens administrativas do protocolo.

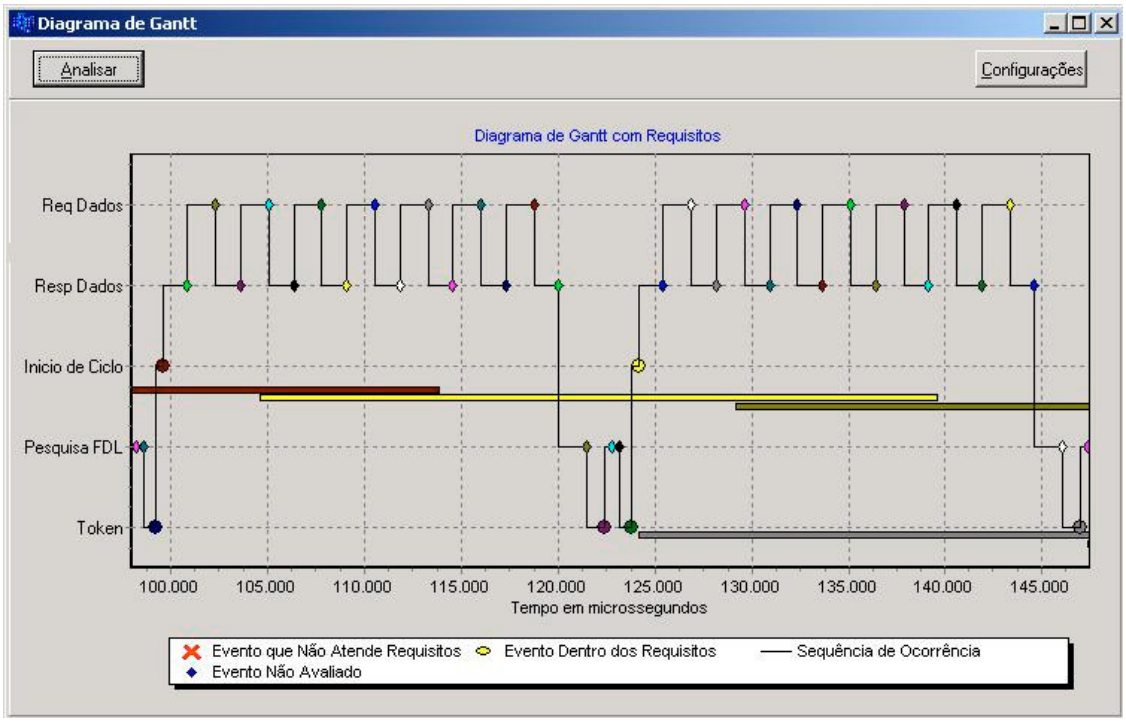



**Figura 32 - Diagrama de Gantt de um trecho da aplicação PROFIBUS-DP desenvolvida**

Nota-se em comparação com o estudo de caso anterior (Figura 28) que esta aplicação, mais complexa, faz uma utilização bem mais efetiva da rede para os eventos de troca de dados. Uma comparação desta diferença é apresentada na Figura 33, a partir dos gráficos de análise comparativa (*chart*) gerados para os dois estudos de caso.

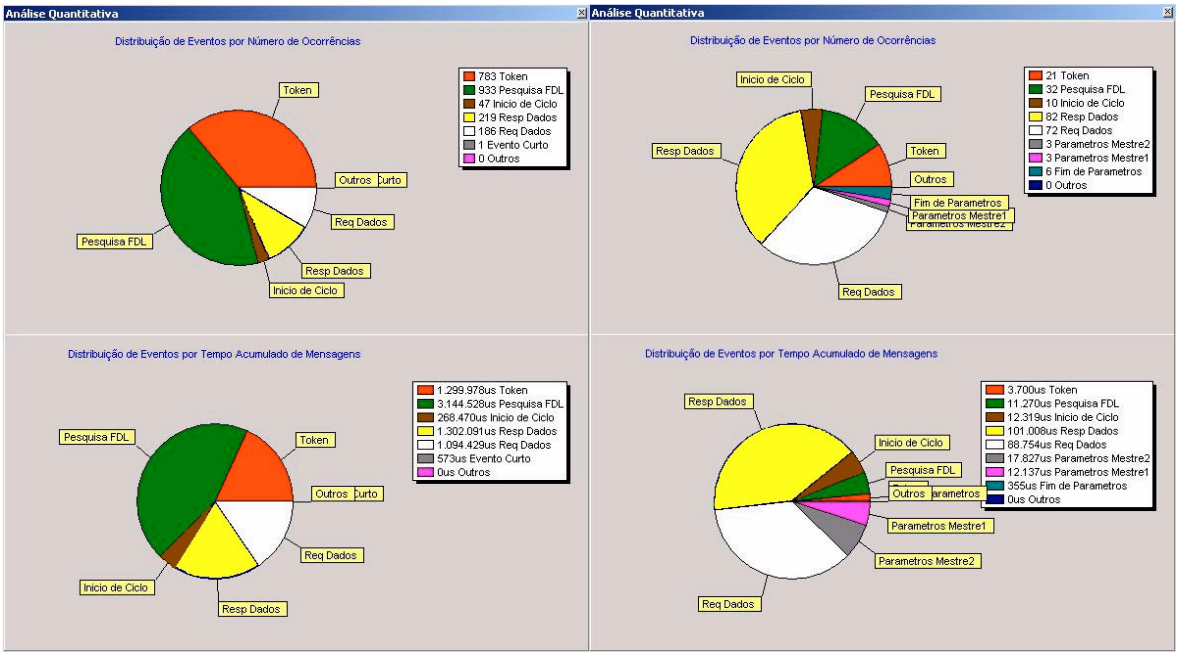



**Figura 33 - Análise quantitativa (*chart*) dos estudo de caso 1 e 2 respectivamente**

Conforme já descrito, o evento "Inicio de Ciclo" indica o começo de um novo ciclo de troca de dados entre a estação e seus escravos. O período de mensagens administrativas PROFIBUS é principalmente representado pela ocorrência dos eventos "Token" e "Pesquisa FDL".

A análise feita pelo BR-Tool demonstra que os resultados registrados de periodicidade de troca de dados e tempo de rotação de *token* do mestre atendem aos requisitos previstos. No histograma de periodicidade de troca de dados, destacam-se entretanto dois casos em que os períodos registrados se aproximam do limite superior do intervalo permitido (Figura 34).

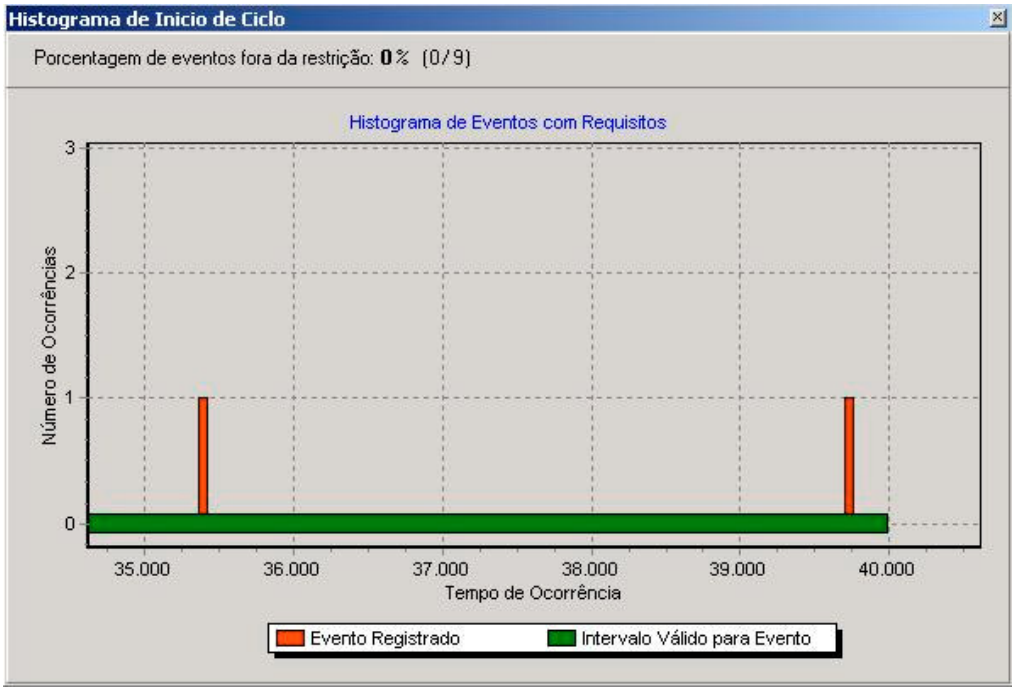



**Figura 34 - Trecho do histograma de periodicidade do ciclo de troca de dados**

Analisando o instante mais crítico (quando o intervalo entre troca de dados atingiu o valor de 39,7 ms), pode-se localizar na tela do diagrama de Gantt o exato instante em que este ocorre (73200 μs).

Pode-se observar graficamente que durante a ocorrência deste ciclo de troca de dados mais atrasado, o primeiro evento do ciclo ("Inicio de Ciclo"), representado por um círculo preto, se encontra no extremo do seu intervalo de validade, representado pela barra preta abaixo deste (Figura 35).

