INOVANDO A TESTABILIDADE DE MÓDULOS MICROPROCESSADOS

Miguel Antonio Sovierzoski
e Aurélio Charão

RESUMO

O artigo apresenta uma nova maneira de testar módulos microprocessados. Graças ao "Special Bootstrap Mode" (modo especial de partida), que é apenas um dos recursos oferecidos pela família de microcontroladores M68HC11 da Motorola, foi possível implementar um teste funcional que se mostrou muito flexível no chão de fábrica. Essa técnica está em uso no equipamento de teste do PABX Euroset Line 8 da Equitel. O artigo foi apresentado pelos autores no II Seminário Brasileiro de Caracterização em Microeletrônica, em Curitiba, no dia 7 de dezembro de 1995.

Miguel Antonio Sovierzoski, *engenheiro da EQUITEL - Equipamentos e Sistemas de Telecomunicações SA, professor do CEFET/PR.*
Aurélio Charão, *professor do Curso de Pós-Graduação em Engenharia Elétrica e Informática Industrial do CEFET/PR.*

1. INTRODUÇÃO

Na área de engenharia de testes, deparamo-nos com certos impasses ao elaborar a estratégia de teste de módulos microprocessados. Através de diversos fatores, dentre os quais o tempo estimado de teste, o custo estimado do teste por módulo, e a escala de produção, determinamos a filosofia de teste do módulo.

Apresentamos a técnica inovadora e extremamente versátil, utilizada no equipamento de teste que realiza o teste funcional do PABX Euroset Line 8 (ESL8) da Equitel.

A estratégia de teste adotada deveu-se aos recursos de testabilidade disponibilizados na família de microcontroladores M68HC11 da Motorola. Tais recursos permitem inicializar o microcontrolador em quatro modos de operação. A seleção do modo de operação ocorre durante o *reset*. Utilizamo-nos do *special bootstrap mode* que carrega um programa pelo canal serial.

Na sequência descrevemos as técnicas tradicionais para testes funcionais de módulos microprocessados, apresentamos o microcontrolador MC68HCP11A1 e descrevemos a técnica utilizada para realizar o teste funcional nos módulos baseados neste microcontrolador.

2. ESTRATÉGIAS PARA O TESTE FUNCIONAL DE MÓDULOS MICROPROCESSADOS

Diversas técnicas são utilizadas para realizar o teste funcional em módulos microprocessados. Abordamos de forma sucinta cada técnica para fornecer uma visão geral.

2.1 Teste funcional com o *firmware* do módulo

Nesta técnica, o módulo em teste está executando o seu *firmware* (o *firmware* do produto). O equipamento de teste acessa os sinais periféricos do módulo e na realização dos programas de teste deve-se observar as condições de uso, respeitando os protocolos e temporizações. Normalmente esta técnica apresenta um tempo de teste mais elevado.

2.2 Teste funcional com *firmware* de teste

Esta técnica pressupõe que o processador do módulo executará um *firmware* de teste, estabelecendo algum protocolo com o equipamento de teste. As restrições anteriores de execução dos protocolos do produto não existem, devendo ser estabelecido algum protocolo que simule as condições de uso e de testabilidade desejadas.

Este teste pode ser implementado de duas maneiras, dependendo da estratégia do produto:

1 - Programa de teste incluso. *O programa de teste está junto com o programa do produto na memória. O programa de teste é executado através da seleção de algum estrape ou opção de configuração do módulo. A memória do produto deve ser dimensionada para comportar também o programa de teste, elevando o custo.*

2 - Substituição do firmware. *Esta alternativa propõe a troca do firmware do módulo pelo firmware de teste. Aparentemente é mais viável que a solução anterior, pois não é necessário reservar memória para comportar o programa de teste. Porém, há uma tendência para a utilização de PROM com encapsulamento PLCC sem soquetes, sendo soldadas diretamente na placa de circuito impresso, o que inviabiliza a utilização desta técnica nos dias atuais.*

2.3 Teste funcional com emulador

A alternativa de utilizar um equipamento emulador para realizar os testes está entrando em desuso. Além da tendência de utilizar componentes com encapsulamento PLCC sem soquete, sendo soldado diretamente na placa, há o custo elevado do emulador e da sua manutenção. E o equipamento deve ser operado por pessoal técnico qualificado.

3. MICROCONTROLADOR M68HC11

A referência M68HC11 descreve uma família de microcontroladores de 8 bits da Motorola. De forma geral, todos os componentes da família apresentam em um único chip a CPU, memória RAM e EEPROM, *timers*, canal serial, fontes de interrupção e *ports* bidirecionais. As diferenças entre os componentes estão na capacidade dos circuitos de memória RAM, EEPROM e EPROM/ROM, a existência do circuito de *watch-dog* (COP), do conversor A/D, da interface periférica serial, e de barramento de dados e endereços multiplexados ou não.

