Universidade de São Paulo

Escola de Engenharia de São Carlos

DESENVOLVIMENTO DE ROTINAS DE MEDIÇÃO COM INTERFACES GRÁFICAS PARA SIMPLIFICAR E EXPANDIR O USO DE "TOUCH-TRIGGER PROBES" EM MÁQUINAS-FERRAMENTAS CNC.

Mateus Carniatto

São Carlos, SP
2010

MATEUS CARNIATTO

DESENVOLVIMENTO DE ROTINAS DE MEDIÇÃO COM INTERFACES GRÁFICAS PARA SIMPLIFICAR E EXPANDIR O USO DE "TOUCH-TRIGGER PROBES" EM MÁQUINAS-FERRAMENTAS CNC.

Trabalho de conclusão de curso em Engenharia Elétrica ênfase em Eletrônica

Orientador: Prof. Dr. Reginaldo Teixeira Coelho

São Carlos, SP
2010

Ficha catalográfica preparada pela Seção de Tratamento
da Informação do Serviço de Biblioteca – EESC/USP

Carniatto, Mateus
C289d Desenvolvimento de rotinas de medição com interfaces gráficas para simplificar e expandir o uso de "*Touch-Trigger Probes*" em máquinas-ferramentas CNC / Mateus Carniatto ; orientador Reginaldo Teixeira Coelho. -- São Carlos, 2010.

Trabalho de Conclusão de Curso (Graduação em Engenharia Elétrica com Ênfase em Eletrônica) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2010.

1. Usinagem. 2. Medição mecânica. 3. Máquina-ferramenta. 4. Apalpadores. I. Título.

Agradecimentos

Em especial gostaria de agradecer ao Prof. Dr. Reginaldo Teixeira Coelho pela oportunidade de desenvolver este projeto sob sua orientação, ao Dr. Ricardo Arai por todo apoio, auxílio, paciência e incentivo durante este desenvolvimento.

Gostaria de agradecer aos funcionários do NUMA e de uma forma especial a Adolfo e Ariel por toda ajuda e apoio para realização e concretização deste desenvolvimento. Não se esquecendo de agradecer à cada integrante do NUMA EESC-USP, Professores, pós-doutorandos, doutorandos, mestrandos, estudantes ou funcionários administrativos, à todos estes muito obrigado.

Demais gostaria de agradecer à minha família que me apoiou durante todo o curso de engenharia elétrica e à dos os funcionários e professores do departamento de engenharia elétrica da EESC que possibilitaram meu progresso durante todos este período.

Enfim gostaria de agradecer ao Dr. Marcelo Del Guerra e ao Prof. Dr. Evandro Luís Linhari Rodrigues por aceitarem participar desta banca de TCC.

Resumo

CARNIATTO, M. Desenvolvimento de rotinas de medição com Interfaces gráficas para simplificar e expandir o uso de "Touch-Trigger Probes" em Máquinas-Ferramentas CNC. 2010. Dissertação (Trabalho de conclusão de curso) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2010.

Os "touch-trigger probes", ou apalpadores, juntamente com o comando CNC, são utilizados para a realização de controle dimensional das peças fabricadas, como ferramenta para a compensação de desgastes de flanco de ferramentas, na realização de ”setup” simplificado e rápido, na criação de dados estatísticos de processos para possíveis otimizações nos processos de manufatura, na realização de medições e calibrações em processo mediante as especificações de projeto, dentre muitas outras. Desenvolveu-se neste trabalho rotinas que possibilitam utilizar estes medidores de forma fácil e intuitiva de maneira a ampliar e facilitar sua aplicação. Através de um ensaio utilizando os resultados deste trabalho pode-se constar que o uso de medidores em máquinas CNC apresenta melhorias na precisão dimensional mesmo em casos onde o desgaste de ferramenta é desprezível.

1. Usinagem. 2. Medição Mecânica. 3. Apalpadores. 4. Maquinas-ferramenta.

Abstract

CARNIATTO, M. Desenvolvimento de rotinas de medição com Interfaces gráficas para simplificar e expandir o uso de "Touch-Trigger Probes" em Máquinas-Ferramentas CNC. 2010. Dissertação (Trabalho de conclusão de curso) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2010.

The touch-trigger probes (TTP), together with the CNC command, are utilized to dimensional control of manufactured parts, to compensate the tool wear errors, to setup a tool easily and faster, to collect data for statistical analysis used to manufacturing process optimization, to make measurements and calibrations in process in spite of projects specifications, and many others. In this work, routines were developed to make possible to use this kind of measurement easier and more intuitively to expand its application. With the trial using the results of this work was possible to conclude that the use of touch-triggers in CNC machines improve the dimensional accuracy even in cases where the wear is extreme low.

1. Machining. 2. Mechanical measurement. 3. Touch-trigger probes. 4. CNC machines

Lista de Figuras

Sumário

1 Introdução

Métodos de inspeção automáticos podem ser a solução para o aumento da demanda por qualidade dimensional nas peças usinadas e a redução dos custos envolvidos. Tais métodos de inspeção devem ser capazes de realizar o controle dimensional durante o processo de usinagem e re-alimentar o sistema com as informações necessárias para as correções. A estratégia para atingir alta qualidade, produtividade e “lead times” reduzidos é trazer o controle da qualidade para próximo da máquina-ferramenta, o que pode ser atingido através da utilização de medições em processo, inspecionando as partes na medida em que são produzidas (YANDAYAN, T; BURDEKIN, M. 1997 apud HUA JONG, D, 2006, *Relatório Processo FAPESP: 2006/02728-8*).

Segundo Smith e Vemuganti, “Muitos processos de manufatura contém pontos ou arestas cortantes que estão sujeitas ao desgaste de modo que a re-calibração da ferramenta torna-se necessária”. Mediante a grande exigência do mercado as linhas de produção têm necessitado de um aumento da sua precisão para atender a demanda, pois com a redução das peças de refugo ocorre também um aumento da produtividade.

Durante um processo de usinagem ocorre inevitavelmente o desgaste da ferramenta, que acarreta na perca de precisão na medida da peça em máquinas CNC, ainda podemos citar de acordo com J.Y. Wanga, C.R. Liua e K.K. Wang, o desgaste de ferramenta é o principal causador de danos térmicos na camada da superfície feita ã maquina, pode-se notar então a importância que o desgaste de ferramenta tem assumido nos processos de manufatura, especialmente quando utilizamos máquinas ferramentas no intuito de obter maior receptibilidade.

Diante desse fato tornam-se cada vez mais viável a utilização de apalpadores para garantir a precisão das máquinas CNC, neste projeto desenvolveu-se uma rotina em linguagem G para setup da ferramenta e uma sub-rotina (que pode ser executada a partir de qualquer programa e qualquer numero de vezes) que calcula o desgaste de ferramenta e insere este fator no corretor de ferramenta presente em máquinas ferramenta CNC, assim torna-se possível manter a ferramenta sempre calibrada para minimizar os erros obtidos por desgaste de ferramenta.

Para maior praticidade na utilização do processo de calibração da ferramenta podemos desenvolver um interface gráfica de modo a tornar a utilização da rotina de setup de ferramenta mais intuitiva e menos propensa a erro na inserção dos parâmetros iniciais. Esta interface pode propiciar ao usuário uma maior interação com o processo de calibração permitindo a configuração de alguns parâmetros para adaptação do processo à necessidade do usuário.

Este trabalho de iniciação científica objetiva o desenvolvimento de uma rotina de medição em linguagem G para gerenciamento do “Touch Trigger Probes” para a realização de rotina de setup automático e da sub-rotina calibração de desgaste da ferramenta a ser inspecionada com uma interface fácil e amigável com o usuário.

2 Desenvolvimento

2.1 Revisão Bibliográfica

2.1.1 Procedimento de medição

Ao realizar um processo de usinagem utilizando máquinas CNC são necessários alguns procedimentos iniciais de preparação da máquina, que consistem em: referenciar a máquina (inicializar o posicionamento da máquina em cada eixo cartesiano), verificar a necessidade de fluído de corte, calibrar as ferramentas que serão utilizas, entre outras.

O procedimento de calibrar as ferramentas consiste em determinar a altura da ferramenta. Este procedimento, na maioria dos casos, é realizado manualmente, sendo assim influenciado pela experiência e pelo julgamento do operador, fato que pode acarretar em erros e redução da produtividade. A altura da ferramenta deve ser determinada para que este valor seja considerado durante o posicionamento.

Na Figura 1 no caso (A) mostra-se a que sem a compensação de comprimento de ferramenta o CNC posiciona a extremidade do cone de fixação na coordenada desejada. Na Figura 1 (B) é demostrado o funcionamento utilizando-se a compensação do comprimento da ferramenta, ou seja, o CNC posiciona a extremidade da ferramenta na posição programada.

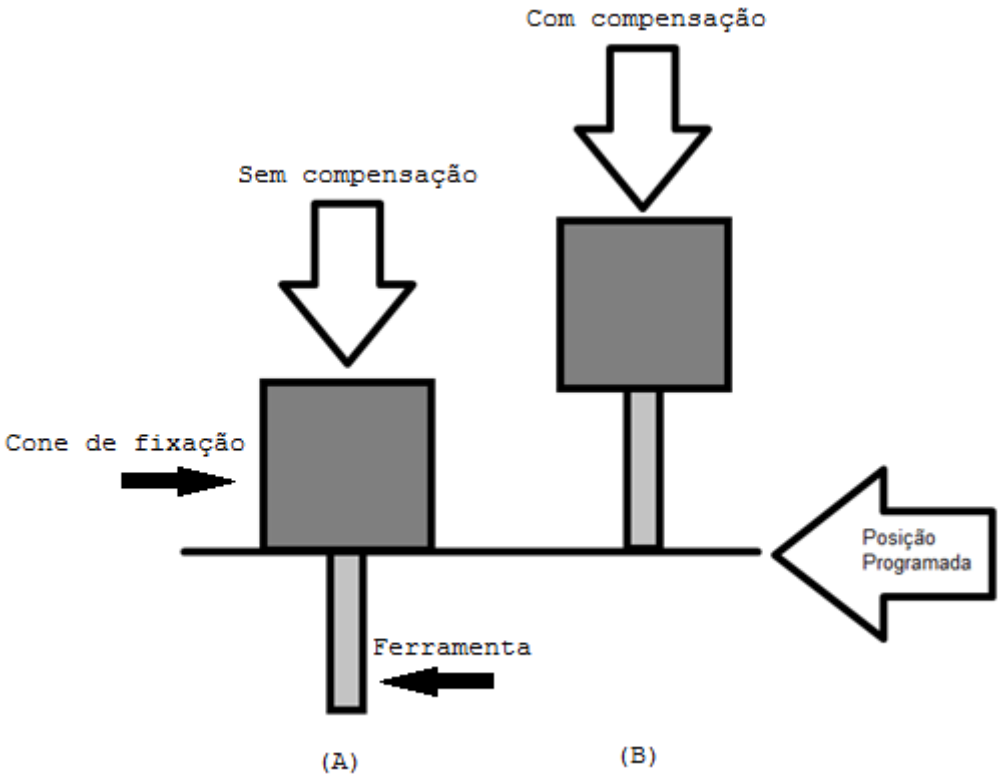

Figura 1 - Demonstração da compensação de ferramenta

Para aperfeiçoar este procedimento fundamental buscou-se desenvolver um sistema capaz de automatizar o ajuste inicial da ferramenta, possibilitando assim uma melhor precisão em um tempo reduzido. Para tanto utilizou-se um sensor apalpador (“Touch-Trigger Probes”) para realizar as medições em conjunto com o sistema de posicionamento da máquina-ferramenta. Este método de medição é denominado medição “in-situ”, ou seja, medição realizada no local, uma vez que a ferramenta é calibrada com o auxílio da máquina-ferramenta em que será utilizada.

Quando se realiza a medição da altura de uma ferramenta utilizando sensores apalpadores é executada uma sequência padrão de ações, estas podem ser observadas no fluxograma da Figura 2.

Após realizar esta calibração não é mais necessário considerar o tamanho da ferramenta, pois este tamanho já está referenciado através do corretor de ferramenta, sendo assim torna-se mais fácil o desenvolvimento de rotinas de usinagem.

Durante um processo de usinagem com altas velocidades de corte ou com materiais de alta dureza torna-se eminente o surgimento do desgaste da ferramenta, que embora frequentemente seja pequeno, se considerado ao longo de uma linha de produção pode causar um desvio dimensional altamente indesejável nas peças produzidas.

Para minimizar os efeitos do desgaste de ferramenta é possível, caso a ferramenta tenha sido calibrada previamente, realizar uma nova calibração entre cada etapa do processo de usinagem, comparando a altura anterior com a altura atual, obtendo assim o valor do desgaste da ferramenta.

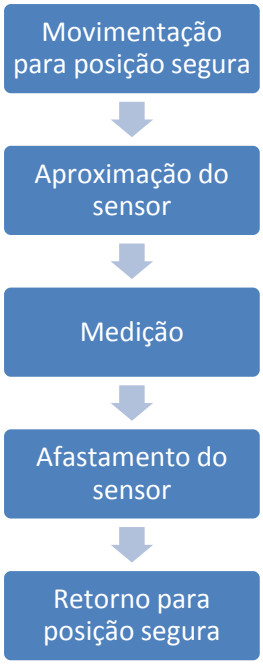

Figura 2 - Processo padrão de medição

2.1.2 Comando Numérico (CN)

Desde o surgimento do Comando Numérico (CN), em meados do século XX, diversas indústrias, em especial a aeronáutica e a automotiva, vêm auferindo ganhos significativos com a utilização dessa tecnologia. Sua aplicação no controle de máquinas-ferramenta permite a realização de tarefas repetitivas e de grande complexidade cinemática. Isto possibilita a reprodutibilidade de produtos de variadas formas geométricas. Além disso, empresas que produzem com alta diversificação e em pequenos lotes usufruem muito da flexibilidade inerente a esses equipamentos (DIAS COSTA, D; GLEBER PEREIRA, A, 2005, P. 02).

Tratando-se de processos de usinagem, a tecnologia CN (Comando Numérico) pode ser interpretada como um sistema de informações que transforma uma descrição geométrica do componente a ser usinado, a partir de uma entrada simbólica da mesma, no controle de posição e velocidade de um ou mais servo-motores (PRESSMAN & WILLIAMS, 1977; CHANG, 1998; apud DIAS COSTA, D; GLEBER PEREIRA, A, 2005, P. 03).

O comando numérico computadorizado tem se tornado cada vez mais frequente em empresas que necessitam de reprodutibilidade aliada à flexibilidade oferecida por estas máquinas. Durante este projeto usou-se uma maquina ferramenta que dispunha de um interpretador de linguagem G no modelo Siemens e uma com possibilidade de uso da linguagem G para o modelo FANUC.

2.1.3 Conceito de modularização de software

O conceito de software modularizado é aplicado em várias áreas do desenvolvimento de software, em especial estudado pela engenharia de software. Este conceito se resume em dividir o problema inicial em vários problemas menores e tratá-los separadamente.

Aplicando-se a modularização pode-se tratar cada caso isoladamente utilizando-se uma sub-rotina, gerando assim várias sub-rotinas especializadas em tarefas simples e deste modo possibilitando o desenvolvimento de softwares mais complexos a partir de pequenos módulos para realizar as funcionalidades simples.

Quando cria-se um conjunto de sub-rotinas que têm por função desempenhar atividades que serão realizadas muitas vezes em um processo maior, denomina-se este conjunto de biblioteca de funções. Quando se trata de desenvolvimento estruturado de softwares em geral este modo de programação é utilizado.

Após o desenvolvimento da biblioteca de funções básicas, é possível utilizar cada função implementada através do nome da sub-rotina sem preocupar-se com a implementação.

O processo de modularização propicia, com principal vantagem, a facilidade de depuração do código (detecção de erros). Uma vez que cada unidade da biblioteca já foi implementada e pode ser assumida como livre de erros, então o processo de depuração se limita aos trechos do programa principal que não fazem referência as sub-rotinas contidas na biblioteca. Outra vantagem que pode ser observada quando se utiliza modularização de software é a facilidade de desenvolver softwares personalizados para as diferentes situações apresentadas, tornando assim o projeto mais flexível.

2.2 Materiais e métodos

2.2.1 Estratégia adotada

Foi traçada uma estratégia para a medição do comprimento da ferramenta que consiste em medir a altura (posição no eixo Z) quando a ferramenta aciona o sensor unidimensional e comparar essa medida com a altura em que o eixo arvore toca o sensor sem ferramenta.

Primeiramente foi elaborado um processo para a determinação da posição em relação ao eixo Z na qual o sensor é acionado, para tanto foi executada a medição com auxilio de um relógio comparador. Para realização desta medição encontrou um problema que se tratava de que o eixo arvore não era capaz de atingir a posição Z em que o sensor era acionado sem que tivesse alguma ferramenta inserida, portanto utilizou-se um relógio comparador para determinar o tamanho de um pino padrão e posteriormente determinou-se a altura (posição Z) em que o conjunto eixo arvore mais pino padrão acionavam o sensor, como mostrado na Figura 3. Com esses dados obtivemos, por meio de cálculos simples, a posição em relação ao eixo Z em que o sensor é acionado.

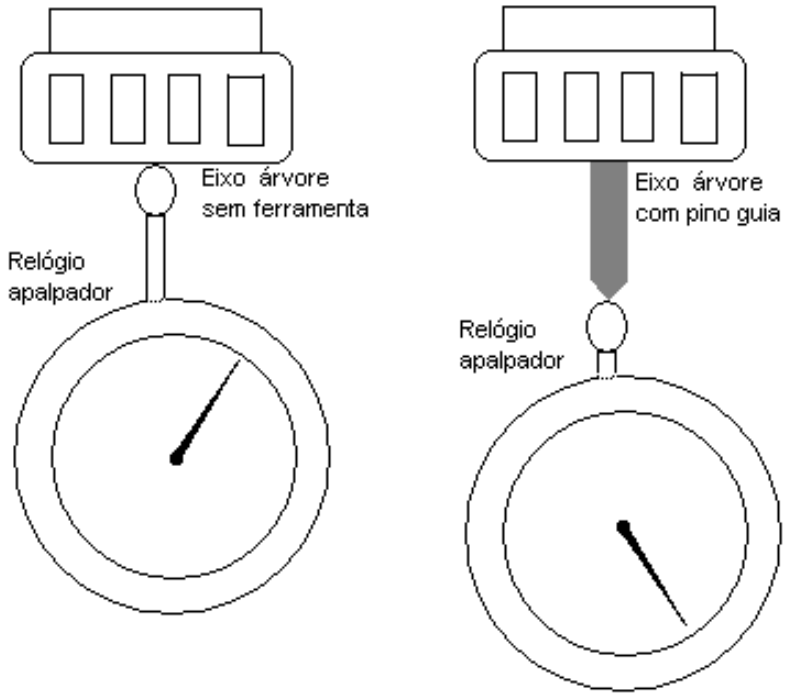

Figura 3 - Ajuste do pino padrão auxílio de um relógio comparador

Uma vez já conhecida à altura em que o eixo árvore acionaria o sensor podemos iniciar o processo de medição no qual medimos a posição em Z em que o sensor é acionado com a ferramenta que desejamos medir e comparamos com a posição em Z do ponto em que o sensor é acionado pelo eixo arvore sem ferramenta e desse modo obtemos o tamanho da ferramenta.