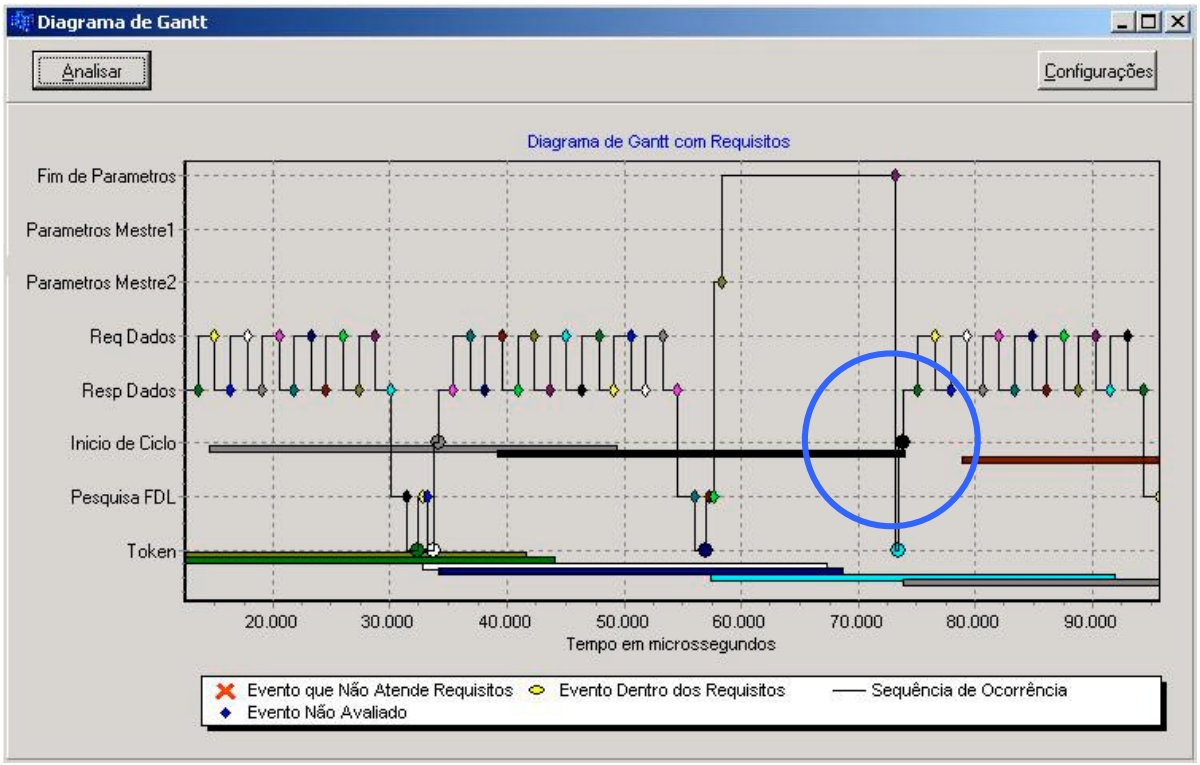



**Figura 35 - Diagrama de Gantt indicando evento de envio de parâmetros ao mestre 2**

A justificativa para este caso extremo se deve à ocorrência de uma mensagem acíclica utilizando serviços de SAP para envio de parâmetros ao mestre 2. O atraso registrado é tão grande, pois a mensagem transmitida contém 244 bytes, quase o limite da norma PROFIBUS que define um campo máximo de 246 bytes para mensagens de tamanho variável.

## 8.2 Estudo de caso 3: barramento CANOpen

A ferramenta BR-Tool foi também utilizada para avaliar o determinismo temporal de uma de rede de automação CANOpen real.

### 8.2.1 Breve descrição do protocolo CANOpen

O protocolo CAN implementa o modelo produtor/consumidor fazendo uso de mensagens do tipo broadcast e multicast. Neste contexto, não se utilizam campos de endereçamento para as mensagens. Um identificador (ID) único informa o tipo e prioridade de cada mensagem. São duas classes de identificadores possíveis: formato padrão, que utiliza um campo de identificação de 11 bits, e formato extendido, com identificadores de 29 bits (TINDELL, 1995).

Uma mensagem CAN consiste de campos de arbitração, controle, dados, CRC e confirmação, conforme Figura 36. O tamanho máximo do pacote de dados é de oito bytes.

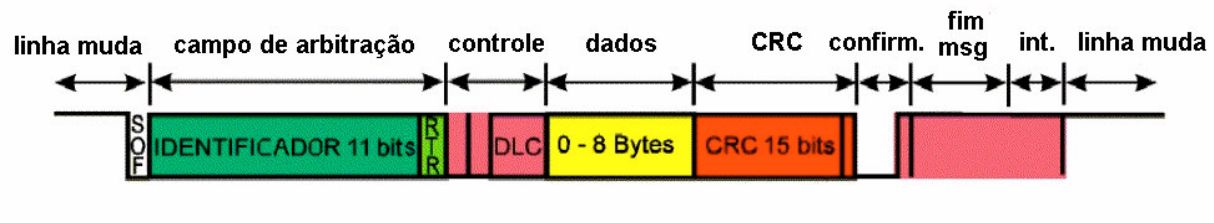



**Figura 36 - Formato de uma mensagem CANOpen padrão**

CANOpen é um protocolo de alto nível que utiliza o barramento CAN como base (BOTERENBROOD, 2000). O protocolo CANOpen define quatro tipos de mensagens padrões: mensagens de administração da rede, mensagens pré-definidas (como sincronização de tempo), mensagens de dados do processo e mensagens de serviço (por exemplo mensagens de confirmação).

Os quatro bits superiores do identificador informam a função da mensagem e os sete bits restantes o identificador da mensagem (prioridade), permitindo assim até 127 identificadores (o identificador 0 indica uma mensagem enviada para todas estações).

Durante o funcionamento de uma rede CANOpen as estações são periodicamente varridas para identificação de seus estados de operação. São quatro os possíveis estados de uma estação:

a) inicialização: no instante que a estação é energizada;

b) pré-operacional: enquanto aguarda configuração;

c) operacional: quando está rodando normalmente;

d) parada: quando desabilitada.

A mensagem de varredura (*Node Guarding*) de estado de um dado nó tem o seguinte formato no barramento:

| | | |
|---|---|---|
| 11 | iii | iiii |

onde: iii.iiii representa o endereço do nó de destino.

A resposta do nó endereçado segue o formato:

| | | | | |
|---|---|---|---|---|
| 111 | 0iii | iiii | ttsss | sssss |

onde: iii.iiii é o endereço do nó

t é um bit alternado (toggle)

sss.ssss é o valor do estado do nó

O processo de comunicação adotado, *Process Data Object* (PDO), visa a comunicação tempo real suportando a transmissão de mensagens síncronas e assíncronas. Uma mensagem de sincronização de alta prioridade (SYNC) é enviada para a rede a cada novo ciclo de dados. As estações com dados de entrada amostram e transmitem suas informações no barramento logo após a deteção desta mensagem. Estações de saída que recebem novos valores mantém estes estocados até a recepção da mensagem SYNC, quando então enviam os valores recebidos para suas saídas.

### 8.2.2 Estudo de caso 3: sistema de controle de transmissão de energia elétrica utilizando CANOpen

A aplicação utilizada neste estudo de caso foi desenvolvida por um pesquisador da empresa Eletrosul para utilização no setor de transmissão de energia elétrica. A aplicação visa a supervisão e controle de tapes de autotransformadores operando em sistemas de alta e extra-alta tensão, permitindo a regulação de tensão elétrica e o controle de fluxo de potência reativa. A mudança de tapes nesta aplicação emprega o esquema de mestre seguidor, ou seja, um elemento da rede será o mestre e os demais procuram manter os tapes na mesma posição do mestre. Fazem parte desta aplicação quatro tipos possíveis de nós: configurador, mestre, seguidor e individual.

O configurador é responsável, entre outras tarefas, por supervisionar o controle de nós, utilizando o esquema de *Node Guarding*. Outra função importante é a geração de pulsos de sincronismo para toda a rede.

Quando o configurador necessita realizar uma mudança de tapes, este gera uma mensagem com objeto de comunicação, *Communication OBject* (COB), específico e a envia para o barramento CANOpen. O nó que desempenha a função de mestre dos bancos, por sua vez, é um consumidor dessa mensagem. Após receber um COB deste tipo, o mestre comanda os seus tapes e, após obter sucesso, informa na rede a nova posição dos seus tapes através do envio de um COB com identificador próprio. Os bancos que estiverem como seguidores são consumidores do COB enviado pelo mestre. Recebendo esse COB, os dispositivos seguidores da rede acompanharão o mestre, colocando os seus tapes na mesma posição.

Na inicialização, todo nó deve fazer a configuração de seu controlador de comunicação, inicializar o dicionário de objetos com valores padrões e enviar um frame (COB) conhecido como *Bootup*, que consiste de um identificador de objetos de comunicação

(COB-ID) com valor 00h e dois bytes de dados, sendo que um dos bytes indica o número do nó, conhecido como NodeID, que pode variar de 1 a 126, e o outro byte indica o estado (que no caso de uma inicialização deve ser pré-operacional). O nó configurador guarda essa informação no seu dicionário de objetos e envia comandos através de objetos de comunicação, colocando o nó em estado operacional, caso ele esteja em modo automático.

