1. Publicação nº	2. Versão	3. Data	5. Distribuição
INPE-2764-PRE/338		*Junho, 1983*	☐ Interna ☒ Externa ☐ Restrita

4. Origem	Programa
DCA/DEA	*SISMAG/PSDA*

6. Palavras chaves - selecionadas pelo(s) autor(es)

PROGRAMA DE TESTE
LINGUAGEM DE TESTE DE CIRCUITOS
TESTE DE CIRCUITOS DIGITAIS E ANALÓGICOS

7. C.D.U.: *681.322.OI:681.34.001.4*

8. Título *INPE-2764-PRE/338*

INTAC - LINGUAGEM DE TESTE DE CIRCUITOS DIGITAIS E ANALÓGICOS

10. Páginas: *22*

11. Última página: *13*

12. Revisada por

Ricardo C.O. Martins

9. Autoria *Marcos Antonio Cardoso Cruz*

13. Autorizada por

Nelson de Jesus Parada
Diretor

Assinatura responsável

14. Resumo/Notas

Apresenta-se uma linguagem do tipo Pascal, denominada INTAC para a descrição de procedimentos de testes de circuitos digitais e analógicos. Enfatiza-se a utilização de novas declarações e comandos da linguagem orientados para a descrição e aplicação de testes, tais como o manuseio de sinais digitais e analógicos.

15. Observações ** Este trabalho será submetido ao 10º Seminário Integrado de Software e Hardware a ser realizado em Campinas, 25-30 julho 1983.*

ABSTRACT

A Pascal like test programming language for digital and analog circuits called INTAC is described. The new statements and declarations oriented for testing purposes, such as digital and analog signal handling, are the main topics of description.

RESUMO ESTENDIDO

As linguagens orientadas para teste de circuito estão relacionadas com os recursos de "hardware" e o modo de execução de teste utilizados no sistema automático de teste. A linguagem apresentada neste trabalho foi desenvolvida para ser utilizada no sistema de teste INTAC. Neste sistema, a execução do teste baseia-se em ciclos do controlador do sistema, sendo composto por fases de síntese dos estímulos e fases de aplicação do teste. A estrutura da linguagem INTAC baseia-se nas declarações e nos comandos da linguagem PASCAL padrão, onde foram incluídas novas declarações e comandos que permitem a manipulação de sinais dentro de programas de teste. Informações relativas aos elementos do circuito envolvidos no teste são fornecidas através das declarações de sinais. Elas permitem descrever sinais digitais, caracterizando as entradas, saídas, barramentos do circuito em teste e também os sinais analógicos desse circuito. Além destes sinais, pode-se declarar o relógio a ser utilizado na execução do teste e as fontes de alimentação necessárias e disponíveis no sistema de teste. Além dos comandos da linguagem PASCAL, foram acrescentados comandos específicos para controle da execução do teste e para geração e aquisição de sinais analógicos. Por ser uma linguagem orientada para teste, expandiu-se o seu conjunto de operadores a fim de incluir funções básicas de circuitos lógicos como, por exemplo, o complemento, não-e, ou-exclusivo e outras. Manipulação parcial de palavras também foi incluída e tem grande utilidade na manipulação de bits no teste de um circuito digital. Outra característica da linguagem é a possibilidade de operação em outras bases além da decimal como a binária, a octal e a hexadecimal. Um compilador para a linguagem de teste INTAC foi implementado gerando uma forma intermediária pós-fixa. A avaliação final da linguagem deverá ser obtida com a implementação do interpretador, ou do gerador de códigos, para o controlador do sistema INTAC, em fase de desenvolvimento.

SUMÁRIO

LISTA DE FIGURAS

Pág.

1 - INTRODUÇÃO

Sistemas CAD/CAM para testes de circuitos eletrônicos, principalmente os digitais, têm sido largamente empregados nos últimos anos, em decorrência das vantagens que oferecem, tais como a redução de tempo e custo no desenvolvimento do projeto. Estes sistemas automáticos de teste envolvem tanto "hardware" como "software", incluindo as linguagens utilizadas para descrição dos programas de teste. Neste trabalho é apresentada uma linguagem para descrição de programas para teste de circuitos digitais e analógicos, a ser utilizada no sistema de teste INTAC (CRUZ, no prelo), atualmente em desenvolvimento no Instituto de Pesquisas Espaciais de São José dos Campos.