Esse valor de tamanho da ferramenta é inserido no corretor de ferramenta e assim a ferramenta esta calibrada e ativando o corretor em qualquer programa podemos utilizar a ferramenta utilizada.

2.2.2 Cálculo da altura da ferramenta

A definição da altura da ferramenta utilizando-se o sistema de medição com apalpador é realizada através da comparação entre a posição de acionamento do sensor sem nenhuma ferramenta fixada no “spindle”, veja a Figura 4, com a posição registrada quando a ferramenta está fixada no “spindle”, veja a Figura 5.

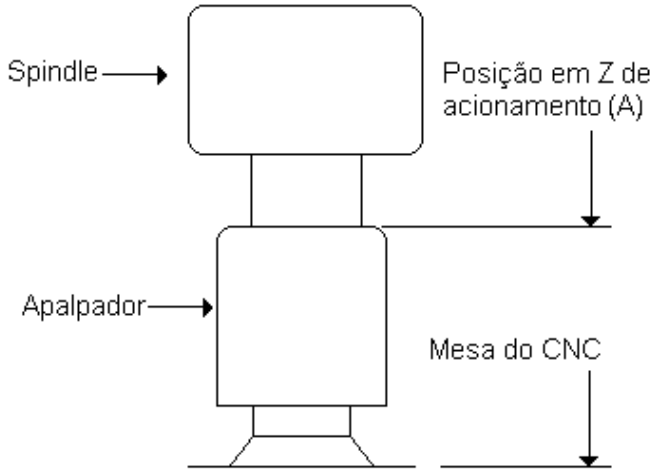

Figura 4 - Acionamento do apalpador sem ferramenta

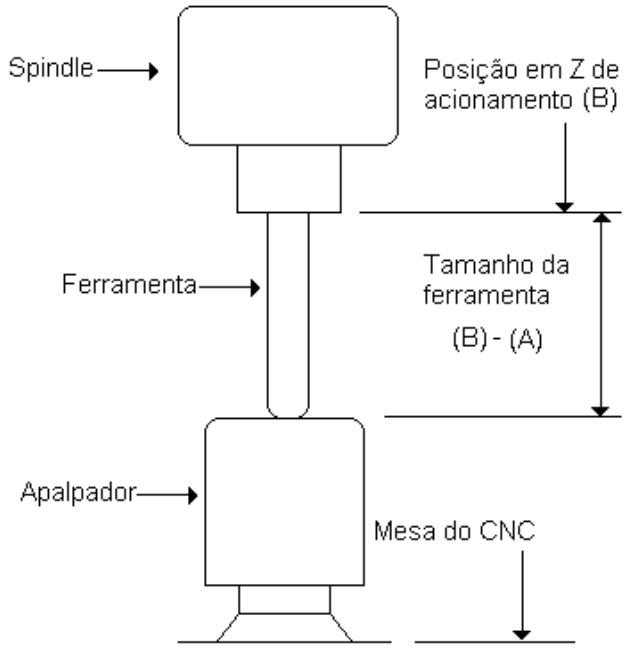

Figura 5 - Acionamento do apalpador com a ferramenta a ser medida

2.2.3 Comando Siemens

2.2.3.1 Linguagem G

Para os comandos Siemens a linguagem G apresenta sua sintaxe baseada em comandos de movimentação, comandos de controle de fluxo e variáveis locais e globais. O código é executado sequencialmente e apresenta grandes limitações se comparado a linguagens convencionais. No apêndice A apresenta-se detalhadamente a estrutura de cada um dos comandos da linguagem G para comandos Siemens.

2.2.3.2 Telas Gráficas

Para o desenvolvimento de telas gráficas em comandos Siemens é utilizada uma linguagem própria com formato parecido com o HTML, linguagem esta que é interpretada e não necessita de compilação. No Apêndice B são detalhadas todas as funções e estruturas disponíveis na programação de telas gráficas em máquinas Siemens através de arquivos de texto.

2.2.3.3 Rotina de Medição

O procedimento de medição realizado que utiliza o comando Siemens é baseado no processo de medição padrão. Para realizar a medição são utilizados os parâmetros $AA_PROBE[n], $AA_MM[A], $TC_DP3[T,D], variáveis GUD e o comando MEAS.

A rotina de medição recebe como entrada T, D e o modo (0 para “Single” e 1 para “Multiple”). A rotina desenvolvida é apresentada abaixo:

```
;Inicialização
N5 PROC CALIBRA(INT T,INT D,BOOL MODE) SAVE
N10 DEF INT DIV ;VARIAVEL DE DIVISÃO
N20 DEF INT COUNT ;VARIAVEL DE MODO
N25 PDT = 0 ;INICIALIZA A ALTURA
N30 G90 G71 ;POSICIONAMENTO
N40 G01 F1500 Z200 ;AFASTAMENTO DE SEGURANÇA
N50 X=CENTROX Y=CENTROY ;CENTRALIZAÇÃO
N60 Z=(ZPROBE+10+ESTIMAAPROXIMAÇÃO EM Z)
N70 G91 MEAS=-2 G01 F250 Z-10
N80 IF $AA_PROBE[2] == 0
N90 ALARME = "Estimativa incorreta"
N100 ELSE
N110 Z5
N120 IF MODE == 0
N130 DIV = 1
N140 ELSE
N150 DIV = 3
N160 ENDIF
N170 FOR COUNT = 0 TO DIV
```

```
N180 G91 MEAS=1 G01 F100 Z-10
N190 STOPRE
N200 PDT = PDT + $AA_MM[Z]
N210 ENDFOR
N220 PDT = PDT/DIV
N230 ALTURA = ABS(PDT-ZPROBE)
N240 $TC_DP3[T,D] = ALTURA 20
N250 ALARME = "Ferramenta calibrada!"
N260 RET
```

2.2.3.4 Sub-rotina de cálculo do desgaste

A sub-rotina de desgaste foi desenvolvida de modo que possa ser utilizado a cada etapa do processo de usinagem e, assim aperfeiçoar o processo reduzindo o erro dimensional e por consequência o refugo. O código da sub-rotina é apresentado abaixo:

```
;Inicialização
N5 PROC DESGASTE(INT T, INT D, REAL TOL, BOOL MODE) SAVE
N10 DEF INT DIV ;VARIAVEL DE DIVISÃO
N20 DEF INT COUNT ;VARIAVEL DE MODO
N25 PDT = 0 ;INICIALIZA A ALTURA
N30 G90 G71
;Posicionamento
N40 G01 F1500 Z200 ;AFASTAMENTO DE SEGURANÇA
N50 X=CENTROX Y=CENTROY ;CENTRALIZAÇÃO
N60 Z=(ZPROBE+10+ESTIMA) ;APROXIMAÇÃO EM Z
N70 G91 MEAS=-2 G01 F250 Z-10
N80 IF $AA_PROBE[2] == 0
N90 ALARME = "Estimativa incorreta"
N100 ELSE
N110 Z5
N120 IF MODE == 0
N130 DIV = 1
N140 ELSE
N150 DIV = 3
N160 ENDIF
N170 FOR COUNT = 0 TO DIV
N180 G91 MEAS=-2 G01 F100 Z-10
N190 STOPRE
N200 PDT = PDT + $AA_MM[Z]
N210 ENDFOR
N220 PDT = PDT/DIV
N230 WEAR = ABS(PDT-ZPROBE)
N232 IF WEAR > TOL
N235 ALARME = "Desgaste maior que a tolerância"
N238 ELSE
```

```
N240 $TC_DP12[T,D] = WEAR
N250 ALARME = "Desgaste ajustado!"
N260 ENDIF
N265 G01 Z200 F250
N270 RET
```

A sub-rotina apresentada acima recebe como entrada T, D, a tolerância e o modo. Ao término da execução desta sub-rotina a máquina se posiciona em uma posição segura em Z.

2.2.3.5 Interface gráfica

Foi desenvolvida uma interface gráfica utilizando uma programação interna dos comandos Siemens que tem como função facilitar a utilização da rotina de medição para o ajuste inicial da ferramenta. A interface pode ser observada na Figura 6, as descrições dos campos são apresentadas abaixo:

1. Campo destinado à estimativa do tamanho da ferramenta;
2. T e D correspondem ao número da ferramenta e do corretor de ferramenta respectivamente;
3. Esta opção pode assumir os valores “Single” ou “Multiple”. Quando selecionada a opção “Single” apenas uma medição é realizada, no caso da opção selecionada ser “Multiple” serão realizadas três medições, a altura será obtida através da médias destas;
4. Uma vez preenchidos os campos acima se utiliza este botão para verificar se os valores dos campos estão corretos. Caso sim libera a execução da rotina de medição e caso contrário sinaliza o erro;
5. Este botão permanece desabilitado até que os campos de entrada sejam validados corretamente, uma vez feito isso, permite executar a medição.
6. Botão que Cancela o procedimento.

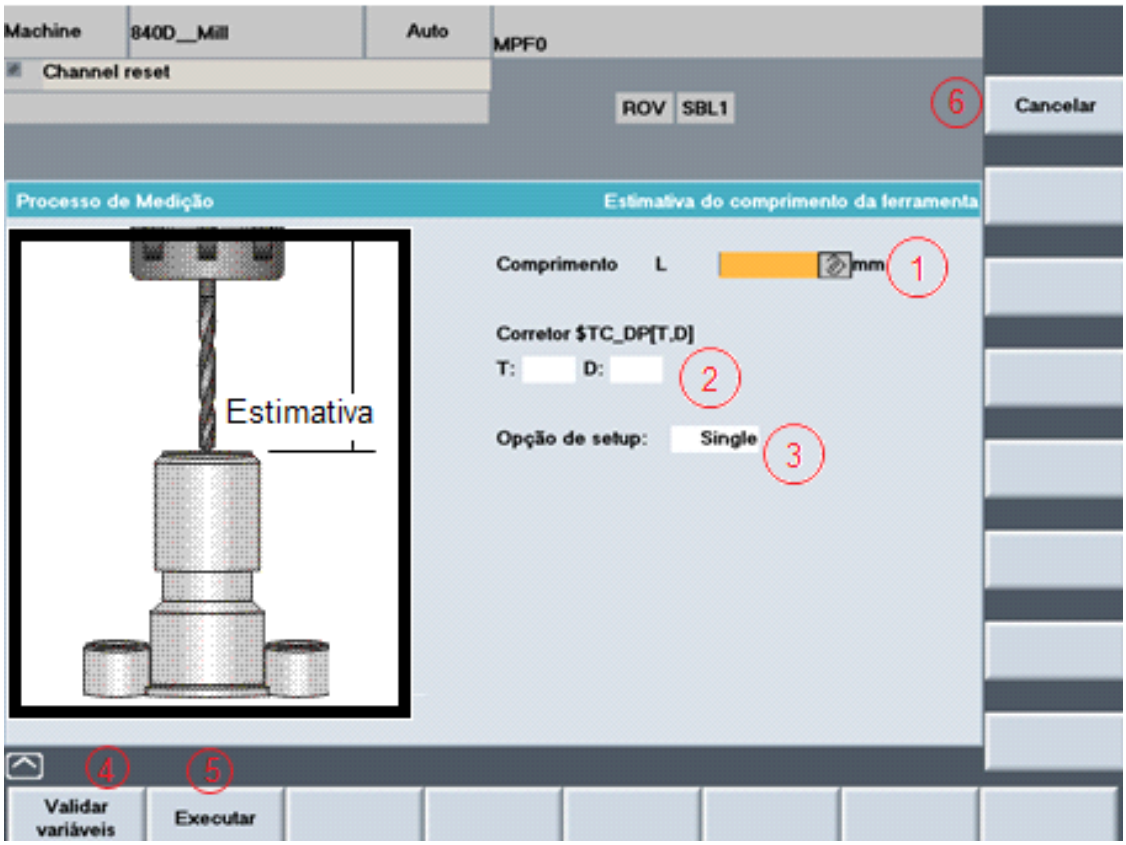

Figura 6 - Tela gráfica desenvolvida

O código desenvolvido para esta interface gráfica é apresentado abaixo:

```
;--------Definindo a start key---------------------------------------
//S(START)
HS2=("Probe Setup",ac7,se1)
PRESS(HS2)
DLGL("")
LM("form1") ;Carrega Tela Form1
END_PRESS
//END
;----------Tela form1------
//M(form1/"Processo de Medição"/"logo.bmp"/)
;--Definindo as variáveis
DEF Estima = (R///"Estimativa do comprimento da ferramenta",
"Comprimento","L","mm"///"ESTIMA"/280,40/,40/)
DEF LABEL = (S///,"Corretor $TC_DP[T,D]"////280,80/,80,0,0/)
DEF ToolT = (I/0,9//"Numero da Ferramenta","T:",,///"_T"/280,100,,50/295,100,50/)
DEF ToolD = (I/0,9//"Corretor da Ferramenta","D:",,///"_D"/330,100,,50/345,100,50/)
DEF Modo = (S/* "Single", "Multi"//,"Opção de setup:"////280,140/380,140,70)
;----------Definindo as teclas de menu--------
HS1=("Validar variáveis")
HS2=("Executar")
HS3=("")
```

```
HS4=("")
HS5=("")
HS6=("")
HS7=("")
HS8=("")
VS1=("Cancelar")
VS2=("")
VS3=("")
VS4=("")
VS5=("")
VS6=("")
VS7=("")
VS8=("")
;--Definindo os métodos e funções
LOAD
END_LOAD
UNLOAD
END_UNLOAD
PRESS(HS1)
IF CVAR == TRUE
        IF ESTIMA >= 100
                DLGL("Variáveis OK!!")
        ELSE
                DLGL(“O valor de L é muito pequeno”)
        ENDIF
ELSE
        DLGL("Há variáveis incorretas")
ENDIF
END_PRESS
PRESS(VS1)
EXIT
END_PRESS
PRESS(RECALL)
EXIT
END_PRESS
//END
```

2.2.4 Comando FANUC

2.2.4.1 Linguagem G

Para comandos FANUC a linguagem G é composta de comandos G, M e T, bem como seus parâmetros de sistema e suas variáveis auxiliares. No apêndice C são descritas todas as funções presentes na linguagem G para comandos FANUC.

2.2.4.2 Estruturação da Rotina de Medição

Para modularizar a rotina de medição primeiramente foi necessário dividi-la em tarefas simplificadas. Este procedimento pode ser realizado através da representação 23 0-Inicializa Parâmetros da medição1-Posicionamento seguro em Z2-Posicionamento seguro em X3-Posicioamento seguro em Y4-Posiciona no centro do Probe5-Aproximação em Z6-Realiza a medição7-Verifica a existência de erro e se existir sinaliza-o8-Calcula a altura da ferramenta e armazena no corretor

da rotina na forma de um fluxograma de tarefas que devem ser executadas baseado no procedimento padrão de medição. O fluxograma desenvolvido é demonstrado na Figura 7:

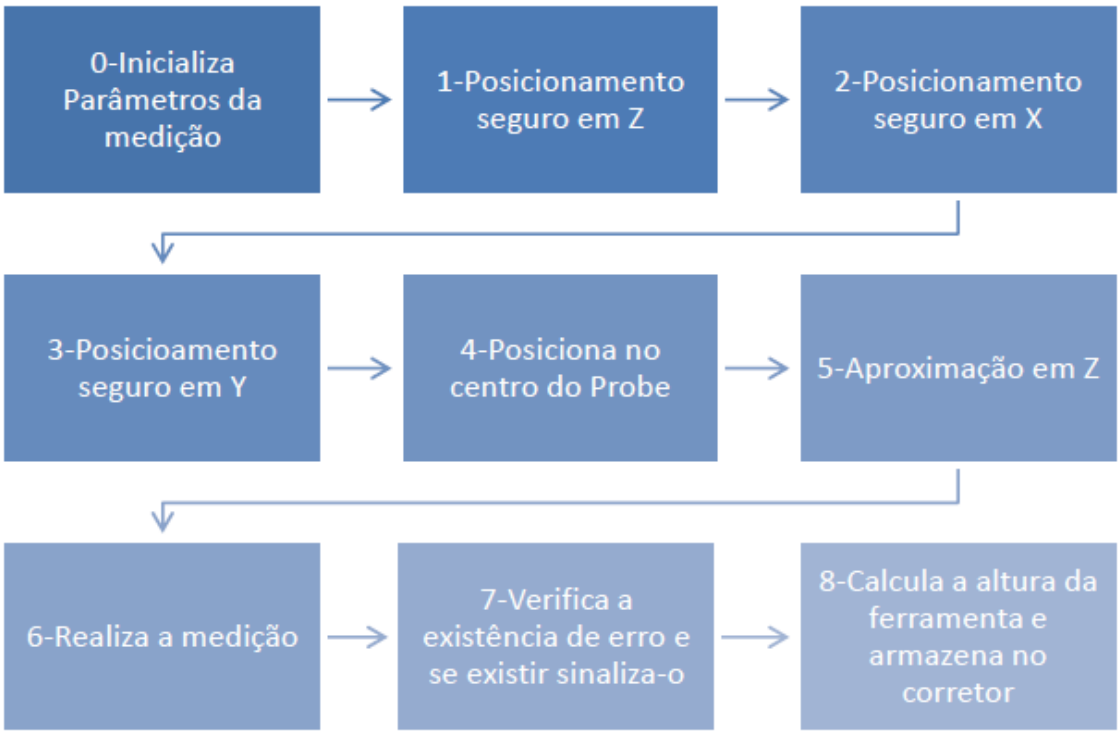

Figura 7 - Fluxograma de tarefas executadas na medição de ferramenta

2.2.4.3 Inicialização dos Parâmetros

Primeiramente detectou-se a necessidade de uma sub-rotina capaz de inicializar os parâmetros utilizados no processo de medição e definir as funções modais que devem estar ativas no início do processo. A sub-rotina de inicialização pode ser utilizada sempre que se deseja alterar os parâmetros e funções modais para as configurações iniciais. O código desta sub-rotina é apresentado abaixo e corresponde à etapa 0 do fluxograma da Figura 4.

```
%CONFIG
(IN: none); 24
O9116(configura o CNC) ;código utilizado para chamada da sub-rotina
G71 G54 G90 G94 ;inicializa as funções globais
#131=XPROBE ;armazena a posição X do sensor
#132=YPROBE ;armazena a posição Y do sensor
#133=ZPROBE ;armazena a posição Z do sensor
#116=3000 ;armazena parâmetro de velocidade 1
```

```
#126=1000 ;armazena parâmetro de velocidade 2
#136=250 ;armazena parâmetro de velocidade 3
M99 ;fim de sub-rotina
%
```