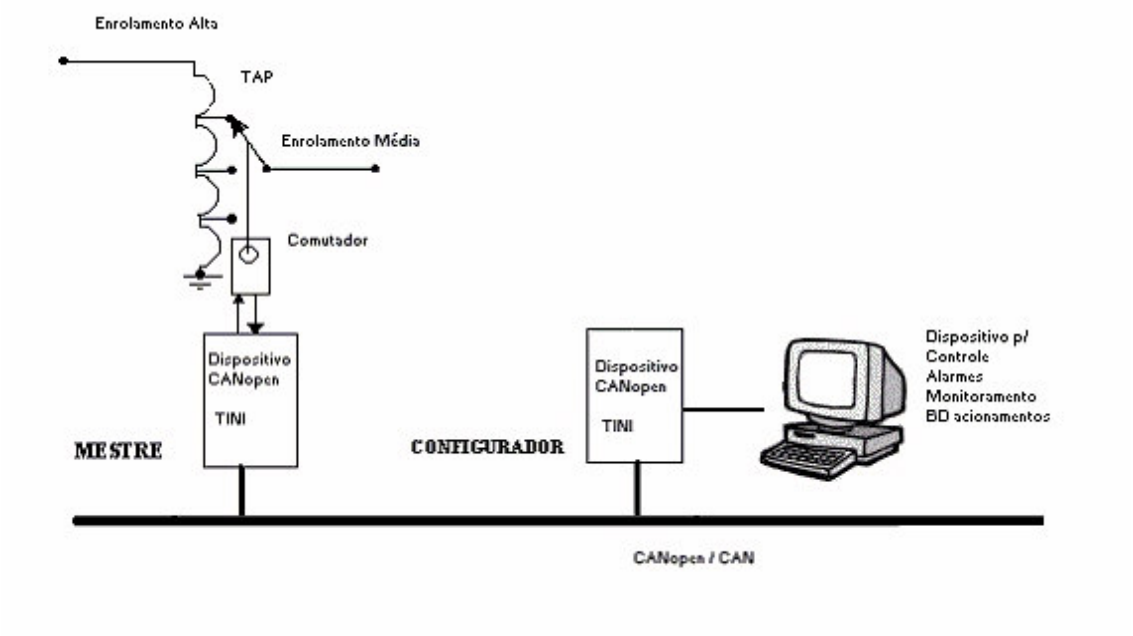



**Figura 37 - Representação da aplicação CANOpen**

Para os ensaios realizados neste estudo de caso (Figura 37) somente dois elementos da rede foram utilizados, (configurador e mestre):

a) nó configurador: Possui as seguintes funções: Produzir o objeto de sincronismo, com identificador COB-ID 128; Supervisionar em memória para cada nó: estado operacional, posição dos tapes do nó, tipo do nó, existência de bloqueio, existência de mestre na rede, identificador do nó, erros do nós, etc; Produzir o objeto módulo de controle, para mudar o estado dos nós com identificador de objeto de comunicação COB-ID 00;

b) nó mestre: Possui as seguintes funções: Consumir o objeto de sincronismo, com identificador de objeto de comunicação COB-ID 128. Consumir o objeto RAISE (COB-ID 385); Consumir o objeto LOWER (COB-ID 386); Produzir o objeto posição de tapes (fase A,B,C) com COB-ID 641 (esse objeto é disparado após o nó mudar de tape ou após o nó receber um número de objetos SYNC equivalente a seu NodeID); Produzir o objeto estado pré-operacional, com o objetivo de avisar a rede sobre estado pré-operacional do nó, com identificador de objeto de comunicação COB-ID 00.

Dentre as mensagens possíveis neste barramento destacam-se:

a) SYNC - COB-ID 80 hexa;
b) Node Guarding - COB-ID 701;
c) Raise - Subir tape COB-ID 181;
d) Lower - Descer tape COB-ID 182;
e) Tape position – Informar posição ao mestre - COB-ID 281, pos1 e pos2.

As especificações usadas para esta aplicação procuram validar dois COB's (SYNC e Node guarding). As mensagens de Node Guarding (usadas para supervisionar os nós) devem apresentar ciclicidade de 10 segundos ± 20%. Já as mensagens de SYNC (usadas para sincronizar a troca de dados) devem ocorrer ciclicamente a cada 4 segundos ± 20%. Na Figura 38 são apresentadas as restrições especificadas para a ferramenta de validação.

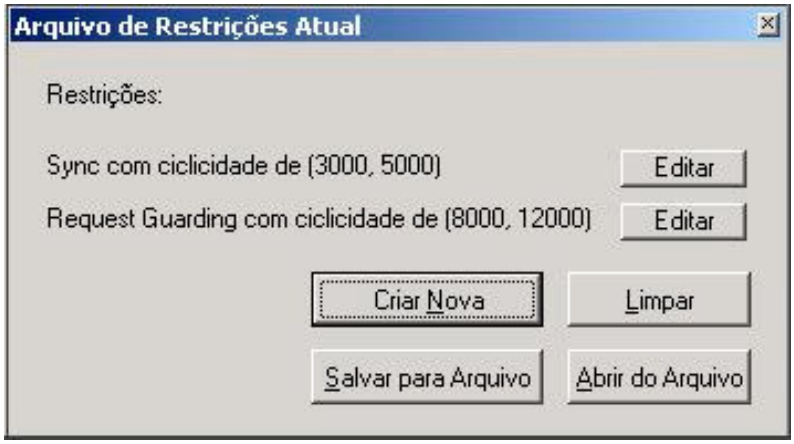



**Figura 38 - Restrições da aplicação CANOpen**

Inicialmente foram feitos alguns ensaios em laboratório implementando-se a aplicação sobre uma rede simples.

Os resultados obtidos com este ensaio mostravam a falha no atendimento às restrições temporais definidas (o que pode ser observado na Figura 39). Para as mensagens de sincronismo (SYNC), a porcentagem de mensagens que não atendem os requisitos chegam a 6,67%.

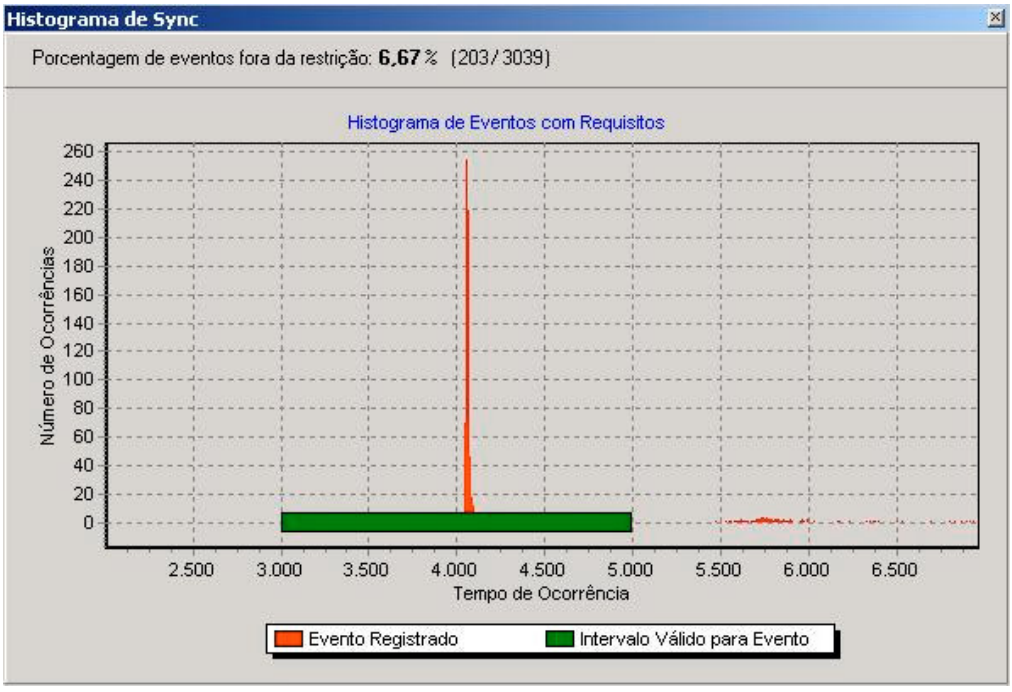



**Figura 39 - Histograma de mensagens de sincronismo (SYNC)**

Para as mensagens de supervisão a performance registrada é ainda pior, com 10,68% das mensagens não atendendo aos requisitos. Mesmo as mensagens que atendem se encontram muito próximas do limite superior especificado, conforme pode ser evidenciado na Figura 40.