As linguagens orientadas para programação de testes de circuitos estão relacionadas com os recursos de "hardware" e com o modo de execução de testes utilizados no sistema automático de teste, provendo, principalmente, meios para descrever os procedimentos de teste de um circuito a ser avaliado.

Os procedimentos de teste (HART, 1980) são sequências de eventos necessários para avaliar o circuito em teste. Estes eventos incluem a aplicação de estímulos bem definidos em pontos específicos do circuito em teste, bem como a medição de respostas associadas em outros pontos desse circuito. Um procedimento também pode incluir cálculos, comparações, avaliações e mensagens para o operador e interagir com o teste em curso.

Um programa de teste, executado no sistema de teste, controla a instrumentação para enviar estímulos e receber respostas, além de efetuar cálculos e comparações, podendo comunicar-se com o operador através de mensagens e comandos.

Comparando um procedimento de teste com um programa de teste, verifica-se que o procedimento especifica o que deve ser feito

para testar um circuito, enquanto o programa de teste incorpora estas instruções e também fornece informações a cerca da implementação do teste no sistema automático de teste.

A execução do teste no sistema INTAC é baseada em ciclos do controlador, onde cada ciclo compreende duas fases distintas: a de síntese e a de execução. Na fase de síntese, o controlador do sistema de teste prepara os estímulos, seguindo o fluxo descrito no programa de teste e podendo utilizar os resultados obtidos em ciclos anteriores, os quais serão aplicados ao circuito em teste durante a fase de execução.

A linguagem de teste INTAC permite a descrição de procedimentos para teste de circuitos digitais e analógicos, provendo meios para a utilização desses elementos dentro de um programa de teste e delimitando os ciclos do controlador dentro desse programa.

Os primeiros sistemas automáticos de teste utilizavam em geral a linguagem "assembly". Atualmente, dispõem-se de linguagens específicas para a descrição de testes como a ATLAS ("Abbreviated Test Language for Avionics Systems") ou ainda as linguagens baseadas no BASIC, FORTRAN, PASCAL, além de linguagens específicas dos próprios fabricantes de sistemas de teste.

A linguagem INTAC foi desenvolvida, com base na estrutura da linguagem PASCAL (JENSEN, 1978), a fim de ser utilizada na geração de programas de teste para o sistema INTAC. Novas declarações e comandos foram acrescentados a esta linguagem para permitir a utilização de elementos empregados nos testes de circuitos e o controle de sua execução.

2 - ESTRUTURA DA LINGUAGEM INTAC

A linguagem INTAC é uma linguagem estruturada com base nas declarações e comandos da linguagem PASCAL. A escolha da linguagem PASCAL como modelo deve-se aos seguintes fatos:

1) ser uma linguagem estruturada e voltada para a descrição de algoritmos;

2) ser uma linguagem onde os atributos associados aos elementos empregados no programa precisam ser declarados, o que se torna bastante útil para descrever sinais de circuitos eletrônicos;

3) apresentar familiaridade e boa divulgação, notadamente em meios acadêmicos e científicos.

Algumas estruturas do PASCAL padrão, como TYPE e POINTER, de grande aplicação em programas que utilizam estruturas de dados (árvores, tabelas, filas, etc.) podem não ter grande interesse em programas que descrevem fluxo e manipulam bits, como num programa de teste, não sendo incluídas na linguagem de teste.

Um programa de teste é composto por duas partes, conforme se vê na Figura 1.

```
<programa> ::= PROGRAM <identificador> ;
               <declaracao de sinais>
               <bloco> .

<bloco> ::= <declaracoes do bloco>
            <lista de comandos>
```

Fig. 1 - Estrutura de um programa de teste.

A < declaração de sinais > permite a descrição dos sinais utilizados no teste. Devido ao fato de os sinais serem entidades reais e, portanto, estarem presente durante a execução de todo o programa de teste, as suas declarações deverão ser sempre globais.