A **figura 1** apresenta um diagrama em blocos da família M68HC11.

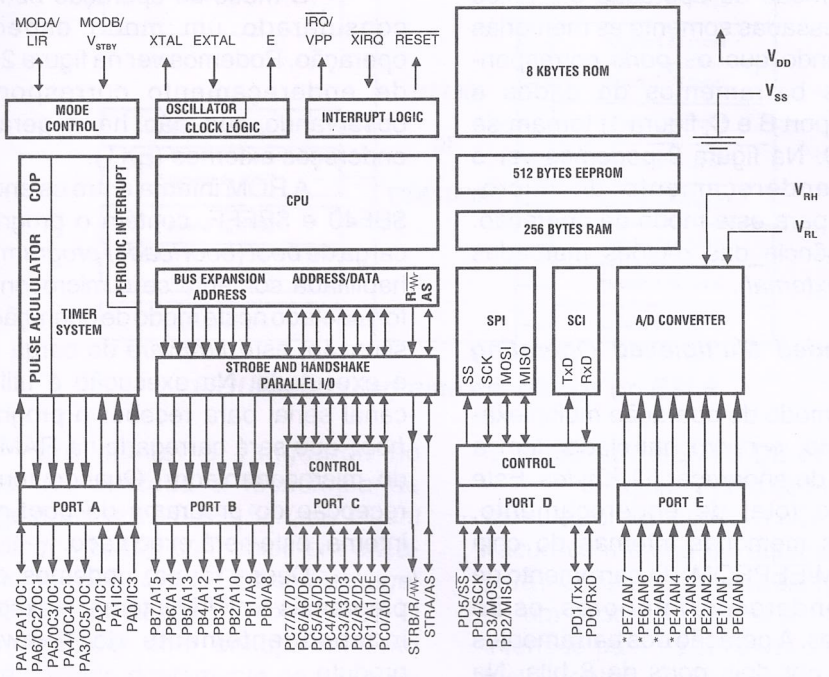

FIGURA 1 - Diagrama em blocos da família M68HC11

A família M68HC11 tem hoje 46 componentes. Podemos citar como exemplo o microcontrolador utilizado no ESL8. O microcontrolador MC68HCP11A1 possui 256 bytes de RAM interna, 512 bytes de EEPROM, timer de 16 bits, oito canais para conversor A/D de 8 bits, interface serial e circuito de *watch-dog*.

3.1 Modos de Operação Selecionados Durante o *Reset*

Os componentes da família M68HC11 apresentam quatro modos de operação: *single-chip operating mode*, *expanded multiplexed operating mode*, *special bootstrap operating mode*, e *special test operating mode*.

A seleção por *hardware* de cada modo de operação é realizada através de níveis digitais em dois sinais de entrada do microcontrolador (MODB e MODA, **figura 1**). Estes sinais devem estar selecionados durante o *reset*. O modo de operação selecionado durante o *reset* pode ser alterado por *software*, realizando escritas em um registrador interno do microcontrolador. Porém, nem todas as possibilidades de alteração de modo de operação são permitidas.

A **figura 2** apresenta os mapas de endereçamento de memória para a família M68HC11, nos vários modos de operação. As áreas indicadas como EXT (*external*) são as áreas de endereçamento externo do microcontrolador neste modo de operação. Caso o componente utilizado não apresente determinado recurso interno de memória, a área de endereçamento correspondente estará disponível.

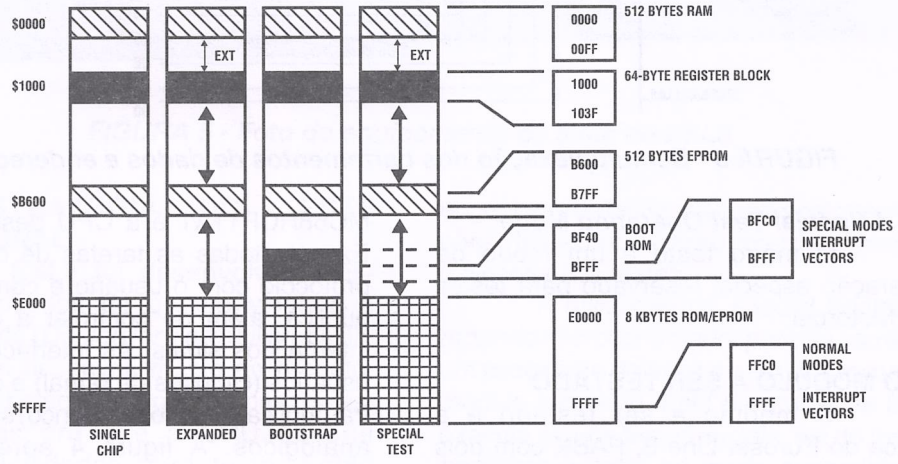

FIGURA 2 - Mapeamento de memória da família M68HC11

3.1.1 *Single-Chip Operating Mode*

No modo de operação em único chip são acessadas somente as memórias internas, sendo que os *ports* correspondentes aos barramentos de dados e endereços (port B e C, figura 1) tornam-se ports de I/O. Na figura 2 podemos ver o mapa de endereçamento do microcontrolador para este modo de operação. Note a ausência das regiões marcadas com EXT (*external*).