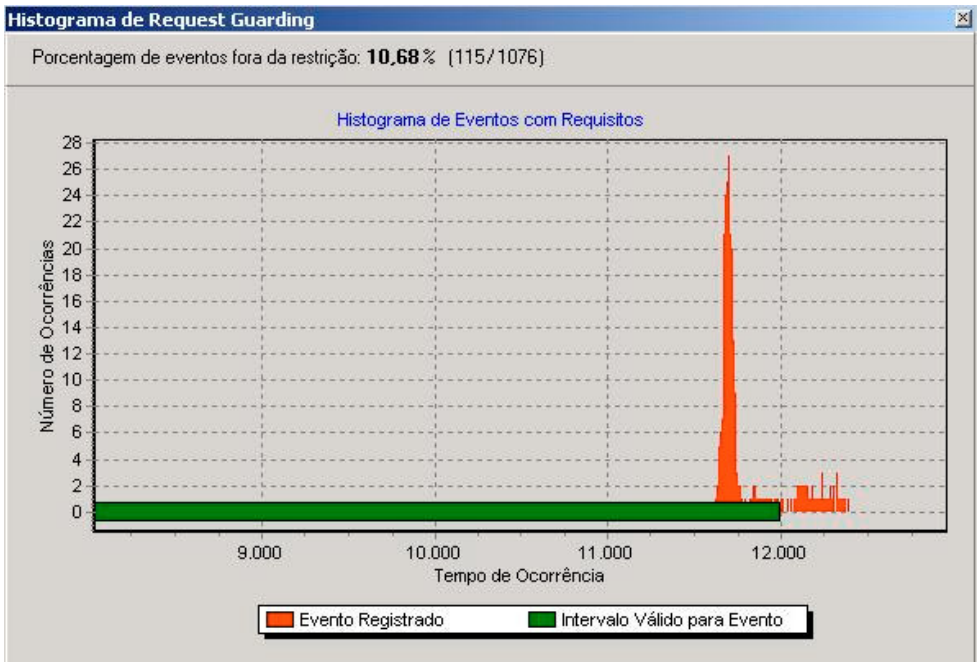



**Figura 40 - Histograma de mensagens de supervisão (Node Guarding)**

Esta rede serviu como objeto de estudo de uma dissertação de mestrado no Programa de Pós-Graduação em Computação (PPGC) da UFRGS (BENEDETTI, 2003), onde diversas alterações foram implementadas nos códigos fonte da aplicação de controle das placas TINI, principalmente procurando garantir as latências dos objetos de comunicação(COBs).

Para analisar o resultado destas alterações implementadas pelo trabalho, as mesmas medidas foram refeitas com a versão atualizada da aplicação rodando dentro de uma subestação de força. Os resultados obtidos indicam o atendimento dos requisitos com uma precisão bastante grande (aproximadamente dentro de ± 2% de desvio). Na Figura 41 tem-se

os valores medidos de ciclicidade de serviços de sincronismo e na Figura 42 a periodicidade registrada para os serviços de supervisão.

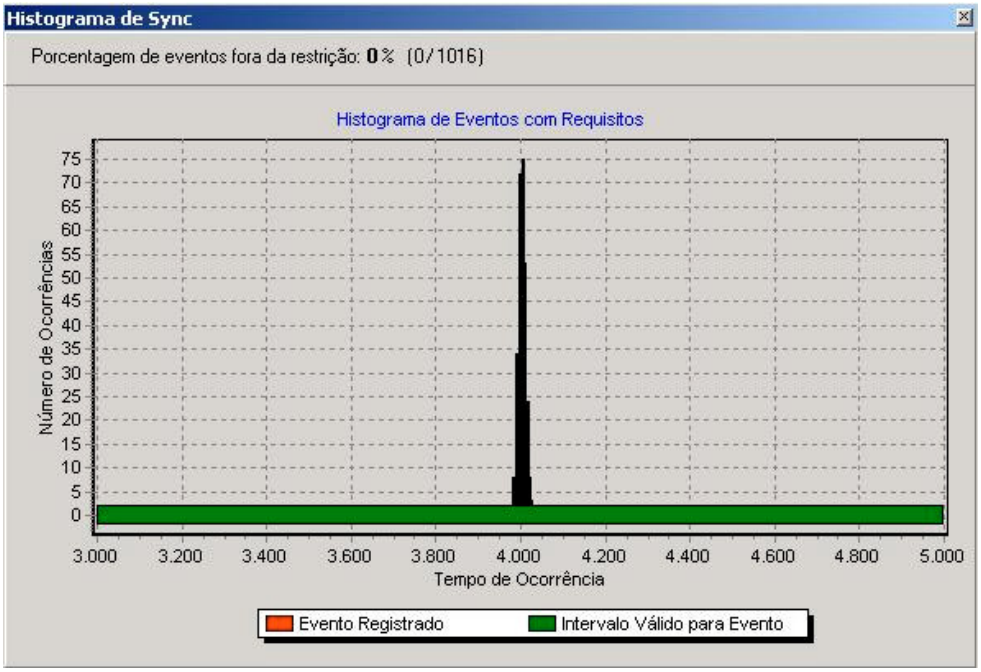



**Figura 41 - Histograma atualizado de mensagens de sincronismo (SYNC)**

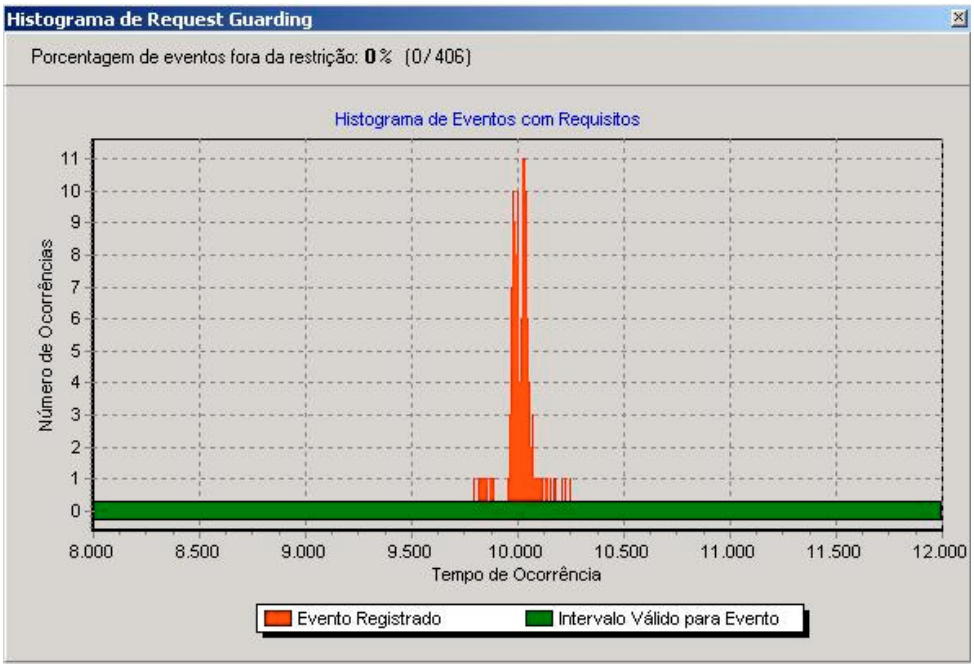



**Figura 42 - Histograma atualizado de mensagens de supervisão (Node Guarding)**

# 9 CONCLUSÕES E TRABALHOS FUTUROS

A presente dissertação abordou a especificação, o projeto, a implementação e a verificação de um sistema de avaliação temporal para redes de barramento de campos. O trabalho abordou tanto modernos conceitos de *hardware* (uso de FPGAs e de interfaces para protocolos de comunicação industrial recentemente padronizados pela IEC), bem como algoritmos de validação e visualização de requisitos temporais. Apresentou-se uma verificação dos aspectos temporais e elétricos do projeto desenvolvido e adicionalmente validou-se o sistema desenvolvido através da realização de três estudos de caso. Com isto, considera-se que o trabalho desenvolvido, além de apresentar comprovada aplicação para o setor industrial, incorpora discussões e desenvolve conceitos de relevância científica e acadêmica.

A ferramenta desenvolvida permite não somente analisar o atendimento ou não dos requisitos temporais de uma dada aplicação, como também mensurar o quão próximo esta se encontra de seus limites de validade, permitindo identificar gargalos do projeto e possíveis pontos de falha (HUSEMANN; PEREIRA; SCHMIDT, 2002).

Além disso, a ferramenta é útil para auxílio na decisão de escolha do protocolo de comunicação mais apropriado para uma determinada aplicação. Pode-se, por exemplo, comparar os desempenhos obtidos por uma mesma aplicação desenvolvida para diferentes barramentos de campo. Em função de poder-se utilizar a mesma ferramenta para estas análises e portanto com os mesmos parâmetros de medida, a comparação entre os resultados é facilitada podendo identificar realmente qual solução melhor se adequa para a aplicação pretendida.

Os estudos de casos apresentados ilustram a aplicabilidade do sistema BR-Tool para lidar com diferentes tecnologias de barramentos de campo. A deteção de irregularidades no

atendimento de restrições temporais como as obtidas nas primeiras análises da aplicação CANOpen comprovam a importância da ferramenta desenvolvida para auxílio a desenvolvedores.

Dentre as possíveis atividades futuras e que visam a dar continuidade a este trabalho destaca-se a realização de ensaios sobre o meio físico IEC1158-2, principalmente, procurando-se validar uma rede de dispositivos PROFIBUS-PA, ainda não testados com esta ferramenta (trabalhos com o barramento de campo Foundation Fieldbus já foram desenvolvido por (WILD, 2000). Outros barramentos importantes como WorldFIP, P-NET e Interbus-S poderiam também ser incorporados e validados.

Entre sugestões de aprimoramento para a ferramenta BR-Tool sugere-se a implementação de um recurso de "gatilho" que possibilite a seleção de eventos importantes que poderiam iniciar um processo de aquisição. Assim a placa BUSMON poderia ser deixada em processo de aquisição contínua, recebendo e transferindo eventos para o *software* de validação no PC, o qual manteria os dados adquiridos em um buffer circular temporário.

