UNIVERSIDADE LUTERANA DO BRASIL

PRÓ-REITORIA DE GRADUAÇÃO

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

FABIO JUNIOR MELO GUIMARÃES

PARQUÍMETRO ELETRÔNICO

Canoas, Julho de 2010

FABIO JUNIOR MELO GUIMARÃES

PARQUÍMETRO ELETRÔNICO

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

Departamento:

Engenharia Elétrica

Área de Concentração

Micro-controladores

Professor Orientador:

MSc.Eng. Eletr. Augusto Alexandre Durgante de Mattos – CREA-RS: 8.800-D

Canoas

2010

FOLHA DE APROVAÇÃO

Nome do Autor: Fabio Junior Melo Guimarães

Matrícula: 021007916-9

Título: Parquímetro eletrônico

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

Professor Orientador:

MSc. Eng. Eletr. Eletr. Augusto Alexandre Durgante de Mattos

CREA- RS: 8.800-D

Banca Avaliadora:

[MSc.] Eng. Eletr. Dalton Luiz Rech Vidor

CREA-RS: 79.005-D

Conceito Atribuído (A-B-C-D):

[MSc.] Eng. Eletr. Paulo César Cardoso Godoy

CREA-RS: 111.682-D

Conceito Atribuído (A-B-C-D):

Assinaturas:

Autor Fabio Junior Melo Guimarães	Orientador Augusto Alexandre Durgante de Mattos
Avaliador Dalton Luiz Rech Vidor	Avaliador Paulo César Cardoso Godoy

Relatório Aprovado em:

DEDICATÓRIA

Dedico aos meus pais:

Neli e

Jadir Guimarães.

AGRADECIMENTOS

A Deus que me deu forças para vencer os obstáculos.

À minha mãe Neli, meu pai Jadir e minha irmã Elisangela pelo incentivo, carinho e compreensão.

Ao colega Adilson Ribeiro dos Santos pelas sugestões e observações valiosas.

Ao Professor Augusto Alexandre Durgante de Mattos pelas valiosas contribuições.

Aos meus familiares e amigos pela compreensão nos momentos de ausência.

Aos professores pelos ensinamentos fornecidos.

Aos colegas de trabalho pelo apoio e compreensão.

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

RESUMO

GUIMARÃES, Fabio. **Parquímetro eletrônico.** Trabalho de Conclusão de Curso em Engenharia Elétrica - Departamento de Engenharia Elétrica. Universidade Luterana do Brasil. Canoas, RS. 2010.

Este trabalho tem como objetivo o desenvolvimento de um parquímetro eletrônico que possui as mesmas características dos utilizados no mercado atual. O projeto foi desenvolvido utilizando um micro-controlador da família 8051 programado em linguagem C utilizando o compilador Keil. A unidade central de processamento do parquímetro foi desenvolvida e interligada com os periféricos por meio de canais de comunicação serial. Cada periférico da CPU foi estudado através de ensaios com o software teste de comunicação serial chamado Device Tester e após ajustes e configurações necessárias todos foram integrados efetivando o funcionamento da CPU. Os resultados foram obtidos por simulações de funcionamento do sistema. Foi testada confiabilidade nos dados fornecidos pelo moedeiro, a geração de relatórios, impressão de tíquetes e testes com horários. O projeto funcionou de forma semelhante a um equipamento comercial.

Palavras chave: Parquímetro. Micro-controladores.

ABSTRACT

GUIMARÃES, Fabio. Electronic Parking meter. Work of Conclusion of Course in Electrical Engineering - Electrical Engineering Department. Lutheran University of Brazil. Canoas, RS. 2010.

This work has as its objective the development of an electronic parking meter which has the same characteristics as the ones used nowadays. The project was developed with the use of a micro-controller from the 8051 family programmed on C language, using a Keil compiler. The parking meter CPU was developed and interconnected with the peripherals through serial communication channels. Each CPU peripheral was studied through rehearsals with the communication test software called Device Tester and after adjustments and necessary configuration; all of them were integrated making the CPU work. The results were obtained by simulation of the working system. The reliability in the data provided by the coin acceptor, the report lineage, printing of tickets and schedule tests were tested. The project has worked in a similar way to commercial equipment.

Keywords: Parking meter. Micro-controller.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

USB: *Universal Serial Bus*

LAN: *Local Area Network*

PDA: *Personal Digital Assistants*

LED: *Light Emissor Diode*

CPU: *Central Processing Unit*

I/O: *Input/Output*

EEPROM: *Electrically Erasable Programmable Read Only Memory*

CI: Circuito Integrado

RTC: *Real Time Clock*

CPU: *Central Process Unit.*

LISTA DE SÍMBOLOS

V – Volts

A – Ampère

Hz – Hertz

SUMÁRIO

1.INTRODUÇÃO

O aumento vertiginoso da frota de veículos de passeio nos últimos anos tem gerado superlotação nos estacionamentos públicos, principalmente, nas grandes cidades do mundo. Mediante este problema as prefeituras determinaram que locais estratégicos fossem transformados em Áreas azuis. Esta medida melhora a rotatividade nestas áreas e consequentemente aumenta as ofertas de vagas.

As vagas de estacionamento da Área Azul são controladas por parquímetros eletrônicos. Mesmo que a utilização deste equipamento seja antiga, ainda existem cidades que não utilizam este recurso, utilizam os talões de preenchimento manual, no qual motorista é obrigado a pagar pelo talão o valor correspondente ao tempo máximo de permanência mesmo que use uma fração do crédito total do talão. O método manual possui outras desvantagens como dificuldade de obter um talão e a possibilidade de fraude do mesmo.

O presente projeto tem como objetivo o desenvolvimento de um parquímetro eletrônico com as mesmas características dos utilizados no mercado atual. Este permitirá o pagamento por meio de moedas. Todas as operações comuns ao parquímetro estarão presentes. O mesmo será capaz de fornecer comprovante de pagamento, assim como emitir um relatório de operações. O projeto tem como problema de engenharia a utilização de um micro-controlador, desenvolvimento do algoritmo de controle do mesmo, estudo e desenvolvimento de protocolo de comunicação entre os periféricos e micro-controlador.

O moedeiro utilizado neste projeto é um dispositivo que depende da implementação de um protocolo de comunicação com o micro-controlador. O moedeiro é capaz de distinguir diferentes tipos de moedas e interpretar falhas. Os comprovantes de pagamentos serão obtidos a partir de uma impressora térmica.

O equipamento tem como unidade de processamento um micro-controlador da família 8051. O parquímetro oferecerá o recurso de relatório

digital: através de uma porta de comunicação será possível obter as ocorrências no equipamento em determinado período além de registrar o valor em dinheiro armazenado no cofre. O projeto prevê um painel de operação composto por botões de comando, um *display* de interface entre o usuário e o parquímetro além de compartimento de devolução das moedas caso ocorra algum problema durante o procedimento de pagamento pelo usuário ou cancelamento da operação.

Neste trabalho o capítulo 2 conta um pouco sobre a evolução dos parquímetros e situa o presente projeto dentro do mercado atual. O capítulo 3 expõe os conhecimentos necessários para integração dos periféricos do parquímetro, são abordadas as características das interligações eletrônicas dos periféricos que compõe o parquímetro. No capítulo 4 será exposto o desenvolvimento do *software* do micro-controlador, a geração de relatórios de operações e o funcionamento do parquímetro durante a operação do usuário. Serão apresentados os resultados de desempenho dos pontos vitais do parquímetro. O capítulo 5 faz um fechamento do projeto através de conclusões obtidas e sugere algumas melhorias futuras.

2.EVOLUÇÃO DOS PARQUÍMETROS

O presente projeto é caracterizado por ser um desenvolvimento de produto já existente no mercado, portanto, é interessante sintetizar a evolução do mercado de parquímetros desde seu surgimento até os dias atuais. O parquímetro foi inventado em 1935 por Carl C. Magee na cidade de Oklahoma [1]. A produção industrial começou em 1936 e foi ampliada até meados dos anos 80. Os primeiros modelos foram baseados em um moedeiro com um botão para iniciar o mecanismo e um ponteiro visível, além de um bilhete para indicar a validade do período de estacionamento conforme figura 2-1[1].

Figura 2-1: Um dos primeiros parquímetros [1].

Esta configuração durou mais de 40 anos, com apenas algumas alterações em sua forma exterior, como a versão de duas cabeças e a

incorporação de novos materiais e técnicas de produção [1]. Em meados dos anos 80 uma versão digital foi introduzida, substituindo os componentes mecânicos por componentes eletrônicos: placas, teclados e monitores. Isto permitiu uma maior flexibilidade ao parquímetro, pois componentes como EEPROM, por exemplo, podem ser configurados com mais facilidade do que componentes mecânicos [1]. No início dos anos 90, milhões de unidades de parquímetros foram vendidas em todo mundo, mas o mercado já estava à procura de novas soluções, como máquinas coletivas com *tickets* e novas formas de pagamentos que apareceram ao longo do tempo através de tecnologias como o “dinheiro eletrônico” e de comunicações. [1]

Na virada do século XXI os parquímetros com pagamentos através de *smart cards* ganham força no mercado e auxiliam na implantação das Áreas azuis nas grandes cidades. Em 2003 foram adotados parquímetros pela prefeitura municipal de Campo Grande MS, por exemplo, para substituir os cartões de estacionamento [2]. O sistema tem chaveiro *i buttons* onde são carregados eletronicamente os créditos em uma unidade qualquer de correspondência monetária para utilização do usuário [2].

No caso de créditos restantes registrados no equipamento, estes poderão ser recuperados com a mesma operação de inserção do chaveiro. No controle de operações diário e registros de dados, o sistema oferece armazenamento dos mesmos, e o acesso é por meio de chaveiro específico [2].

Na cidade de Seattle em Washington, Estados Unidos, pode-se notar um fato de mudança da filosofia de parquímetros ocorrido entre 2004 e 2006. A modernização do sistema de estacionamento consistia em ampliar as opções de pagamento aos clientes e melhorar o ambiente de estacionamento [3]. As estações de pagamento foram projetadas para aceitar cartões de crédito, de débito e moedas. Se não foi utilizado todo tempo, os condutores podem utilizar o tempo restante do estacionamento que pagaram em outro parquímetro levando o comprovante de pagamento [3].

Em 2007, na Espanha uma das maiores empresas chamada Cale, lançou o parquímetro modelo MP104, figura 2-2. O equipamento foi desenvolvido com a idéia de ser uma das primeiras com sistema de controle remoto. [6] O equipamento pode emitir alarmes instantâneos ou avisos de todo tipo, desde cofre cheio até rolo de papel da impressora quase vazio [6].

Figura 2-2: Parquímetro modelo MP140 [6].

Os gabinetes podem funcionar com energia solar. A forma de pagamento pode ser feita através de moedas, fichas, cartões magnéticos ou cartões com chip, estes recursos possibilitam que o usuário pague a tarifa na desocupação da vaga. O software deste equipamento foi preparado para funcionar com esquema de fidelização de clientes.

Atualmente existem diversas empresas fabricante de parquímetros. No Brasil a Estepar e a Rek Parking estão entre as maiores empresas do ramo de estacionamento, elas gerenciam Áreas Azuis nas quais possuem parquímetros do Fabricante Parkeon (antiga Schulumberger), figura 2-3, e Digicon, figura 2-4, que funcionam de maneira similar.

Figura 2-3: Parquímetro Parkeon.

Figura 2-4: Parquímetro Digicon.

O pagamento pode ser feito em moedas de R$0,05 a R$1,00 ou por meio de cartão inteligente recarregável [5]. O cartão inteligente contém créditos pré-gravados que são debitados automaticamente de acordo com o tempo de cada utilização. A recarga de créditos do cartão pode ser realizada no próprio parquímetro ou em postos autorizados no comércio [5].

Os parquímetros possuem energia própria, através de painel solar e bateria interna, não gerando custos aos municípios. O equipamento possui um *software* de gestão próprio bloqueado que não permite alterações nos registros já efetuados.

Estatísticas são emitidas automaticamente pelo parquímetro, após cada interferência da gerência ou do departamento técnico. O equipamento permite o fracionamento da hora, dentro do tempo permitido para o estacionamento [4].

Outro tipo de sistema de estacionamento público atuante no mercado é o sistema *Eparq*, mostrado na figura 2-5, desenvolvido pela empresa Lapaza. Este equipamento está presente em oito estados brasileiros e controlando mais de 7000 vagas de estacionamentos [7].

Figura 2-5: Parquímetro modelo Eparq.

O parquímetro funciona através de um dispositivo de acionamento chamado "*ibutton*", figura 2-6, recarregável destinado para uma vaga da área azul gerenciada pelo parquímetro. Através dos "*ibuttons*" é feita a inserção de tempos nos parquímetros em resolução de minuto, e são recarregáveis nos próprios parquímetros ou em máquinas de recarga distribuídas no comércio [7].

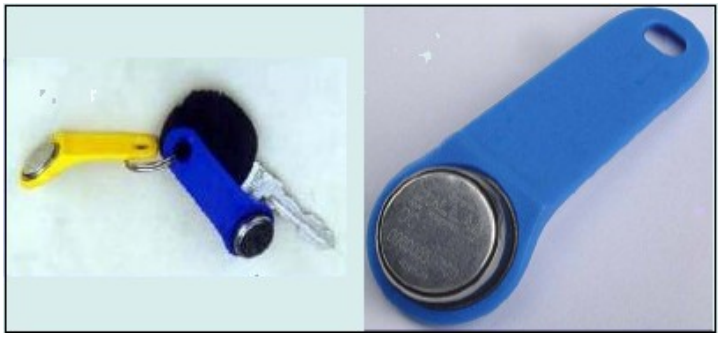


Figura 2-6: Dispositivo de acionamento *ibuttons*.

No procedimento de estacionamento, o usuário estaciona o veículo, em seguida vai até o gabinete e pressiona o "*ibutton*" no local indicado, então a vaga é selecionada automaticamente e carregado o tempo máximo de permanência de 120 minutos, na volta ao pressionar o "i*button*" novamente, o usuário recupera os créditos remanescentes. Este equipamento está disponível no mercado para controle de quatro ou oito vagas [7].

De posse da situação atual do mercado de parquímetros com relação a suas características e recursos, a Tabela 2-1 mostra um panorama comparativo entre o parquímetro desenvolvido neste projeto e os parquímetros de alguns fabricantes atuantes no brasil.

Tabela 2-1: Comparação do parquímetro desenvolvido com o mercado atual

Recurso	Parkeon	Rek Parking	Lapaza	**Parquímetro TCC**
Pagamento por moedas	X	X		X
Pagamento por Smart Card	X	X		
Pagamento por i-buttons			X	
Emissão de Comprovante	X	X		X
Relatório eletrônico	X	X	X	X

3. MATERIAIS E MÉTODOS

3.1. Descrição Geral do Sistema

A estrutura do parquímetro é composta por um moedeiro, uma impressora térmica, um *display* de interface com o usuário, bem como um canal de exportação do relatório de operações. A figura 3-1 apresenta um diagrama em blocos para ilustrar como estes periféricos estão interligados com a unidade central de processamento do parquímetro desenvolvido. O Apêndice A mostra o esquema elétrico geral do parquímetro.