3.1.2 *Expanded Multiplexed Operating Mode*

No modo de operação multiplexado expandido, o microcontrolador tem a capacidade de endereçar 64 *Kbytes*. Este é o espaço total de endereçamento, incluindo as memórias internas do chip (RAM e ROM/EEPROM). O barramento de dados e endereços inferiores estão multiplexados. A geração dos barramentos é realizada por dois *ports* de 8 bits. Na figura 3 podemos ver como realizar a demultiplexação dos barramentos de dados e endereços.

3.1.3 *Special Bootstrap Operating Mode*

O modo de operação *bootstrap* é considerado um modo especial de operação. Podemos ver na figura 2 o mapa de endereçamento correspondente, observando que não há a geração de endereços externos (EXT).

A ROM interna entre os endereços $BF40 e $BFFF, contém o programa de carga de *boot* (*boot loader program*), sendo habilitada somente se o microcontrolador for resetado neste modo de operação. Nesta situação, este *firmware* de carga de boot é executado. Na execução é utilizado o canal serial para receber o programa de *boot*, que será carregado na RAM interna do microcontrolador. Quando terminar a recepção do programa de *boot* na RAM interna, este será executado.

Dessa forma, podemos carregar programas na RAM interna, e executá-los, independentemente do *firmware* do produto.

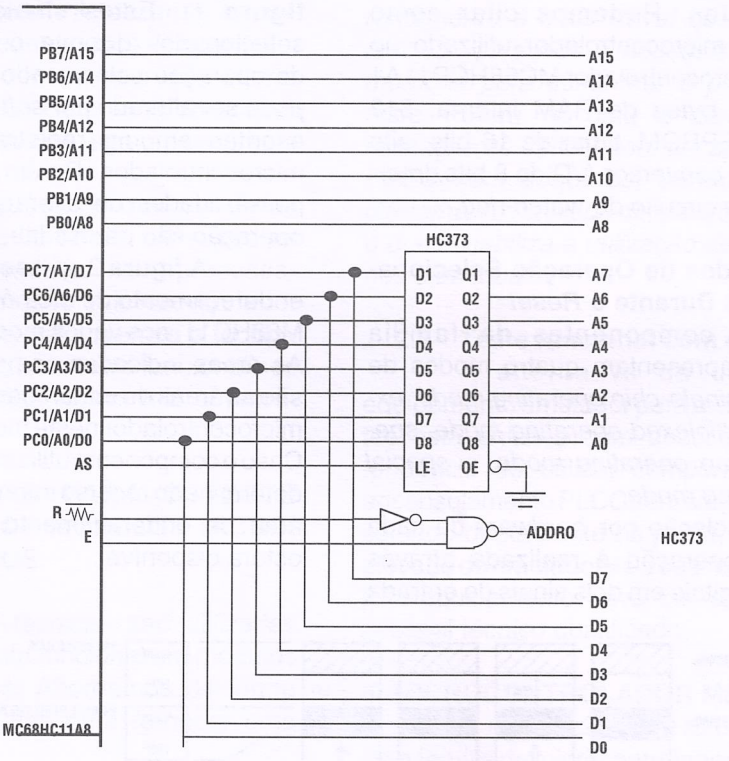

FIGURA 3 - Demultiplexação dos barramentos de dados e endereços

3.1.4 *Special Test Operating Mode*

O modo teste é um modo de operação especial reservado para testes na Motorola.

4. O MÓDULO A SER TESTADO

O módulo a ser testado é a placa do Euroset Line 8, PABX com dois troncos e oito ramais analógicos, produzido pela Equitel. O microcontrolador MC68HCP11A1 é a CPU deste módulo. Executa todas as tarefas de controle do protocolo com o usuário e com a central pública, além de controlar a comutação interna dos sinais. As interfaces com os usuários (circuitos de ramal) e com a central pública (circuitos de tronco) são circuitos analógicos. A figura 4 apresenta um diagrama em blocos com os principais circuitos do ESL8.