Por ser aplicável a sistemas distribuídos a ferramenta BR-Tool poderia ser utilizada para monitoração simultânea de diferentes pontos da rede. Os dados obtidos nos diferentes pontos poderiam ser utilizados para determinação do comportamento global de soluções que utilizam mais de uma rede (utilizando o mesmo tipo de barramento ou não). Para a realização deste levantamento global de forma consistente entretanto é necessário que as informações temporais de cada unidade de monitoração estejam sincronizadas. Para a sincronização entre diferentes placas BUSMON algumas opções podem ser consideradas: a) a utilização de uma rede proprietária entre as placas, implementada através de uma das interfaces disponíveis ou pelo barramento de expansão da placa; b) soluções que utilizem um sistema global de referência por GPS para difusão de tempos entre sistemas isolados, similar ao proposto em (VERISSIMO; RODRIGUES; CASIMIRO, 1997), ou c) uso de algoritmos de sincronização

de relógio de alta precisão através do envio de mensagens via redes de comunicação padronizadas, como por exemplo, (SCHMID; HOURAUER; KERO, 2000), que descreve uma proposta de sincronização que usa rede Ethernet.

Outra proposta de aprimoramento da ferramenta se daria no emprego desta para operações com sistemas operacionais tempo real, de forma a garantir que os tempos de leitura da placa possam ser feitos de forma periódica. Entre as possíveis soluções pode-se destacar a utilização de um pacote comercial como o WinRT, que provê suporte tempo real em ambientes Windows NT e 2000 (LEE; MAVROIDIS, 2000), ou a adaptação dos *software*s desenvolvidos para operação com sistemas operacionais tempo real como RT Linux e QNX.

# REFERÊNCIAS


AL-ANBUKY, A. H.; DANIAL, K. H. A Flexible Data Acquisition and Monitoring System. **Desalination.** Amsterdam: Elsevier Science Publishers B. V., n. 92, p. 271-280, 1993.

ALVES, M. J. A. F. **Real-Time Communications over Hybrid Wired/Wireless PROFIBUS-Based Networks**, 2003, 185 f. Tese (Doutorado) – Faculdade de Engenharia, Universidade do Porto, Portugal, 2003.

BAKER, A. **Windows NT Device Drivers Book:** a guide for programmers**.** Nova Jersei: Prentice Hall PTR, 1996. 544p.

BENEDETTI, M. **CANOpen: aplicado a aquisição e controle de processos distribuídos em tempo real no setor elétrico**, 2003, 101 f. Dissertação (Mestrado) – Programa de Pós-Graduação em Ciências da Computação, Universidade Federal do Rio Grande do Sul, 2003.

BEUS-DUKIC, L.; WELLINGS, A. J. Scheduling Time Constrained Messages on MiniMAP. In: WORKSHOP ON REAL-TIME SYSTEMS, 1991, Paris-Orsay, França. **Proceedings …** Paris, França: IEEE, jun. 1991. p. 106-110.

BOTERENBROOD, H. **CANOpen:** high-level protocol for CAN-bus, Nikhef, Amsterdam, 2000, 23 p. Disponível em: < http://www.nikhef.nl/pub/departments/ct/po/doc/ CANopen20.pdf>. Acesso em: 10 jan. 2003.

CATANIA, V. et al. **Computer Communications.** Norwell: Elsevier Science Publishers B. V. 1996, v. 19, p. 788-803, mar. 1996.

CHODROW, S. E.; JAHANIAN, F.; DONNER M. Run-Time Monitoring of Real-Time Systems, In: IEEE REAL-TIME SYSTEMS SYMPOSIUM, 1991**,** Texas. **Proceedings …** San Antonio, Texas: IEEE, dec. 1991. p. 74-83.

DRIVER DEVELOPMENT KIT FOR WINDOWS 2000 (DDK). Microsoft Corporation, 2000. Disponível em: <http://support.microsoft.com/support/ddk/Win2000ddk/>. Acesso em: 10 mar. 2002.

HABAN, D.; WYBRANIETZ D. A Hybrid Monitor for Behavior and Performance Analysis of Distributed Systems. California. **IEEE Transactions on Software Engineering**. v. 16, n. 2, p. 197-211, fev. 1990.

HMS. **AnyBus-S**: catalog, HMS Industrial Networks Incorporation, 2002, 4 p. Disponível em: <http://209.15.171.122/hms/downloads/Shortforms/ABSBrochere.PDF>. Acesso em: 15 ago. 2001.

HONG, H. H.; KO S. J. Analysis of Real-Time Data Transmission in the DLL of IEC/ISA Fieldbus. In: INTERNATIONAL SYMPOSIUM ON INDUSTRIAL ELECTRONICS, 1998**,** Pretoria, África do Sul. **Proceedings …** Pretória, África do Sul: IEEE, jul. 1998, p. 694-699.

HONG, S. H. Experimental Performance Evaluation of Profibus-FMS. **IEEE Robotics and Automation Magazine**, New York, N.Y., v. 7, n. 4, p. 64 -72, dec. 2000.

HUSEMANN R.; PEREIRA C. E.; SCHMIDT R. L. Sistema Monitorador para Aplicações Baseadas em Comunicação por Barramentos Industriais. In: CONGRESSO BRASILEIRO DE AUTOMÁTICA, 14., 2002, Natal, RN. **Anais …** Natal: UFRN, 2002a, p. 2780-2785.

INSTITUTE OF ELECTRICAL AND ELETRONICS ENGINEERS (IEEE): **Std 610.12-1990:** IEEE standard glossary of software engineering terminology. California: IEEE, 1990.

INTERNATIONAL ELECTROTECHNICAL COMMISION (IEC). **IEC61158:** digital data communication for measurement and control. 3. ed. Geneva: IEC, 2001.

INTERNATIONAL STARDARDIZATION FOR ORGANIZATION (ISO). **ISO11898-2**: road vehicles - interchange of digital information - Part 2: High Speed Medium Access Unit and Medium Dependant Interface. 3. ed. Geneva: ISO, sept. 1999.

JAHANIAN, F.; MOK, A. K. Safety of Timing Properties in Real-Time Systems. **IEEE Transactions on Software Engineering.** California, v. 12, n. 9, p. 890-904, sept. 1986.

JAHANIAN, F.; RAJKUMAR, R.; RAJU, S. C. V. Runtime Monitoring of Timing Constraints in Distributed Real-Time Systems. **Real-Time Systems.** Norwell, EUA: Kluwer Academic Publishers, v. 7, n. 3, p. 247-273, nov. 1994.

JOHNK E.; DIETMAYER K. Determination of Bit Timing Parameters for the CAN Controller SJA 1000. Alemanha: Philips Electronics, 1997, 26 p. Disponível em: <http://www-eu3.semiconductors.com/acrobat/applicationnotes/AN97046.pdf>. Acesso em: 10 out. 2002.

JUNDI, K.; MOON, D. F. Real-Time Monitoring for Software Development and Testing. In: NATIONAL AEROSPACE AND ELECTRONICS CONFERENCE, 1992. **Proceedings …** Dayton, USA: IEEE, v. 2, 1992. p. 572-580.

KAISER, J.; LIVANI, A. M.; JIA, W. Scheduling Hard and Soft Real-Time Communication in a Controller Area Network. **Journal of Control Engineering Practice.** Norwell: Elsevier Science Ltd., n. 7, p. 1515-1523, IEEE, dec. 1999.

KOPETZ, H. **Real Time Systems.** Massachusetts, EUA: Kluwer Academic Publishers, 1997. 338p., ISBN 0-7923-9894-7.

KUUSILINNA, K.; HAMALAINEN, T.; SAARINEN, J. Field Programmable Gate Array-Based PCI Interface for a Coprocessor System. **Microprocessors and Microsystems.** Norwell: Elsevier Science Ltd., n. 22, p. 373-388, jan. 1999.

LEE, C. J.; MAVROIDIS, C. WinReC V.1: real-time control software for windows NT and its applications. In: AMERICAN CONTROL CONFERENCE (AOC'00), 2000, Chicago: **Proceedings …** Chicago, IEEE, jun. 2000. p. 651-655.

LIU, G.; MOK, A. K. Efficient Run-Time Monitoring of Timing Constraints. In: REAL-TIME TECHNOLOGY AND APPLICATIONS SYMPOSIUM, 3., 1997**,** Montreal. **Proceedings …** Montreal, Canadá: IEEE, jun. 1997.

LIU, J. W. S.; HÁ, R. Methods for Validating Real-Time Constraints. **Journal of Systems Software.** Nova Iorque: Elsevier Science Ltd., n. 30, p. 85-98. 1995

MARINO, P. et. al. Link Level Formal Specification for Industrial Communication Networks. In: ANNUAL CONFERENCE OF THE IEEE INDUSTRIAL ELECTRONICS SOCIETY, 24., 1998**,** Alemanha. **Proceedings…** Aachen, Alemanha: IEEE, v. 2, sept. 1998. p. 226 -231.

MENDONÇA, A.; ZELENOVSKY, R. **PC:** um guia prático de hardware e interfaceamento. 3. ed. Rio de Janeiro: MZ Editora, abr. 2002. 1032 p. ISBN 85-87385-09-7.