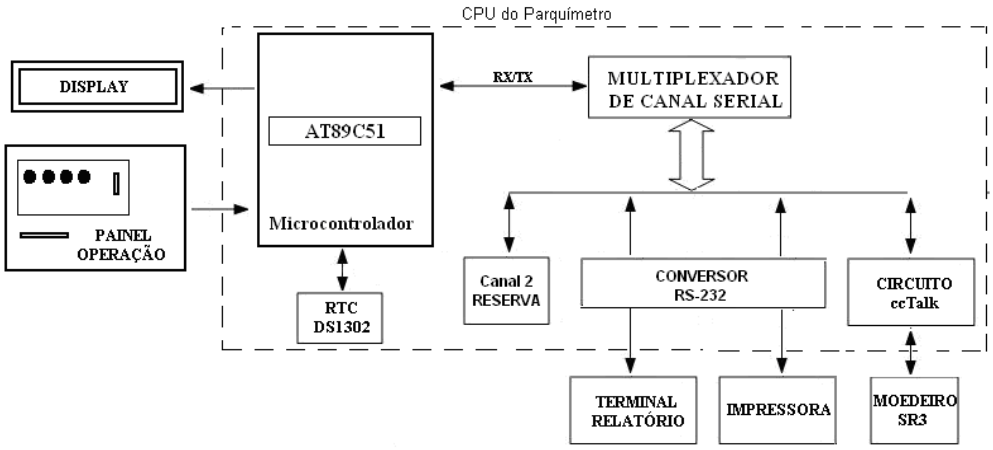


Figura 3-7: Estrutura do parquímetro em blocos.

Os periféricos do parquímetro bem como a CPU são detalhados a seguir com suas respectivas características, princípios de funcionamento, e suas interligações através de diagramas.

3.2. Moedeiro

O moedeiro utilizado, mostrado na figura 3-2, é da série SR3 tipo 2 da empresa Money Controls Ltda. É o principal equipamento do parquímetro, pois é o responsável pela identificação das moedas inseridas pelo usuário. As leituras de moedas ocorrem através de uma matriz de pequenos sensores indutivos de precisão posicionados de maneira a analisar diferentes áreas de diferentes

moedas. Devido o manuseio de moedas bicolores e prevenção contra fraudes do metal, os sensores de precisão concentram em pequenas áreas para obter medições precisas do núcleo da moeda e do anel externo. Este equipamento é capaz de receber 12 tipos diferentes de moedas de 15 mm até 31 mm de diâmetro, sendo que o moedeiro instalado está preparado para trabalhar com moedas brasileiras de R$0,05 até R$1,00. É alimentado com 12 a 24 Vdc com tolerância de +/- 10%, consome normalmente 70 miliampères e pode chegar a 450 miliampères. [8].

Figura 3-8: Moedeiro SR3 Tipo 2.

Sua configuração mecânica, mostrada na figura 3-3, é projetada para que seja inserida uma moeda por vez na parte superior. Caso a moeda seja válida, um solenóide é acionado e desvia a moeda para a saída do lado esquerdo, caso contrário a solenóide não liga e a moeda é rejeitada pelo lado direito.

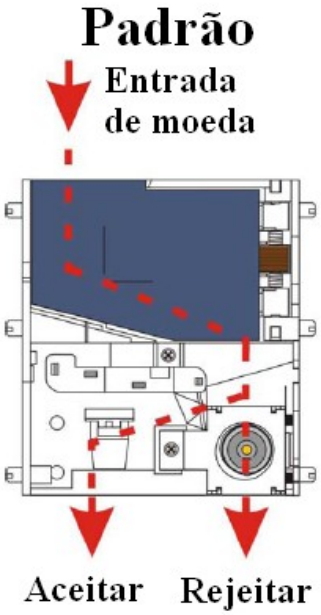


Figura 3-9: Separação de moeda válida e inválida no moedeiro.

3.2.1 Dispositivos do moedeiro

A parte traseira do moedeiro, mostrada na figura 3-4, possui diversos dispositivos do moedeiro, referenciados na tabela 3-1 [8]. Os dispositivos utilizados neste projeto estão detalhados a seguir, os demais estão detalhados no manual técnico conforme referência bibliográfica [8].

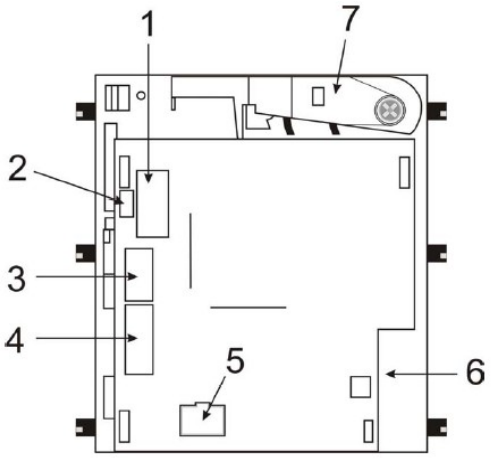


Figura 3-10: Parte traseira do SR3 Tipo 2.

Tabela 3-2: SR3 Tipo 2 detalhes da tampa traseira

Número	Característica/Função
1	Opcional/chaves totalizadoras (não usado)
2	LED
3	Interface serial
4	Saídas digitais de funções (não usado)
5	Chaves de inibição de moedas
6	Porta paralela (não usado)
7	Alavanca de rejeição

Led - Tem a finalidade de identificar os motivos de rejeição de moeda, cada motivo é identificado pelo número de vezes que o *led* vermelho liga. (8), conforme lista abaixo.

1 pulso vermelho: moeda aceita/alavanca de rejeição acionou.

2 pulsos vermelhos: moeda fora do faixa programada (inválida).

3 pulsos vermelhos: moeda válida, mas inibida (chaves totalizadoras).

4 pulsos vermelhos: moeda inibida pelo equipamento de controle.

Interface Serial com moedeiro - A comunicação entre o micro-controlador e o moedeiro é feita através do circuito eletrônico chamado *ccTalk*

mostrado na figura 3-5. Este circuito é fornecido pelo fabricante Money Controls sem exigências de taxa de licença. Este tipo de circuito é largamente usado em setores de tarifação de máquinas de diversão, vídeo, jogos e transportes. Seu funcionamento está baseado em um barramento de comunicação serial bidirecional identificado como Dado_SR3. Este barramento está conectado ao moedeiro e tem a função de enviar e receber os dados do moedeiro. Além deste barramento há o canal TX e RX que estão conectados ao multiplexador de canal serial da unidade de processamento.

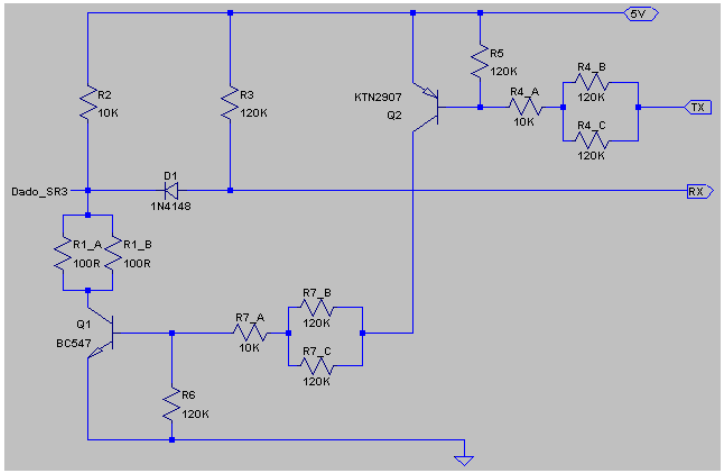


Figura 3-11: Circuito de comunicação serial ccTalk.

Chaves de inibição de moedas - É um conjunto de oito chaves com a função de inibir moedas indesejáveis. As chaves 1 a 6 controlam, a aceitação de cada dupla de doze moedas diferentes, onde ON = moedas inibidas e OFF = moedas aceitas. A chave 7 habilita o grupo de 6 moedas do BANCO 1 mostrado na segunda coluna da tabela 3-2. Enquanto que a chave 8 habilita o grupo das outras 6 moedas do BANCO 2 mostrado na terceira coluna da tabela abaixo.

Tabela 3-3: Inibição de moedas.

CHAVE	BANCO1	BANCO2
1	R$0,05 prateada	R$0,25 dourada
2	R$0,05 dourada	R$0,50 fina
3	R$0,10 prateada	R$0,50 grossa
4	R$0,10 dourada	R$1,00 desativada
5	R$0,25 desativada	R$1,00 fina
6	R$0,25 prateada	R$1,00 grossa
7	ON = desabilita BANCO 1	OFF = habilita BANCO 1
8	ON = desabilita BANCO 2	OFF = habilita BANCO 2

Em uma situação que se deseja inibir as moedas de R$0,10 prateada e a de R$0,50 grossa, por exemplo, é necessário posicionar a chave 3 em OFF e as chaves 7 e 8 em OFF para que as demais moedas sejam aceitas pelo moedeiro.

Alavanca de rejeição - É um dispositivo mecânico acionado por um solenóide, cuja função é de segregar moedas inválidas.

3.2.2 Reconhecimento de moedas.

O reconhecimento de moedas é feito através do comando de leitura de créditos ou erros acumulados no *buffer* do moedeiro, denominado comando 229. A figura 3-6 mostra como o comando 229 é enviado ao moedeiro e o tipo de informação de cada byte enviado. Este comando deve ser enviado a cada 200 milissegundos a fim de evitar perdas de leitura em uma sequência rápida de moedas inseridas.

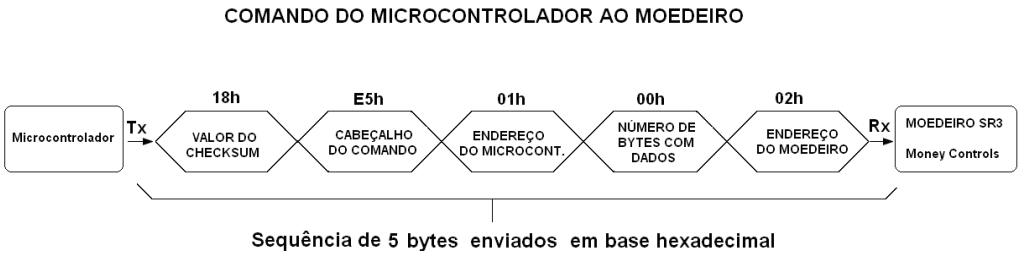


Figura 3-12: Comando de leitura do moedeiro via comunicação serial.

Após a recepção destes dados pelo moedeiro, o mesmo retorna uma resposta através de uma sequência de 16 bytes, os resultados “A” e “B” dos 5 eventos contem os códigos de moedas válidas inseridas ou códigos de erros. A figura 3-7 mostra o formato de resposta do moedeiro ao micro-controlador.

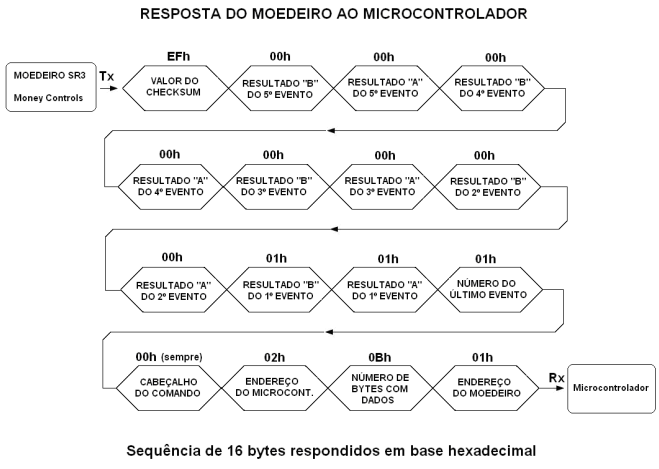


Figura 3-13: Resposta do moedeiro via comunicação serial.

A figura 3-8 mostra o *buffer* do moedeiro no *display* com o conteúdo de cada byte e seu respectivo significado.

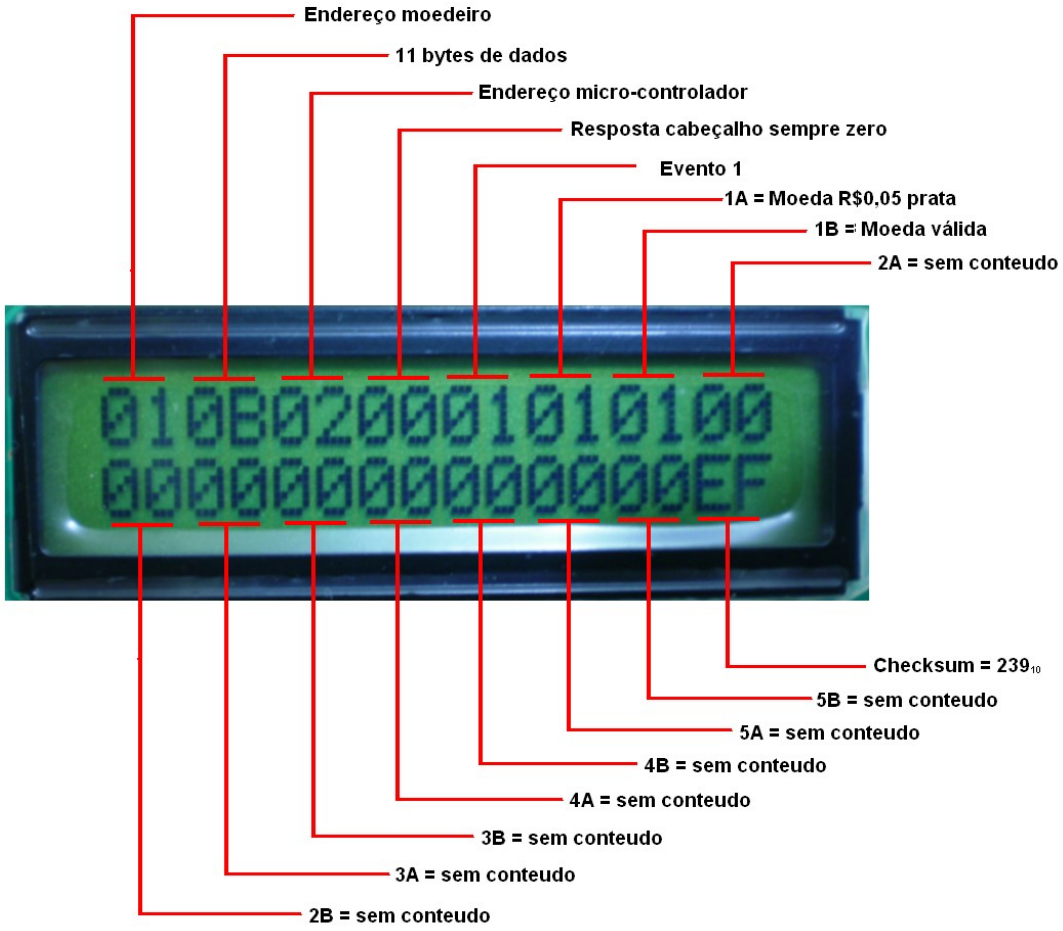
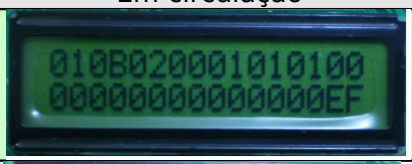
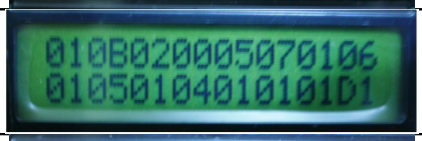
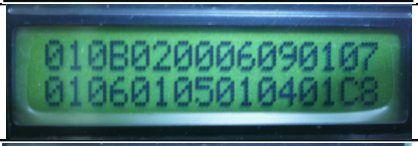
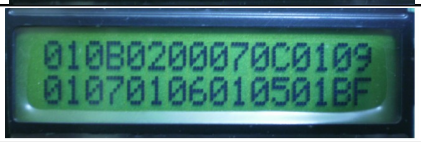


Figura 3-14: *Buffer* do moedeiro exibido no *display* do parquímetro.