2.2.4.4 Teste de Erros da Movimentação

Nos comandos FANUC para verificar se uma movimentação foi realizada por completo ou se ocorreu algum erro deve-se executar um conjunto de testes lógicos. Portanto desenvolveu-se uma sub-rotina que recebe como parâmetros: a posição de origem (A), a posição do eixo no fim da movimentação (B) e posição de destino programada (C). Com estes parâmetros é possível determinar se o sensor está acionado no início da movimentação (A = B) ou se a movimentação terminou antes de tocar o sensor (B = C). Esta rotina é aplicada somente após uma movimentação de medição. Esta sub-rotina representa a etapa 7 e é apresentada abaixo:

```
%TESTE;
(IN: A B C);
O9114(verifica movimentação) ; código utilizado para chamada da sub-rotina
IF[#1 EQ #2]GOTO 1 ;testa se A = B caso sim vai para N1 25
IF[#2 EQ #3]GOTO 2 ;testa se B = C caso sim vai para N2
N1 #3000=1(PROBE OPEN);
N2 #3000=2(PROBE FAIL);
M99 ;fim de sub-rotina
%
```

2.2.4.5 Movimentação Segura

Como pôde ser observado no fluxograma da Figura 4, o processo de medição de uma ferramenta pode ser dividido em onze etapas básicas, onde cada uma destas etapas representa uma tarefa simples. Observou-se que as etapas 1, 2, 3, 4 e 5 tinham características muito similares sendo apenas diferenciadas pelas coordenadas da posição de destino da movimentação. Outro ponto importante é que estes cinco passos constituem em movimentações de segurança, ou seja, uma movimentação que é interrompida caso o sensor de medição seja acionado. Com base nas observações acima desenvolveu-se uma sub-rotina capaz de executar a movimentação para um ponto (X,Y,Z) qualquer e que pare a movimentação caso o sensor seja tocado (este procedimento previne, em partes, que o sensor seja danificado). O código desta sub-rotina é apresentada abaixo:

```
%SAFE_MOVE;
(IN: X Y Z F A);
O9113(movimentação segura);
S0 ;certifica-se que o eixo de rotação esta desligado
G31 X#24 Y#25 Z#26 F#6 ;realiza a movimentação e para se o sensor for acionado 26
IF[#1 EQ 1] GOTO 2 ;se A=1 é medição e não realiza os testes abaixo, caso contrário realiza
os testes
IF[#24 LT #5021]GOTO 1 ;testa se foi acionado em X
```

```
IF[#25 LT #5022]GOTO 1 ;testa se foi acionado em Y
IF[#26 LT #5023]GOTO 1 ;testa se foi acionado em Z
N2 M99 ;caso nenhum dos casos acima finaliza a sub-rotina
N1 #3000=7(PROBE TRIGGERED) ;caso contrário sinaliza que o sensor foi acionado e
interrompe o processo.
M99;
%
```

Onde os parâmetros de entrada X, Y, Z são coordenadas do ponto de destido da movimentação segura, F é o valor da velocidade de movimentação e A indica que é um movimento de medição caso A=1 ou uma movimentação normal para qualquer outro valor (mesmo se oculta a entrada A).

2.2.4.6 Cálculo da Altura e Validação do Número da Ferramenta

A etapa 8 é responsável por calcular a altura da ferramenta com base no posicionamento do sensor (previamente armazenado em variáveis globais utilizando a sub-rotina de inicialização) e dos valores calibrados utilizando o mesmo. Após calculado este valor este deve ser armazenado em uma variável de sistema que corresponda ao tamanho da ferramenta, este parâmetro é indicado pelo valor inicial 2800 somado ao número da ferramenta que está sendo medida, por exemplo, ferramenta 1 tem seu corretor na variável #[2801] (2800 +1).

Como o indicador do número da ferramenta é um parâmetro de entrada do sistema, para evitar erros devido a entrada incorreta deste valor, desenvolveu-se também uma sub-rotina (O9117) que será utilizada em conjunto da sub-rotina de 27

cálculo da altura (O9115) da ferramenta que determina se o número da ferramenta (T) é compatível Deste modo desenvolveu-se a sub-rotina de cálculo da altura como demonstrada abaixo:

```
%CALC_ALT;
(IN: V T)
O9115(calcula a altura) ;código utilizado para chamada da sub-rotina
G65 O9117 T#20 ;chama a sub-rotina de validação do parâmetro T (número da ferramenta)
#[2800+#20]=ABS[#22-#133] ;armazena o valor da altura da ferramenta no corretor
correspondente ao T se este for válido
M99 ;fim de sub-rotina
%
%TEST_T
(IN: T);
O9117(checa se T é válido) ;código utilizado para chamada da sub-rotina
IF[#20 GT 49]GOTO 1 ;caso T > 49 ou
IF[#20 LT 1]GOTO 1 ;caso T < 1 vai para N1
M99 ;caso contrário finaliza a sub-rotina sem parar o processo
N1 #3000=3(Corretor T inválido) ;para o processo e sinaliza o erro
M99 ;fim de sub-rotina
%
```

Nas sub-rotinas acima os parâmetros de entrada V e T são respectivamente a posição em que a ferramenta toca o sensor em Z e o número da ferramenta que está sendo medida (este valor deve estar entre 1 e 49).

2.2.4.7 Rotina Principal

Uma vez concluídas as sub-rotinas responsáveis por cada uma das tarefas que compõem o procedimento de medição, pode-se desenvolver uma rotina principal que utiliza as sub-rotinas criadas na ordem e quantidade adequada para efetuar a medição da ferramenta. A rotina principal recebe como entrada a estimativa do tamanho e o número da ferramenta, representados por C e T respectivamente. Apresenta-se abaixo o código da rotina desenvolvida:

```
%CALIBRA
(IN: C T)
O9111(calibra a ferramenta) ; código utilizado para chamada da sub-rotina
G65 O9113 Z200 F#126 ;realiza a movimentação segura para Z = 200
G65 O9113 X100 F#126 ; realiza a movimentação segura para X = 100
G65 O9113 Y100 F#126 ; realiza a movimentação segura para Y = 100
G65 O9113 X#131 Y#132 F#116 ; centraliza o spindle no centro (X e Y) do sensor
G65 O9113 Z[#133-#3-5] ; aproxima-se do sensor utilizando a estimativa de entrada
#3004 = 2 ;desativa o controle de avanço do usuário 29
G65 09113 Z[#133-#3+5] F#136 A1 ;realiza a medição
#3004 = 0 ;ativa o controle de avanço do usuário
G65 09114 A#1 B#5023 C[#133-#3+5] ;
G65 O9115 V#5063 T#20 ;calcula a altura e salva no corretor correspondente
G65 O9113 Z[#133-#3-5] ;recua 5mm em Z
;vai para a posição segura
G65 O9113 Z200 F#126 ;realiza a movimentação segura para Z = 200
G65 O9113 X100 F#126 ; realiza a movimentação segura para X = 100
G65 O9113 Y100 F#126 ; realiza a movimentação segura para Y = 100
#3000 = 4(ferramenta calibrada!) ;sinaliza que a ferramenta foi calibrada com sucesso
M99 ;fim da rotina de medição
%
```

2.3 Ensaio experimental

Foi realizado um ensaio experimental para teste da repetitividade de um apalpador utilizando a rotina e sub-rotina desenvolvidas por esse projeto de iniciação científica. O objetivo deste ensaio é a avaliação do sistema de medição em processo para uma situação de compensação dos erros de máquina. Neste foram realizadas várias operações de fresamento de 10 colos em alumínio nas seguintes situações:

- Sem o sistema de medição em processo a frio
- Com o sistema de medição em processo a frio
- Sem o sistema de medição em processo a quente

- Com o sistema de medição em processo a quente

Os corpos de prova eram de alumínio 7075, veja a Figura 8, o inserto era uma fresa de 10 mm de diâmetro, com quatro facas. Utilizaram-se as condições de usinagem abaixo:

Velocidade de corte: 100,0 m/min

Avanço: 0,05 mm/rotação

Figura 8 - Peça usinada durante o ensaio

Figura 9 - Fotos do processo realizado no ensaio

Observa-se na Figura 9 o procedimento de usinagem realizado durante o ensaio. A cada etapa de usinagem a calibração da ferramenta era realizada.

Depois de usinados os corpos de prova foram submetidos a uma medição utilizando um perfilômetro (medidor de perfil), para detectar a variação das medidas em cada caso. Da análise dos dados traçaram-se as curvas para os erros em cada um dos quatro processos. Como mostrado na Figura 11:

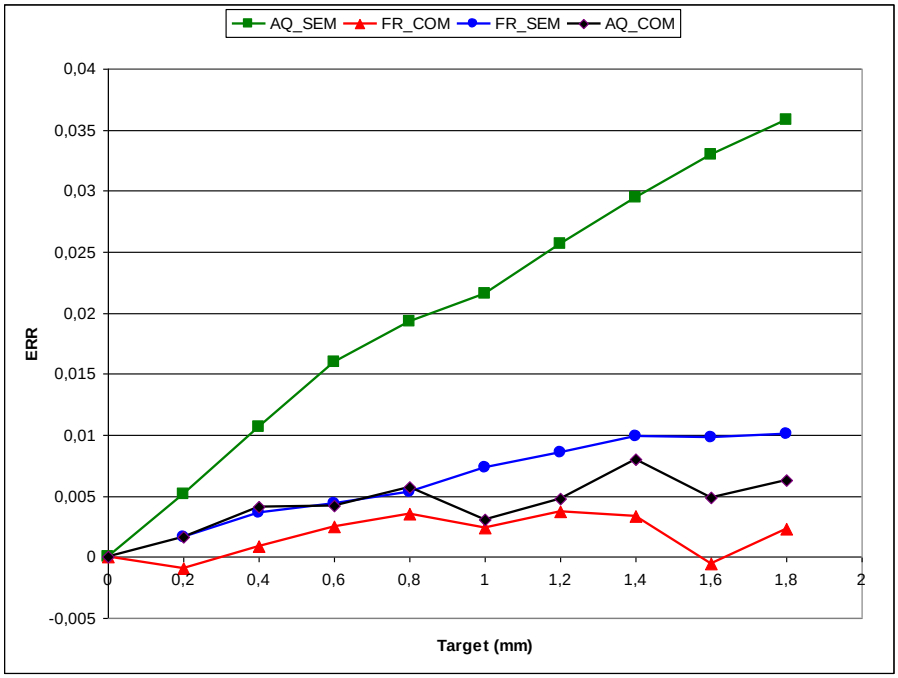

Figura 10 - Gráfico de erro por valor desejado

Podemos concluir a partir do gráfico da Figura 10 que com a utilização da compensação via apalpador o erro se mostra de modo aleatório, ou seja, este erro pode ser atribuído apenas à repetitividade do apalpador. Já nos processos sem compensação notamos que os erros são cumulativos, deste modo proporcionando erros de medida maiores, em especial no processo a quente.

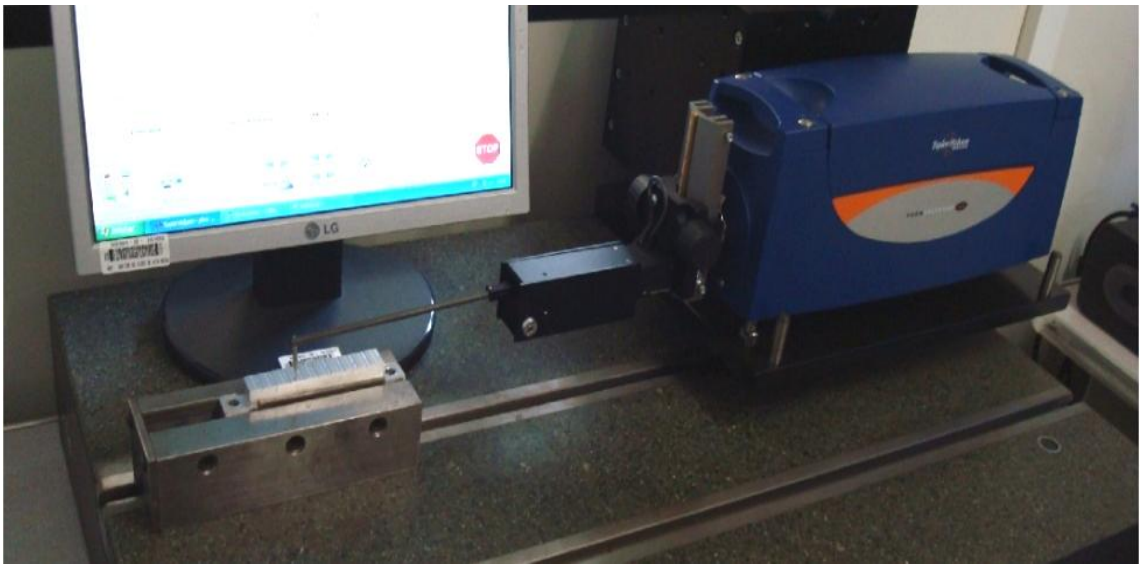

Figura 11 - Perfilômetro utilizado no ensaio

O processo a quente apresenta erros maiores uma vez que ocorrem dilatações térmicas (que não são compensadas automaticamente pela máquina uma vez que esta máquina não possui este sistema), mas estas são minimizadas pelo uso de um apalpador juntamente com os softwares desenvolvidos neste projeto.

3 Conclusões

O desenvolvimento deste projeto buscou utilizar ao máximo os conhecimentos adquiridos ao longo do curso de engenharia como, por exemplo, a estruturação de um projeto de software por meio de fluxogramas, o entendimento do funcionamento do hardware (CLP) contido na máquina-ferramenta que faz interface com as rotinas desenvolvidas, princípios de automação de processos entre outros.

Durante este trabalho de conclusão de curso pode-se observar o crescente desenvolvimento de aplicações na área de inspeção “in-situ” para máquinas ferramentas. Com os resultados apresentados pelo ensaio realizado pode-se concluir que mesmo em um caso onde o desgaste é próximo de zero, a utilização de um apalpador para eliminar erros dimensionais nas peças fabricadas é indicada. Como apresentado pelo gráfico de meta X erro na Figura 10, apenas a dilatação térmica da ferramenta já pode propiciar erros dimensionais em máquinas CNC, podendo estes ser minimizados utilizando-se o sistema proposto por este projeto de conclusão de curso.

Neste projeto os objetivos principais, que se resumem em facilitar e tornar intuitiva a utilização de sensores de medição, foram atingidos com sucesso e em algumas medidas superados como, por exemplo, a utilização de variáveis GUD que possibilita que dados internos de calibração dos sensores sejam protegidos contra a alteração indevida.

Acredita-se que este trabalho seja apenas um primeiro passo em uma cadeia de possibilidades para o desenvolvimento de medições em máquinas ferramentas. Deve-se observar que opções similares são fornecidas por empresas multinacionais fornecedoras de sensores deste tipo, mas estas tem alto custo e requerem, em geral, treinamento para utilização e manuseio que deve ser pago pelo cliente. Os resultados obtidos por este projeto auxiliam de maneira direta o desenvolvimento de sensores de medição para máquinas ferramentas sejam eles 2D ou 3D pois mostra a vantagem de sua aplicação mesmo em casos menos críticos.

O comando Siemens apresenta uma linguagem G com grande liberdade de sintaxe como, por exemplo, a definição de variáveis com nomes definidos pelo usuário, recurso este que não está disponível no comando FANUC. Contudo a programação utilizando-se o comando FANUC permite a utilização de variáveis globais de maneira direta enquanto na linguagem para comandos Siemens a implementação de variáveis globais requer um procedimento extenso de configuração.

Quanto ao desenvolvimento de interfaces gráficas, no comando Siemens é possível criar telas customizadas utilizando-se apenas um editor de texto convencional. No comando FANUC a criação de telas customizadas só é possível utilizando-se o software fornecido pelo fabricante.

Apesar da possibilidade de modularização das rotinas, que é possível com ambos os comandos, a linguagem G apresenta grandes limitações estruturais. Desde a criação dos comandos CNC a linguagem G evoluiu muito pouco comparada com outras linguagens

disponíveis atualmente. Os fabricantes de máquinas CNC optaram por manter a linguagem G simples ao custo de limitar suas capacidades.

Enfim este trabalho teve grande importância para o desenvolvimento de habilidades de pesquisa, planejamento de projeto, prevenção e solução de erros, entre outros conhecimentos bastante desejados para profissionais formados em engenharia.

Referências

[1] YANDAYAN, T.; BURDEKIN, M. (1997). In-process dimensional measurement and control of workpiece accuracy, International Journal of Machine Tools and Manufacture, V. 37, No. 10, P. 1423-1439.

[2] HUA JONG, D, (2006), Relatório Processo FAPESP:2006/02728-8

[3] BARNARD E. SMITH AND RAMAKRISHNA R. VEMUGANTI. A Learning Model for Processes with Tool Wear, Technometrics, Vol. 10, No. 2 (Maio, 1968), pp. 379-387. Publicado por: American Statistical Association and American Society for Quality

[4] GUERRA, M. D. Desenvolvimento de apalpador de contato elétrico (“Touch Trigger Probe”) para atuação no processo de torneamento. 2004. 92f. Dissertação (Mestrado em Engenharia Mecânica) – Departamento de Engenharia Mecânica, Universidade de São Paulo, São Carlos, 2004.