MINK, A. et. al. Performance Measurement using Low Perturbation and High Precision Hardware Assists. In: REAL-TIME SYSTEMS SYMPOSIUM, 3., 1998**,** Madri. **Proceedings …** Madri, Espanha: IEEE, dec. 1998, p. 379-388.

PCI Local Bus Specification Revision 2.1. Portland, Estados Unidos: PCI Special Interest Group, IEEE: jun. 1995. 281 p.

PEREIRA, C. E.; WILD, R. Tool for Validating Timing Requirements of Industrial Application based on the Foundation Fieldbus Protocol. In: WORKSHOP ON REAL-TIME PROGRAMMING. 24., 1999, Schloss Dagstuhl, Germany. **Proceedings …** Schloss Dagstuhl, Germany: 1999. p. 9-14. Disponível em: <http://www.fernuni-hagen.de/IT/ wtrp99/ papers/paper-041.pdf>. Acesso em: 5 mar. 2001.

PEREIRA, C. E. Temporal Reasoning on Object-Oriented Real-Time Specifications by using Constraint Propagation Techniques. In: WORKSHOP ON REAL TIME PROGRAMMING, 20., 1995, Estados Unidos. **Proceedings …** Estados Unidos, IFAC/IFIP, nov. 1995.

POPP, M. **The Rapid Way to PROFIBUS-DP**. Germany: PROFIBUS Nutzerorganisation e.V., 1997. 120 p.

PROFIBUS. **DIN 19245:** PROFIBUS Standard, Alemanha: PROFIBUS Nutzerorganization (PNO). IEEE, apr. 1991. 145 p.

SCHICKHUBER, G.; MCCARTHY, O. Distributed Fieldbus and Control Network Systems. **IEEE Computing and Control Engineering Journal.** Geneva. n. 8, v. 1, p. 21-32. 1997.

SCHMID, U.; HORAUER, M.; KERO, N. A Network Interface for Highly Accurate Clock Synchronization. In: AUSTROCHIP, 2000. Áustria. **Proceedings …** Áustria, p. 93-101. IEEE, jun. 2000.

SHANLEY, T.; ANDERSON, D. **PCI System Architecture**. 4. ed. New Jersey: Addison-Wesley, 1999. 787 p. ISBN 0-201-30974-2.

SHIN, K. G.; CHOU C. C. Design and Evaluation of Real-Time Communication for FieldBus-based Manufacturing Systems, In: CONFERENCE LOCAL COMPUTER NETWORKS, 17., 1992**,** Minneapolis**. Proceedings …** Minneapolis. IEEE, sept. 2001. p. 483-492.

SHOBAKI, M. E.; LINDH, L. A Hardware and Software Monitor for Monitor for High-Level System-on-Chip Verification. In: INTERNATIONAL SYMPOSIUM ON QUALITY ELECTRONIC DESIGN, 2001**,** San Jose. **Proceedings …** San Jose, Califórnia: IEEE Computer Society, 2001. Disponível em: < http://www.mrtc.mdh.se/publications/0378.pdf>. Acesso em: 10 jan. 2003.

SIEMENS. **Profibus Monitor Tutorial**. Siemens Energy & Automation Incorporation, 1998. 13 p. Disponível em: <http://www.sea.siemens.com/profibu/docs/monitr.ppt>. Acesso em: 10 abr. 2002.

STEWART, D. A.; GENTLEMAN, W. M. Non-Stop Monitoring and Debugging on Shared-Memory Multiprocessors. In: INTERNATIONAL WORKSHOP ON SOFTWARE ENGINEERING FOR PARALLEL AND DISTRIBUTED SYSTEMS, 2., 1997, Boston. **Proceedings …** Boston: IEEE Computer Society, IEEE, may 1997, p. 263-269.

TANENBAUM, A. **Computer Networks**. Englewood Cliffs, New Jersei: Prentice-Hall, 1989. 658 p., ISBN 0-13-162959-X.

THANE, H. Design for Deterministic Monitoring of Distributed Real-Time Systems, Technical Report, Mälardalen Real-Time Research Centre., Mälardalen University, Suécia, IEEE, sept. 2000. 32 p. Disponível em: < http://citeseer.nj.nec.com/rd/ 13657653 Download/http://citeseer.nj.nec.com/cache/papers/cs/26645/http:zSzzSzwww.mrtc.mdh.sezSz publicationszSz0242.pdf/thane00monitoring.pdf >. Acesso em: 15 jan. 2002.

THOMAS, F. et al. A Hardware/Software Codesign for Improved Data Acquisition in a Processor-Based Embedded System. **Microprocessors and Microsystems.** Norwell: IEEE Elsevier Science Ltd., n. 24, p. 129-134. Jun. 2000.

TINDELL, K.; BURNS A. Run-Time Monitoring of Real-Time Systems, In: INTERNATIONAL CAN CONFERENCE, 1., 1994**,** Mainz. **Proceedings ...** Mainz, Alemanha: IEEE, sept. 1994. p. 1-11.

TMG **Profibus PA Device Test Tool**: product catalog, TMG i-tec GmbH, 2002. 7 p. Disponível em: <www.tmgitec.de/englisch/Our_Offer/Products/Product_catalog/ PA_Device_Test_Tool_e.pdf>. Acesso em: 10 ago. 2002.

TOKURA, H.; KOTERA, M.; MERCER C. M. A Real-Time Monitor for a Distributed Real-Time Operating System. In: WORKSHOP PARALLEL AND DISTRIBUTED DEBUGGING, 1988**,** Madison. **Proceedings …** Madison: [S.n], 1988, p. 68-77.

TOVAR, E.; VASQUES, F.; BURNS, A. Communication Response Time in P-NET Networks: worst-case analysis considering the actual token utilization, submetido para **Real-Time Systems**. Kluwer Academic Publishers, Norwell, EUA, 1999. Disponível em: <http://www.fe.up.pt/~vasques/research/publi_99/jrts99.pdf>. Acesso em: 5 abr. 2002.

TOVAR, E. et. al. Evaluating Worst-Case Response Time in Mono and Multi-Master Profibus-DP Networks, In: IEEE INTERNATIONAL WORKSHOP ON FACTORY COMMUNICATION SYSTEMS, 4., 2002, Vasteras. **Proceedings …** Vasteras, Suécia: IEEE, oct. 2002, p. 233-240. Disponível em: < http://www.fe.up.pt/~vasques/research/ publi_02/wfcs2002.pdf>. Acesso em: 10 set. 2002.

TSAI, J. J. P.; FANG, K., CHEN H. A Noninvasive Architecture to Monitor Real-Time Distributed Systems. **Computer**. California. v. 23, n. 3, p. 11-23. IEEE, mar. 1990.

TSAI, J. J. P.; YANG, S. J. H. **Monitoring and Debugging of Distributed Real-Time Systems.** Los Alamitos, California: IEEE Computer Society Press, 1995. 430 p., ISBN 0-8186-6537-6.

ULLOA, G. et al. Characterizing Temporal Consistency in a Fieldbus Environment. In: SINGAPORE INTERNATIONAL CONFERENCE ON INTELLIGENT CONTROL AND IINSTRUMENTATION (SICICI), 1992, Singapura. **Proceedings …** Singapura: SICICI, 1992, p. 766-771.

UNIVERSAL SERIAL BUS (USB) **Universal Serial Bus Revision 1.1 Specification**. Portland, Estados Unidos: USB Organization, IEEE, sept. 1998. 327 p. Disponível em: <http://www.usb.org/ developers/docs/usbspec.zip>. Acesso em: 10 abr. 2001.

UNIVERSAL SERIAL BUS (USB) **Universal Serial Bus Revision 2.0 Specification**. Portland, Estados Unidos: USB Organization, IEEE, apr. 2000. 650 p. Disponível em: <http://www.usb.org/ developers/docs/usb_20.zip >. Acesso em: 20 abr. 2001.

VECTOR **CANalyzer Manual** Alemanha: Vector Informatik GmbH, 1999. 169 p. Disponível em: <ftp.vector-informatik.de/pub/support/manuals/CANalyzerV4.0_ Manual_EN.pdf>. Acesso em: 10 nov. 2002.