Do 6º byte ao 15º têm-se os últimos 5 dados dos últimos 5 eventos, sendo que os bytes 1A 2A 3A, 4A e 5A quando inserida uma moeda válida mostra o código da moeda em hexadecimal. Quando houver algum o erro mostra o valor 00 em hexadecimal. Os bytes 1B, 2B, 3B, 4B e 5B mostram o valor 01 quando foi inserida alguma moeda válida, quando houver alguma falha mostra o código do erro em hexadecimal. A relação de código de erros está no anexo B e estrutura do *buffer* está no Anexo C.

Neste caso a resposta do moedeiro está informando que ocorreu um evento no moedeiro após sua energização e que neste evento foi inserida uma moeda de R$0,05 conforme destacado na tabela 3-3.

Tabela 3-4: Conteúdo do *buffer* de cada moeda válida para o moedeiro.

Moeda	Em circulação	Resultado 1A	Resultado 1B
		01	01
		02	01
		03	01
		04	01
		05	01
		06	01
		07	01
		09	01
		0C	01

3.2.3 Verificação de *Checksum*

Visando a confiabilidade dos dados enviados pelo moedeiro ao micro-controlador foi desenvolvida uma rotina no *software* de certificação da integridade dos dados, denominada *checksum* conforme figura 3-9. O dado relacionado ao *checksum* é enviado pelo moedeiro sempre no último byte da *string.*

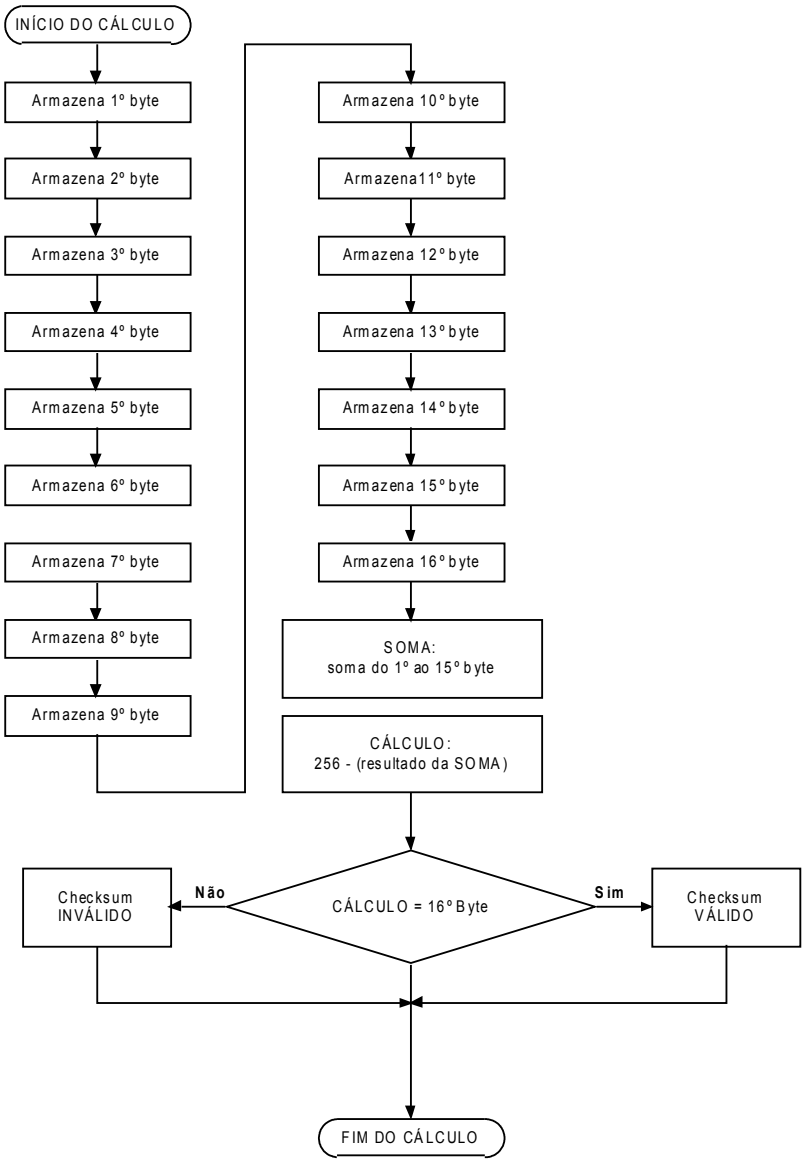


Figura 3-15: Trecho do *software* para verificação do *checksum.*

A equação abaixo mostra a equação implementada no fluxograma do *software* para o cálculo de *checksum*

$$CKS = 256 - [D1 + D2 + D3 + (Dn - 1)]$$

Onde: CKS = resultado do calculo de *cheksum;* D1, D2 e D3 são os primeiros, segundo e o terceiro dados, respectivamente, da string enviada pelo moedeiro; Dn-1 é o penúltimo dado enviado pelo moedeiro.

Aplicando o fluxograma da figura 3-9 com os dados da figura 3-7, temos:

$$CKS = 256 - [1 + 11 + 2 + 0 + 1 + 1 + 1 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + (0)]$$

$$CKS = 256 - [17] = 239$$

Após a realização do cálculo de *checksum* por parte do software, o resultado CKS é comparado com o último byte enviado pelo moedeiro. Caso a comparação seja verdadeira, as tarefas seguintes são habilitadas, caso contrário, um alarme de sistema é gerado no *display* e as moedas inseridas pelo usuário são devolvidas.

3.3. Impressora

A impressora térmica para geração dos bilhetes é o modelo CP205-MRS fabricada pela APS mostrada na figura 3-10. A alimentação elétrica é de 5 até 8,5 volts DC e corrente máxima de 5 ampère. Possui dois conectores: um de alimentação e outro do canal serial RS-232 [9].

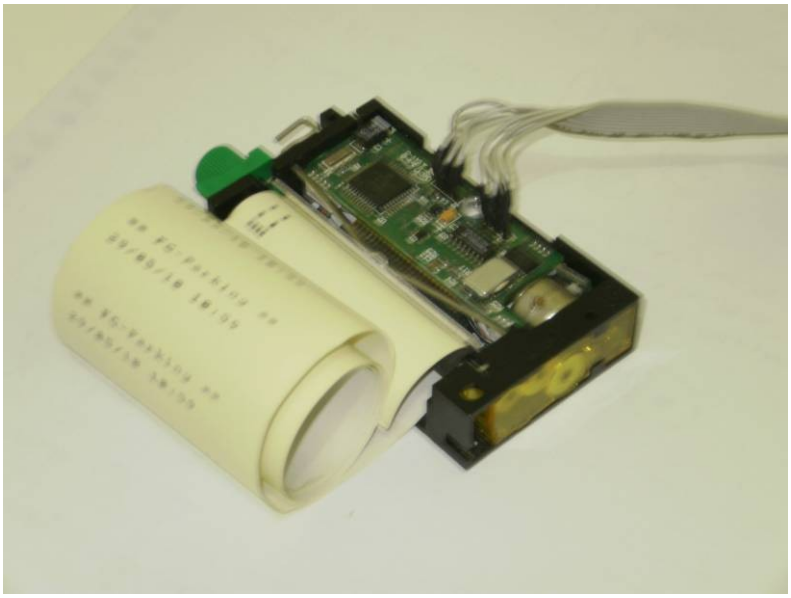


Figura 3-16: Impressora térmica CP205 – MRS.

O método de impressão é por linhas e pontos em um papel térmico de aproximadamente 58 milímetros, quando em contato a cabeça térmica da impressora, pode gerar caracteres de até três tamanhos de fontes diferentes [9].

A comunicação serial se dá através de um conector de cinco pinos, o protocolo de comunicação permite enviar códigos de parametrização da impressora, requisitar estado atual da impressora (alarmes, falhas, etc.), bem como enviar comandos de impressão de caracteres a uma velocidade de até 115200 bits por segundo (9).

A impressora utilizada no projeto contempla a geração de caracteres em formato de texto, de código de barras e formas gráficas [9]. Os caracteres em

formato de texto podem ser gerados em três tamanhos de fontes. Os caracteres estão disponíveis pela impressora em forma de tabela conforme a figura 3-11.

A impressão de caracteres em formato de texto é o mais utilizado no parquímetro. O caractere é gerado quando a impressora recebe, pelo canal serial, seu respectivo número ASCII em código hexadecimal, como por exemplo: Para geração da letra ‘A’’A’, o micro-controlador envia à impressora 41h, no qual o *nible* menos significativo representa coluna e o *nible* mais significativo representa linha

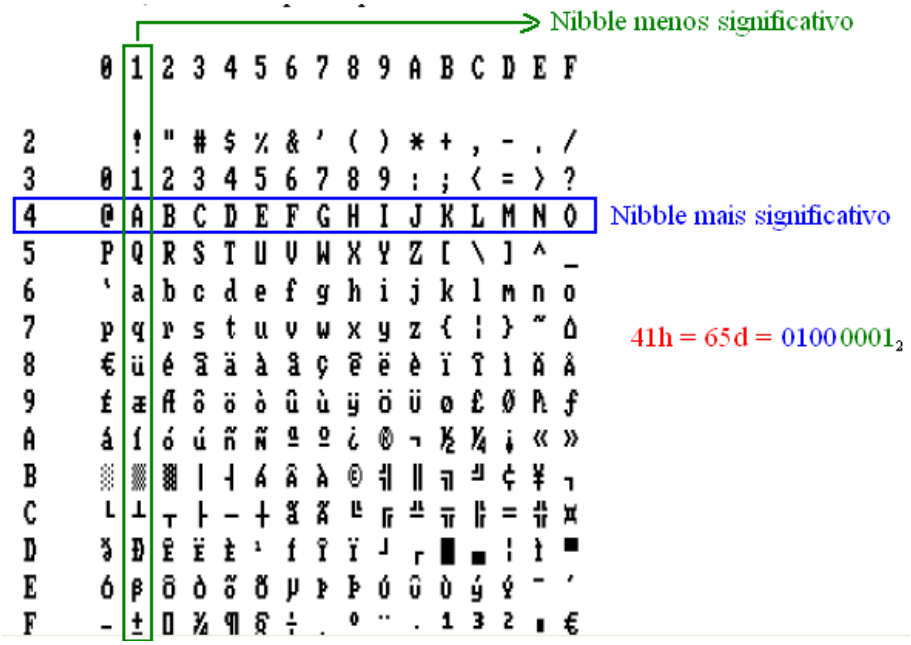


Figura 3-17: Mapa de caracteres da impressora.

3.2.1. Inicialização da Impressora

Além de comandos de escrita a impressora possui comandos de parametrização, estes comandos são necessários após a energização do equipamento para possibilitar o correto funcionamento. A figura 3-12 mostra a sequência de comandos que foram desenvolvidos no *software* do micro-controlador.

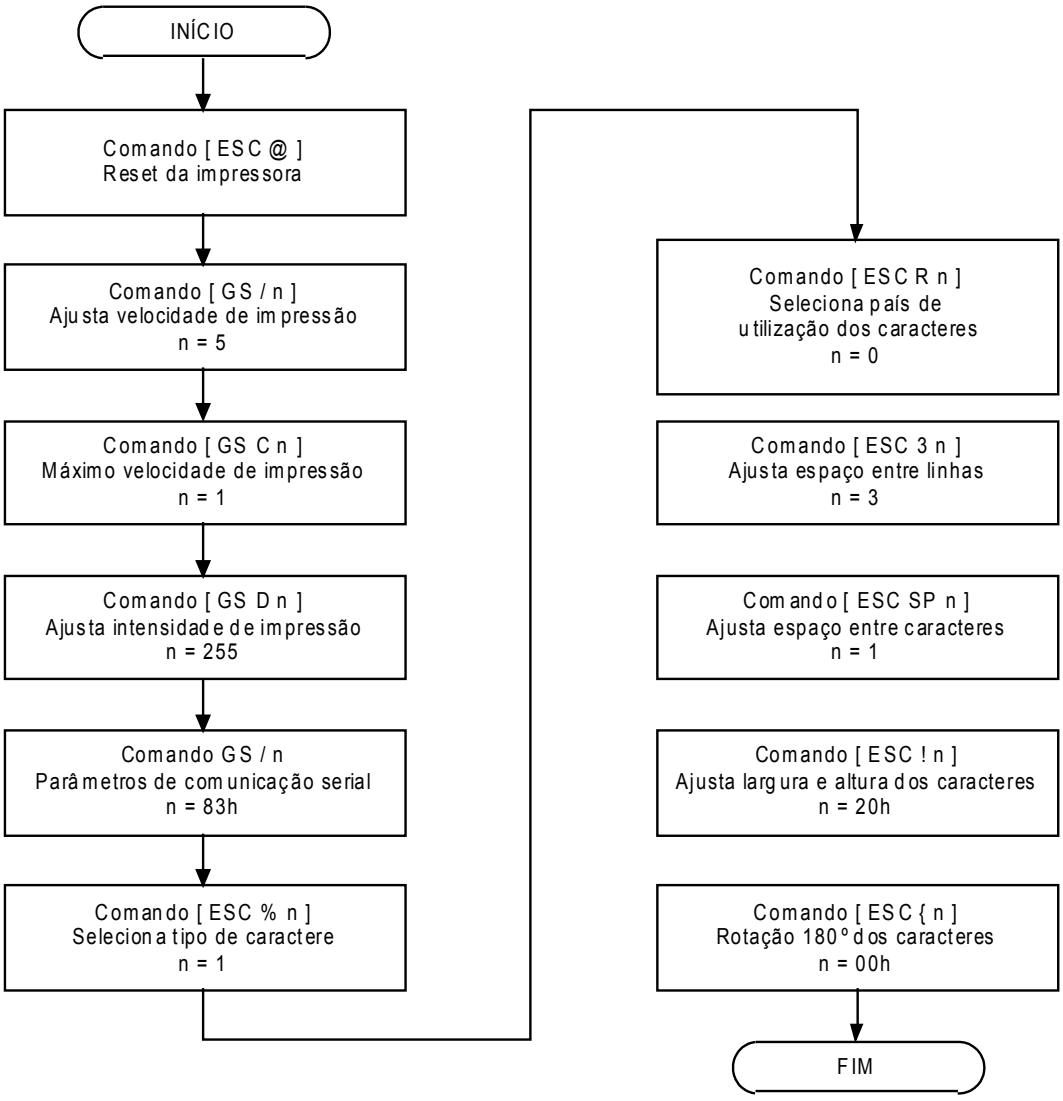


Figura 3-18: Comandos de inicialização da impressora

O protocolo de comunicação entre o micro-processador e a impressora foi desenvolvido de maneira que cada letra do comando seja enviada à impressora em código ASCII. O valor do parâmetro está contido em “n” de cada comando em base hexadecimal.

3.4. Unidade central de processamento do parquímetro

O micro-controlador é o elemento responsável pelas tarefas de gerenciamento dos periféricos. Neste projeto está sendo utilizado o AT89C51RD2 de oito bits de barramento, sessenta e quatro *kilobytes* de memória *flash* e de programação. A figura 3-13 mostra o encapsulamento do micro-controlador.

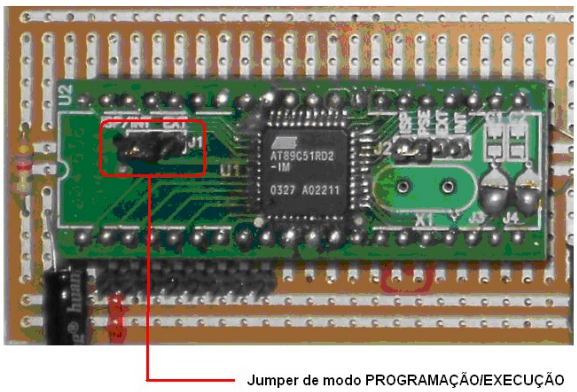


Figura 3-19: Micro controlador AT89C51RD2.