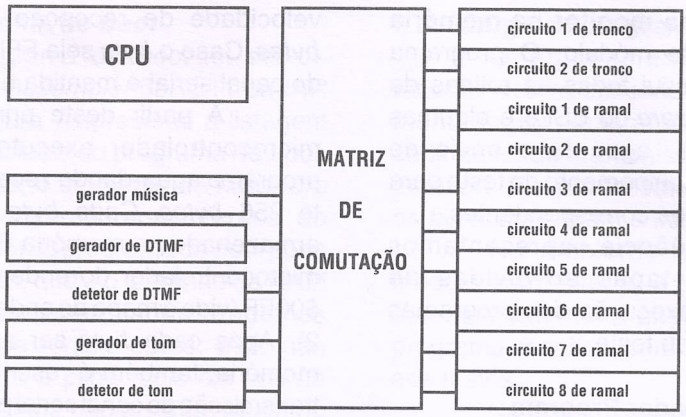

FIGURA 4 - Diagrama em blocos do ESL8

Para manter a vanguarda no produto, foi utilizada a tecnologia de montagem em superfície (*Surface Mount Technology* - SMT), com chips em encapsulamento SOP (*Small Outline Package*) e PLCC (*Plastic Leaded Chip Carrier*). O microcontrolador com encapsulamento PLCC é soldado diretamente na placa de circuito impresso. A PROM com o *firmware* do produto, também no encapsulamento PLCC ainda está sendo soqueteada, mas assim que for verificada em campo a estabilidade do *software* do produto, esta deverá ser soldada diretamente na placa de circuito impresso.

5. O EQUIPAMENTO DE TESTE

O equipamento de teste para este módulo foi concebido com um *fixture* a vácuo com cama de agulhas de contato para realizar as conexões elétricas. Consiste de um *rack* com a CPU e os módulos de interface para os circuitos do módulo sob teste, fontes programáveis e a interface homem-máquina, composta de *display* de cristal líquido, teclado e impressora. A figura 5 apresenta uma foto do equipamento de teste, mostrando o *rack* com as interfaces, o *fixture* com a IHM e o módulo a ser testado.

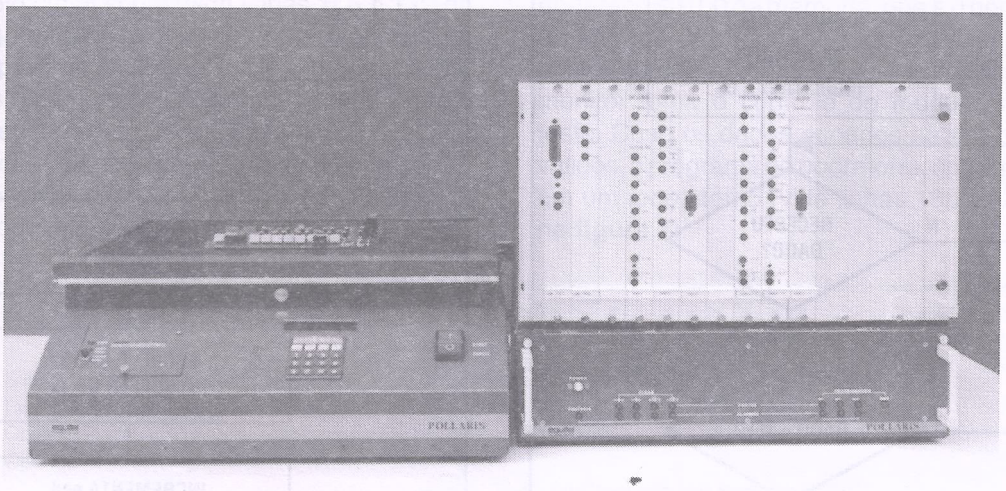

FIGURA 5 - Foto do equipamento de teste do ESL8

6. A ESTRATÉGIA DO TESTE FUNCIONAL

O ESL8 foi desenvolvido utilizando conceitos de projeto visando à testabilidade (*Design for Testability* - DFT). Na fase de desenvolvimento do produto, foram estabelecidas algumas premissas. Uma delas era que a CPU deveria possuir facilidades para teste. Então foram estabelecidos, em conjunto com a engenharia de testes, os pontos estratégicos para o teste de circuitos e ficou determinada a filosofia dos testes.

A utilização do *special bootstrap mode* do microcontrolador permite carregar um **programa de boot na RAM interna** do módulo sob teste. Este programa de *boot* executa alguns testes iniciais, sai do modo *bootstrap* e entra no modo expandido, e então carrega outro programa denominado

programa monitor na memória RAM externa do módulo. O programa monitor, que possui todas as rotinas de acesso ao *hardware* do ESL8 e algumas rotinas de teste, aguarda o envio de comandos pelo equipamento de teste para executar as rotinas correspondentes.

Na seqüência, apresentamos detalhes das etapas envolvidas na transferência e execução dos programas para o módulo sob teste.