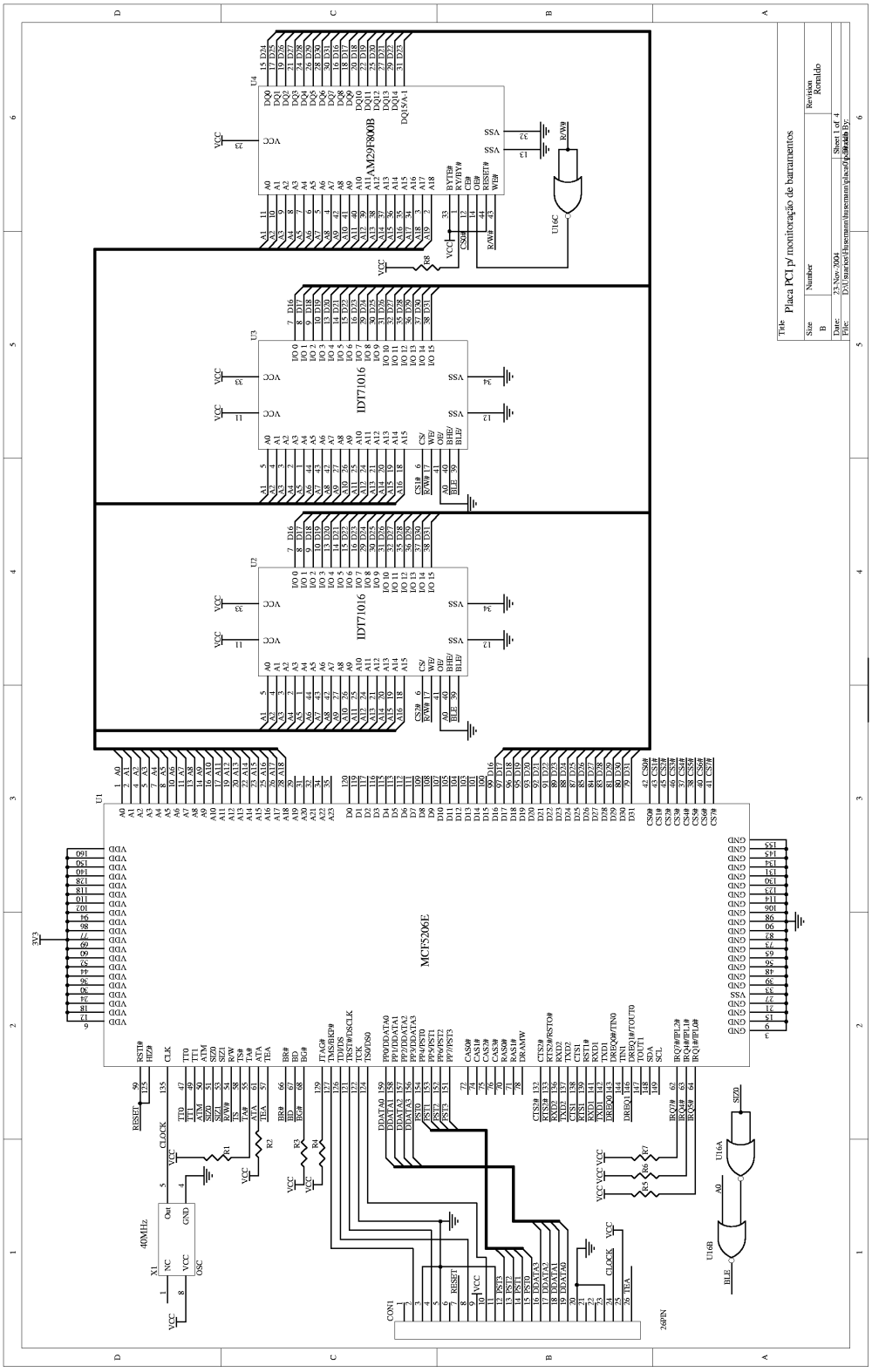
VERISSIMO, P.; RODRIGUES, L.; CASIMIRO, A. CESIUMSPRAY: a precise and accurate global time service for large-scale systems. **Journal of Real-Time Systems**. New York: Kluwer Academic Publishers, v. 12, p. 243-294. 1997.

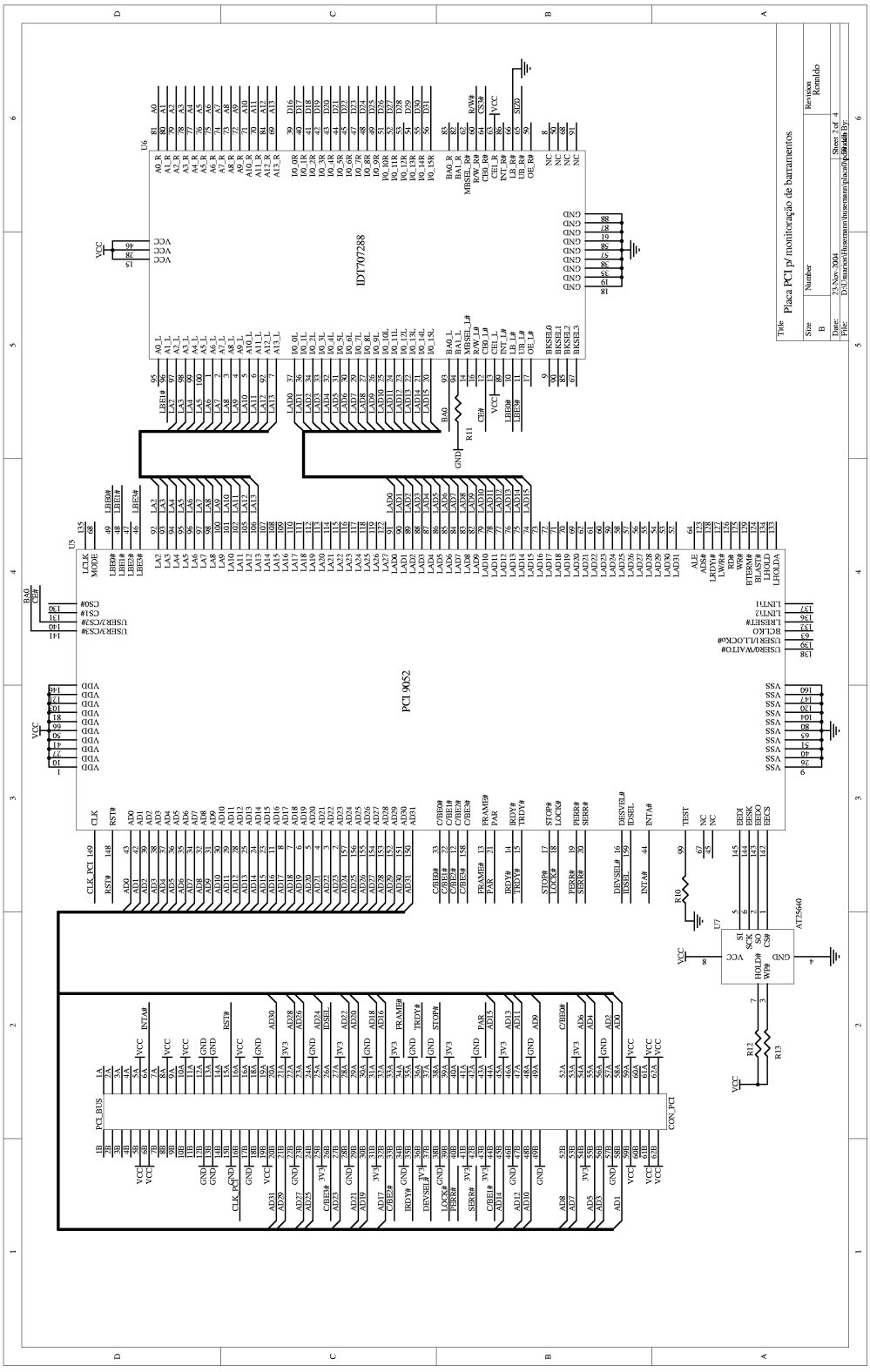
WILD, R. **Proposta de Ferramenta para Validação Temporal em Barramentos de Campo,** 2000, 92 f. Dissertação (Mestrado) – Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2000.

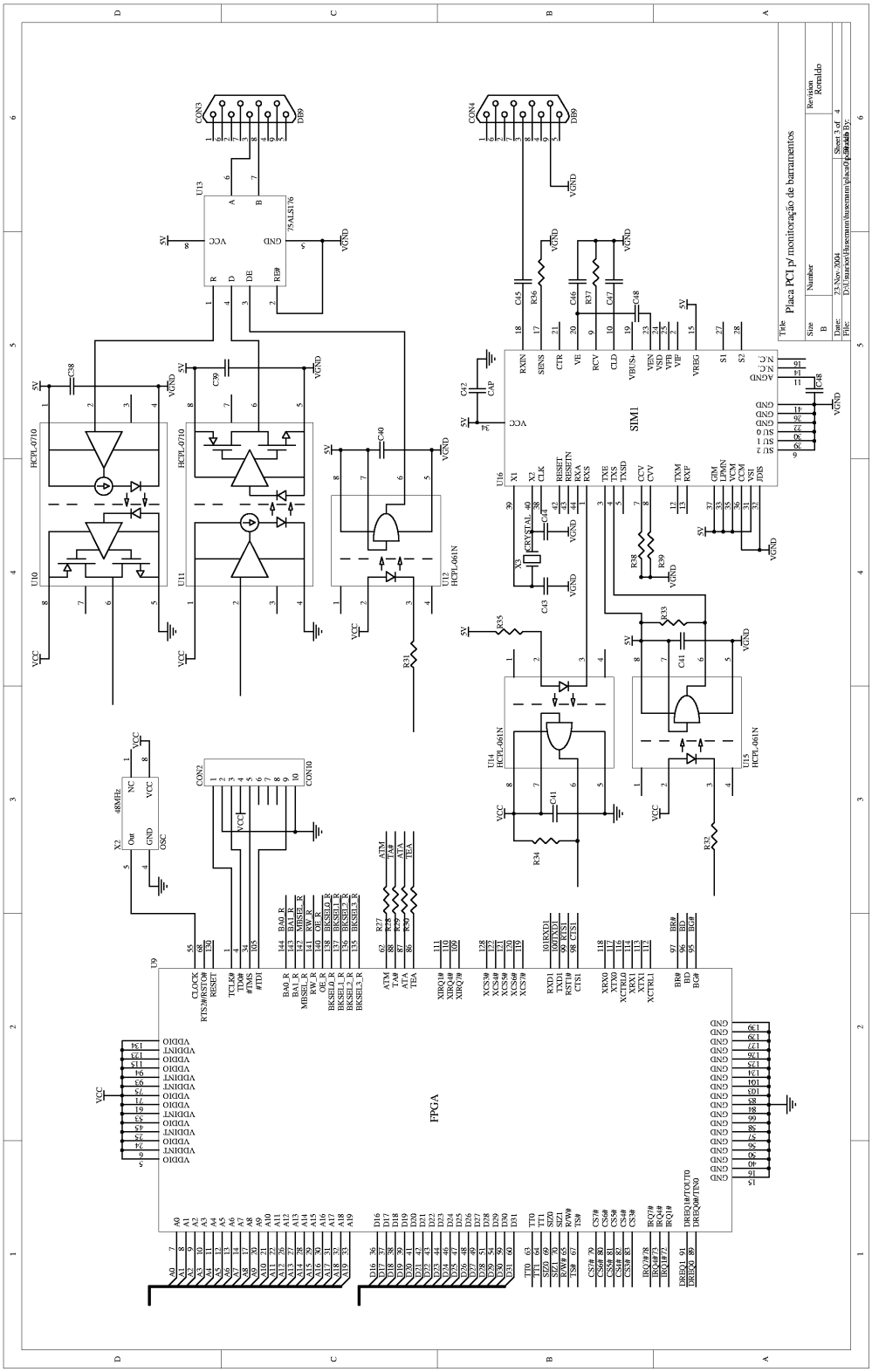
XILINX. **Spartan and Spartan-XL Families Field Programmable Gate Arrays**: product specification, 2000. 82 p. Disponível em: <http://direct.xilinx.com/bvdocs/ publications/ ds060.pdf>. Acesso em: 18 set. 2002.