[6] SIEMENS AG, SINUMERIK 840D/840Di/810D Programming Guide Fundamentals, 2002, ed. 11.02

[7] SIEMENS AG, SINUMERIK 840D/840Di/810D Programming Guide Advanced, 2002, ed. 11.02

[8] SIEMENS AG, SINUMERIK 840D/810D/FM-NC Operator's Guide, 2002, ed. 10.00

[9] SIEMENS AG, SINUMERIK 840D/840Di/810D Commissioning HMI (IAM) – 01/2006 Edition

[10] J.Y. WANGA, C.R. LIUA AND K.K. WANG. The Effect of Tool Flank Wear on the Heat Transfer, Thermal Damage and Cutting Mechanics in Finish Hard Tuning. Escola de engenharia industrial, Purdue University West Lafayette, USA, Janeiro de 1999

[11] BONIFACIO, M. E. R.; DINIZ, A. E. O desgaste correlacionando da ferramenta, vida da ferramenta, aplaina aspereza e vibração da ferramenta no revestimento que gira com as ferramentas revestidas do carbide - Universidade estadual de Campinas, Desgaste (ISSN 0043-1648), vol. 173, no. 1-2, p. 137-144, abril de 1994

[12] DIAS COSTA, D;GLEBER PEREIRA, A, 2005, no. P. 02-03. Desenvolvimento e avaliação de uma tecnologia de baixo custo para programação CNC em pequenas empresas.

APÊNDICE A – Linguagem G para comandos Siemens

Linguagem G

A linguagem utilizada para a programação dos CNC's é a linguagem G que se trata de uma linguagem em modo txt com comandos que determinam a movimentação do eixo arvore do CNC e pode executar operações lógicas para controle de variáveis de sistema. Além de suas funções de posicionamento é possível controlar outras funções do centro de usinagem como velocidade de desbaste, controle de fluido e velocidade de rotação.

Posicionamento Absoluto ou Incremental

Utilizamos como sistema de posicionamento espacial um conjunto de três coordenadas referentes às distâncias de três eixos ortogonais X, Y, Z do sistema cartesiano de posicionamento como mostrado na Figura 12.

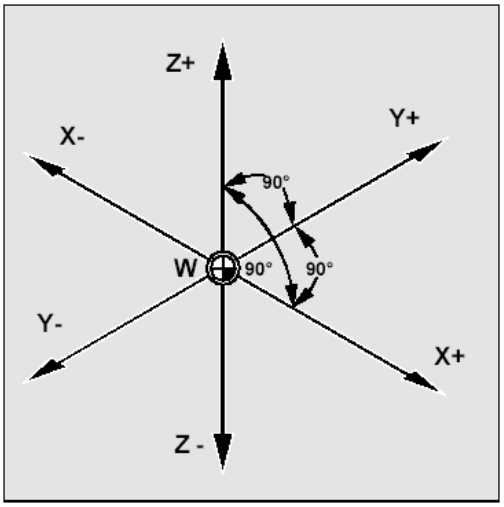

Figura 12 – Sistema de Coordenadas utilizado em máquinas CNS.

Quando usamos um sistema de posicionamento absoluto tomamos todas as posições em relação a um zero pré-definido e fixo. Deste modo um ponto localizado na posição (2;5;3), de modo que corresponda à (X;Y;Z), que é deslocado em três unidades na direção X será representado por (5;5;3), ou seja, seu posicionamento sempre se relaciona com as coordenadas com relação ao zero inicialmente definido.

Ao usarmos o sistema de posicionamento incremental sempre indicaremos a posição de um ponto em relação ao zero que se encontra na posição anterior ocupada por esse ponto, por exemplo, um ponto localizado na posição (2;5;3), de modo que corresponda à (X;Y;Z), que é deslocado em três unidades na direção Z será representado por (0;0;3), ou seja,não houve deslocamento em X e Y, mas somente em Z e de três unidades.

Parâmetros de Sistema

Existem três classes básicas de parâmetros na linguagem G, em especial voltada para máquinas ferramenta Siemens, que são parâmetros de sistema, parâmetros aritméticos e parâmetros definidos pelo usuário. Para nosso estudo são de fundamental utilidade os parâmetros de sistema que indicam estados do hardware, possibilitando-nos fazer uma interface hardware-software mais funcional.

Os parâmetros de sistema utilizados e sua respectiva descrição encontram se descritas abaixo:

$AA_PROBE[n] →parâmetro de deflexão do probe n (em geral n = 1)

$AA_MM[W] → parâmetro que armazena a posição no eixo W (X, Y ou Z)

$TC_DP3[t,d] → parâmetro que armazena o corretor de ferramenta para dimensão da altura para a ferramenta t e corretor d (em geral (t,d) → (1,1))

(SIEMENS, SINUMERIK 840D/840Di/810D Programming Guide Advanced, 2002, ed. 11.02)

Instruções de Inicialização

Algumas instruções têm por objetivo definir a interpretação dos dados inseridos na rotina em G, como por exemplo, os comandos G90 e G91 eles são usados para alternar entre os modos de posicionamento absoluto e incremental. Quando utilizamos o comando G90 definimos que os comandos de posicionamento recebem dados em coordenadas absolutas como apresentado acima, já quando usamos G91 indicamos a utilização de coordenadas incrementais. Deste modo podemos alternar livremente entre os dois modos de posicionamento. Na Figura 13 apresenta-se uma ilustração explicativa da utilização dos diferentes modos de posicionamento.

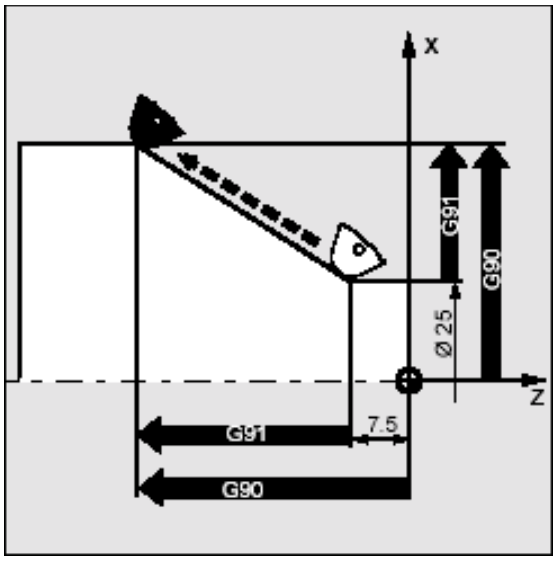

Figura 13 - Ilustração sobre movimentação incremental e absoluta

As instruções G70 e G71 são também usadas para configurar os parâmetros de entrada da rotina e nesse caso o G70 indica que as medidas serão consideradas dadas e polegadas e com G71 utiliza-se milímetros.

Todas as instruções demonstradas acima apresentam um comportamento característico denominado Modalidade, neste caso as funções são modais, ou seja, quando usadas suas configurações continuam validas até que sejam alteradas ou até o fim do programa.

(SIEMENS AG, SINUMERIK 840D/840Di/810D Programming Guide Fundamentals, 2002, ed. 11.02, P. 84-92)

Variáveis de Usuário

Para tornar o programa mais flexível podemos utilizar variáveis de usuário ou registradores da máquina já pré-estabelecidos chamados R. As variáveis são criadas na inicialização da rotina antes de qualquer instrução e podem ter características diferentes.

A declaração de variáveis de usuário é simples, utiliza-se a seguinte estrutura:

DEF {tipo} {nome}

Onde {tipo} refere-se ao tipo de dado armazenado pela variavel, por exemplo, inteira (INT), real (REAL), booleana (BOOL), entre outras, e {nome} indica o nome com o qual irá se referir a variável futuramente na rotina. Por exemplo:

```
DEF INT VAR
VAR = 32
G0 X20 Y=VAR
```

O exemplo acima demonstra uma implementação muito simples de variáveis de usuário, na primeira linha definimos a variável de nome VAR, na segunda atribuímos a ela um valor segundo seu tipo, inteiro, 32 e na terceira movemos o eixo arvore para a posição (20,32,0) no modo de posicionamento que estiver ativo.

Em processos automatizados podemos precisar de um conjunto de variáveis que possam ser acessadas seqüencialmente ou um conjunto de dados onde cada elemento possa ser acessado com maior praticidade a partir de um indicie, a este tipo de variável damos o nome array. Uma array nada mais é que um vetor de variáveis simples, ou seja, um conjunto ordenado de elementos que contem informações independentes armazenada em cada uma de suas componentes. Sua forma básica de declaração é mostrada abaixo:

DEF {nome}[n]

Para acessar as componentes desta variável durante a rotina utilizamos a forma {nome}[componente desejada] onde o a componente desejada é um numero que varia de 0 até n, que é o numero de componentes. Podemos também utilizar arrays de arrays assim criamos ao invés de um vetor uma matriz de elementos, isto permite o armazenamento de dados de um processo de modo estruturado. /segue abaixo um exemplo de aplicação de arrays:

```
DEF INT VAR[3]
VAR[2] = 5
VAR[1} = 45
G0 X20 Y=VAR[0]
G1 f250 Z=VAR[1]
G1 X20 Y=VAR [2] Z2
```

Neste exemplo temos a criação de uma array, seguida de sua inicialização, observe que os indicies para uma array de 3 componentes variam de 0 até 2, observe também que a instrução f250 que determina o avanço é modal portanto uma vez configurada permanece até que seja alterada ou até o fim do programa. Os parâmetros R da máquina podem ser acessados como array também onde R1 nada mais é R[1], R2 é R[2] e assim por diante.

Este tipo de variável permite que suas componentes sejam acessadas de maneira mais pratica em caso de funções de laço que serão descritas a seguir.

Funções de laço (loop's)

Em processos de usinagem é comum a necessidade de repetir uma tarefa pré-estabelecida por certo número de vezes para isso utilizou-se instruções denominadas funções de laço, são estas "For", "While", "Repeat-Until", entre outras. Iremos demonstrar o funcionamento de cada uma destas funções a seguir.

A função For permite que uma sequencia de instruções sejam executadas por um numero inteiro de vezes, cada iteração da função For automaticamente incrementa uma variável de usuário, que deve ser criada previamente, quando esta variável atingir um valor definido a rotina sai do laço. Abaixo se mostra a estrutura básica do laço "For":

FOR Variável = {valor inicial} **TO** {valor final}
Tarefa a ser executada
ENDFOR

Deste modo a tarefa a ser executada será realizada o numero de vezes que for necessário para que a variável utilizada no For atinja o valor final, lembrando que o incremento é de um por iteração.

A função "Repeat-Until" tem funcionamento semelhante, mas repete uma sequência de instruções por tempo indeterminado até que seja atingida uma condição então o laço termina. Sua forma básica é:

REPEAT
Tarefa a ser executada
UNTIL {Condição}

Outro comando é o comando "While" que tem seu funcionamento semelhante aos anteriores mas tem maior proximidade com o "Repeat-Until" ao passo de que este executa a tarefa e faz a verificação da condição após terminada a tarefa, entretanto "While"-Do verifica a condição antes de executar a tarefa, em alguns caso essas propriedade podem fazer grande diferença na rotina. Abaixo a forma básica do comando "While":

WHILE {condição}
Tarefa a ser executada
ENDWHILE

Estas funções são de grande utilidade quando realiza-se uma programação automatizada, ou seja, com a menor interferência do operador possível, deste modo podemos obter maior praticidade, flexibilidade e economia de tempo. Um exemplo de aplicação destas funções pode ser o seguinte:

```
DEF VAR
FOR VAR = 1 TO 3
G0 X10 Y10 Z=VAR
```

A cada iteração do For a variável VAR é acrescida de uma unidade sendo assim essa rotina executara o G0 apresentado acima três vezes. As condições citadas acima devem ser escritas na forma de expressões, com o auxilio dos sinais abaixo:

- Igualdade (==)
- Desigualdade (<>)
- Maior ou igual à (>=)
- Menor ou igual à (<=)
- AND Função lógica e (variáveis Booleanas)
- OR Função lógica ou (variáveis Booleanas)
- NOT Negação (variáveis Booleanas)
- XOR Função lógica ou exclusive (variáveis Booleanas)

Abaixo mostramos alguns exemplos de expressões baseadas em A = 10, B = 4 e C = 0, para as condições:

```
A == C  -> Retorna falso
NOT C  -> Retorna verdadeiro
B <= A  -> Retorna verdadeiro
B <> C  -> Retorna verdadeiro
```

Funções de Controle

Normalmente em processos de usinagem avançados necessitamos de testes de condição para que a partir de uma variável, ou parâmetro de sistema ou um evento ocorrido anteriormente a rotina se ajuste para melhor solução. A classe de funções que permite esta verificação é denominada funções de controle, por sua vez permitem que um parâmetro seja testado e que uma decisão seja tomada. A função mais utilizada nesta classe é a função “If-Else-Endif” que permite que se escolha entre duas alternativas, sua forma básica é mostrada abaixo:

```
IF {condição1}
Tarefa se a condição 1 for satisfeita
ELSE
Tarefa a ser executado caso contrário
ENDIF
```

Esta estrutura de controle é muito utilizada quando trabalhamos com parâmetros de sistema e temos que chegar a situação atual de um deles, para tomar uma decisão. Abaixo segue um exemplo pratico:

```
DEF VAR
VAR = 30
IF (VAR == 30)
G1 f250 X2 Z = VAR
```

```
ELSE
G0 Y = VAR
ENDIF
```

Neste caso a condição é sempre satisfeita mas esse exemplo é meramente ilustrativo, sempre que VAR = 30 executamos um G1 como acima e quando VAR diferente de 30 executamos o G0.

Funções de Posicionamento

A linguagem G tem como comandos principais de movimentação G01 G00 que executam movimentos com propriedades pré-definidas e velocidade máxima respectivamente. O comando G00 é utilizado na maioria das vezes para movimentações quando o spindle encontra-se distante da peça. Por outro lado, a função G01 permite controle de velocidade de avanço, sendo assim é muito utilizado durante o processo de usinagem.

O comando de movimentação tem a forma básica G{num1} f{avanço} X{p1} X{p1} Y{p2} Z{p3}, onde {num1} indica qual função G está sendo usada, {avanço} indica a velocidade de avanço nas unidades descritas pelo G70 ou G71 como descritos acima, {p1}, {p2}, {p3} são as coordenadas de movimentação nos três eixos de acordo com o sistema de posicionamento definido por G90 ou G91. Observe que o comando G0 não apresenta parâmetro de avanço, pois realiza a movimentação com avanço máximo. Por exemplo:

```
G90 G71
G1 f250 X20 Y20
G0 Z2
```

Este exemplo utiliza coordenadas absolutas e medida em milímetros devido a G90 e G71 respectivamente, e depois movimenta com avanço de 250 mm/min para a posição (20,20,0) em relação ao zero absoluto. Então recua para a posição (0,0,2) com relação ao zero absoluto com velocidade máxima de avanço.

Existem outros comandos de usinagem denominados ciclos que permitem a execução de sequências de comandos pré-estabelecidos a partir de parâmetros inseridos pelo programador na chamada do ciclo. Ciclos são sub-rotinas que podem ser chamadas de rotinas principais ou de sub-rotinas a qualquer momento.

Instrução Meas

A linguagem G apresenta uma instrução de medição chamada Meas que permite a movimentação do eixo arvore em uma direção determinada até que o parâmetro de sistema que indica que houve deflexão do “Touch-Trigger Probe” seja acionado. Possibilitando assim que possamos saber quando o sensor foi ativo e memorizar esta posição.

O comando Meas é utilizado com uma estrutura pré-estabelecida que apresente a utilização das funções de posicionamento conjuntamente. O comando Meas pode ser entendido como um comando de posicionamento que se movimenta até que seja acionado o sensor apalpador. A estrutura básica é apresentada abaixo:

MEAS = n {Comando de posicionamento};

Enfim o comando Meas é utilizado precedendo uma instrução de movimentação, onde essa movimentação estará sujeita ao fim de curso que será fornecido pelo apalpador indicado por n. Por exemplo, se tivermos apenas um apalpador e desejamos movimentar-nos em Z até que a ferramenta toque o este apalpador, com a velocidade de avanço de 100mm/volta e descendo no máximo 5 mm, a seqüência de instruções seria:

G91 G

MEAS = 1 G01 f100 z-5

Sub-rotinas e variáveis de usuário em linguagem G

Sub-rotinas

Nas linguagens de programação é comum a utilização de rotinas que utilizam sub-rotinas que executam tarefas especificas. A utilização de uma sub-rotina torna-se interessante quando as instruções contidas nela precisam ser utilizadas em mais de uma rotina, no meio de um processo, ou quando esta será utilizada mais de uma vez em uma mesma rotina.

As sub-rotinas na linguagem de programação utilizada em máquinas ferramentas CNC, ou seja, a linguagem G tem extensão diferente dos arquivos de rotinas principais. Os arquivos de programação de rotinas principais tem a extensão .MPF (main program file), uma vez que os arquivos de sub-rotina tem sua extensão como .SPF (sub-program file). Para a programação desses arquivos utilizamos um editor de texto padrão de modo que possamos salvar esses arquivos em .txt e posteriormente antes da inserção dos arquivos na memória da máquina CNC podemos alterar a sua extensão para .MPF e .SPF dependendo do tipo de rotina.

As sub-rotinas em máquinas CNC com comando Siemens apresentam um limite de onze rotinas aninhadas o que significa que ao cascatear uma seqüência de sub-rotinas chamando sub-rotinas dentro de sub-rotinas podemos formar no máximo 11 níveis de cascateamento, isso é ilustrado na Figura 14.