6.1 Boot Loader Program

O *boot loader program* que está disponível na ROM interna nos endereços $BF40 a $BFFF é executado após a seleção do *special bootstrap mode* durante o estado de *reset* do microcontrolador.

Na figura 6 vemos o fluxograma do *boot loader program* para o microcontrolador utilizado. Na execução do *boot loader program*, após as inicializações e a programação da velocidade de operação do canal serial para 7812 bps, é aguardado receber dados pelo canal serial. O primeiro *byte* recebido pelo microcontrolador não é ecoado e determina a velocidade de recepção dos próximos *bytes*. Caso o *byte* seja FFh, a velocidade do canal serial é mantida em 7812 bps.

A partir deste primeiro dado, o microcontrolador executa um *loop* de programa aguardando receber exatamente 256 *bytes*. Cada *byte* recebido será armazenado na memória RAM interna do microcontrolador do endereço $0000 até $00FF (vide o mapa de endereços na figura 2). Após cada *byte* ser armazenado na memória, também é escrito no *buffer* de transmissão do canal serial para ser ecoado. A técnica de ecoar os dados recebidos pelo microcontrolador permite ao equipamento de teste verificar a comunicação. Caso ocorra algum erro na comunicação, não é possível recuperar, devendo ser iniciado todo o procedimento de teste novamente.

Após ser ecoado o último *byte* recebido, temos o programa de *boot* montado na RAM interna. O programa de carga de *boot* passa o controle para o programa de *boot* ao executar um salto para o endereço $0000 da RAM interna.

Detalhes do programa de *boot* são abordados na seqüência.