O chip é alimentado com 5 Vdc e possui um canal serial com pinos RX e TX, dois pinos de interrupção INT0/INT1, 2 temporizadores. O chip tem 34 pinos de I/O entre outros conforme a figura 3-14. Para programar o micro-controlador é utilizado o compilador Keil Versão 2.40 e para carregar o programa compilado foi utilizado o *software* Flip Atmel 2.4.6.

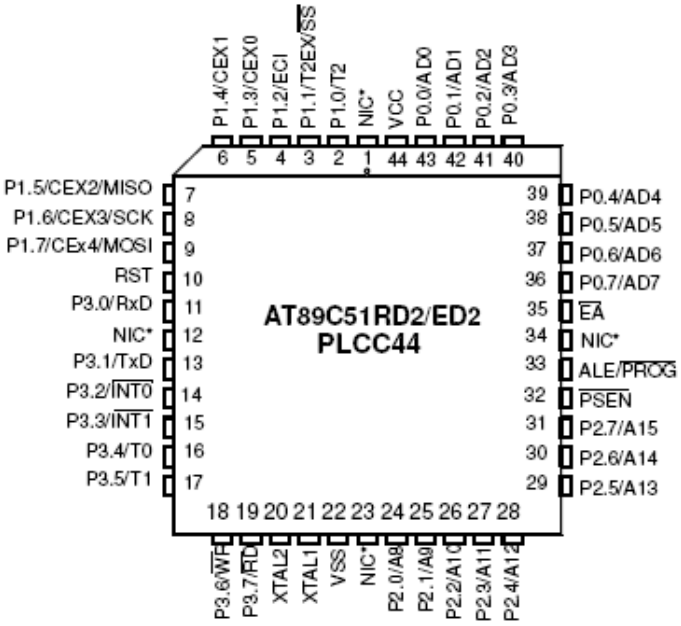


Figura 3-20: Pinos do micro-controlador.

Além do micro-controlador a CPU é constituída de circuitos de controle dos periféricos, conforme mostrado na figura 3-15. A CPU desenvolvida gerencia o *display*, o moedeiro SR3, a impressora térmica, e o painel de operação composto por dois botões. Todos os periféricos estão ligados por conectores soldados na placa de circuito impresso.

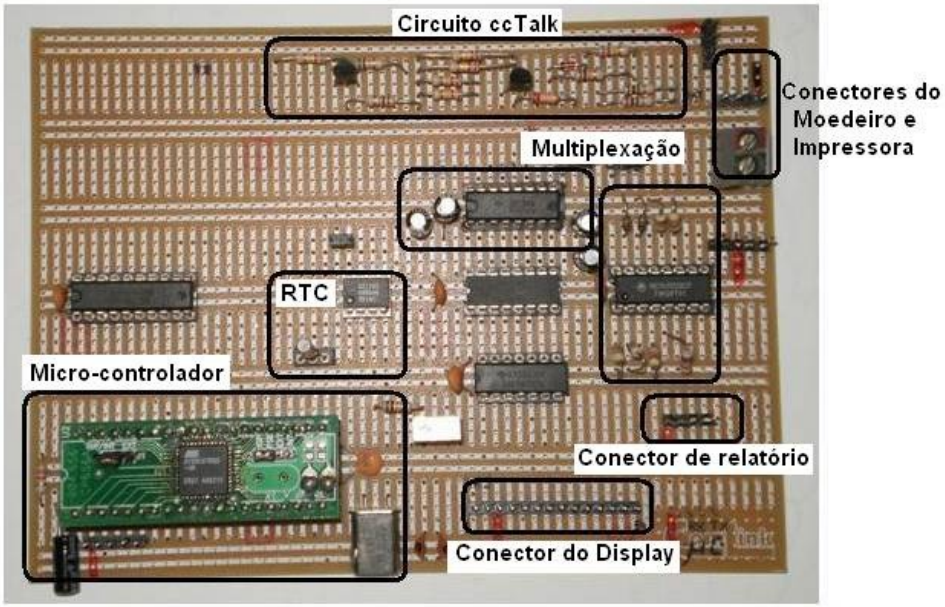


Figura 3-21: Placa da unidade central de processamento.

3.4.1 Circuito de Multiplexação serial

O micro-controlador recebe e envia dados aos periféricos via canal serial. Porém ele dispõe de apenas um canal serial, por isso foi desenvolvido um circuito de multiplexação a fim de viabilizar o trafego de dados a todos os periféricos.

O circuito de multiplexação do canal de comunicação serial da figura 3-16 está baseado em um circuito integrado MC14052B que consiste em quatro canais analógicos, nos quais cada canal possui uma chave de entrada e outra de saída. A seleção de cada canal é feita através de dois pinos chamados A e B. A tabela 3-4 explica de que maneira a seleção dos canais acontecem.

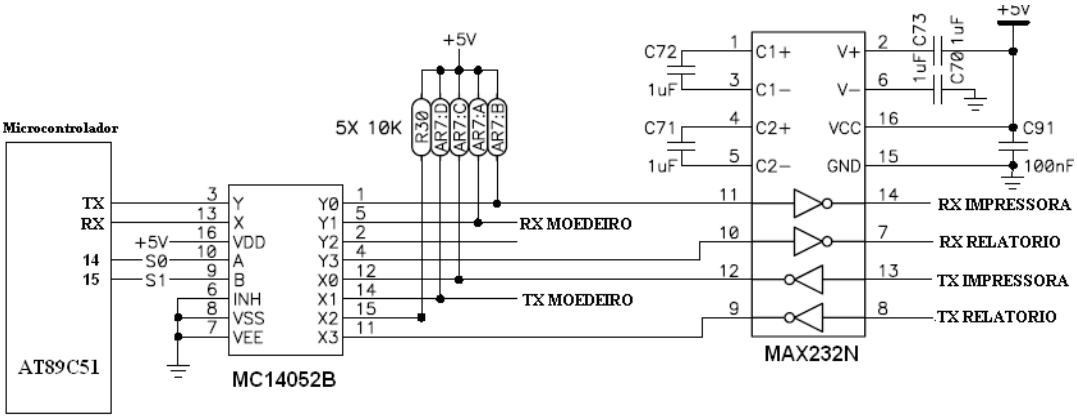


Figura 3-22: Circuito de multiplexação serial.

Tabela 3-5: Funcionamento do Multiplexador/Demultiplexador.

Tabela verdade MC14052				
Pinos de controle			Canais ativos	
Inibe chip	Seleção dos canais			
	B	A		
0	0	0	Y0	X0
0	0	1	Y1	X1
0	1	0	Y2	X2
0	1	1	Y3	X3
1	x	x	nenhum	nenhum

Desta forma foi definido um canal para cada periférico:

Canal 1 – Impressora

Canal 2 – Moedeiro

Canal 3 – Reserva

Canal 4 – Relatório digital

A escolha do canal 4 para transmitir o relatório digital ao PC com Hyperterminal é estratégica. O motivo está no processo de desenvolvimento do parquímetro quando há a necessidade de transmitir o *software* compilado do PC para o micro-controlador. As portas P3.2 e P3.3 possuem *pull-up* interno, esta característica faz com que ao alimentar o micro-controlador em modo de programação os pinos 14 e 15 ficam com nível alto (5V), consequentemente, os pinos 9(B) e 10 (A) do *mux/demux* selecionam o canal 4. Com o micro-controlador em modo de execução o controle dos canais é feito pelo *software*. Este recurso permite deixar um canal reserva para futuras modificações no parquímetro.

3.5. Relógio

Um dos itens fundamentais do parquímetro é o chip RTC DS1302 da figura 3-17, este componente é responsável pelo fornecimento de tempo e data do parquímetro. Este chip conta segundos, minutos, hora, data do mês, mês, dia da semana, e ano incluindo anos bissextos até 2100. (10).

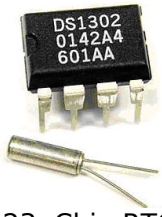


Figura 3-23: Chip RTC DS1302.

Através dele é possível prever dias, feriados e finais de semanas até o ano de 2100, este dado é essencial para o funcionamento em situações em que o

usuário utilizará a Área azul em períodos próximos de fim de expediente, no qual deverá reservar o restante do crédito para que o usuário possa utilizar no próximo dia útil.

O RTC faz a comunicação serial bidirecional através de um único pino. Para a interface de comunicação entre o micro-controlador e o RTC foram utilizados três pinos de I/O. A interligação entre o RTC e o micro-controlador foi feita conforme a figura 3-18.

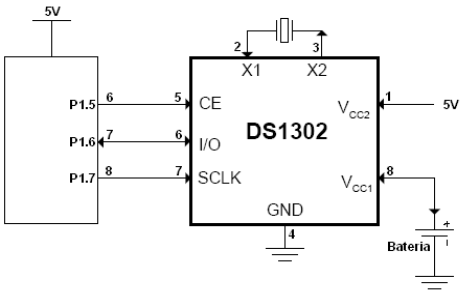


Figura 3-24: Circuito de ligação do RTC com o micro-controlador.

O pino CE habilita a comunicação do RTC, o pino I/O é por onde trafega o dado de entrada e de saída, o pino SCLK sincroniza o dado. Durante a transmissão ou recepção de dados este pino é comandado pelo micro-controlador de maneira que o mesmo suba para nível 5V e volte para 0V a cada bit da palavra digital no pino I/O. No Anexo A uma tabela exibe detalhadamente como cada registrador do RTC deve ser configurado para o processo de escrita de dados.

De acordo com o diagrama da figura 3-19, o procedimento adotado para escrita de dados ao RTC consiste em ligar o pino CE, oscilar pino de SCLK e enviar o dado através do pino I/O. O pino I/O envia duas palavras de 8 bits ao RTC, a primeira refere-se ao tipo de grandeza a ser escrita no RTC, a segunda refere-se ao valor da grandeza escrita.

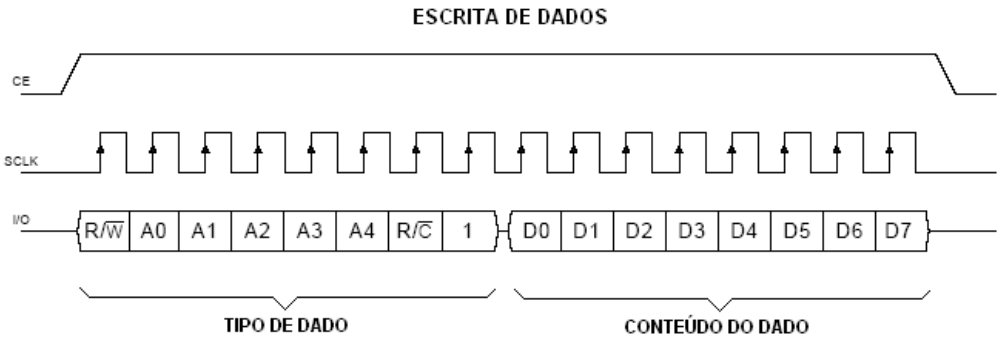


Figura 3-25: Diagrama de escrita de dados no RTC.

A figura 3-20 mostra o procedimento de recepção de uma grandeza do RTC, embora a estrutura seja semelhante à de escrita, pode-se notar que a palavra digital do conteúdo da grandeza é enviada ao micro-processador no momento de descida do pulso SCLK.

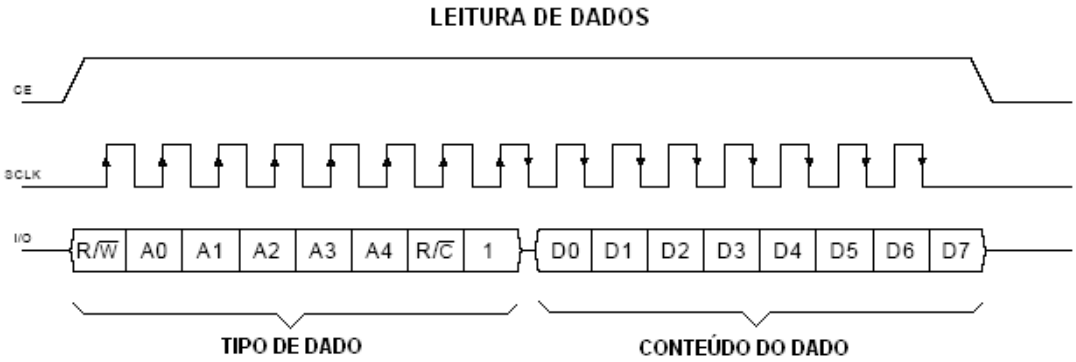


Figura 3-26: Diagrama de recepção de dados do RTC.

Exemplo de escrita e leitura do RTC:

De acordo com o Anexo A, para escrever no RTC o dado 23 a primeira palavra digital a ser enviada é 82 hexadecimal = 10000010 binário. A segunda palavra digital é 23 hexadecimal = 00100011 binário.

Para ler o dia do mês do RTC, a primeira palavra é 87 hexadecimal = 010000111 binário. Caso a data corrente no RTC seja 16 de Junho de 2009, a segunda palavra será enviada pelo o RTC como o conteúdo 2C hexadecimal = 000101100 binário.

3.6. Relatório eletrônico

Um *log* eletrônico foi desenvolvido a fim de viabilizar análise de horários de picos de utilização do parquímetro. O algoritmo de armazenamento do LOG foi desenvolvido para atender uma autonomia de 340 operações o que significa 1 operação de usuário a cada 2 minutos, suficiente para um dia de operação.

Para fazer a exportação dos dados foi utilizado o aplicativo Hyperterminal. Esta tarefa é necessária cada vez que o cofre for acessado, além disto, um bilhete é impresso contendo o valor financeiro armazenado no cofre. A figura 3-21, explica como foi estruturada a montagem do LOG do parquímetro.