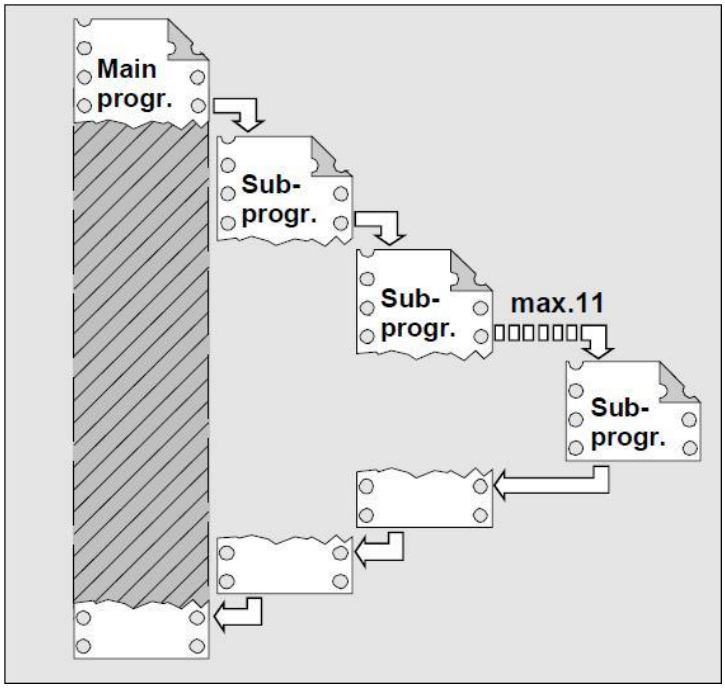

Figura 14 - Cascateamento de sub-rotinas

No desenvolvimento das sub-rotinas a programação se mostra similar com a programação de rotinas principais apenas alterando alguns parâmetros. Nas sub-rotinas ao invés de sinalizarmos fim de código com a instrução M30 terminamos com M02 ou com RET e para utilizarmos diferentes maneiras de posicionamento devemos sinalizar a cada linha qual o modo de posicionamento, por exemplo, para do spindle na posição X = 0, Y = 100, Z = 0 e posteriormente para a posição X = 20, Y = 0, Z = -50 a partir da posição inicial X = 15, Y = 0, Z = -3 devemos escrever a linha abaixo:

Exemplo 1 - Seleção do modo de posicionamento em sub-rotinas

```
;Modo de posicionamento absoluto ( com relação à origem)
G90 G1 X0 Y100 Z0 F1500
G90 X20 Y0 Z-50
;Modo de posicionamento incremental (com relação ao posicionamento anterior)
G91 G1 X-15 Y100 Z3 F1500
G91 X20 Y-100 Z-50
```

Observe que nos exemplos acima utilizamos a movimentação de todos os eixos ao mesmo tempo em cada uma das linhas, desse modo a maquina interpola os eixos de modo a percorrer o menor caminho entre os dois pontos, mas em alguns casos isso pode se tornar um problema, por exemplo, quando geramos sub-rotinas que serão usadas em vários processos diferentes e em que não é possível prever quais obstáculos estarão presentes no trajeto do spindle, neste caso devemos fazer a movimentação de cada eixo separadamente de modo a minimizar a possibilidade de colisão. Para realizar os movimentos dos eixos separadamente devemos escrever os comandos de posição em linhas separadas, lembrando de utilizar o seletor de modo de posicionamento em cada linha (G90 ou G91), como mostrado no trecho de código abaixo:

Exemplo 2 – Movimentação de cada eixo sem interpolação

```
;Modo de posicionamento absoluto (com relação à origem)
G90 G1 X0 F1500
G90 Y100
G90 Z0
G90 X20
G90 Y0
G90 Z-50
;Modo de posicionamento incremental (com relação ao posicionamento anterior)
G91 G1 X-15 F1500
G91 Y100
G91 Z3
G91 X20
G91 Y-100
G91 Z-50
```

Chamada de sub-rotinas

Quando desenvolvemos sub-rotinas em maquinas CNC podemos utilizar essa sub-rotina de dentro de qualquer outra rotina principal ou até mesmo sub-rotina. Quando uma sub-rotina é chamada em outra rotina o CNC carrega a sub-rotina e executa ela enquanto a rotina que executou a chamada permanece parada no ponto em que a sub-rotina foi chamada, ao término da execução da sub-rotina o CNC recarrega a rotina principal e continua a executá-la da linha seguinte da chamada da sub-rotina.

Para invocar uma sub-rotina utilizamos o comando CALL em conjunto com o nome da sub-rotina, lembrando que a sub-rotina deve estar na pasta SPF e deve ter sido pré-carregada. A instrução CALL tem a forma abaixo é exemplificada na Figura 15:

```
CALL <nome da sub-rotina>
```

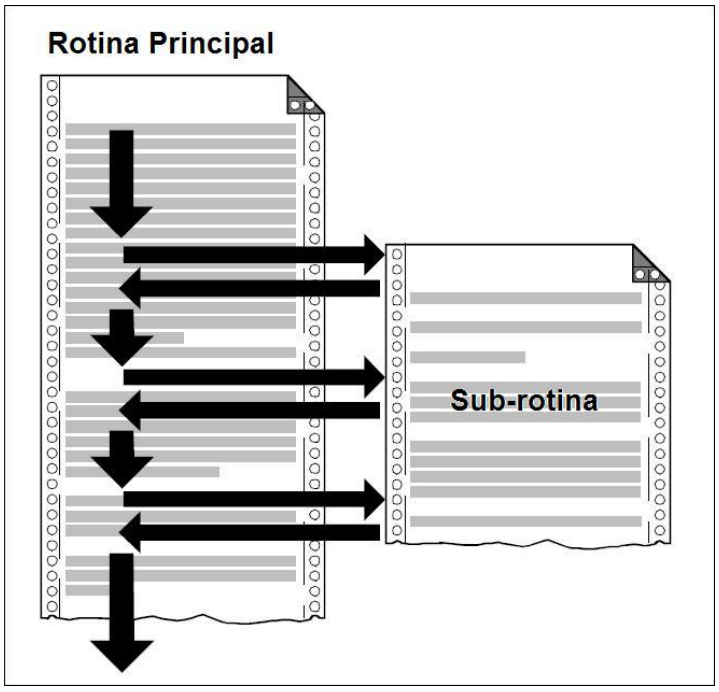

Figura 15 - Exemplo de chamada de sub-rotina

A chamada de uma sub-rotina fica bem representada na imagem acima. Para exemplificar uma chamada de sub-rotina vamos supor que tenha o seguinte conjunto de sub-rotina e rotina abaixo:

Exemplo 3 - Chamada de sub-rotina utilizando CALL

```
;Sub-rotina de posicionamento do próximo furo
;next.SPF
M05
G91 G0 Z10F1500
G91 Y15
G91 Z-10
M02
;Rotina de furação
;furar.MPF
VAR INT COUNT
G90 G1 Z0 F1500
X10 Y10
FOR COUNT = 5 DOWNTO 1
G91 Z-30 S5000 F350
Z30
CALL next
ENDFOR
G90 X0 Y0 Z0 F1500
M30
```

No exemplo acima temos uma sub-rotina que posiciona o spindle na posição onde o próximo furo será realizado, na rotina principal o spindle é posicionado na posição do primeiro

furo e então se inicia um loop que será executado quatro vezes e a cada vez irá ligar o spindle (instrução S5000) descer 30 mm, subir 30 mm, para retirar a broca de dentro do furo, depois chama a sub-rotina que posiciona para o próximo furo e ao término de cada posicionamento retorna-se para a rotina principal e continuando a sua execução do passo seguinte ao de chamada da sub-rotina. O exemplo acima resultaria em uma peça como a demonstrada na Figura 16:

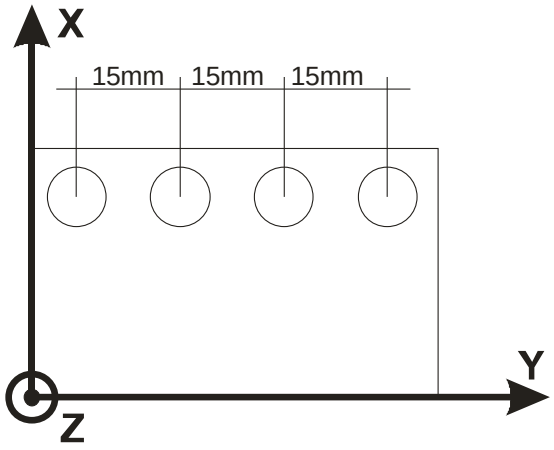

Figura 16 - peça gerada a partir do Exemplo 3

No caso de necessitar chamar uma sub-rotina mais de uma vez em seguida podemos utilizar o comando de chamada seguido do modificador de quantidade de execuções a serem realizadas em seqüência, deste modo o formato de chamada ficaria como a seguir e é exemplificado na Figura 17:

CALL <nome da sub-rotina> P <numero de vezes a ser realizado>

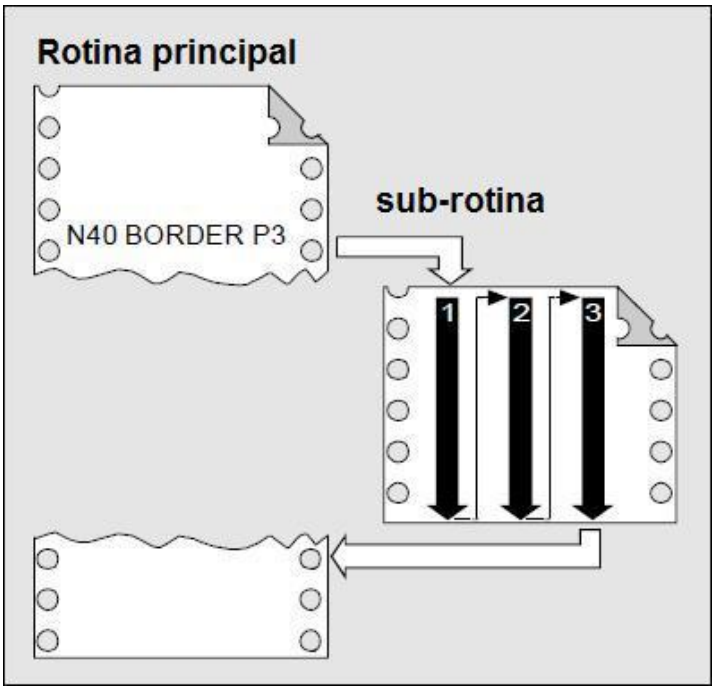

Figura 17 - Exemplo de execução repetitiva

Deste modo podemos chamar a sub-rotina mais de uma vez consecutiva facilmente, mas quando desejamos realizar essa sub-rotina sempre após cada movimentação podemos ainda usar o comando MCALL que caracteriza uma chamada de sub-rotina modal, ou seja, uma chamada de sub-rotina que permanece ativa até ser desativada, utilizando o comando MCALL sozinho, sendo assim ao termino de cada movimentação nos eixos a sub-rotina é chamada. O exemplo abaixo ilustra a utilização do comando MCALL, assim como o trecho de programação seguinte.

Exemplo 4 - Rotina utilizando MCALL

```
;sub-rotina – movex.SPF
G91 G1 X2
M02

;rotina principal – desbaste.MPF
G91 G53 S5000
G91 G1 Z-2 F2000
MCALL MOVEX
Z-3
Z-2
MCALL
M30
```

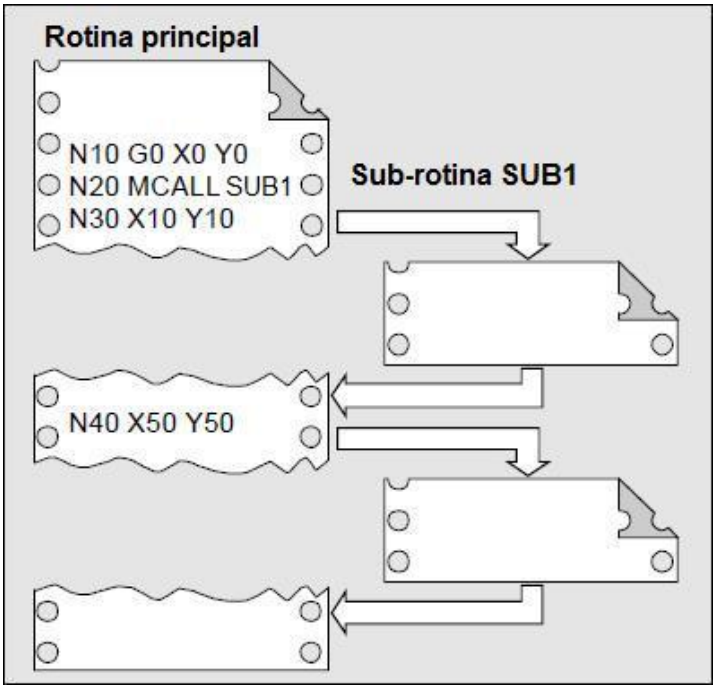

Figura 18 - Exemplo de utilização do comando MCALL

Na Figura 18 mostra-se o funcionamento de uma chamada de subrotina através de MCALL.

Nas rotinas acima executamos três degraus com profundidades diferentes com 2 mm de comprimento e essa dimensão é assegurada pela execução da sub-rotina movex.SPF no fim

de cada movimentação na rotina principal, observe que no fim da rotina principal utilizamos um MCALL sem identificador de sub-rotina para desativar a chamada modal de sub-rotina.

Podemos também chamar sub-rotinas de um dispositivo externo por rede, por disquete, ou comunicação serial utilizando o comando EXTCALL, junto a este existem muitos outros modos de chamada de sub-rotinas que podem ser utilizados em outras aplicações e suas especificações podem ser encontradas em **Programming Guide Advanced, 2002, ed. 11.02, seção 2**.

Variáveis globais de usuário (GUD)

Quando tratamos de variáveis de usuário no relatório anterior apresentamos uma estrutura de variáveis que facilitavam muito o processo de programação, mas apresentavam a limitação de serem exclusivas das rotinas que as criavam. Para resolução desses problemas estudou-se a possibilidade de criar variáveis globais, ou seja, que possam ser acessadas de qualquer rotina executada na maquina ferramenta e esta solução são as GUD`s (global user data).

As GUD`s apresentam como principal vantagem, além da possibilidade de ser acessada por qualquer rotina, ter um comportamento idêntico a uma variável comum quando se trata de programação. Desta forma tornou-se muito fácil adaptar os programas já desenvolvidos para utilização de GUD`s, permitindo maior flexibilidade para estes. O procedimento de criação das GUD`s é realizado na própria maquina CNC em poucos passos simples como descrito da Figura 19 à Figura 22:

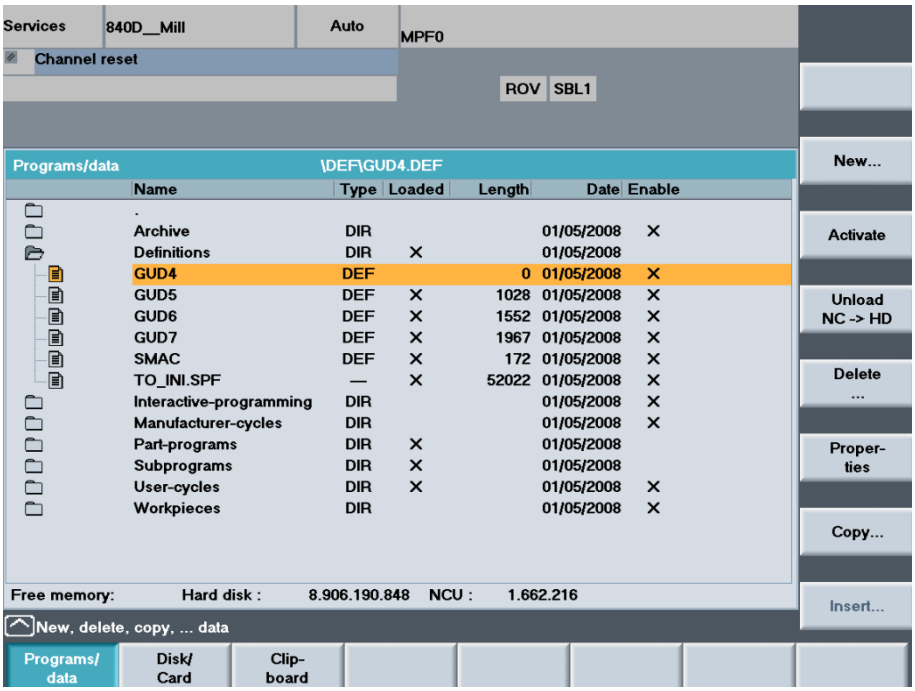

Figura 19 - Tela de arquivos de definição

Estando na tela de “Menu select” e escolhendo a opção “Service” entramos na tela mostrada acima, onde os principais arquivos de configuração da máquina CNC ficam armazenados. Entrando na pasta “Definitions” encontramos alguns arquivos de definição,

inclusive GUD`s, que são criados pelo fabricante, utilizando o comando “New...” surge a tela abaixo.

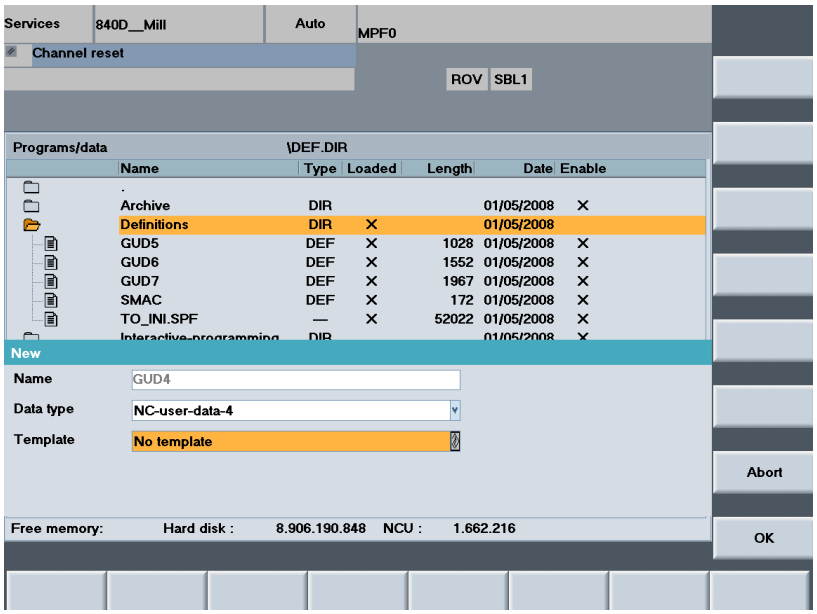

Figura 20 - Criando um arquivo de definição de GUD

Devemos selecionar na guia “Data type” a opção “NC-user-data-n”, onde n é o numero do conjunto de GUD`s, uma vez que podemos ter até nove arquivos de GUD`s diferentes. Após escolhido o tipo de arquivo, teclando “OK”, vemos a tela de criação do arquivo de definição das GUD`s como mostrada abaixo.