O controle de fluxo no programa é executado através dos comandos inseridos em cada < bloco > do programa fonte. As < declarações do bloco > possibilitam a descrição de elementos de programação, tais como rótulos, constantes, variáveis e sub-rotinas, locais ao < bloco >.

Uma característica da linguagem INTAC é a possibilidade de representação dos números em outras bases além da decimal, como a binária, a octal e a hexadecimal. A manipulação parcial de palavras também foi incluída nessa linguagem e torna-se bastante útil quando se trabalha com bits ou campos de bits, situação comum no teste de circuitos digitais.

Os tipos de variáveis admitidas na linguagem INTAC são três: inteira, caractere e booleana. As variáveis caractere assumem os códigos internos, em geral ASCII, correspondentes aos caracteres alfanuméricos; por exemplo, a letra A é representada por 65 no código ASCII.

Para efeito de compatibilidade de tipos, os sinais digitais e analógicos declarados no programa fonte são equivalentes às variáveis inteiras. Assim, operações aritméticas que envolvem sinais e variáveis inteiras fornecerão um resultado do tipo inteiro. Os tipos dos operandos utilizados em uma expressão devem ser verificados, em tempo de compilação, a fim de determinar a compatibilidade entre eles.

3 - DECLARAÇÃO DE SINAIS

A < declaração de sinais > permite descrever os elementos dos circuitos envolvidos no programa de teste. As informações derivadas desta declaração serão utilizadas para configurar as interfaces do sistema de teste antes do início da execução do teste. Os sinais envolvidos no teste são sempre declarados com relação ao circuito em teste ("Unit Under Test - UUT").

A < declaração de sinais > inclui declarações para descrição de:

1) famílias lógicas (LOGIC TYPE) dos componentes da parte digital envolvida;

2) sinais digitais (DIGITAL UUT), os quais podem ser entradas (INPUT), saídas (OUTPUT) ou barramentos bidirecionais (BIDIRECTIONAL) do circuito em teste;

3) sinais analógicos (ANALOG UUT), que podem ser entradas (INPUT) ou saídas (OUTPUT);

4) fontes de alimentação (POWER SUPPLY) utilizadas no teste, disponíveis no sistema de teste;

5) controle do relógio (CLOCK) a ser utilizado na execução do teste.

A caracterização de um sinal digital consta basicamente de dois elementos, ou seja, a sua família lógica e uma lista de atributos associados a ele. Exemplos de declaração destes sinais são dados na Figura 2.

```
LOGIC TYPE
        TTL : (5,1,3) VOLT;
        CMOS : (12,3,9) VOLT;

DIGITAL UUT INPUT
        MEMW = XA(1) ^ 1 ^ ,
        MEMR = XA(2) ^ 1 ^ ,
        ADDR (15..0) = XB(16..1) : TTL;

      OUTPUT
        FLAG = PT(15) : CMOS;

      BIDIRECTIONAL
        DATA (7..0) = YA (22..15) (MEMR=0) :CMOS;
```

Fig. 2 - Declaração de sinal digital.

Uma família lógica é descrita por um identificador, por exemplo TTL, CMOS, seguido dos níveis de tensão que a caracterizam, ou seja, a sua alimentação e os seus limiares de nível baixo e nível alto.

Um sinal digital é descrito por um identificador, por exemplo MEMW, FLAG, DATA(7..0), seguido da sua posição no sistema de teste (XA(1), XB(16..1), PT(15) ...). Opcionalmente pode-se declarar um valor de omissão, representado por uma constante entre colchetes. A declaração do sinal digital é completada identificando a família lógica a qual ele pertence.

A possibilidade de declarar um valor de omissão está relacionada com o fato de alguns sinais terem de manter um dado valor durante todo o teste, exceto em certos ciclos do controlador nos quais

eles assumem valores diferentes, conforme instruções inseridas no programa de teste, retomando o seu valor de omissão nos ciclos seguintes. Diferentemente, os sinais cujo valor de omissão não foi declarado manterão o último valor atribuído a eles, até que uma nova atribuição seja executada.