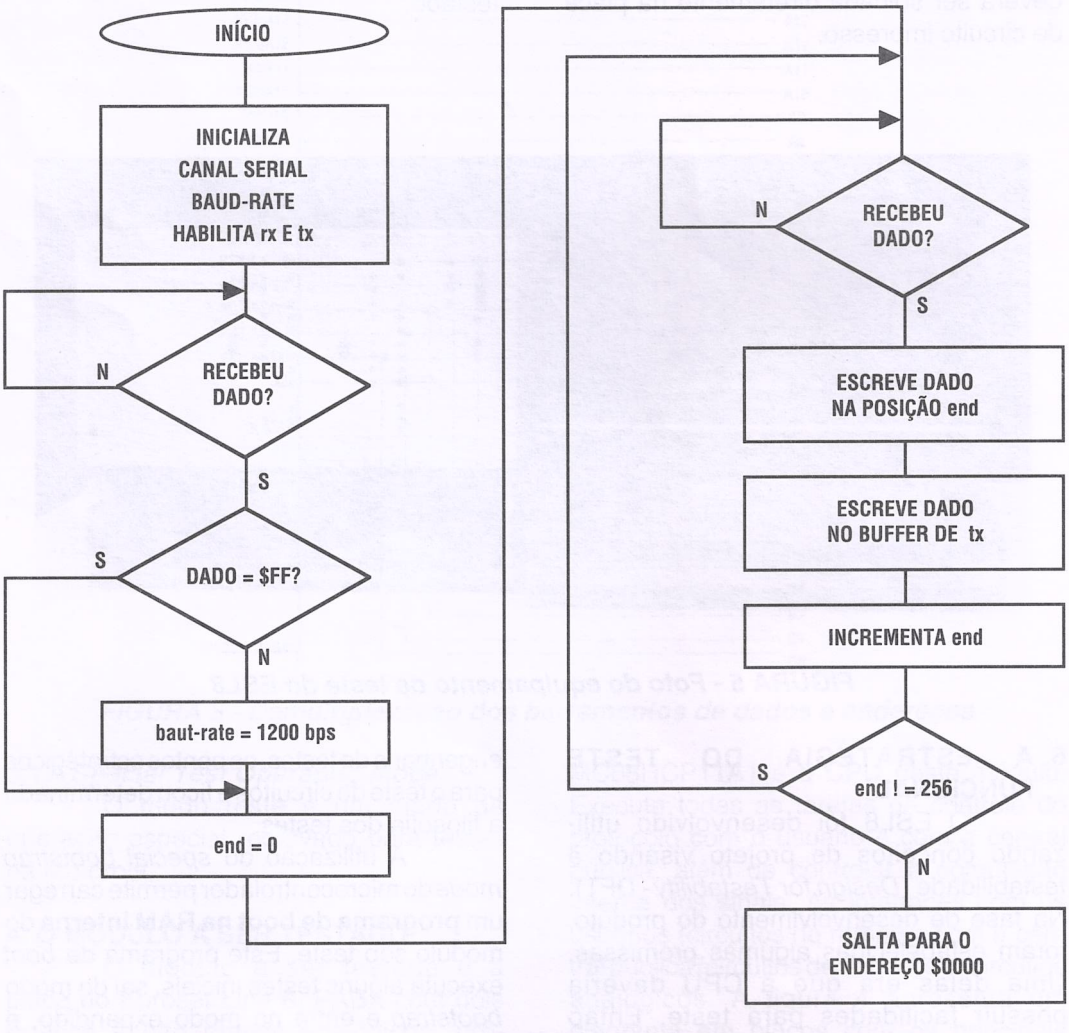

FIGURA 6 - Fluxograma do boot loader program

6.2 Programa de *boot*

O programa de *boot* foi desenvolvido com o *Assembler* 68HC11 da IAR Systems. A figura 7 apresenta a listagem das partes principais do programa de *boot*. Neste momento, o ambiente de *hardware* está restrito aos recursos internos do microcontrolador. O programa de *boot* pode ter até 256 bytes, que é o tamanho da RAM interna do microcontrolador. Podemos utilizar todos os registradores, mas não podemos utilizar a pilha, pois ela fica na RAM externa, ainda não testada. Muda do *special bootstrap mode* para o modo *expanded* para poder gerar os barramentos externos (vide linhas 30 a 33 na figura 7), executa o teste da memória RAM externa, e carrega o programa monitor na memória externa.

6.2.1 Teste da memória externa

Primeiramente, o programa de *boot* executa o teste da memória RAM externa do módulo. Este teste garante a integridade dos barramentos entre o microcontrolador e a memória, bem como a integridade funcional da memória RAM e circuitos auxiliares.

Durante a realização do teste, o programa de *boot* altera a velocidade do canal serial para 125 kbps (vide linha 55 na figura 7). Ao término do teste de memória, o programa de *boot* envia o resultado pelo canal serial (vide linhas 106 e 112 da figura 7). Caso o resultado seja satisfatório, o programa de *boot* inicia o procedimento de recepção do programa monitor. Caso o resultado não seja satisfatório, há algum problema nos circuitos digitais, não sendo possível dar continuidade nos testes do módulo.

6.2.2 Carga do programa monitor

O equipamento de teste ao receber o *byte* indicando que o teste de memória foi satisfatório, inicia o procedimento de transmissão do programa monitor. No processo de comunicação os *bytes* continuam a ser ecoados.

No ambiente de *hardware* do módulo sob teste, já é possível utilizar a memória externa, a qual irá armazenar o programa monitor, para dar continuidade aos testes.

Na figura 7 vemos o procedimento de carga do programa monitor com início na linha 122.

Dois *bytes* com o tamanho do programa monitor são enviados logo no início do procedimento. É executado o *loop* controlado de recepção de dados e montagem na memória externa (linhas 138 até 157). Os dados são armazenados a partir do endereço $0100. Observe que o programa de *boot* está entre $0000 e $00FF.

Como os *bytes* transmitidos são ecoados, o equipamento de teste verifica o que está sendo montado na memória do módulo sob teste.