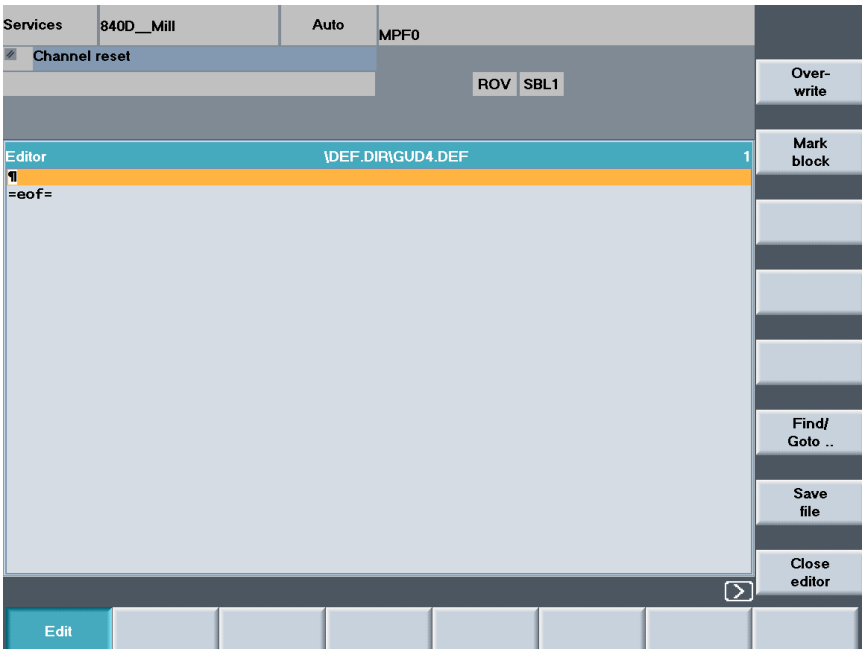

Figura 21 - Editor de código de geração de GUD

Nesta tela iremos inserir o código de definição das GUD`s e de definição dos níveis de acesso destas, como proceder na criação desse código trataremos mais adiante. Quando o

código estiver pronto devemos teclar “Save file” e então “Close editor” assim retornaremos para a tela “Program data” devemos então acionar a tecla active e a seguinte tela surgirá.

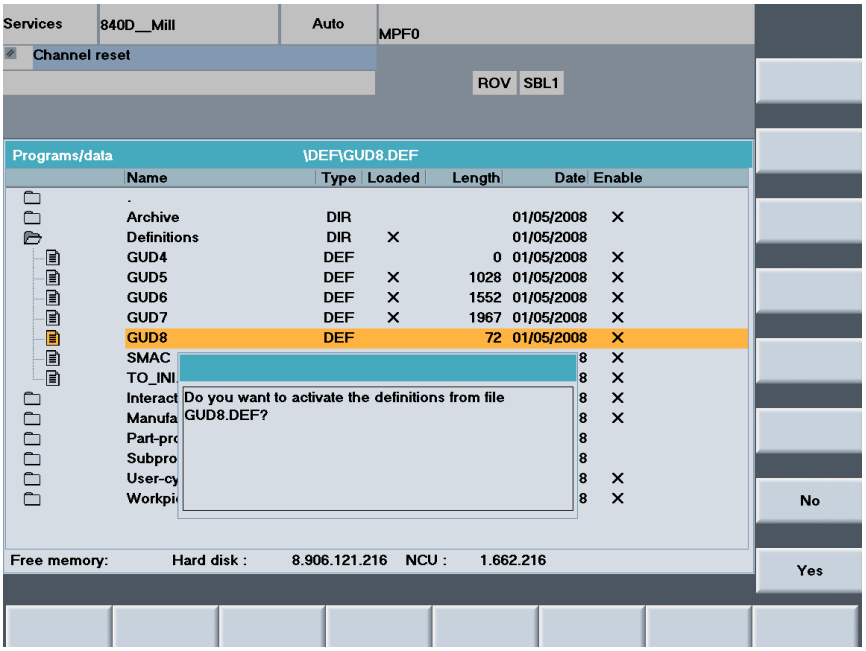

Figura 22 - Ativando as definições de GUD criadas

Pressionando a tecla Yes o arquivo de GUD criado será ativado e as GUD`s definidas nele estarão prontas para serem utilizadas após a reinicialização do CNC.

Para criar o arquivo de configuração das GUD`s podemos utilizar dois métodos, um deles é criar o arquivo de definição e depois carregá-lo via serial, disquete ou ethernet e o outro método é criando na própria máquina como ilustrado acima. Para tanto devemos utilizar o comando DEF NCK que deve ser utilizado da mesma maneira que utilizou-se o comando VAR para variáveis normais, abaixo exibi-se o formato de utilização deste comando:

DEF NCK <tipo> <identificador>

Onde <tipo> representa o tipo da variável à ser criada, por exemplo INT para inteiro, utilizando os mesmos tipos definidos para variáveis normais, e <identificador> representa o nome pelo qual a variável será acessada. Observe que as GUD`s podem ser declaradas como arrays de variáveis assim como as variáveis convencionais, inclusive com o mesmo modo de declaração exceto pelo DEF NCK. Deste modo declaramos variáveis que podem ser acessadas por qualquer programa executado no CNC, mas ainda não realizamos a definição de segurança que permite restringir o acesso das GUD`s de acordo com o modo de acesso da máquina CNC.

As maquinas CNC apresentam níveis de acesso que variam de 0 (acesso total) à 7 (acesso mais limitado), eles podem ser mudados de duas maneiras, os níveis de acesso de 7 a 5 são normalmente acessados através de um seletor via chave externo à máquina e os outros níveis podem ser acessados mediante a inserção de senhas na seção “Start-up” encontrada na tela “Program Menu” (elas devem ser obtidas com o fornecedor da máquina ferramenta). As GUD`s podem ser configuradas para que restrinjam seu acesso a níveis de acesso utilizando os comandos APR e APW como mostrado abaixo:

0 ou 10	Siemens
1 ou 11	OEM_HIGH
2 ou 12	OEM_LOW
3 ou 13	Usuário final
4 ou 14	Chave na posição 3
5 ou 15	Chave na posição 2
6 ou 16	Chave na posição 1
7 ou 17	Chave na posição 0

Significado dos níveis de acesso

APW <nível de acesso> APR <nível de acesso>

Devemos observar que é necessário que haja memória suficiente na máquina CNC para pré-carregar os arquivos de definição das GUD`s, caso este espaço não seja suficiente as definições das GUD`s não irão ser criadas. Abaixo mostramos um exemplo de arquivo de definição de GUD`s.

```
APW 7 APR 7
DEF NCK INT VAR1
DEF NCK BOOL VAR2
DEF NCK REAL VAR3[1,4]
```

Exemplo 5 - Criação de variáveis GUD

Depois de criadas as variáveis globais podem utilizá-las em qualquer rotina ou sub-rotina, sempre respeitando os níveis de segurança impostos, na criação, para leitura e escrita das mesmas. Outro fator que torna as GUD`s interessantes é o fato de poderem ser acessadas diretamente na máquina CNC (de acordo com o nível de segurança) por uma tela pratica que permite a inserção de dados nas GUD`s e/ou leitura destes conforme configurado no processo de criação. Para acessar as GUD`s através da máquina CNC devemos seguir os passos abaixo.

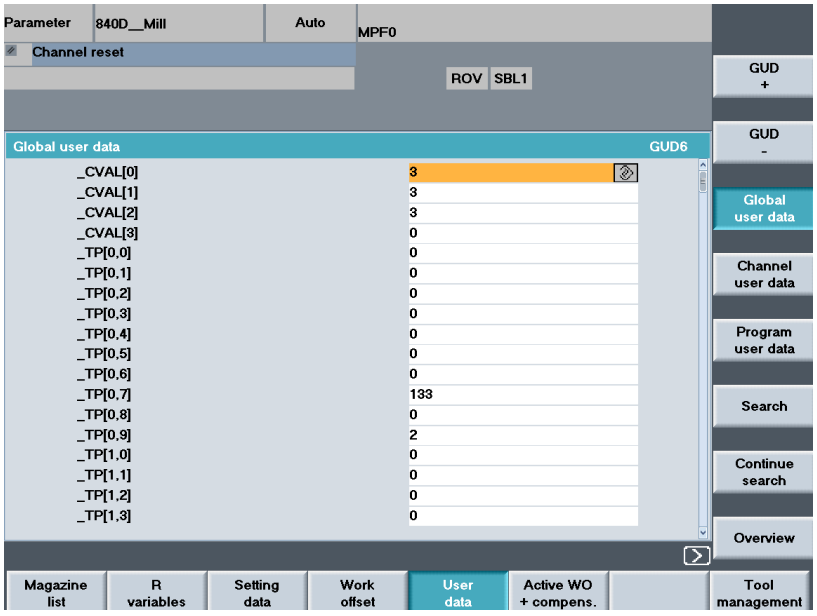

Figura 23 - Tela de visualização das GUD`s

Na tela de “Menu select” acionando a tecla “Parameters” e depois a tecla “User data” chegamos na tela mostrada na Figura 23. Para acessar o arquivo de GUD correto devemos utilizar as teclas “GUD +” e “GUD -” na figura acima estamos no arquivo de definição de GUD numero 6, esta informação pode ser vista no canto superior direito da tela mostrada acima. Observe que as GUD`s não estarão disponíveis para alteração caso o nível de acesso seja inferior ao definido no arquivo de definição quanto a escrita (comando APW) e nem se quer apareceram na tela casos o nível de acesso for menor que o definido quanto a leitura.

Para mudar o nível de acesso deve-se mudar a posição da chave externa (níveis 7 a 5) ou seguir os passo abaixo caso possua as passwords adequadas.

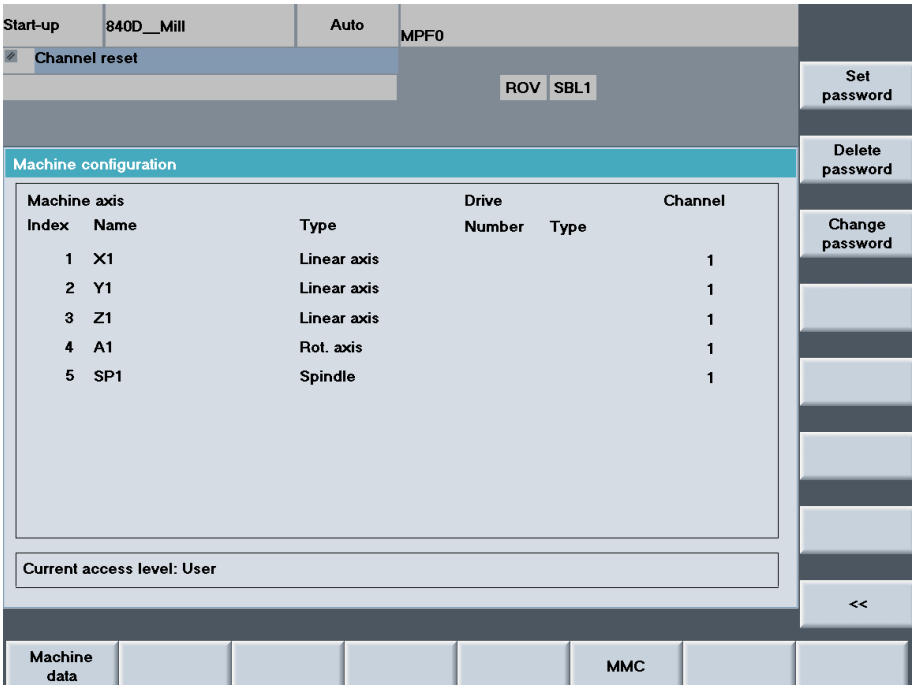

Figura 24 - Tela de service

Ao teclar “Service” na tela de “Menu select” surge a tela apresentada na Figura 24 onde devemos teclar “Set password” então a janela da Figura 25 abrirá.

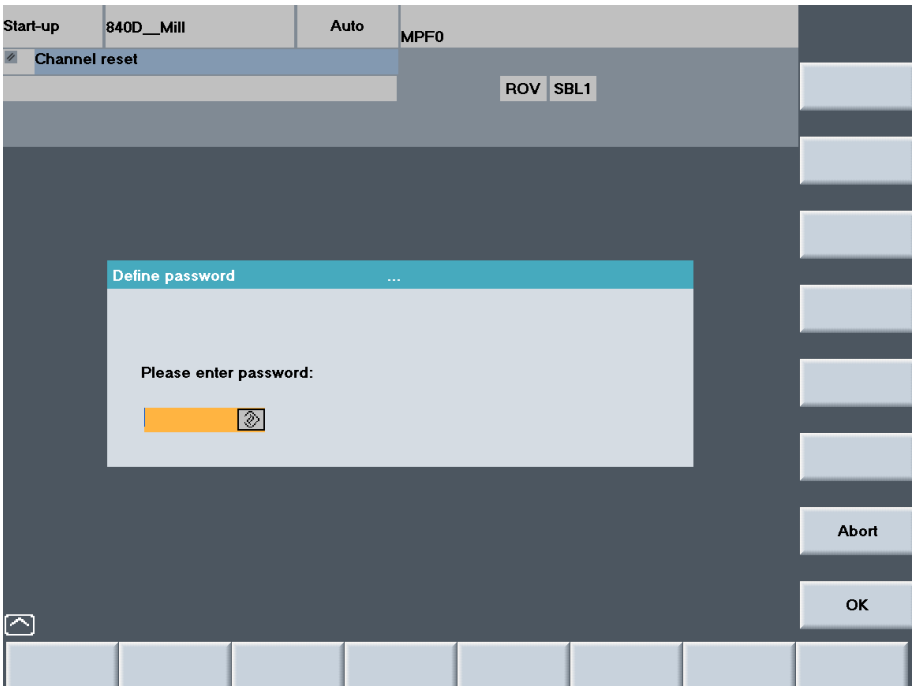

Figura 25 - Tela de inserção de password

Digitando a password adequada e teclando “OK” o nivel de acesso mostrado no campo “Current Access level” irá mudar para o nivel de acesso atual. Para resetar o nível de acesso utilizamos a tecla “Delete password”.

A principio todas as maquinas equipadas com 810D ou 840D suportam GUDS. A maneira de criá-los e ativá-los que é diferente. Depende do tipo de HMI (MMC102, MMC103, MMC100.2, PCU20, PCU50...) e da versão de System Software. Para obter essas informações é preciso consultar o manual ou entrar em contato com a Siemens. Na máquina utilizada neste projeto a compatibilidade foi verificada.

APÊNDICE B - Telas gráficas via arquivo de texto (.com) para comandos Siemens

A programação gráfica "wizard" ou "telas .com" funcionam a partir da versão de HMI 5.3. (com ou sem Hard Disk). Para versões mais antigas é preciso consultar o manual.

A criação de telas gráficas pode ser realizada de três maneiras que são:

- Programação via arquivo de texto
- Programação via visual basic
- Programação via Protools, software distribuído pelo fabricante

Mediante a viabilidade e visando o objetivo deste projeto, que pretende desenvolver um sistema de monitoramento de baixo custo, foi considerada a melhor opção a programação via arquivo de texto uma vez que a tecnologia de desenvolvimento (Bloco de notas) é acessível gratuitamente e também por diminuir a complexidade do desenvolvimento das telas gráficas. As duas ultimas opções apresentadas, além de serem geradas por softwares não gratuitos que necessitariam da compra de licença, demonstraram grande utilidade quando se deseja utilizar acionamento mecânico a partir de uma tela, por exemplo um pistão pneumático, mas para a nossa meta a primeira opção se mostrou mais favorável.

Diante disto nos próximos parágrafos iremos descrever o funcionamento da programação via arquivos de texto, que são chamados arquivos de configuração e que têm extensão .COM. Existem dois modos de interpretação das telas gráficas em máquinas Siemens que são HMI embarcado ou HMI avançado. Basicamente a diferença entre os modos de interpretação é o modo de se inserir os arquivos de configuração no sistema da máquina CNC. A máquina CNC utilizada neste projeto apresenta interpretador do tipo HMI avançado, utilizado em máquinas que contém hard disk, ou seja, o arquivo pode ser inserido via ethernet, caso contrário, se o interpretador fosse HMI embarcado o processo seria mais complexo, envolvendo modificar os arquivos de inicialização (.INI) da máquina via conexão serial.

Regras de sintaxe

Na linguagem de programação utilizada nestes arquivos de texto deve-se seguir algumas regras simples de sintaxe assim como todas as linguagens de programação. A tabela abaixo mostra alguns símbolos e suas funções.

;	Comentário
/	Separador de grupo de parâmetros, um grupo de parâmetros compreende um ou mais parâmetros
,	Separador de parâmetros individuais em um grupo de parâmetros
(e)	Sinalizadores de inicio e fim de argumentos

Quando desejamos utilizar a configuração padrão de um parâmetro ou grupo de parâmetros, basta apenas omitir seu valor, observe o exemplo abaixo:

Exemplo 6 - Formato de sintaxe

```
//M(identificador/[Cabeçalho]/[Gráfico]/[Dimensão]/[Variável de sistema ou  usuário],[Posição do gráfico]/[Atributos])
```

No exemplo acima os grupos de parâmetros são separados por “/” mas cada um dos grupos de sintaxe acima têm seus parâmetros individuais que devem ser separados por “,”. Observe o exemplo abaixo:

Exemplo 7 - Declaração com omissão de parâmetros

```
//M(mascara/”ciclo1”/“c1.bmp”/,,300,200//10)
;Nome do identificador: mascara
;Cabeçalho: ciclo1
;Gráfico: c1.bmp
;Dimensão: distancia da esquerda e do topo omitidas portanto padrão (dependem do sistema) e largura: 300 e altura: 200
;Variável de sistema ou usuário: nenhuma
;Posição x do Help: 10 pixels
;Atributos: nenhum
```

Estes exemplos ficaram mais claros quando definirmos quais os grupos de parâmetros e o significado de cada um de seus parâmetros individuais. Nesta linguagem as telas são chamadas de formulários.

Arquitetura de formulários (telas)

Ao criar um formulário ele contém elementos que podem ser utilizados apara interação do usuário com o processo, dependendo da complexidade dos formulários desenvolvidos pode se tornar interessante criar mais formulários que serão chamados a partir dos primeiros e assim por diante. Tanto a primeira tela customizada com todas as outras que podem ser chamadas a partir dela serão chamadas utilizando as teclas horizontais e verticais presentes na interface de máquinas CNC.

Cada formulário pode tanto chamar um outro formulário depois dele quanto anterior, podendo também chamar diretamente um formulário que seria chamado por etapas em um outro, ou seja, o acesso de formulários é tal que qualquer formulário pode acessar qualquer formulário dependendo exclusivamente da configuração das teclas horizontais e verticais.