Os sinais bidirecionais necessitam ainda que seja declarada a sua condição de ativação, dada por uma expressão booleana. No exemplo da Figura 2, a condição de ativação do barramento DATA é MEMR = 0, ou seja, o sistema de teste só poderá enviar dados através desse barramento quando o sinal MEMR estiver no nível baixo.

```
ANALOG UUT INPUT
        RAMP = CHNL (1) (-2,6,1) VOLT ^15 KHZ^;

OUTPUT
        SIG23 = CHNL (9) (1,10,5)VOLT ^197 HZ^;
        DELTA = CHNL (2) (-8,0,-5)VOLT ^63 KHZ^;
```

Fig. 3 - Declaração de sinal analógico.

Os sinais analógicos, mostrados na Figura 3, são descritos por um identificador seguido de seus atributos. Estes atributos compreendem a especificação da sua posição no sistema de teste (por exemplo, CHNL(1), CHNL(9)..) e as informações de tensões e frequência do sinal. As informações de tensões incluem os níveis mínimo e máximo do sinal e o nível de gatilho ("TRIGGER"), enquanto a informação de frequência especifica a sua taxa de variação.

Completando a descrição de sinais, deve-se declarar como será o relógio utilizado na execução do teste e as fontes de alimentação que serão necessárias para sua realização. Exemplos destas declarações são fornecidas na Figura 4.

```
CLOCK EXT (NEG) ^ 2 MHZ ^ : CMOS;

POWER SUPPLY -12, +5, +12, +15 VOLT;
```

Fig. 4 - Declaração de relógio e fontes de alimentação.

A declaração de relógio indica se a base de tempo usada no teste será fornecida pelo circuito em teste (INT), ou se ela será gerada pelo sistema de teste (EXT). Além disto, deve-se informar a polaridade do sinal, se positiva (POS) ou negativa (NEG). No caso do relógio ser fornecido pelo sistema de teste, é preciso declarar, entre colchetes, qual o valor da frequência desejada. A descrição do relógio é completada acrescentando-lhe a família lógica.

Já a declaração de fontes relaciona as fontes de alimentação disponíveis no sistema de teste, que serão utilizadas na alimentação do circuito durante sua execução.

4 - DECLARAÇÕES DO BLOCO

As declarações do bloco, incluídas na linguagem de teste, oferecem facilidades para descrever entidades utilizadas no desenvolvimento dos programas de teste e compreendem as declarações de rótulo (LABEL), constantes (CONST), variáveis (VAR) e sub-rotinas (PROCEDURE). Estas declarações podem aparecer no início de cada bloco, e as entidades nelas relacionadas só têm validade dentro do respectivo bloco onde foram declaradas. Um exemplo destas declarações é dado na Figura 5.

```
LABEL LOOP, FIM;

CONST
      TRES = 3;
      DEZ = 1010B ;
      JOTA = 'J' ;
      FLAG = TRUE ;

VAR
    DIGITO,NUMERO : INTEGER;
    NOME, LETRA : CHAR;
    OVERFLOW, SIGNAL : BOOLEAN;

PROCEDURE ALFA;
   :
   BEGIN
      :
   END
```

Fig. 5 - Exemplo de declarações do bloco.

5 - EXPRESSÕES

As expressões empregadas no programa de teste podem envolver constantes, variáveis, sinais digitais e analógicos. Basicamente têm-se dois tipos de expressões, a aritmética e a booleana. A primeira fornece como resultado um valor do tipo inteiro, enquanto a última só pode assumir duas condições, ou seja, TRUE ou FALSE.

Na Figura 6 têm-se alguns exemplos de expressões. Por ser uma linguagem orientada para teste de circuitos, principalmente os digitais, expandiu-se o conjunto de operadores para incluir funções básicas de circuitos lógicos, tais como complemento ($NOT), e ($AND), não-e ($NAND), ou ($OR), não-ou ($NOR), ou-exclusivo ($XOR) e não-ou-exclusivo ($XNOR).

```
DATA ^3..0^ := A.^3..0^;
ADDR := 2 * B + C;
MEMW := (Y $AND MEM^1^ ) $XOR MEMR;
IF ((DATA AND 15) OR (ADDR <> Y) THEN ...
WHILE AADR <= 0F3E2H DO ...
```

Fig. 6 - Exemplo de expressões.