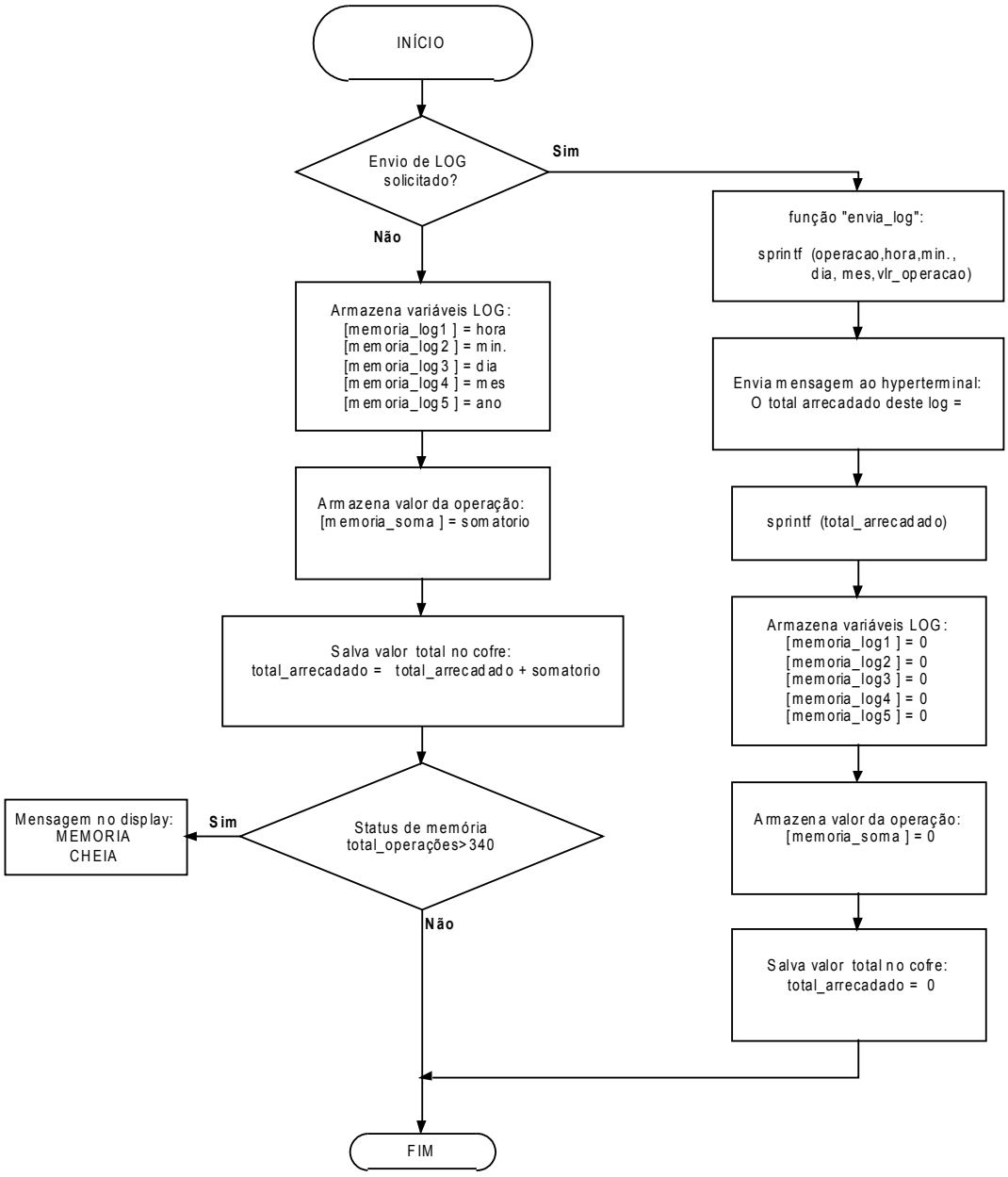


Figura 3-27: Estrutura do relatório eletrônico.

4.APRESENTAÇÃO E DISCUSSÃO DOS RESULTADOS

A seguir seguem os resultados de *hardware* e *software* do parquímetro bem como testes realizados e seus resultados a fim de visualizar o funcionamento de itens importantes. Também é apresentado o ensaio de verificação da confiabilidade de comunicação entre o moedeiro e o micro-controlador, o funcionamento do relógio (RTC), os alarmes de operação e sistema bem como a geração do relatório através de simulações de LOG.

4.1. Estrutura física completa do parquímetro

A figura 4-1 mostra o resultado do desenvolvimento do *hardware* do parquímetro no qual é composto pelas calhas e recipientes de moedas, a placa da unidade de processamento, o painel de operação além dos periféricos.

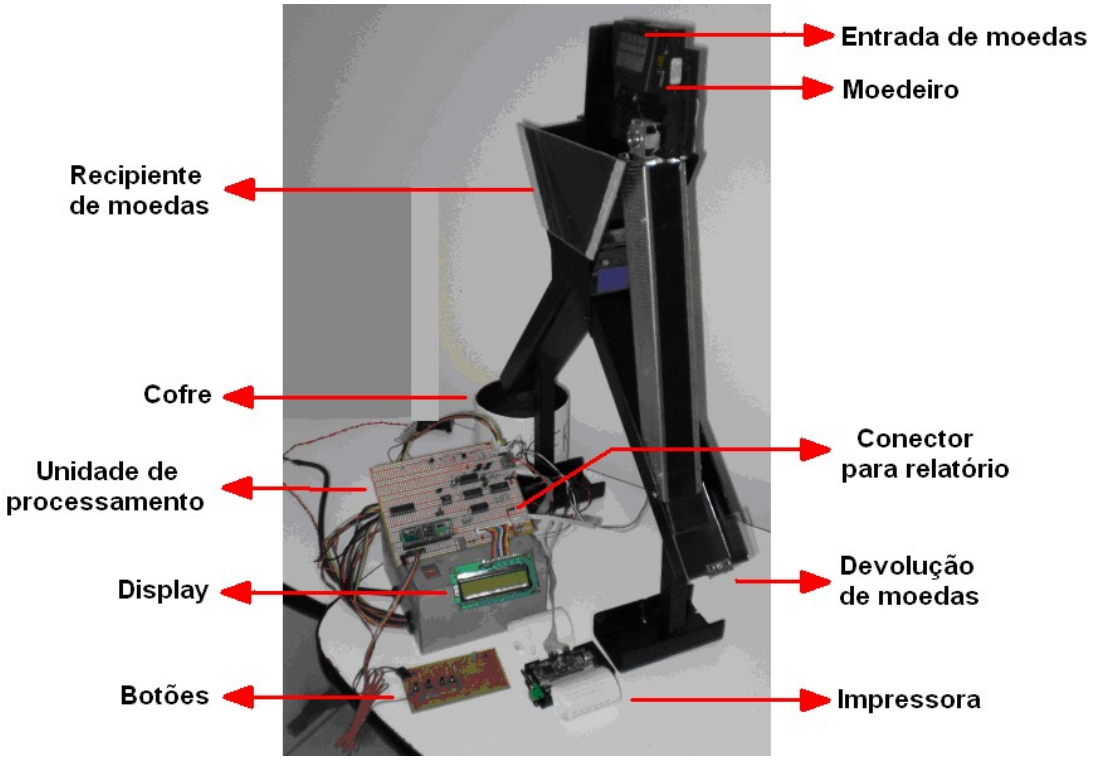


Figura 4-28: Estrutura mecânica do parquímetro.

As moedas inseridas pelo usuário seguem dois destinos possíveis (cofre ou devolução) em três situações distintas (moeda inválida, confirmação ou cancelamento da operação). Por isso foi desenvolvido uma estrutura mecânica a fim de viabilizar esta necessidade.

A estrutura possui calhas de fluxo de moedas na qual é composta por um recipiente principal. Este recipiente possui dois alçapões comandados por dois solenóides de 12 Volts, elas permanecem fechadas a fim de armazenar, temporariamente, todas as moedas válidas inseridas pelo usuário durante a operação. Após a operação ser efetuada com sucesso, o alçapão referente à calha do cofre abre e fecha após 2 segundos.

Caso a operação seja cancelada pelo usuário o alçapão de devolução de moedas é aberto e fecha após 2 segundos. Na situação de inserção de moeda inválida, o moedeiro segrega esta moeda para uma terceira calha que direciona a moeda diretamente para a calha de devolução de moedas ao usuário mostrado na figura 4-2.

Figura 4-29: Compartimento de devolução de moedas ao usuário.

4.2. Software de gerenciamento do parquímetro

O software do micro-controlador foi desenvolvido para gerenciar tarefas como fluxo de dados entre o micro-controlador e periféricos, controle do relógio do parquímetro, controle das calhas e cofre de moedas, armazenamento e geração de relatório. A figura 4-3 dá uma idéia geral do *software* desenvolvido. Os algoritmos detalhados estão no Apêndice B.

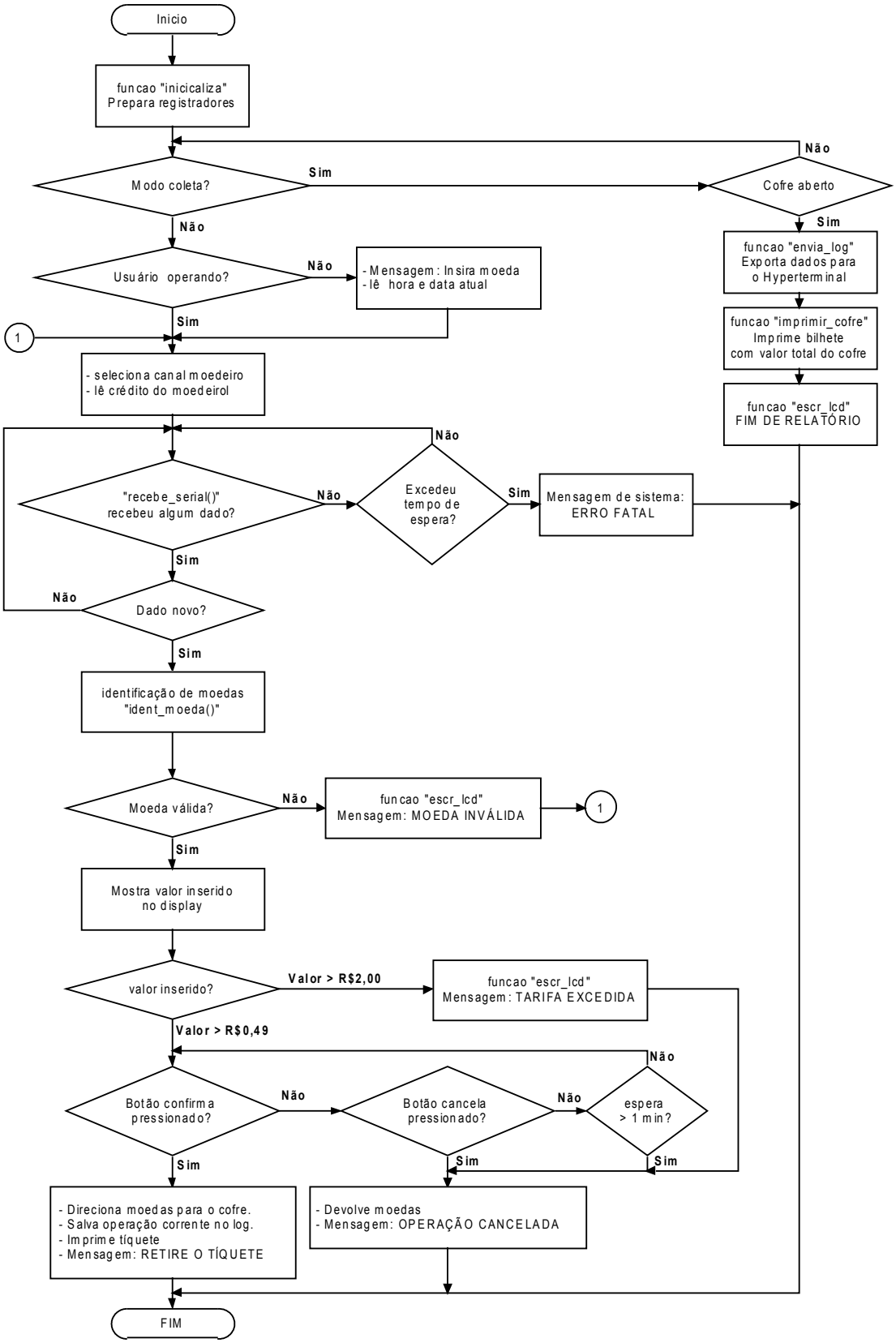


Figura 4-30: Resumo da estrutura do *software*.

4.3. Teste de eficiência do moedeiro

Através de um elenco de 10 moedas de cada tipo foram realizados ensaios a fim de verificar a eficiência do moedeiro. O ensaio como grupo de

moedas foi feito cinco vezes, o que significa o total de cinquenta inserções de moedas de cada valor, conforme Tabela4-1 e figura 4-4.

Tabela 4-6: Tabela de ensaio de eficiência do moedeiro

Moeda		Moedas rejeitadas	Moedas Aceitas	Eficiência
1	R$0,05 prata	2	48	96%
2	R$0,05 dourada	4	46	92%
3	R$0,10 prata	3	47	94%
4	R$0,10 dourada	2	48	96%
5	R$0,25 prata	4	46	92%
6	R$0,25 dourada	0	50	100%
7	R$0,50 fina	1	49	98%
8	R$0,50 grossa	0	50	100%
9	R$1,00	3	47	94%

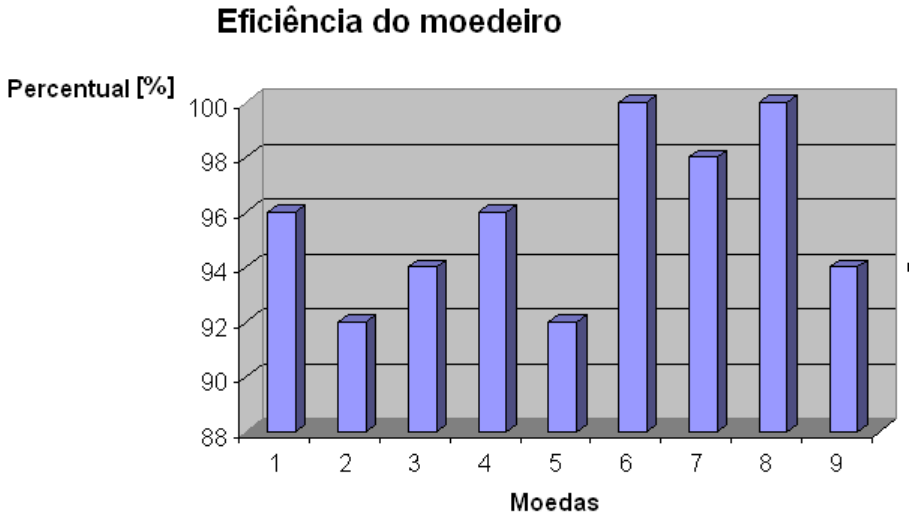


Figura 4-31: Gráfico de eficiência do moedeiro.

4.4. Impressão de bilhetes

O bilhete do parquímetro segue o padrão conforme figura 4-5. No exemplo abaixo o usuário retirou o bilhete comprovante às 8hs e 59 minutos do dia 21 de Junho de 2010 e inseriu R$0,55 o que lhe deu direito de permanecer 33 minutos, ou seja, até as 9 horas e 32 minutos.

Figura 4-32: Exemplo de bilhete gerado pelo parquímetro.

4.5. Teste de Ckecksum do moedeiro

Para teste do cálculo de *checksum* da comunicação entre o moedeiro e o micro-controlador foi utilizado um programa de comunicação serial chamada Device Tester que assume o papel do moedeiro conforme figura 4-6. O Device Tester permite enviar um código manipulado para o micro-controlador do parquímetro. A partir deste recurso foi preparado o comando de resposta similar ao do moedeiro, mas com o último *byte* (*checksum*) diferente do que o moedeiro enviaria ao micro-controlador.

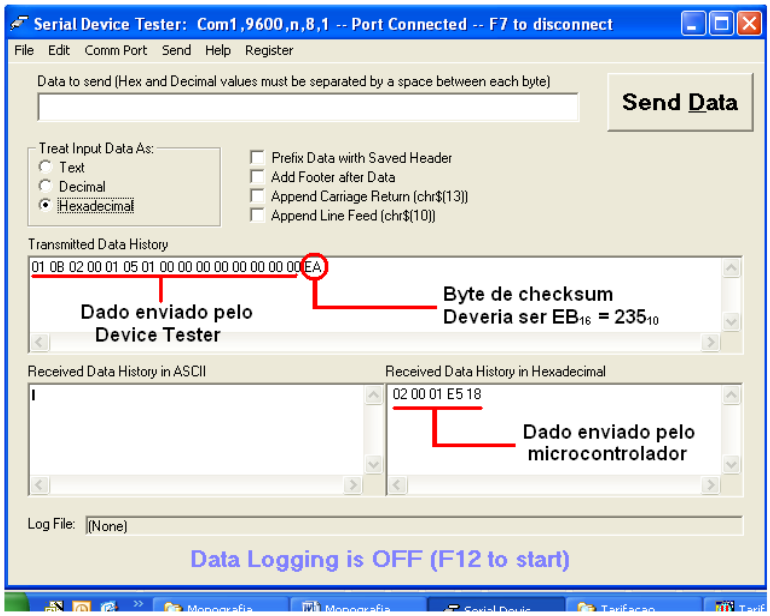


Figura 4-33: Teste de *checksum.*

De acordo com o dado retornado pelo Device Tester pode-se notar que o último byte está incorreto, pois a equação abaixo mostra o cálculo com números decimais que o valor correto é 235.

$$CKS = 256 - [1 + 11 + 2 + 0 + 1 + 5 + 1 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + (0)]$$

$$CKS = 256 - [21] = 235 = EBh$$

O teste gerou a mensagem de erro no *checksum* conforme figura 4-7.