Estrutura básica do arquivo de configuração

O arquivo de configuração consiste nos elementos abaixo:

Definição das Start Keys
Definição do formulário de tela
Discrição das teclas de menu
Descrição dos blocos

Exemplo 8 - Estrutura do arquivo de configuração

```
; Definição das Start Keys
//S (START)
....
//END
;Definição do formulário de tela
//M (.....)
;Definição das variáveis
DEF .....
; Definição dos blocos:
LOAD
...
END_LOAD
UNLOAD
...
END_UNLOAD
ACTIVATE
...
END_ACTIVATE
...
//END
; Definição das teclas de menu
//S (...)
//END
```

Definindo uma Start key

Para acessar um formulário customizado temos que configurar uma tecla que possibilite o acesso à esse formulário, esta tecla chamamos de start key, para definir uma start key devemos declará-la entre os identificadores de inicio (//S(START) e //END) sempre utilizando uma estrutura como abaixo:

Exemplo 9 - Estrutura básica de definição de start key

```
//S(START)                ;identificador de inicio
HSn=(“texto”)             ;rotulo para a tecla horizontal n
PRESS(HSn)                ;evento de pressionar a tecla horizontal n
LM(“Form”)          ;carrega o Form, onde Form tem que ser definido neste mesmo
arquivo (segundo argumento omitido)
END_PRESS                 ;fim do evento
//END                     ;fim do identificador de inicio
```

Observe que se desejarmos definir uma start key vertical devemos utilizar ao invés de HSn o comando VSn do mesmo modo, devemos também notar que podemos definir mais de uma start key em um mesmo arquivo de configuração assim como mais de um formulário.

Definindo o formulário de tela

O formulário de tela é o trecho do arquivo de configuração em que definimos o conteúdo da tela que estamos criando. Os formulários são compostos por variáveis e teclas (verticais e horizontais), no entanto as variáveis nesta linguagem de programação representam todos os elementos presentes em uma tela de CNC. Deste modo as variáveis podem representar campos para inserção de valores ou leitura dos mesmos, campos de opções do tipo “Dropdown Box”, rótulos com textos ou figuras. Neste campo também definimos quais as funções das teclas presentes nesta tela sendo elas oito verticais e oito horizontais.

Para declararmos os formulários de tela devemos fazê-lo dentro dos identificadores de formulário (//M e //END) e sua estrutura básica é mostrada abaixo:

Exemplo 10 – Estrutura básica do formulário de tela

```
//M...                    ;identificador de inicio de formulário de tela
DEF var1=...              ;definição da variável var1
HS1=(...)                 ;declaração da tecla horizontal 1
...
PRESS(HS1)                ;definição do evento de teclar HS1
      LM…
END_PRESS                 ;fim do evento
//END                     ;fim do formulário de tela
```

Alguns detalhes devem ser observados na criação dos formulários de telas:

- Os textos, nomes de arquivos e nomes de variáveis devem sempre ser apresentados entre aspas.
- Os identificadores sempre devem ser únicos e ter no mínimo 2 caracteres e no máximo 32. Os dois primeiros caracteres têm que ser letras ou o sinal de underline.
- O sistema é case insensitive, ou seja, não distingue maiúsculas de minúsculas.

As propriedades do formulário gerado são definidas no identificador de formulário de tela, seu formato básico é como mostrado abaixo:

```
//M(identificador/[Cabeçalho]/[Gráfico]/[Dimensão]/[Variável de sistema ou  usuário],[Posição do gráfico]/[Atributos])
```

Agora iremos definir cada um dos grupos de parâmetros apresentados acima, de modo a tornar mais compreensível como criar os identificadores de formulários de tela.

Identificador: nome pelo o qual o formulário será chamado

Cabeçalho: nome que aparece no titulo do formulário

Gráfico: imagem que desejamos utilizar !!!

Dimensão: posicionamento do formulário com quatro parâmetros individuais (distância da esquerda, distância do topo e largura, altura) separados por vírgula.

Variável: variável que receberá o foco ao abrir o formulário

Posição do gráfico: posição da imagem utilizada com dois parâmetros individuais (distância da esquerda e distância do topo) separados por vírgula.

Atributos: usado para configuração avançada (checar manual **Commissioning CNC HMI for Siemens seção 2**).

Depois de definida a estrutura básica do formulário, vamos definir os elementos que compõe este formulário, para variáveis veremos uma estrutura muito similar à apresentada acima, mas com algumas divergências em alguns grupos de parâmetros. Para iniciar a declaração utilizamos o comando DEF seguido do identificador igualado aos grupos de parâmetros como mostrado abaixo:

Exemplo 11 – Estrutura básica de declaração de variável

DEF identificador=
[1](Tipo
[2]/[limites ou campos de escolha ou identificador de tabela]
[3]/[Padrão]
[4]/[Textos]
[5]/[Atributos]
[6]/[Tela de help]
[7]/[Variável de usuário ou sistema]
[8]/[Posição da texto curto]
[9]/[Posição do campo de I/O]
[10]/[Cores]
[11]/[Help])

Tipo: podem ser:

R[x]	Real com x casas decimais
I	Inteiro
S[x]	Texto de comprimento x (String)
C	Caractere
B	Boleana

Limite: define os limites mínimo e máximo separados por virgula e nessa ordem. Define a variável como um campo de entrada de dados.

ou

Campos de escolha: as opções devem ser inseridas separadas por vírgula e o campo deve ser começado em * para sinalizar que é uma variável de Dropdown Box.

ou

Identificadores de tabela: iniciado por % para indicar uma variável de tabela, utilizada para armazenar uma grande serie de valores.

Padrão: o valor padrão da variável

Textos: composto de quatro parâmetros individuais que são:

Texto longo: Texto mostrado na tela

Texto curto: Nome do elemento

Texto gráfico: Texto referente ao termo na imagem

Texto de unidade: Texto de unidade

Podemos inserir uma imagem no lugar do texto curto.

Atributos: apresenta sete parâmetros individuais que são:

Modo de entrada: o valor padrão é wr2

Wr0 o campo de I/O invisível e texto curto visível

Wr1 leitura (não é possível entrada de dados)

Wr2 leitura e escrita, o valor tem q ser confirmado

Wr3 o mesmo que wr1 mas permite foco

Wr4 tudo invisível

Wr5 o valor digitado é salvo assim que digitado sem confirmação

Nível de acesso: o padrão é ac7

ac0 ... ac7 níveis de acesso de 0 à 7

Alinhamento do texto curto: o padrão é al0

al0 alinhado à esquerda

al1 alinhado à direita

al2 centralizado

Tamanho da fonte: o padrão é fs1

fs1 fonte no tamanho normal (8 pts)

fs2 fonte no tamanho dobrado

Checagem de máximo e mínimo: padrão é li3

li0 Sem checagem

li1 checa o mínimo

li2 checa o máximo

li3 checa o mínimo e o máximo

Tela de help: nome do arquivo de help entre aspas

Variável: os valores das variáveis de formulário podem ser associados à variáveis de sistemas ou de usuário (GUD).

Posição do texto curto: o posicionamento do texto curto ou imagem inserida como texto curto contém dois parâmetros individuais distância da esquerda e distância do topo, que devem ser inseridos separados por vírgula medida em pixels.

Posição do campo de I/O: o posicionamento do campo de entrada e leitura de dados contém dois parâmetros individuais distância da esquerda e distância do topo, que devem ser inseridos separados por vírgula medida em pixels.

Cores: composto por dois parâmetros individuais, cor de escrita e cor de fundo, que devem ser inseridas separadas por virgula e somente se aplicam ao campo de I/O.

Os valores de cores variam de 1 a 10 de acordo com a tabela abaixo:

Número	COR
1	Preto
2	Vermelho/Marrom
3	Verde escuro
4	Cinza claro
5	Cinza escuro
6	Azul
7	Vermelho
8	Marrom
9	Amarelo
10	Branco

Tabela 1 - Números correspondentes a cada cor

O padrão de cores varia de acordo com o modo de entrada selecionado, veja a tabela abaixo:

Modo de escrita	Cor de escrita	Cor de fundo
Wr0	Cor da janela	Cor da janela
Wr1	Preto	Cor da janela
Wr2	Preto	Branco
Wr3	Como wr1	Como wr1
Wr4	Como wr0	Como wr0

Wr5	Como wr2	Como wr2

Tabela 2 - Modos de entrada

Help: esta opção somente está disponível para máquinas que tem HMI avançado. Possui três parâmetros individuais que devem ser escritos separados por vírgula:

Arquivo de ajuda: caminho do arquivo PDF de ajuda.
Index: número de índice à ser carregado.
Texto de ajuda: caminho do arquivo de texto de ajuda.

Os arquivos PDF e TXT devem ter nomes completamente idênticos e serem armazenados nos diretórios, CUS.DIR\hlp.dir ou CST.DIR\hlp.dir. Os arquivos de ajuda são mostrados quando o cursor é posicionado sobre a variável e aciona-se a tecla help.

Abaixo se apresenta alguns exemplos de declaração de variáveis utilizando os grupos de parâmetros mostrados acima:

Exemplo 12 – Declaração de variáveis 1

DEF var1(R//45////"$TC_DP1[1,1]"//300,10,200)
Tipo: Real
Limites ou campos de escolha: nenhum
Padrão: 45
Textos: Nenhum
Atributos: Nenhum, aplicam-se os padrões
Tela de ajuda: Nenhum
Variável: variável de sistema $TC_DP1[1,1]
Posição do texto curto: Padrão
Posição do campo de I/O:
Distância da esquerda: 300
Distância da direita: 10
Largura: 200
Altura: Padrão segundo o sistema
Cor: Padrão
Help: Nenhum

Observe que quando queremos utilizar o valor padrão de um parâmetro apenas o omitimos, o posicionamento de tantos parâmetros juntos se demonstra um pouco confuso, mas depois de realizados alguns exemplos a compreensão torna-se mais fácil. Vamos a mais um exemplo:

Exemplo 13 - Declaração de variáveis 2

DEF Var2 = (I/30,50/23//wr1,al1///,,300)
Tipo: Inteiro

Limite: MIN: 30; MAX: 50
Padrão: 23
Textos: Nenhum
Atributos:
Modo de entrada: só leitura
Alinhamento do texto curto: alinhado à esquerda
Tela de ajuda: Nenhum
Variável: Nenhum
Posição do texto curto:
Distância da esquerda: Nenhum
Distância do topo: Nenhum, aplicam-se os padrões
Largura: 300
Altura: Nenhum
Posição do campo de I/O: Nenhum, aplicam-se os padrões
Cor: Nenhum, aplicam-se os padrões
Help: Nenhum

Propriedades de variáveis

As propriedades das variáveis contidas em formulários podem ser alteradas ou lidas utilizando a seguinte notação:

```
Identificador.(propriedade) = valor                    ;alterando propriedade
ou
var = Identificador.(propriedade)          ;lendo valor da propriedade
```

Mostraremos agora algumas propriedades e como acessá-las.

Tipo de variável

Para acessar utilizamos a forma Identificador.typ e assim podemos muda o tipo de uma variável, assim como mostrado abaixo:

Exemplo 14 – Alterando o tipo de uma variável

```
DEF var1 = (R//10)
HS1 = (“mudar tipo”)
PRESS(HS1)
        var1.typ = “I”              ;muda var1 de real para inteiro
END_PRESS
```

Valor de variável

Podemos alterar o valor de uma variável utilizando identificador.val e igualando com um valor correspondente com o tipo desta variável, do mesmo modo podemos utilizar esta instrução para ler o valor de uma variável.

Exemplo 15 – Alterando o valor de uma variável

```
DEF var2 = (S)
PRESS(RECALL)
var2.val = “valor2”
END_PRESS
```

Estado de variável

O estado de uma variável pode ser verificado a partir da instrução identificador.vld, assim podemos verificar se o valor inserido para uma variável é válido ou não.

Exemplo 16 – Checando estado de uma variável

```
CHANGE
        IF var3.vld == FALSE                ;se o valor for inválido insere o valor 10
                var3.val = 10
        ENDIF
END_CHANGE
```

Outras propriedades

Abaixo iremos listar algumas das outras propriedades que podem ser lidas ou alteradas:

identificador.**min** = Valor minimo
identificador.**max** = Valor máximo
identificador.**lt** = Texto longo
identificador.**st** = Texto curto
identificador.**gt** = Texto gráfico
identificador.**ut** = Texto de unidade
identificador.**wr** = Modo de entrada
identificador.**ac** = Nível de acesso
identificador.**al** = Alinhamento do texto da tela
identificador.**fs** = Tamanho da fonte
identificador.**hlp** = Tela de display
identificador.**var** = Variável
identificador.**fc** = Cor de escrita
identificador.**bc** = Cor de fundo
identificador.**htx** = Arquivo de ajuda, Index, Texto de ajuda

As propriedades das teclas de menu podem ser alteradas da mesma maneira que as propriedades das variáveis, utilizando SK.se e/ou SK.ac e/ou SK.st, que permitem a alteração do estado, nível de acesso e texto da tecla SK respectivamente, onde SK pode ser qualquer uma das teclas horizontais e verticais.

Definição de teclas de menu

As teclas de menu são usadas para detectar uma seleção de opção do usuário, ou seja, quando uma tecla é acionada podemos definir uma resposta da interface. A tela de máquinas CNC Siemens contém oito teclas horizontais, oito teclas verticais e uma tecla de recall

(retorno), que podem ser configuradas para desenvolver funções específicas, mas por enquanto vamos nos limitar à declaração destas teclas.

Para declararmos as teclas de menu utilizamos uma estrutura como a apresentada abaixo:

SK=(Texto,Nível de acesso,Estado)

Observe que SK deve ser substituído por HSn ou VSn, onde n é o número da tecla e H ou V indicam horizontais e verticais respectivamente. A tecla recall não deve ser declarada mas pode ser utilizada com os métodos que serão executados na definição dos blocos. Os parâmetros individuais de declaração das teclas de menu são definidos como abaixo:

Texto: Texto que rotula a tecla

Nível de acesso: nível de acesso necessário para utilizar a tecla, definido de ac0 até ac7 e o padrão é ac7

Estado: o estado tem como padrão é se1 e as opções são:

- se1: visível
- se2: desativado (texto em cinza)
- se3 visível (ultima opção usada)

Para serem utilizadas em métodos as teclas horizontais e verticais devem ser declaradas como descrito acima. Por exemplo podemos escrever:

Exemplo 17 – Definição de teclas de menu

```
VS1 = (“Tecla1”,ac5,se1)
HS6 = (“Tecla2”,ac2)
```

No exemplo acima duas teclas são definidas a primeira (tecla vertical 1) tem o texto “Tecla1”, requer nível de acesso 5 e seu estado é disponível, a segunda (tecla horizontal 6) tem o texto “Tecla2”, requer nível de acesso 2 e seus estado é o padrão (se1). Caso seja interessante nomear um grupo de teclas de menu podemos delcara-las entre os identificadores de bloco de start key, como abaixo:

```
//S(menu1)
        HS1=(“tecla1”)
        HS2=(“tecla2”)
        VS1=(“tecla3”)
        VS2=(“tecla4”)
//END
```

Métodos

Os métodos são utilizados para definir a resposta do sistema a alguns eventos que ocorrem ao se interagir com as telas gráficas. Esses eventos incluem o acionamento de teclas, carregamento do formulário, descarregamento do formulário entre outros que serão descritos aqui.

Juntamente com os métodos são utilizadas estruturas denominadas funções que serão discutidas posteriormente, mas além das funções podemos utilizar os comandos de controle de

fluxo e loop que se utilizou na linguagem G (IF,ELSE,FOR,etc). Abaixo mostramos a o padrão de utilização de alguns métodos.

Método PRESS

Este método é acionado quando alguma tecla é acionada. Sua forma básica é mostrada abaixo:

```
PRESS(SK)
(Código)
END_PRESS
```

Onde SK pode ser uma das 16 teclas (HS1 à HS8 e VS1 à VS8), a tecla de recall (RECALL), ou uma das teclas mostradas na tabela abaixo:

PU	PageUp
PD	PageDown
SL	Esquerda
SR	Direita
SU	Cima
SD	Baixo

Tabela 3 - Codificação das teclas utilizadas co o método PRESS

O método PRESS é utilizado para executar uma ação em resposta um acionamento de tecla, este método permite tornar o formulário mais interativo e fácil de ser utilizado. Segue um exemplo da utilização do método PRESS:

Exemplo 18 – Utilização do método PRESS

```
DEF var1 = (I//10/”contador”
HS1 = (“Tecla1”)
PRESS(HS1)
LM(“form1”)
END_PRESS
PRESS(SU)
var1 = var1 +1
END_PRESS
```

No exemplo acima quando acionamos a tecla seta pra cima a variável var1 é incrementada e quando acionamos a tecla HS1 o formulário form1 é chamado. Observe que utilizamos o comando LM, que é uma função que tem como função carregar formulários, esta e outras funções serão discutidas no tópico Funções.

Método LOAD e UNLOAD

Os métodos LOAD e UNLOAD são métodos opostos o primeiro é acionado quando o formulário esta sendo carregado e o segundo por sua vez é acionado imediatamente antes do formulário ser fechado. Suas formas básicas são as seguintes:

```
LOAD
(Código)
END_LOAD
e
UNLOAD
(Código)
END_UNLOAD
```

Estes métodos apresentam grande utilidade e podem ser usados principalmente para armazenar valores inseridos em campos de entrada ou recarregar estes valores para estes campos a partir de uma variável de sistema ou de usuário, como mostrado no exemplo abaixo:

Exemplo 19 – Utilização dos métodos LOAD e UNLOAD

```
DEF var2 = (R//"valor importante")
LOAD
var2 = $R[1]            ;transfere o valor do registrador R1 para var2
END_LOAD
UNLOAD
$R[1] = var2            ;transfere o valor de var2 para o registrador R1
END_UNLOAD
```

Método CHANGE

O método CHANGE é acionado quando o valor de uma variável especifica é alterado, este método se mostra muito interessante quando deseja-se fazer um controle quanto ao valor inserido pelo usuário em um campo. Sua forma básica é como mostrada a seguir:

```
CHANGE(identificador)
(Código)
END_CHANGE
```

Onde o identificador é o nome associado à variável que acionará o método ao ter seu valor alterado. Abaixo apresentamos um exemplo de utilização do método CHANGE:

Exemplo 20 – Utilização do método CHANGE

```
DEF var3 = (I* "tela1","tela2"///"Opção")
CHANGE(var3)
IF(var3 == "tela1")
        LM(form1)
ELSE
        LM(form2)
ENDIF
```

No trecho de código acima ao mudar o valor da variável var3, o sistema verifica qual das opções foi selecionada e carrega o formulário correspondente. Caso deseje-se utilizar o método CHANGE de maneira global, basta omitir o identificador como mostrado abaixo:

```
CHANGE()
    (Código)
```

```
END_CHANGE
```

Deste modo qualquer alteração nas variáveis presentes na tela fará com que o método CHANGE acima seja chamado.