6 - COMANDOS

Os comandos da linguagem de teste, além de incluir os comandos normalmente utilizados em linguagens estruturadas, como IF, WHILE, REPEAT, FOR e CASE, incorporam comandos especiais utilizados na execução do teste. Alguns exemplos de comandos podem ser vistos na Figura 7.

```
IF A > B THEN BEGIN
                ADDR^3..0^ := A.^7..4^ ;
                MEMW := 0
              END
         ELSE
            GOTO LOOP;
#
$SAMPLE (9,100);
LOOP: FLAG:= FALSE;
CASE I OF

0,1: ADDR:= (A/B + D) $AND 101B;
2: RAMP:= MEM ^1^;
3,4,5: GOTO FIM
END;  % end CASE
```

Fig. 7 - Exemplos de comandos.

A delimitação dos camandos a serem processados em cada ciclo do controlador é dada no programa fonte através de um caractere (##).

Há comandos específicos para manipulação de sinais analógicos com $SAMPLE e $RANDON para aquisição ce dados, e $SIGNAL para geração de formas de onda. Estes comandos necessitam de dois parâmetros, o primeiro para especificar o canal analógico a ser utilizado e o último para determinar o número de dados a ser amostrados ou gerados.

A entrada e saída são feitas através dos comandos READ e WRITE, respectivamente, conforme são mostrados na Figura 8. Tanto a entrada como a saída manipulam basicamente caracteres alfanuméricos e números inteiros.

```
WRITE ('END =',ADDR : H4, 'DADO =', DATA)$

WRITE (LETRA, K*5 + 12 :D3 )$


READ (NOME, A, LETRA)$

READ (ADDR, MEMR)$
```

Fig. 8 - Exemplos de entrada e saída.

O comando WRITE permite a saída de cordões de caracteres, como por exemplo END, DADO, ou de valores das variáveis e expressões. Neste último caso o formato de saída pode ser livre, ou seja, é dado implicitamente pelo tipo da variável (inteiro ou caractere) ou formatada, onde se especifica a base desejada e o número de campos a serem impressos. Por exemplo, ADDR: H4 significa que o valor de ADDR será impresso na base hexadecimal com 4 dígitos.

No comando READ, o formato de entrada será livre, determinado pelo tipo dado na declaração da variável, podendo ser do tipo inteiro (por exemplo, ADDR, A, MEMR) ou do tipo caractere (como LETRA, NOME).

7 - CONCLUSÕES

A primeira versão do compilador da linguagem INTAC foi implementada no computador Burroughs B-6800, escrito em linguagem PASCAL (INPE, 1982). Na sua construção foram utilizadas algumas idéias propostas por Welsh (1980), especialmente na organização da tabela de símbolos em árvores binárias. Um dos cuidados tomados na implementação foi a utilização das estruturas da linguagem PASCAL padrão, de modo a permitir a sua portabilidade para outras máquinas com mínimas alterações.

O compilador construído gera atualmente uma forma intermediária pós-fixa. A avaliação final da linguagem deverá ser obtida com a implementação do interpretador, ou do gerador de códigos, para o controlador do sistema de teste INTAC, em desenvolvimento neste Instituto.

8 - REFERÊNCIAS BIBLIOGRÁFICAS

CRUZ, M. A. C., *INTAC - uma interface de teste auxiliada por computador*. São José dos Campos, INPE (no prelo).

HART, D., *Test programming languages*. Electronic Test, 3(6): 48-57, Jun. 1980.

INSTITUTO DE PESQUISAS ESPACIAIS - INPE, *Departamento de Processamento de Dados*. Manual de PASCAL. São José dos Campos, INPE, Mar. 1982..

JENSEN, K., WIRTH, N. PASCAL, *user manual and report*. 2ed. New York, Springer-Verlas. 1978.

WELSH, J., McKEAG, M., *Strutured system programming*. London, Prentice-Hall, 1980.