Após transferir todo o programa monitor, o equipamento de teste envia um *byte* de validação dos dados ecoados. Caso os dados ecoados sejam válidos, o programa de *boot* executa um salto para o endereço $0100 da memória, que é o ponto de entrada do programa monitor. A partir deste momento, a execução do programa monitor toma o controle do módulo sob teste. Caso os dados ecoados não sejam válidos, o programa de *boot* morre, entrando em um *loop* eterno (vide linhas 166 e 167 **na figura 7**).

```
Micro Series 6801 Assembler V2.00/DOS                    05/Nov/95  09:17:58

Source          = bootsbm.asm
List            = bootsbm.lst
Object          = bootsbm.obj
Options         =

                                                         (c) Copyright IAR Systems 1990

  ...
  12 0000              main:
  13 0000                          P68H11                ; Diretiva do tipo de microcontrolador
  14 0000 8603                     LDAA #$03             ;  Aumenta o tempo de guarda do circuito
  15 0002 B71039                   STAA $1039            ;  de Watch-dog para 1,049 segundos.
  16
  17 0005 8608                     LDAA #$08             ;  Reposiciona o banco de registros internos
  18 0007 B7103D                   STAA $103D            ;  a partir do endereço $8000
  19 000A 18CE8000                 LDY #$8000            ;  O registro IY aponta para o banco de registros
  ...
  30 0017 C6F5                     LDAB #$F5             ;  Carrega B com F5h e escreve no registro HPRIO
  31 0019 F7803C                   STAB $803C            ;  Sai do  special bootstrap mode
  32 001C C6B5                     LDAB #$B5             ;  Carrega B com B5h e escreve no registro HPRIO
  33 001E F7803C                   STAB $803C            ;  Vai para o expanded multiplexed mode
```

```
...
 55 0039 5F              CLRB                    ; Escreve no registro BAUD do canal serial
 56 003A F7802B          STAB $802B              ; alterando o Baud Rate para 125 kbps
...
106 007E                 teste_RAM_ok:           ; Resposta de teste de RAM OK!
107 007E 8640            LDAA #$40               ; Carrega A com 40h
108 0080 181F2EC0        BRCLR $2E,Y,#$C0,*-4    ; Espera esvaziar o buffer de transmissão
    0084 F7
109 0085 B7802F          STAA $802F              ; Escreve A no buffer de transmissão
110 0088 7E0098          JMP carga_monitor       ; Salta para a carga do programa monitor
111 008B                 monitor_erro:           ; Resposta de erro de carga do monitor
112 008B                 teste_RAM_erro:         ; Resposta de teste de RAM ERRO!
113 008B 863F            LDAA #$3F               ; Carrega A com 3Fh
114 008D                 resposta:
115 008D 181F2EC0        BRCLR $2E,Y,#$C0,*-4    ; Espera esvaziar o buffer de transmissão
    0091 F7
116 0092 B7802F          STAA $802F              ; Escreve A no buffer de transmissão
117 0095 7E00EF          JMP loop_eterno         ; O programa de boot morre
118
119
120                      ========== CARGA DO PROGRAMA MONITOR ==========
121
122 0098                 carga_monitor:
123 0098 CE0100          LDX #$0100              ; Ponteiro início do programa monitor
124 009B 18CE8000        LDY #$8000              ; Ponteiro dos registros
125
126 009F                 le_tamanho:
127 009F 181F2E20        BRCLR $2E,Y,#$20,*-4    ; Espera a recepcao de dado pelo canal serial
    00A3 F7
128 00A4 B6802F          LDAA $802F              ; Lê o primeiro byte do tamanho do monitor
129 00A7 B77FFE          STAA $7FFE              ; Armazena este dado no endereço 7FFEh
130 00AA B7802F          STAA $802F              ; Escreve este dado no canal serial (ECO)
131 00AD 01              NOP
132 00AE 01              NOP
133 00AF 181F2E20        BRCLR $2E,Y,#$20,*-4    ; Espera a recepção de dado pelo canal serial
    00B3 F7
134 00B4 B6802F          LDAA $802F              ; Lê o segundo byte do tamanho do monitor
135 00B7 B77FFF          STAA $7FFF              ; Armazena este dado no endereço 7FFFh
136 00BA B7802F          STAA $802F              ; Escreve este dado no canal serial (ECO)
137 00BD 01              NOP
138 00BE                 recebe_dado:
139
140
141                      ========== RESETA O COP ==========
142 00BE 8655            LDAA #$55               ; procedimento para resetar o timer do COP
143 00C0 B7803A          STAA $803A              ; escreve 55h no registrador do COP
144 00C3 43              COMA                    ; complementa o registro A
145 00C4 B7803A          STAA $803A              ; escreve AAh  no registrador do COP
146
147
148 00C7 181F2E20        BRCLR $2E,Y,#$20,*-4    ; Espera a recepção de dado pelo canal serial
    00CB F7
149 00CC F6802F          LDAB $802F              ; Lê o dado recebido
150 00CF E700            STAB $00,X              ; Grava o dado no endereço apontado por IX
151 00D1 01              NOP
152 00D2 E600            LDAB $00,X              ; Lê o dado da memória
153 00D4 01              NOP
154 00D5 F7802F          STAB $802F              ; Escreve o dado no canal serial (ECO)
155 00D8 08              INX                     ; Incrementa o endereço
156 00D9 BC7FFE          CPX $7FFE               ; Compara com o tamanho
157 00DC 26E0            BNE recebe_dado         ;
158 00DE 01              NOP
159 00DF 01              NOP
160 00E0                 valida_dados:           ; Espera a validação dos dados ecoados
161 00E0 181F2E20        BRCLR $2E,Y,#$20,*-4    ; Espera a recepção de dados pelo canal serial
    00E4 F7
162 00E5 F6802F          LDAB $802F              ; Lê o dado do canal  serial (dado de validação)
163 00E8 C150            CMPB #$50               ; Verifica a validade dos dados do monitor
164 00EA 2706            BEQ  monitor_valido:    ; Se correto, desvia para monitor_válido
165 00EC 7E008B          JMP monitor_erro        ; Se ERRO, desvia para monitor_erro
166 00EF                 loop_eterno:            ; LOOP ETERNO
167 00EF 7E00EF          JMP loop_eterno         ; LOOP ETERNO
```

```
168 00F2                monitor_válido:          ; Os dados do programa monitor são válidos
169 00F2 8660                   LDAA #$60        ; Carrega 60h no acumulador
170 00F4                termino:                 ; Finalização do programa de boot
171 00F4 01                     NOP
172 00F5 B7802F                 STAA $802F       ; Envia resposta via canal serial
173 00F8 181F2EC0               BRCLR $2E,Y,#$C0,*-4  ; Espera esvaziar o buffer de transmissão
    00FC F7
174 00FD 7E0100                 JMP $0100        ; Salta para o início do programa monitor
175 0100                        END              ; FIM DO PROGRAMA DE BOOT

Errors: None
Bytes: 256          bootsbm
CRC:  75BA
```