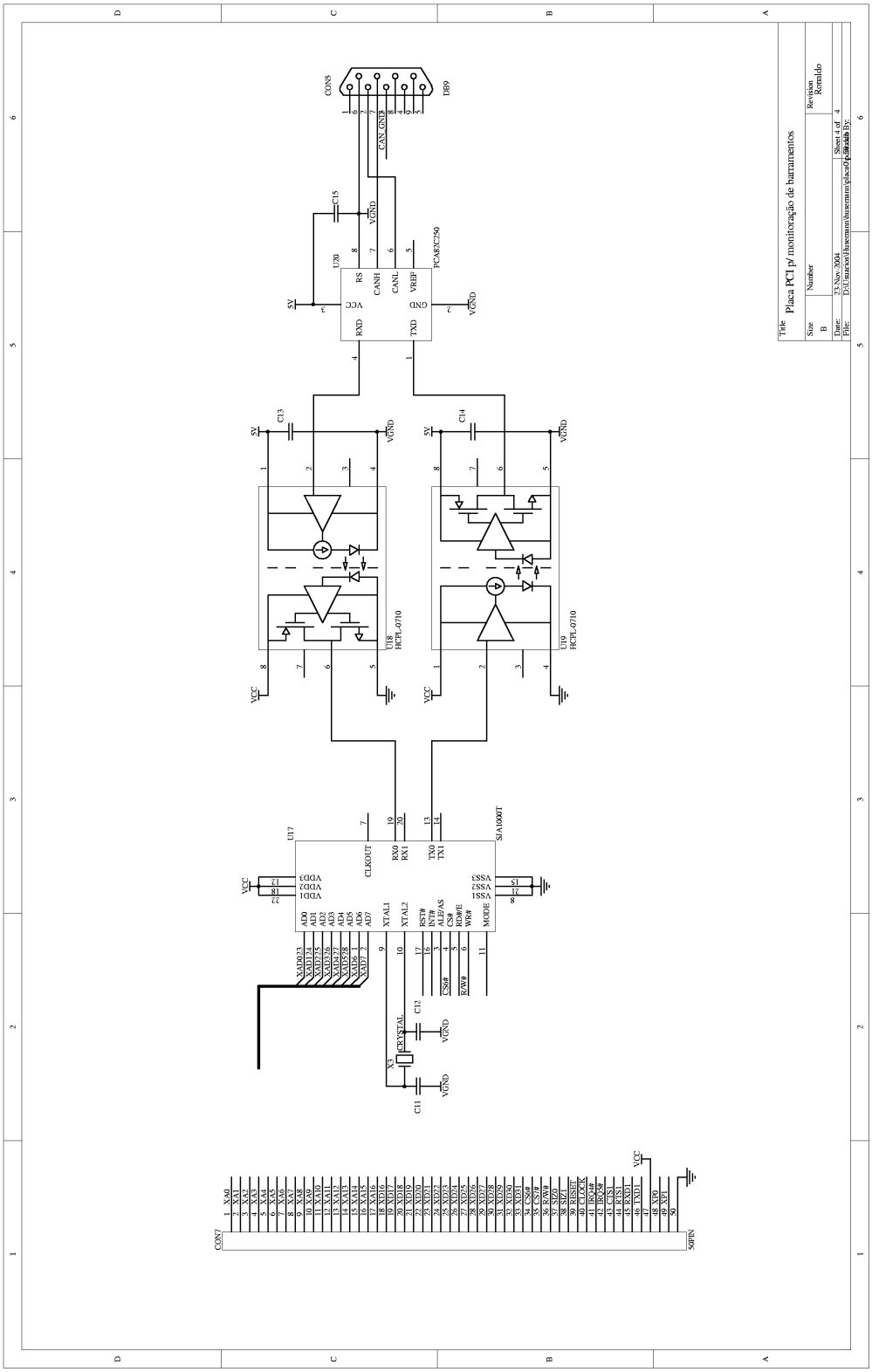
# APÊNDICE:

Diagramas Esquemáticos da Placa BUSMON.


Placa PCI p/ monitoração de barramentos
Revisão
Ronaldo
MCF5206E
IDT71016
AM29F800B
40MHz
CLOCK
RESET
TEA
BLE
CON1
248PIN


Placa PCI p/ monitoração de barramentos
Romualdo
PCI 9052
IDT707288
AT25640
CON_PCI


Placa PCI p/ monitoração de barramentos
Revision
Ronaldo
Sheet 1 of 4
FPGA
STM1
HCPL-0710
HCPL-061N
75ALS176
CON3
CON4
CON10
U9
U10
U11
U12
U13
U14
U15
U16
X1
X2
X3
C38
C39
C40
C41
C42
C43
C44
C45
C46
C47
C48
R27
R28
R29
R30
R31
R32
R33
R34
R35
R36
R37
R38
R39
5V
VCC
VGND
48MHz
CRYSTAL
CLOCK
RESET
TCLK#
TDO#
#TMS
#TDI
BA0_R
BA1_R
MBSEL_R
RW_R
OE_R
BKSEL0_R
BKSEL1_R
BKSEL2_R
BKSEL3_R
ATM
TA#
ATA
TEA
XIRQ1#
XIRQ4#
XIRQ7#
XCS3#
XCS4#
XCS5#
XCS6#
XCS7#
RXD1
TXD1
RST1#
CTS1
XTX0
XRX0
XCTRL0
XRX1
XTX1
XCTRL1
BR#
BD
BG#
TT0
TT1
SIZ0
SIZ1
R/W#
TS#
CS7#
CS6#
CS5#
CS4#
CS3#
IRQ7#
IRQ4#
IRQ1#
DREQ1#
DREQ0#
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
A13
A14
A15
A16
A17
A18
A19
D16
D17
D18
D19
D20
D21
D22
D23
D24
D25
D26
D27
D28
D29
D30
D31
GND
VDD
VDDINT
VDDIO


Placa PCI p/ monitoração de barramentos
Revision
Ronaldo
Size
B
Number
27-Nov-2004
Sheet 4 of 4
D:\Projetos\Sistemas\Sistemas\placa\placa.ddb
Drawn By:
CON5
DB9
CAN_GND
C15
VGND
U20
PCA82C250
RS
CANH
CANL
VREF
VCC
GND
RXD
TXD
5V
C13
C14
U18
HCPL-0710
U19
HCPL-0710
VCC
U17
SJA1000T
CLKOUT
RX0
RX1
TX0
TX1
VDD1
VDD2
VDD3
VSS1
VSS2
VSS3
AD0
AD1
AD2
AD3
AD4
AD5
AD6
AD7
XTAL1
XTAL2
RST#
INT#
ALE/AS
CS#
RD/E#
WR/#
MODE
XAD0/D3
XAD1/D4
XAD2/D5
XAD3/D6
XAD4/D7
XAD5/D0
XAD6/D1
XAD7/D2
CRYSTAL
C12
X3
C11
CON7
50PIN
XA0
XA1
XA2
XA3
XA4
XA5
XA6
XA7
XA8
XA9
XA10
XA11
XA12
XA13
XA14
XA15
XA16
XA17
XD18
XD17
XD16
XD15
XD14
XD13
XD12
XD11
XD10
XD9
XD8
RESET
CLOCK
IRQ#
WR#
CS1
RST1
RD1
XD1
XD0
XP1
XP0