Figura 4-34: Mensagem de erro de *checksum*.

4.6. Testes do relógio

O parquímetro exibe a hora e data atual do parquímetro na tela, conforme figura 4-8. Para testar as previsões de feriados, finais de semanas bem como virada de ano foi ajustado o RTC do parquímetro para as 18 horas e 26 minutos do dia 31 de Dezembro de 2009, e logo foi inserido R$2,00. Conforme tabela 4-2 esta tarifa dá o direito de estacionar por 2 horas. O resultado obtido está exposto na figura 4-9.

Figura 4-35: Tela do parquímetro em funcionamento.

Figura 4-36: Bilhete de estacionamento de véspera de feriado.

Como era previsto o bilhete impresso permite o estacionamento até as 9 horas e 26 minutos do dia 02 de Janeiro de 2010, pois no momento da retirada do bilhete faltavam 34 minutos para acabar o horário de funcionamento do parquímetro e o dia seguinte é feriado. Portanto, pode-se perceber que o parquímetro comportou-se de maneira correta e que a previsão de feriado foi atendida.

4.7. Operação do parquímetro

Por se tratar de um equipamento acessado por diversos tipos de pessoas o procedimento de retirada de bilhete foi desenvolvido para se tornar rápido e com o mínimo de comandos necessário por parte do usuário. A figura 4-10 mostra um fluxograma explicativo das decisões que o parquímetro adotará durante o acesso ao parquímetro pelo usuário.

O fluxograma leva em conta que o parquímetro está operando dentro do horário e data de funcionamento, em caso de feriados, fora de horário comercial ou em finais de semana, o display exibe a mensagem FORA DE OPERAÇÃO. Caso alguma moeda seja inserida nesta condição a mesma será devolvida imediatamente. A tabela 4-2 exibe as condições de operação do parquímetro baseado nas Áreas Azuis de Porto Alegre – RS para servir de parâmetro de análise dos resultados

Tabela 4-7: Condições de funcionamento do parquímetro.

Dados do parquímetro				
Tempo mínimo	Tempo máximo	Fração de tempo	Período de funcionamento	
30 minutos R$0,50	120 minutos R$2,00	3 minutos R$0,05	Segunda-feira até Sexta-feira (Exceto Feriados)	8:00 às 19:00

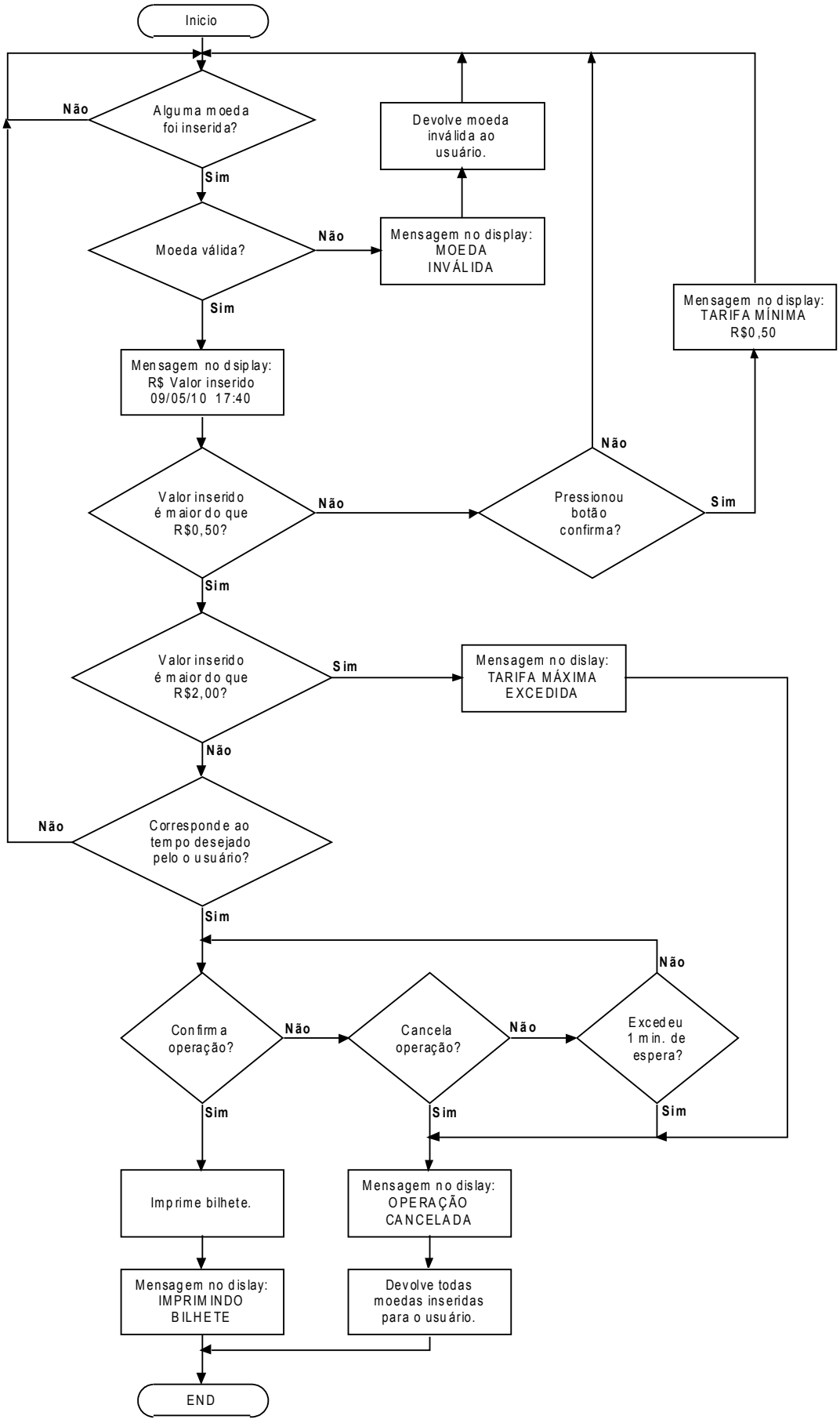


Figura 4-37: Fluxograma de funcionamento do parquímetro.

4.7.1 Exemplo de uma seção completa

A seguir uma operação de retirada de bilhete é demonstrada para ilustrar os testes de recursos que o parquímetro oferece quando manipulado por um usuário

1 - Tela inicial: o parquímetro em repouso exibe a tela com a data e hora atual, além de uma mensagem para inserir moedas conforme figura 4-11.

Figura 4-38: Tela inicial da operação de teste.

2 – Moedas utilizadas: Foram inseridas 2 moedas de R$0,25 e uma moeda R$0,01, mostradas na figura 4-12, para testar condições de moedas inválidas.

Figura 4-39: Moedas utilizadas na operação de teste.

3- Inserção da moeda de R$0,01: Ao inserir a moeda, uma mensagem de moeda inválida foi exibida conforme figura 4-13.

Figura 4-40: Aviso de moeda inválida na operação de teste.

4- Tentativa de confirmar operação com uma moeda de R$0,25: Ao tentar confirmar a operação com apenas 25 centavos inseridos o parquímetro avisou ao usuário que o valor mínimo da tarifa é de R$0,50 conforme figura 4-14.

Figura 4-41: Aviso de valor mínimo não inserido na operação de teste.

5- Inserção das 2 moedas de R$ 0,25: Após a inserção das 2 moedas a tela exibiu o valor total inserido e o tempo correspondente de estacionamento, além da data e hora atual mostrado na figura 4-15.

Figura 4-42: Tela de exibição de dados da operação de teste.

6- Confirmação da operação: Ao confirmar a operação, o parquímetro exibiu a mensagem de aviso de impressão do bilhete, mostrada na figura 4-16 e forneceu o tíquete comprovante da figura 4-17.

Figura 4-43: Aviso de bilhete sendo impresso.

Figura 4-44: Bilhete impresso da operação de teste.

Portanto, o usuário de posse do tíquete pode deixar seu veículo estacionado até as 9 horas e 54 minutos do dia 18 de Junho de 2010.

4.8. Geração do relatório

Para teste do funcionamento do *log* foram efetuadas 28 seções de compra de bilhete. Considerando o horário de funcionamento dos parquímetros de Porto Alegre, o mesmo foi configurado para funcionar das 8hs até as 19hs do dia 21 de Junho de 2010. Neste período ocorreu uma variação na quantidade de retiradas de bilhetes que está detalhado no gráfico da figura 4-18 e no *log* da figura 4-19.

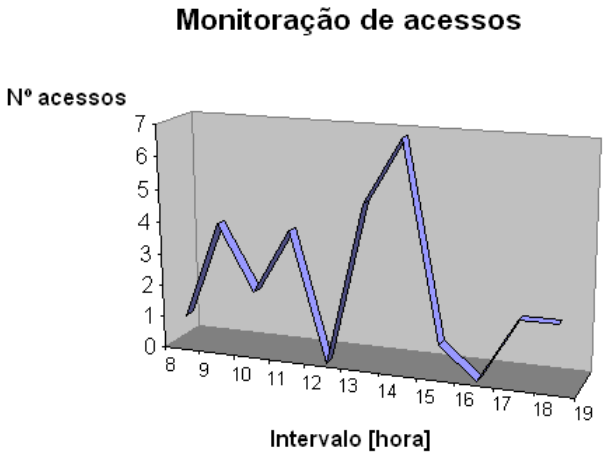


Figura 4-45: Gráfico de número de acessos ao parquímetro.

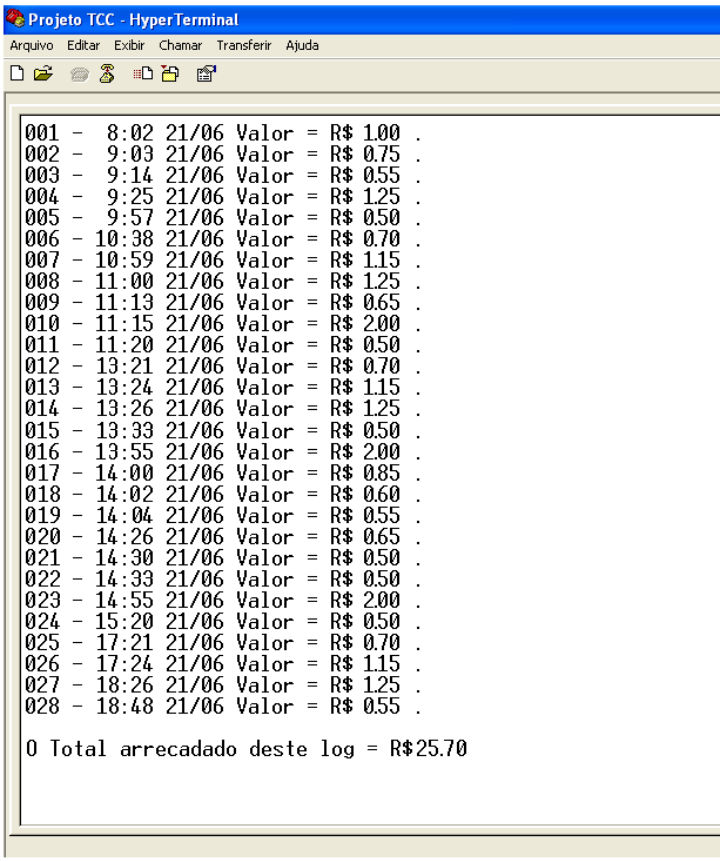


Figura 4-46: Exemplo de *log* gerado.

Após a geração do *log* no Hyperterminal o parquímetro imprimiu um recibo contendo a data de abertura do cofre, horário e o valor total coletado do cofre conforme figura 4-20.

Figura 4-47: Comprovante de coleta de cofre.

5.CONSIDERAÇÕES FINAIS

Os resultados dos testes realizados no parquímetro indicam que o mesmo atingiu o objetivo proposto. As expectativas de desempenho do equipamento, a qualidade bem como o tempo de execução foi atendida. Avaliando a eficiência do sistema e os recursos que o presente projeto apresentou após sua finalização, faz concluir que o mesmo possui características de funcionamento similares aos parquímetros do mercado atual.

O parquímetro foi desenvolvido baseado em uma unidade central de processamento na qual foi possível integrar três periféricos eletrônicos com diferentes protocolos de comunicação a um micro-controlador de um canal de comunicação serial. A verificação de *checksum* desenvolvida no *software* do micro-controlador fez com que a comunicação entre o moedeiro e o micro-controlador seja confiável. A qualidade do tíquete desenvolvido é satisfatória, pois exibe informações essenciais de forma nítida para a visualização do agente fiscalizador da Área azul. O parquímetro oferece flexibilidade ao usuário, pois é possível cancelar a operação corrente sem perder as moedas inseridas, além disto, os créditos restantes após horário de funcionamento do parquímetro é transferido para o próximo dia útil.

Através de ensaios realizados verificou-se que a eficiência de leitura de moedas é satisfatória, e que se houver alguma falha na leitura de uma moeda válida, esta é devolvida ao usuário. Alguns itens de segurança do parquímetro apresentam grande potencial de melhorias para evitar fraudes. Um item importante a ser aprimorado é acesso ao canal de relatório via PC durante a coleta de moedas, para viabilizar esta segurança sugere-se o cadastro de senhas no *software* do parquímetro. Ao solicitar a coleta de moedas, o parquímetro solicitará uma senha ao técnico a fim de certificar que há uma pessoa autorizada efetuando a coleta.

O *log* de operações realizadas no parquímetro é coletado por um PC com um aplicativo de comunicação serial RS-232. Os dados coletados possibilitam estudos estatísticos e podem ser mais bem disponibilizados para a empresa administradora a fim de aperfeiçoar o processo de montagem dos gráficos de análise.

Outra sugestão de melhoria do parquímetro é a adição de uma nova forma de tarifação através da instalação de leitor de cédulas, pois o circuito de multiplexação serial desenvolvido tem um canal disponível. Para este recurso é necessário instalar um sistema de devolução de troco a fim de atender situações de estacionamento com tarifação menor do que R$2,00. Também é sugerida a instalação de uma memória EEPROM na unidade de processamento. Esta melhoria garante a conservação dos dados importantes como, por exemplo, o valor armazenado no cofre em uma eventual falta de energia elétrica, além disto, possibilita maior capacidade de armazenamento de *log.*

REFERÊNCIAS

[1] Wikipédia. História do Parquímetro. 2010 Disponível em: <http://en.wikipedia.org/wiki/Parking_meter >. Acesso em: 2 maio 2010.

[2] PINTO, Luis Fernando Mirault BARBATO, Augusto César Ribeiro. Controle Metrológico em parquímetros. Disponível em: <http://www.inmetro.gov.br/producaointelectual/obras_intelectuais/321_obraIntelectual.pdf>. Acesso em: 1 maio 2010.

[3] HEREDIA,Miguel. Parquímetro centralmente manejado, Seattle, WA, los E.E.U.U. Disponível em: <http://www.arqhys.com/contenidos/parquimetro-manejado.html>. Acesso em: 1 maio 2010.