FIGURA 7 - Listagem parcial do programa de boot, mostrando o procedimento de carga e execução do programa monitor

6.3 Programa monitor

O programa monitor, que está sendo executado na memória RAM externa do módulo sob teste, possui as rotinas para acessar todos os dispositivos de entrada e saída do módulo.

A partir deste momento, o equipamento de teste possui o controle do módulo sob teste, pois através de comandos enviados pelo canal serial, é possível controlar todos os circuitos periféricos do módulo, bem como realizar os testes digitais restantes.

Como comandos do programa monitor, podemos citar o comando de teste da PROM, que verifica a integridade das conexões entre o microcontrolador e o componente e a correta gravação do *firmware*, executando a leitura de todas as posicões da memória e geração de *checksum* de 16 bits. O *checksum,* após gerado, é devolvido como resposta do comando ao equipamento de teste que determina se está correto ou não.

O comando de teste da EEPROM verifica através de operações de escrita/leitura o correto funcionamento deste dispositivo, e devolve via canal serial a resposta do comando de teste.

A partir daí, seguem os comandos que acessam circuitos periféricos do módulo, devendo ser acionados à medida que o equipamento de teste necessitar que algum circuito seja atuado ou que algum sinal seja lido.

O programa monitor possui um total de 41 comandos para controle de circuitos periféricos e realização de testes. Foi desenvolvido em linguagem C ANSI e utilizado o compilador C 68HC11 da IAR Systems, sendo que o programa executável está com 3 *Kbytes*.

7. CONCLUSÕES

O teste funcional com a execução do *firmware* do módulo foi descartado devido ao tempo de teste do módulo. As estimativas indicavam que o tempo de teste nesta filosofia estaria próximo de oito minutos. Nesta técnica, temos a necessidade de respeitar todos os protocolos de acesso ao módulo.

No caso do teste funcional com um *firmware* de teste, ambas as alternativas foram vetadas. Após o desenvolvimento do *software* do produto, não sobrou espaço suficiente para a inclusão de rotinas de teste. A troca de *firmware* também não foi possível devido ao encarecimento do produto ao inserir um soquete PLCC SMD para receber a memória PROM.

A técnica de utilizar um emulador para realizar o teste funcional foi descartada por diversas razões: elevado custo do emulador, elevado custo de manutenção, operação do equipamento por pessoal técnico qualificado, e também porque seria necessária a colocação de um soquete PLCC SMD para receber a POD do emulador.

Dessa forma, fomos direcionados a utilizar a facilidade de *special bootstrap mode* do microcontrolador. Para a realização dos testes funcionais na fábrica, não foi encarecido o produto e hoje o tempo de execução de todos os testes funcionais no ESL8 está abaixo de três minutos.

A estratégia de carregar um programa de *boot* para testar o ambiente da memória externa e em seguida carregar um programa monitor de comandos nesta memória, mostrou-se extremamente eficiente, permitindo que o equipamento de teste tivesse o controle de todo o *hardware* do módulo sob teste.

Em outros equipamentos de teste desenvolvidos, já havíamos utilizado as outras estratégias de teste funcional nos módulos microprocessados, e acabamos concordando que dentre as estratégias existentes, esta é a mais versátil, sendo técnica e economicamente mais viável na

linha de produção.

Evidentemente a utilização desta técnica deveu-se aos recursos de testabilidade disponíveis no componente. Outros fabricantes de microcontroladores possuem técnicas diferentes para o teste de circuitos, porém não são tão versáteis quanto esta implementada pela Motorola.

Como informação adicional é interessante citar que nos contatos mantidos com o Departamento de Engenharia de Aplicações da Motorola, soubemos que fomos os pioneiros no Brasil a utilizar o recurso de *special bootstrap mode* para testes funcionais.

8. REFERÊNCIAS BIBLIOGRÁFICAS

[1] Cortner, J. Max; *Digital Test Engineering*; John Wiley & Sons; 1987.

[2] Buckroyd, Allen; *Computer Integrated Testing*; John Wiley & Sons; 1989.

[3] Motorola; M68HC11 *Reference Manual*; M68HC11RM/AD REV 3, 1991.

[4] Motorola; MC68HC11A8 *Technical Data*; MC68HC11A8/D REV 5, 1991.