[4] Rek Parking Ltda. Serviços.2009. Disponível em: <http://www.rekparking.com.br/produtos.php>. Acesso em: 1 maio 2010.

[5] ESTAPAR. Parquímetro Eletrônico. 2007. Disponível em: <http://www.estapar.com.br/servicos_area_azul_eletronica.htm>. Acesso em: 1 maio 2010.

[6] AMANO,Time & Parking.S.A. Cale MP104 Compact. 2007. Disponível em: <http://www.amano-spain.com/mainsite/prod_Parquimetro.php>. Acesso em: 1 maio 2010.

[7] Lapaza empreendimentos Ltda. Epark Parking System. 2008 Disponível em: <http://www.lapaza.com.br/eparq.pdf>. Acesso em: 2 maio 2010.

[8] CONTROLS, Money. SR3 Type Technical Manual. 2008 Disponível em: <http://www. moneycontrols.com/en/products/view/38/SR3+Coin+Acceptor >. Acesso em: 8 mar. 2010.

[9] APS. CP205-MRS. 2008 Disponível em: <http://www. aps-printers.com/on-multi/Home/support/technicalmanuals.html>. Acesso em: 29 mar. 2010.

[10] MAXIM,Dallas. DS1302 Trickle-Time-Charge Timekeeping Chip. 2008. Disponível em: <http:// datasheets.maxim-ic.com/en/ds/DS1302.pdf>. Acesso em: 16 maio 2010.

OBRAS CONSULTADAS

Leonardo Peronio Bassin – 2.4 GHz Wireless Modbus - Canoas, 2009.

Ludimila La Rosa Centeno– Desenvolvimento de um sistema para identificação de caracteres utilizando redes neurais artificiais - Canoas, 2006.

GLOSSÁRIO

Mifare: tecnologia de cartões inteligentes sem contato.

Bit menor unidade de informação da computação.

Byte: conjunto de 8 bits.

Nible: quatro bits de um byte.

Memória *Flash*: memória eletrônica não volátil.

Pull up: resistor ligado entre +Vcc e o ponto específico do circuito.

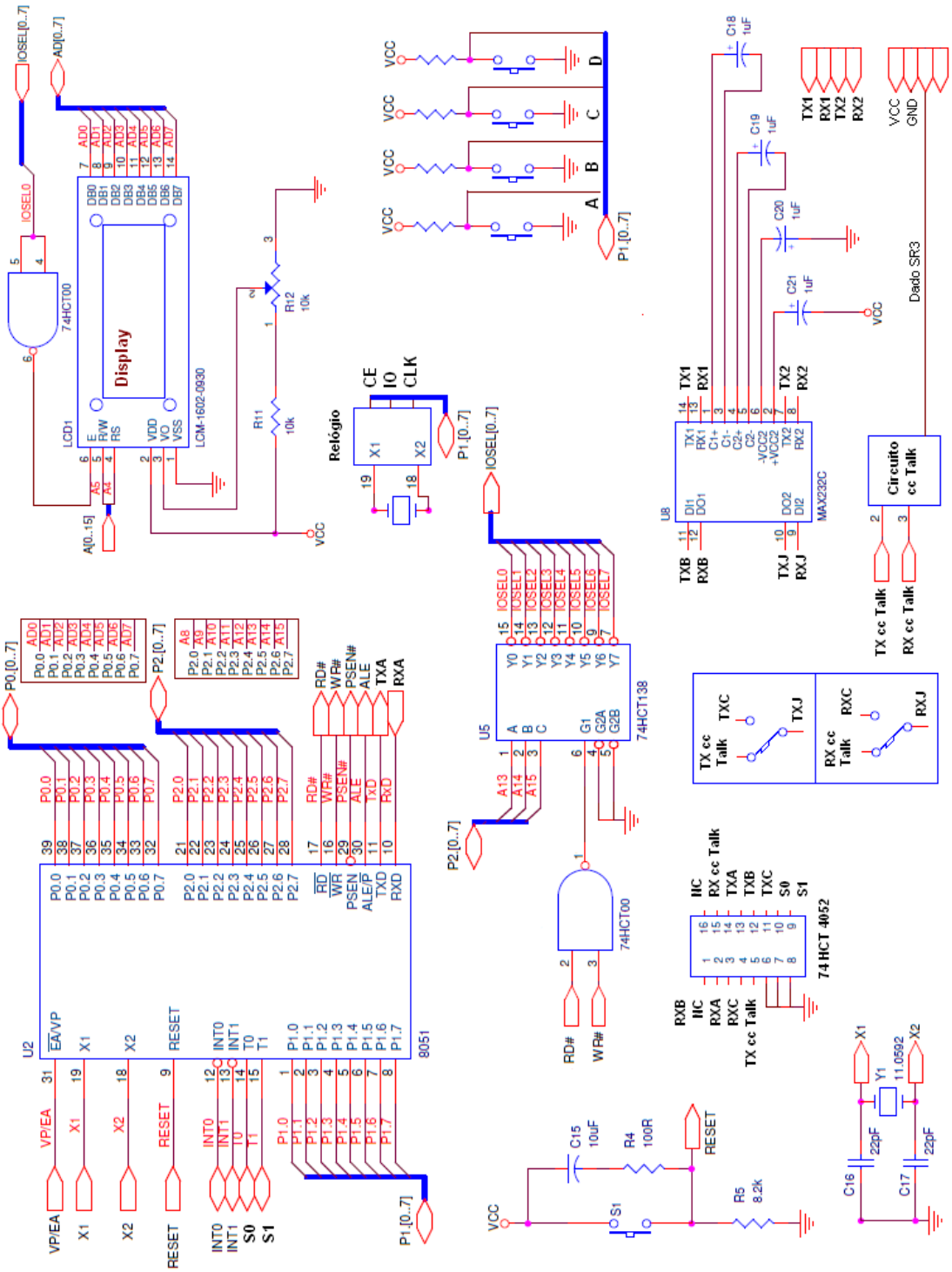
Software: programa de execução de tarefas dentro do micro-controlador.

Log: registro de eventos.

Checksum: verificação da soma dos dados transmitidos ou recebidos.

String: uma série de bytes.

APÊNDICE A – ESQUEMA ELETROELETRÔNICO



APÊNDICE B – SOFTWARE

```
/*-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------*/
//                                                                                                                             //
//                                          PARQUIMETRO ELETRÔNICO                                                             //
//                                    ULBRA - TRABALHO DE CONCLUSÃO - 2010/1                                                   //
//                                                                                                                             //
/*-------------------------------------------------------------------------------------------------------------------------------*/
/*-------------------------------------------------------------------------------------------------------------------------------*/
#include "definicoes.h"
/*-------------------------------------------------------------------------------------------------------------------------------
                                    DECLARAÇÃO DAS VARIÁVEIS GLOBAIS
-------------------------------------------------------------------------------------------------------------------------------*/
bit f_fim_delay;                                        //flag - indica fim do delay
bit memoria_cheia = OFF;                                //indica status da memoria do LOG

unsigned char ano, mes, dia_mes, dia_semana;            //variaveis de data do RTC
unsigned char hora, minuto, segundo;                    //variaveis de tempo do RTC
unsigned char prev_sec = 99;
unsigned char buffer_recebe_uc[17];                     //armazena dado para recepcao serial
unsigned char evento_ul = 0;                            //variavel de ultimo evento no moedeiro
unsigned char somatorio;                                //aramzena valor inserido pelo usuario
unsigned int total_arrecadado;                          //variavel de valor armazenado no cofre
unsigned long tempo_limite;                             //tempo de delay informado
unsigned long tempo_contado;                            //contador de timeout, incrementado cada interrup.
xdata unsigned char buffer_envio[TAM_BUFFER_ENVIO];     //armazena dado para transmissao serial

//-------------------------------------------------------------------------------------------------------------------------------
void serial_moedeiro (void);            //rotina SELECAO DE CANAL DO MUX p/ comunicar c/ moedeiro
void serial_relatorio (void);           //rotina SELECAO DE CANAL DO MUX p/ comunicar c/ PC

#define POSICAO_FINAL 1700      //limite de gravacoes de operacoes no log
xdata unsigned char memoria_log [POSICAO_FINAL + 1];    //variavel de armazenamento do LOG

/*-------------------------------------------------------------------------------------------------------------------------------
                                            ALGORITMO  PRINCIPAL
-------------------------------------------------------------------------------------------------------------------------------*/
void main (void)
{
        bit controle_loop;
        bit operacao = OFF;         // variavel indica se o usuáro esta operando o parquimetro ou nao.
        bit valor_maior;
        unsigned int i;

        write_clk_regs();                               // inicializa tempo de data do RTC
        escr_lcd (1,1,"  PARQUIMETRO   ");              // mensagem de inicialização
        escr_lcd (2,1,"       ULBRA          ");
        delay(25000);                                   // delay de aproximadamente 0,5 segundos
        controle_loop = ON;                             // ativa loop principal
        i = 0;

        while (controle_loop==ON)
        {
                if(operacao==OFF)
                {
```

```
            escr_lcd (1,1,"  INSIRA MOEDAS ");          // solicita  que insira moedas
            read_clk_regs();                            // chama leitura do rtc e mostra
                                                        //...valores no display
}


serial_moedeiro ();                                     // Chama rotina de seleção de canal do mux p/
                                                        //...comunicar c/ moedeiro
buffer_envio[0]=0x02;                                   // transmite ao moedeiro o endereço destino
buffer_envio[1]=0x00;                                   // transmite ao moedeiro o num. de bytes c/ dados
buffer_envio[2]=0x01;                                   // transmite ao moedeiro o endereço de origem
buffer_envio[3]=0xE5;                                   // transmite ao moedeiro  código 229
buffer_envio[4]=0x18;                                   // transmite ao moedeiro  o valor de checksum
transmite_serial(5);                                    // chama a rotina de transmissao serial.5 bytes
while (recebe_serial() == NO_OK);                       // aguarda retorno dos dados requisitados

if (evento_ul != buffer_recebe_uc[4])                   // verifica se houve um novo evento no
                                                        //...moedeiro
{
            operacao=ON;                                // indica que há um usuario operando o
                                                        // ... parquimetro
            evento_ul = buffer_recebe_uc[4];            // salva evento atual p/ ser monitorando no
                                                        // ... prox. evento do moedeiro
            ident_moeda();                              // chama rotina de identificacao da moeda
                                                        // ...inserida no moedeiro

            if (valor_maior == OFF)                     // verifica se valor inserido eh menor que
                                                        // ... 50 centavos

           {
                        escr_lcd (1,1,"Valor Inserido  ");  // escreve no display o campo "valor
                                                            // ... inserido"
                        display_valor(2,6,somatorio);       // mostra no display o valor inserido
            }
}

if          (somatorio>=50)                             // verifica se valor total inserido e maior ou
                                                        // ...igual a 50 centavos
{
            valor_maior = ON;                           // status de valor minimo atingido
            escr_lcd (1,1,somatorio);                   // mensagem na esquerda do display
                                                        // ... indicando valor inserido
            escr_lcd (1,5,tempo_est);                   // mensagem na direita do display indicando
                                                        // ... tempo de permanencia

            if          (CONFIRMA == 0)                 // verifica se foi pressionado botao
                                                        // ... CONFIRMA
            {
                        armazena_log();                 // chama rotina de armazen. relatorio - LOG
                        operacao=OFF;                   // indica que acabou a sessão
                        somatorio=0;                    // zera o valor total inserido
                        valor_maior = OFF;              // zera o indicador de tarifa minima R$0,050.
}
}


if          (COFRE == 0)                                // Se sensor do cofre for aberto vai rotina
                                                        // ... de impressao do comprovante
                                                        // ... de coleta e emite relatorio eletronico
{

            imprimir();                                 // chama rotina de impressao de tiquete
            line_feed(1);                               // comando de nova linha da impressora
            serial_relatorio ();                        // chama rotina de SELECAO DE CANAL
                                                        // ... DO MUX para comunicar com o PC
            escr_lcd (1,1,"    GERANDO    ");          // informa no display que recebeu comando
            escr_lcd (2,1,"   RELATORIO   ");          // ...para rotina de relatorio.

            delay(50000);                               // delay de aproximadamente 1 segundo
```

```
                        envia_log();
                        escr_lcd (1,1,"       FIM          ");  // informa no display que terminou rotina
                        escr_lcd (2,1,"  RELATORIO   ");  // ... emissao relatorio
                        delay(50000);                          // pausa de aproximadamente 1 segundo

                }
        }
}

void serial_moedeiro (void)          // rotina de SELECAO DE CANAL DO MUX para comunicar com o moedeiro
{
        S0_SERIAL=1;                 // INICIA EM ZERO A SELECAO DE CANAL DO
                                     // ... MUX 74HCT 4052 - coloca A (pino 10) para 0.
        S1_SERIAL=0;                 // INICIA EM ZERO A SELECAO DE CANAL DO
                                     // ... MUX 74HCT 4052 - coloca B (pino 9)  para 0.
}

void serial_relatorio (void)         //rotina de SELECAO DE CANAL DO MUX para comunicar com o PC
{
        S0_SERIAL=1;                 // SELECAO DE CANAL DO MUX 74HCT 4052
                                     // ... PARA O PC - coloca A (pino 10) para 1.
        S1_SERIAL=1;                 // SELECAO DE CANAL DO MUX 74HCT 4052
                                     // ... PARA O PC - coloca B (pino 9 ) para 1.
}

/*-------------------------------------------------------------------------------------------------------------------------------
                                ALGORITMO DE TRATAMENTO DE LOG - Relatorio
-------------------------------------------------------------------------------------------------------------------------------*/

unsigned char formata_log (unsigned int valor)          // funcao que formata valor financeiro R$
{
        unsigned char valor_formatado[5];
        unsigned int num_temporario,num_temporario_1;
        unsigned int teste;

        if (valor > 9999)
        {
                valor = 0;
        }
        valor_formatado[0] = (valor / 1000) + 0x30;
        num_temporario     = (valor % 1000);
        valor_formatado[1] = (num_temporario / 100) + 0x30;
        num_temporario_1 = (num_temporario % 100);
        valor_formatado[2] = (num_temporario_1 / 10) + 0x30;          // pega resto e transforma em ascii
        valor_formatado[3] = (num_temporario_1 % 10) + 0x30;          // pega resto e transforma em ascii
        valor_formatado[4] = '\0';                                    // byte finalizador

        return (OK);
}

bit armazena_log(void)      //algoritmo de armazenagem do LOG
{
        if (memoria_cheia == OFF)
        {
                memoria_log[posicao_log + 0] = hora;
                memoria_log[posicao_log + 1] = minuto;
                memoria_log[posicao_log + 2] = dia_mes;
                memoria_log[posicao_log + 3] = mes;
                memoria_log[posicao_log + 4] = somatorio;
                total_arrecadado = total_arrecadado + (unsigned int) somatorio;

                escr_cent(1,1,total_arrecadado);
                delay(50000);
                posicao_log = posicao_log + 5;
        }

        if (posicao_log == POSICAO_FINAL)
        {
                memoria_cheia = ON;
```

```
		}

		return (OK);
}

void envia_log (void)		//algoritmo de envio de LOG
{
		unsigned int operacao = 1;
		unsigned int envio;
		xdata unsigned char buffer_auxiliar[20];
		int tmp_hora,tmp_minuto,tmp_dia_mes,tmp_mes,tmp_somatorio;
		envio = 0;

		while (envio != posicao_log)
		{
			tmp_hora		= memoria_log[envio + 0];
			tmp_minuto		= memoria_log[envio + 1];
			tmp_dia_mes		= memoria_log[envio + 2];
			tmp_mes			= memoria_log[envio + 3];
			tmp_somatorio	= memoria_log[envio + 4];

			memset(buffer_envio,0,TAM_BUFFER_ENVIO);

			sprintf(buffer_envio,"%.3d - %2d:%.2d %.2d/"//15
		,operacao
		,tmp_hora
		,tmp_minuto
		,tmp_dia_mes);

			sprintf(buffer_auxiliar,"%.2d Valor = R$ %.3d ."//
			  ,tmp_mes
			  ,tmp_somatorio);

			strcat(buffer_envio,buffer_auxiliar);

			buffer_envio[38] = 10;
			buffer_envio[39] = 13;

			transmite_serial(40);

			operacao ++;
			envio = envio + 5;
		}

		buffer_envio[0] = 10;//nOVA LINHA
		buffer_envio[1] = 13;
		transmite_serial(2);

		memset(buffer_envio,0,TAM_BUFFER_ENVIO);
		sprintf(buffer_envio,"O Total arrecadado deste log = R$ %.5d    ",total_arrecadado);

		buffer_envio[40] = 10;
		buffer_envio[41] = 13;
		transmite_serial(42);

		posicao_log = 0;
		memoria_cheia = OFF;
		total_arrecadado = 0;

		return;
}

//-----------------------------------------------------------------------------------------------------------------------------------------------------------*/


/*-----------------------------------------------------------------------------------------------------------------------------------------------------------
		ident_moeda();
		Função para identificar a moeda inserida no moedeiro a cada novo evento ocorrido
-----------------------------------------------------------------------------------------------------------------------------------------------------------*/
```

```
void ident_moeda (void)
{
        unsigned char moeda_at;

                switch (buffer_recebe_uc[5])
                {
                        case(CENT_5_PR):
                        moeda_at=5;
                        somatorio=somatorio+moeda_at;
                        delay(5000);
                        break;

                        case(CENT_5_AM):
                        moeda_at=5;
                        somatorio=somatorio+moeda_at;
                        delay(5000);
                        break;

                        case(CENT_10_PR):
                        moeda_at=10;
                        somatorio=somatorio+moeda_at;
                        delay(5000);
                        break;

                        case(CENT_10_AM):
                        moeda_at=10;
                        somatorio=somatorio+moeda_at;
                        delay(5000);
                        break;

                        case(CENT_25_PR):
                        moeda_at=25;
                        somatorio=somatorio+moeda_at;
                        delay(5000);
                        break;

                        case(CENT_25_AM):
                        moeda_at=25;
                        somatorio=somatorio+moeda_at;
                        delay(5000);
                        break;

                        case(CENT_50_FINA):
                        moeda_at=50;
                        somatorio=somatorio+moeda_at;
                        delay(5000);
                        break;

                        case(CENT_50_GROS):
                        moeda_at=50;
                        somatorio=somatorio+moeda_at;
                        delay(5000);
                        break;

                        case(REAL_1_NOVA):
                        moeda_at=100;
                        somatorio=somatorio+moeda_at;
                        delay(5000);
                        break;

                        default:
                        escr_lcd (1,1,"   MOEDA       ");
                        escr_lcd (2,1,"  INVALIDA     ");
                        delay(50000);
                        break;
                }
}

/*-----------------------------------------------------------------------------------------------------------------------------------------------
```

```
|                                   COMANDOS DA IMPRESSORA - CP205-MRS
+-------------------------------------------------------------------------------------------------------------------------------------------------------*/
void printing_speed (unsigned char n)
{
        buffer_envio[0] = GS;
        buffer_envio[1] = '/';
        buffer_envio[2] = n;                    // n varia de 0 a 47

        transmite_serial(3);
}

void continuous_printing (unsigned char n)
{
        buffer_envio[0] = GS;
        buffer_envio[1] = 'C';
        buffer_envio[2] = n;                    // n pode ser 0 ou 1

        transmite_serial(3);
}

void print_intensity (unsigned char n)
{
        buffer_envio[0] = GS;
        buffer_envio[1] = 'D';
        buffer_envio[2] = n;                    // n varia de 0 a 255

        transmite_serial(3);
}

void resets_printer (void)
{
        buffer_envio[0] = ESC;
        buffer_envio[1] = '@';

        transmite_serial(2);
}

void serial_communication (void)
{
        buffer_envio[0] = GS;
        buffer_envio[1] = 'B';
        buffer_envio[2] = 0x83;                 // DSR/DTR, 1 stopbit, 9600 bps

        transmite_serial(3);
}

void font_bank (unsigned char n)
{
        buffer_envio[0] = ESC;
        buffer_envio[1] = '%';
        buffer_envio[2] = n;                    // n = 0 - 8x8 e n = 1 - 7x8

        transmite_serial(3);
}

void international_character (unsigned char n)
{
        buffer_envio[0] = ESC;
        buffer_envio[1] = 'R';
        buffer_envio[2] = n;                    // n varia de 0 a 12

        transmite_serial(3);
}

void line_spacing (unsigned char n)
{
        buffer_envio[0] = ESC;
        buffer_envio[1] = '3';
        buffer_envio[2] = n;                    // n varia de 3 a 15
```

```
	transmite_serial(3);
}

void character_spacing (unsigned char n)
{
	buffer_envio[0] = ESC;
	buffer_envio[1] = SP;
	buffer_envio[2] = n;			// n varia de 1 a 16

	transmite_serial(3);
}

void print_modeA (unsigned char n)
{
	buffer_envio[0] = ESC;
	buffer_envio[1] = '!';
	buffer_envio[2] = n;

	transmite_serial(3);
}

void rotated_characters (unsigned char n)
{
	buffer_envio[0] = ESC;
	buffer_envio[1] = '{';
	buffer_envio[2] = n;

	transmite_serial(3);
}

void line_feed (void)
{
	buffer_envio[0] = LF;

	transmite_serial(1);
}

void cancel_print (void)
{
	buffer_envio[0] = CAN;
	transmite_serial(1);

}


/*-------------------------------------------------------------------------
|	imprime_texto() - .
+-------------------------------------------------------------------------*/
void imprime_texto (unsigned char *mens)
{
	unsigned char i;

	for (i=0; i<21; i++)
		buffer_envio[i] = *(mens + i);

	transmite_serial(21);
	delay(50000);
}

  void bilhete_teste (void)
  {
		char linha;

		S0_SERIAL=0;		//SELECAO DE CANAL DO MUX 74HCT 4052
					// ... PARA IMPRESSORA - coloca A (pino 10) para 0.
```

```
                S1_SERIAL=0;                //SELECAO DE CANAL DO MUX 74HCT 4052
                                            // ... PARA IMPRESSORA - coloca B (pino 9 ) para 0.


        cancel_print();
        resets_printer();
        printing_speed(5);
        continuous_printing(1);
        print_intensity(255);
        serial_communication();
        font_bank(1);
        international_character(0);
        line_spacing(3);
        character_spacing(2);
        print_modeA(0x20);
        rotated_characters(0);


        delay(100000);

        line_feed();

        for (linha=0; linha<9; linha++)          // ---- chama impressão do logotipo
        {
                buffer_envio[0]=0x09;
                buffer_envio[0]=0x09;
                transmite_serial(2);
                delay(10000);

                imp_logotipo(linha,0);        //----- logotipo --------
                transmite_serial(18);
                delay(10000);
                line_feed();
        }

        line_feed();
        line_feed();
        imprime_texto("      ENTRADA:       ");
        line_feed();

        texto_data (dia_mes,mes,ano);       // imprime data atual
        line_feed();
        texto_entrada(hora,minuto);
        line_feed();
        line_feed();

        imprime_texto("     VALOR PAGO:     ");
        texto_soma(somatorio);

        line_feed();
        line_feed();

        imprime_texto("---------------------");
        imprime_texto("       SAIDA:        ");
        line_feed();
        texto_data (dia_mes,mes,ano);
        line_feed();
}



/*---------------------------------------------------------------------------------------------------------------------------------------------
                                        ALGORITMO DE CONTROLE DO RTC
---------------------------------------------------------------------------------------------------------------------------------------------*/
#include "definicoes.h"
/*---------------------------------------------------------------------------------------------------------------------------------------------
|                       reset_3w()- função utilzada para comando de reset
```

```
|                                                    e habilita o RTC
+-----------------------------------------------------------------------------------------------------------------------------------------------------*/

void reset_3w()    //------- reset e habilita 3 pinos de interface ------
{
  SCLK = 0;
  CE = 0;
  CE = 1;
}

/*-----------------------------------------------------------------------------------------------------------------------------------------------------
|                    wbyte_3w()- função utilzada para ESCREVER um byte no  RTC
+-----------------------------------------------------------------------------------------------------------------------------------------------------*/
void wbyte_3w(uchar W_Byte)        /* ------ escreve um byte no RTC ------- */
{
uchar i;
          for(i = 0; i < 8; ++i)
          {
                    IO = 0;
                    if(W_Byte & 0x01)
                    {
                              IO = 1;
                    }
                    SCLK = 0;
                    SCLK = 1;
                    W_Byte >>= 1;
    }
}
/*-----------------------------------------------------------------------------------------------------------------------------------------------------
|                    wbyte_3w()- função utilzada para LER um byte do  RTC

-----------------------------------------------------------------------------------------------------------------------------------------------------*/
uchar     rbyte_3w()          /* ------- le um byte do RTC -------- */
{
uchar i;
uchar R_Byte;
uchar TmpByte;

          R_Byte = 0x00;
          IO = 1;
          for(i = 0; i < 8; i++)
          {
                    SCLK = 1;
                    SCLK = 0;
                    TmpByte = (uchar)IO;
                    TmpByte <<= 7;
                    R_Byte >>= 1;
                    R_Byte |= TmpByte;
          }
          return R_Byte;
}
/*-----------------------------------------------------------------------------------------------------------------------------------------------------
|                    () - função utilzada para LER em modo BURST
-----------------------------------------------------------------------------------------------------------------------------------------------------*/

void read_clk_regs()          /* ---- le e mostra registradores do RTC ---- */
{
                    reset_3w();
                    wbyte_3w(0xBF);

                    segundo = rbyte_3w();
                    minuto   = rbyte_3w();
                    hora      = rbyte_3w();

                    dia_mes             = rbyte_3w();
                    mes                 = rbyte_3w();
                    dia_semana          = rbyte_3w();
                    ano                 = rbyte_3w();
                    reset_3w();
```

```
                    if(segundo != 1)
                    {
                            display_dezena (2,1,dia_mes);           // escreve dia no display
                            escr_lcd (2,3,"/");
                            display_dezena (2,4,mes);               // escreve mes no display
                            escr_lcd (2,6,"/");
                            display_dezena (2,7,ano);               // escreve ano no display

                            escr_lcd (2,9,"   ");

                            display_dezena (2,12,hora);             // escreve hora no display

                            if (dois_pontos_relogio)
                                    escr_lcd (2,14,":");
                            else
                                    escr_lcd (2,14," ");

                            display_dezena (2,15,minuto);           // escreve minuto no display
                            prev_sec = segundo;
                    }

}

/*---------------------------------------------------------------------------------------------------------------------------------------
|                   write_clk_regs() - função utilzada para inicializar os registradores
                                    do RTC e entrar com com valores    de ANO, MES, DIA, etc.
---------------------------------------------------------------------------------------------------------------------------------------*/
void    write_clk_regs()   /* --- initialize time & date from user entries --- */
{
        ano=10;                         // entra com o ano (0-99):          ANO
        mes=6;                          // entra com o mes (1-12):          MES
        dia_mes=27;                     // entra com dia mes (1-31):        DIA
        dia_semana=5;                   // entra com dia semana (1-7):
        hora=9;                         // entra com a hora (1-24):         HORA

        hora = hora & 0x3f;             // configura para modo 24hs
        minuto=23;                      // entra com o minuto (0-59):       MINUTO
        segundo=00;                     // entra com os segundos (0-59):

        reset_3w();                     //comando de reset e habilita o RTC
        wbyte_3w(0x8e);                 //controle de registro
        wbyte_3w(0);                    //disabilita protecao de escrita
        reset_3w();                     //comando de reset e habilita o RTC
        wbyte_3w(0x90);                 // registrador de carga de baterias
        wbyte_3w(0xab);                 // habiliat, 2 diodos, resistores de 8k
        reset_3w();                     // comando de reset e habilita o RTC

        wbyte_3w(0xbe);
        wbyte_3w(segundo);
        wbyte_3w(minuto);
        wbyte_3w(hora);
        wbyte_3w(dia_mes);
        wbyte_3w(mes);
        wbyte_3w(dia_semana);
        wbyte_3w(ano);
        wbyte_3w(0);
        reset_3w();                     // comando de reset e habilita o RTC
}
/*---------------------------------------------------------------------------------------------------------------------------------------
        calc_checksum();        Função de verificacao do CHECKSUM do moedeiro
---------------------------------------------------------------------------------------------------------------------------------------*/

void calc_checksum (void)
{
        unsigned char j=0;                      //variável de varredura de posição do vetor "buffer_recebe"
        unsigned char cks[17];          //variável de armazenamento dos 16 bytes do moedeiro
        bit st_check= NOK;                      //status do checksum
```

```
                while(j=0; j<16;j++)
                {
                        cks[j]=buffer_recebe_uc[j]  //armazena byte do moedeiro na posicao "j" do vetor cks
                        if(j<16)
                        {
                                soma_cks=soma_cks=cks[j]
                        }
                }

                if(soma_cks != cks[16])
                {
                        st_check=NOK
                        escr_lcd (1,1,"  CHECKSUM    ");
                        escr_lcd (2,1,"  INVALIDO    ");
                }
                st_check=OK
}
```

ANEXO A – REGISTRADORES DO RTC

Table 3. Register Address/Definition

RTC

READ	WRITE	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	RANGE
81h	80h	CH	10 Seconds			Seconds				00–59
83h	82h		10 Minutes			Minutes				00–59
85h	84h	12/$\overline{24}$	0	10 $\overline{\text{AM/PM}}$	Hour	Hour				1–12/0–23
87h	86h	0	0	10 Date		Date				1–31
89h	88h	0	0	0	10 Month	Month				1–12
8Bh	8Ah	0	0	0	0	0	Day			1–7
8Dh	8Ch	10 Year				Year				00–99
8Fh	8Eh	WP	0	0	0	0	0	0	0	—
91h	90h	TCS	TCS	TCS	TCS	DS	DS	RS	RS	—

ANEXO B – TABELA DE ERROS DO MOEDEIRO

Result A	Result B	Event	Type
0	0	Master inhibit active	Status
0	1	Bill returned from escrow	Status
0	2	Invalid bill (due to validation fail)	Reject
0	3	Invalid bill (due to transport problem)	Reject
0	4	Inhibited bill (on serial)	Status
0	5	Inhibited bill (on DIP switches)	Status
0	6	Bill jammed in transport (unsafe mode)	Fatal Error
0	7	Bill jammed in stacker	Fatal Error
0	8	Bill pulled backwards	Fraud Attempt
0	9	Bill tamper	Fraud Attempt
0	10	Stacker OK	Status
0	11	Stacker removed	Status
0	12	Stacker inserted	Status
0	13	Stacker faulty	Fatal Error
0	14	Stacker full	Status
0	15	Stacker jammed	Fatal Error
0	16	Bill jammed in transport (safe mode)	Fatal Error
0	17	Opto fraud detected	Fraud Attempt
0	18	String fraud detected	Fraud Attempt
0	19	Anti-string mechanism faulty	Fatal Error
0	20	Barcode detected	Status

ANEXO C – BUFFER DO MOEDEIRO