Método FOCUS

Este método é utilizado para detectar que uma das variáveis foi selecionada (ganhou foco), mas os comandos ou funções utilizados dentro deste método não podem referir a outras variáveis senão a variável em foco e não é possível chamar outro formulário. A sua forma básica é mostrada abaixo:

```
FOCUS
    (Código)
END_FOCUS
```

Funções

Quando criamos arquivos de configuração temos uma variedade de funções que podem ser definidas para manipular os formulários de tela ou as teclas de menu, estas são usadas em conjunto com os métodos.

Função LM e LS

Esta função é utilizada para carregar um outro formulário, de maneira a fazer possível a navegação pelas telas customizadas. Sua estrutura básica é como mostrada abaixo:

```
LM(identificador,”arquivo”,MSx,VARx)
```

Onde identificador é o nome do formulário que deve ser chamado, arquivo representa o caminho para o arquivo de configuração que contém o formulário que será carregado, caso esse valor seja omitido o formulário será procurado no arquivo de configuração onde a função LM foi utilizada. Os valor de MSx determina o modo de mudança de formulário, sendo 0 ou 1 onde o formulário anterior é descartado no primeiro caso e no segundo ele é interrompido para a execução do próximo. A principal diferença entre os dois modos de chamada é que no primeiro caso o formulário anterior deixa de existir e o método UNLOAD é acionado, já no segundo o formulário anterior permanece pausado permanecendo como mestre durante a execução do próximo formulário, ou seja, não aciona o método UNLOAD, o padrão deste parâmetro é definido como 0. O parâmetro VARx é utilizado no caso em que MSx = 1 e representa as variáveis que serão transmitidas para o próximo formulário, podendo ser até vinte variáveis separadas por virgula.

Observe que apenas os valores, das variáveis transferidas por VARx, estarão disponíveis no próximo formulário e seus elementos não estão visíveis. Apresenta-se abaixo um exemplo

Exemplo 21 – Utilização da função LM

```
PRESS(HS3)
    LM(“form1”,1,var_trans)
```

```
        var_trans = 12
    END_PRESS
```

Observe que no exemplo, a variável var_trans é transferida para form1 e quando o form1 é finalizado a execução do formulário anterior continua do momento posterior a função LM (transferindo o valor 12 para var_trans), isto porque se utilizou MSx = 1, caso contrario não seria nem possível a transferência da variável var_trans.

Quando definimos as teclas de menu podemos nomear um conjunto delas, deste modo podemos se referir a um grupo de teclas de menu por um identificador. Com a função LS podemos carregar um menu através desse identificador, mas isso pode ser realizado de duas maneiras: sobrescrevendo todo o menu anterior ou sobrescrevendo apenas as teclas definidas no novo menu, essas opções são configuradas por 0 e 1 respectivamente colocados no campo merge presente na forma básica.

```
LS(identificador,"arquivo",merge)
```

Onde os dois primeiros parâmetros são análogos aos apresentados para a função LM e o terceiro é definido como descrito acima e seu padrão é 1. Para ilustar esta função propõe-se o exemplo abaixo:

Exemplo 22 – Utilização da função LS

```
PRESS(HS6)
    LS("menu1",,1)
END_PRESS
```

No exemplo as teclas do menu1 são carregadas mas apenas sobrescrevem as teclas de menu definidas no menu1.

Enquanto nenhum formulário for carregado, ou seja, nenhuma função LM for processada, só uma função LM ou uma Função LS e nenhuma outra mais pode ser configurada no método PRESS nos blocos de definição das teclas de menu. Para ambas as funções LM e LS só podem ser utilizadas com o método PRESS.

Função EXIT

Esta função procura por um formulário mestre e caso encontre retorna para este formulário fechando o formulário atual, caso não encontre nenhum formulário mestre, retorna para a tela padrão antes de ser iniciada a tela de usuário. Lembrando que formulário mestre é obtido quando utilizamos a função LM com o parâmetro MSx = 1. Sua forma básica pode ser das duas maneiras apresentadas abaixo:

```
EXIT
ou
EXIT(VARx)
```

No primeiro modo a função apenas executa os passos descritos acima e no segundo a função EXIT retorna para o formulário mestre, caso ele seja encontrado, e transfere as variáveis presentes no campo VARx, separadas por virgula (no máximo 20).

Exemplo 23 – Utilização da função EXIT como transferência de variveis

```
//M(form1)
...
PRESS(HS2)
LM(“form2”,,1,var1,var2,var3)
END_PRESS
...
//END
//M(form2)
...
PRESS(VS7)
EXIT(1,,num)
END_PRESS
```

No exemplo o form2 é chamado mediante o acionamento da tecla HS2 e form1 permanece como mestre e transfere as variáveis var1, var2, var3 para o form2. Quando o form2 recebe o acionamento da tecla VS7 ele é concluído e retorna ao form1 transferindo 1 para var1, o conteúdo de num para var3 enquanto var2 permanece inalterada.

Função EXITLS

Assim como a função EXIT ela finaliza o formulário atual e retorna para o anterior mas ao fazê-lo carrega um conjunto de teclas determinado. Seu formato básico é como mostrado abaixo:

```
EXITLS(identificador, “arquivo”)
```

Onde o identificador se refere ao nome associado a um grupo de teclas e o parâmetro arquivo se refere ao caminho para o arquivo de configuração que contém o grupo de teclas, em caso de omitir esse parâmetro o grupo de teclas de menu é procurado no arquivo de configuração atual.

Exemplo 24 – Utilização do comando EXITLS

```
PRESS(HS1)
        EXITLS(“barra1”,”MEUSMENUS.COM”)
ENS_PRESS
```

Funções RNP e WNP

Estas funções são utilizadas para interação direta com os valores de variáveis de usuário e/ou de sistema. Estes comandos devem ser usados juntamente com o método CHANGE caso o valor da variável varie muitas vezes seguidas, caso contrário se deseja acessá-la apenas uma vez, devem ser utilizados nos métodos LOAD e UNLOAD. Suas formas básicas são:

```
RNP(variável)
WNP(variável, valor)
```

Onde na primeira o valor da variável escolhida é lido e na segunda o conteúdo do campo valor é gravado na variável escolhida. Observe que caso o valor seja um texto devemos escrevê-lo entre aspas.

Exemplo 25 – Utilização das funções RNP e WNP

```
LOAD
        var1 = RNP("$TC_DP[1,1]")
END_LOAD
…
UNLOAD
        WNP(("$TC_DP[1,1]",var1)
END_UNLOAD
```

No exemplo acima, ao carregar o formulário var1 recebe o conteúdo de $TC_DP[1,1] e ao sair deste $TC_DP[1,1] recebe o valor de var1 novamente. As variáveis de sistema disponíveis podem ser encontradas em SIEMENS AG, SINUMERIK 840D/840Di/810D Programming Guide Advanced, 2002, ed. 11.02.

Função DLGL

Esta função permite escrever pequenos textos para informações adicionais sobre os formulários. Na Figura 26 mostra-se o campo onde a mensagem será mostrada.

```
DLGL("sua mensagem")
```

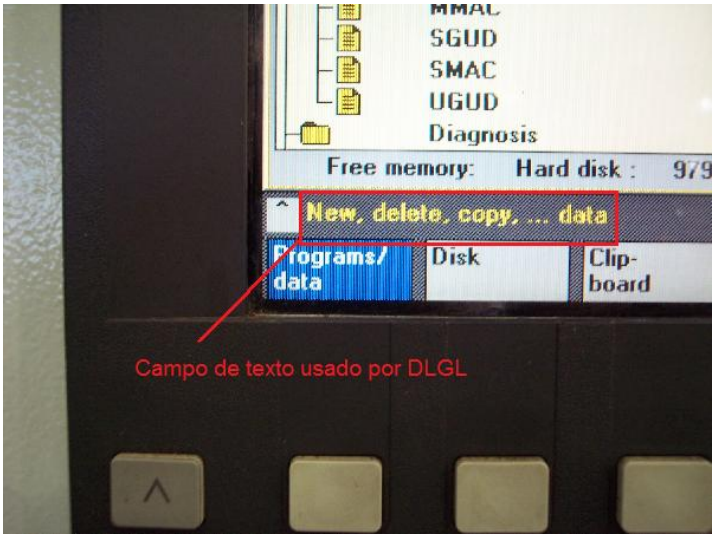

Figura 26 - Campo de texto utilizado pela função DLGL

Onde sua mensagem é apresentada no campo mostrado na figura acima. Podemos utilizar esta função para fornecer ao usuário mais informações sobre as variáveis como, por exemplo, mostrado no exemplo abaixo:

Exemplo 26 – Utilização da função DLGL

```
CHANGE(var1)
```

```
        IF var1 >= 100
                DLGL("o valor de var1 é muito grande")
ENDIF
END_CHANGE
```

Função CVAR

Para checar a validade de uma ou mais variáveis utilizamos a função CVAR, que retorna o valor false caso ao menos uma variável for inválida ou true se todas elas forem válida. Podemos utilizá-la de uma das duas maneiras abaixo:

```
CVAR
ou
CVAR(VARx)
```

Onde no primeiro caso todas as variáveis presentes no formulário são checadas e no segundo apenas as variáveis listadas no campo VARx serão verificadas. Observe que as variáveis inseridas no campo VARx devem ser separadas por virgula e não devem exceder o máximo de 29 variáveis ou 500 caracteres.

Exemplo 27 - Utilização da função CVAR

```
IF CVAR == TRUE
        VS8.se = 1
ELSE
        VS8.se = 2
ENDIF
IF CVAR( var1, var2) == TRUE
        DLGL("As variáveis var1 e var2 estão OK")
ELSE
        DLGL("Há um problema nas variáveis var1 e var2")
ENDIF
```

Nesse exemplo utilizamos CVAR para verificar a validade de todas as variáveis e caso todas sejam válidas libera-se o acesso à tecla VS8, caso contrário impede que ela possa ser acessada. Na segunda parte do exemplo testamos as variáveis var1 e var2 com a função CVAR e informamos com a função DLGL a validade ou não destas.

Função EVAL

A utilização desta função torna-se interessante quando queremos que uma expressão matemática, boleana ou de manipulação de strings (textos) seja processada. Sua forma básica é:

```
EVAL(exp)
```

Onde exp é uma expressão de um dos tipos citados acima. Abaixo mostramos um exemplo de utilização desta função e seu equivalente sem utilizar a função EVAL para que possamos entender sua utilidade:

Exemplo 28 - Utilização da função EVAL

```
Utilizando EVAL
DEF var1 = (I)
DEF var2 = (S)
DEF var3 = (S)
DEF var4 = (S)
DEF var5 = (S)
CHANGE()
        Var4 = EVAL(“var”<<var1)
END_CHANGE

Equivalente sem EVAL
DEF var1 = (I)
DEF var2 = (S)
DEF var3 = (S)
DEF var4 = (S)
CHANGE()
        IF var1 == 2
                var5 = var2
        ELSE
                IF var1 == 3
                        var5 = var3
                ELSE
                        IF var1 == 4
                                var5 = var4
                        ENDIF
                ENDIF
        ENDIF
END_CHANGE
```

No exemplo a variável var5 recebe o valor de var(x) onde x é o valor armazenado na variável var1, ou seja, caso var1 = 3 var5 recebe o conteúdo de var3, por comparação entre os dois trechos de código acima fica claro que a função EVAL simplifica a programação de arquivos de configuração. Observe que utilizamos no trecho do exemplo utilizando EVAL o operador << que é utilizado para concatenar strings, mais informações sobre este e outros operadores de strings serão discutidos posteriormente.

Registradores

Os registradores são utilizados para transferir valores entre formulário que não tem o vinculo mestre-escravo como descrito anteriormente. Existem 20 registradores disponíveis para uso do usuário (os outros são utilizados pela Siemens) e eles são tratados como variáveis globais, ou seja, podem ser acessadas por qualquer formulário. Ao se carregar a primeira tela customizada os registradores são carregados e associados ao valor 0 ou a uma string em branco. Neles podemos armazenar valores numéricos do tipo REAL ou textos do tipo STRING. Quando um

novo formulário é criado os valores dos registradores são automaticamente transferidos para ele. Os registradores podem ser utilizados como a seguir:

Exemplo 29 – Utilização de registradores e suas propriedades

```
REG[x]                          ;onde x é um valor de 0 à 19

IF REG[1].vld == FALSE ;verifica o estado de validez de REG[1]
REG[1].val = 2          ;associa o valor 1 ao valor de REG[1]
ENDIF
```

Algumas variáveis com funções específicas na programação de arquivos de configuração podem ser chamadas de registradores também algumas delas estão listadas abaixo:

```
FOC   ;contém o identificador da variável em foco
ERR   ;FALSE caso a última linha de código tenha sido executada sem erro
      ;TRUE caso a última linha de código não tenha sido executada sem erro
```

Ambas os registradores especiais mostrados acima devem ser utilizados apenas para leitura.

Operações com Strings

Quando desenvolvemos uma programação que tenha interface com o usuário seja qual for a linguagem a manipulação de strings se mostra muito favorável para que possamos customizar textos de acordo com os interesses do programador. Na programação de arquivos de configuração podemos manipular strings com algumas funções específicas que serão descritas abaixo.

Manipulando Strings

As strings (trechos de texto) devem sempre ser apresentadas entre aspas e devemos lembrar que esta linguagem de programação não faz distinção entre letras maiúsculas e minúsculas. Quando manipulamos strings as seguintes operações podem ser executadas:

Exemplo 30 – Manipulando Strings

```
Considere:
var1.val= "Isto é um"
var8.val = 4
var14.val = 8
var2.val = "erro"

Concatenando Strings
Var12.val = var1 << “programa”   ;Resultando em “Isto é um programa”

Deletando um String
Var10.val = “”                          ;Resultando em uma String em branco
```

```
Transferindo valor entre strings
var11.val = var1.val                    ;Resultando em “Isto é um”

Concatenando Strings com números
var13.val = “Manipulação número” << (var14 – var8) << “apresentada”
                                        ;Resultando em “Manipulação número 4 apresentada”

Usando Strings em estruturas IF
IF var2 == “erro”
        Var16.val =18.195               ;Resultando em var16 = 18.195 se var2 = “erro”
ENDIF

Indexando variáveis de sistema dinamicamente
var2.val = “$R[” << var8 << “]”         ;Resultando em var2 = “$R[4]”
```

Esta última manipulação se mostra muito interessante para escolha de variáveis de sistema a partir de uma entrada de usuário ou o acontecimento de algum evento especifico.

Função LEN

Esta função permite obter a quantidades de caracteres presentes em uma string ou uma variável do tipo string, sua forma básica é como mostrado abaixo:

```
LEN(String ou variável)
```

Observe que apenas podemos utilizar um dos parâmetros mostrados na forma básica. A seguir mostramos um exemplo:

Exemplo 31 – Utilização da função LEN

```
DEF var1 = (S)
DEF var2 = (I)
DEF var3 = (I)
LOAD
        var1 = “texto aleatório”
        var2 = LEN(var1)        ;Resulta em var2 = 15
        var3 = LEN(“texto”)             ;Resultando em var3 = 5
```

Observe que a função LEN considera que um espaço é um caractere assim contando-o também.

Função INSTR

Esta função nos permite encontrar uma string dentro de outra string, ou seja, podemos verificar se uma string está contida em uma outra. Sua forma básica é mostrada a seguir:

```
INSTR(começo,string1,string2,direção)
```

Onde começo corresponde à posição em que a busca deve ser iniciada, string1 é a string procurada, string2 é onde a string1 deve ser procurada e direção deve ser 0 ou 1 que correspondem à direção da esquerda para a direita e da direita para esquerda

respectivamente. Observe que esta função retorna o valor zero caso a string1 não for encontrada em string2 e caso seja encontrada retorna o valor da posição onde foi encontrada. Veja o exemplo abaixo:

Exemplo 32 - Utilização da função INSTR

```
DEF var1 = (S)
DEF var2 = (I)
LOAD
        var1 = “texto/exemplo”
        var2 = INSTR(1,”/”,var1,1)          ;Resultando em var2 = 6
END_LOAD
```

Funções LEFT e RIGHT

As funções LEFT e RIGHT permitem cortar uma string partindo da esquerda ou da direita em uma subseção de tamanha definido. Suas formas básicas são apresentadas abaixo:

```
RIGHT(string,tamanho)          ;inicia na direita
LEFT(string,tamanho)                    ;inicia na esquerda
```

Onde string é a string original e tamanho é medido em quantidade de caracteres, lembrando que espaço também é considerado um caractere. Observe o exemplo:

Exemplo 33 -Utilização das funções RIGHT e LEFT

```
DEF var1 = (S)
DEF var2 = (S)
DEF var3 = (S)
LOAD
        var1 = “texto/exemplo”
        var2 = LEFT(var1,5)             ;Resultando em var2 = “texto”
        var3 = RIGHT(var1,7)            ;Resultando em var3 = “exemplo”
END_LOAD
```

Função MIDS

Exerce a mesma função das funções apresentadas no tópico anterior mas extraindo uma subseção do meio de uma string. Sua forma básica é como mostrado abaixo:

```
MIDS(string,começo,tamanho)
```

Diferindo apenas no parâmetro começo que representa a posição do primeiro caractere presente na subseção que será extraida. Veja o exemplo abaixo:

Exemplo 34 - Utilização da função MIDS

```
DEF var1 = (S)
DEF var2 = (S)
LOAD
        var1 = “um/texto/legal”
        var2 = MIDS(var1,4,5)           ;Resultando em var2 = “texto”
```

```
END_LOAD
```

Função REPLACE

Quando desejamos substituir um trecho presente em uma string devemos utilizar a função REPLACE, uma vez que nos possibilita procurar uma string dentro de outra e substituí-la por outra que desejarmos. Abaixo mostra-se a sua forma básica:

```
REPLACE(string1,trechoprocurado,trechodesubstituição,começo,contador)
```

Onde string1 é a string original, trechoprocurado é a string q deve ser encontrada em string1, trechodesubstituição é o texto que será colocado no lugar do texto procurado, começo é posição inicial da busca e contador é a quantidades de caracteres máxima que a busca percorrerá. Observe que os valores que esta função pode retornar podem ser:

Se string1 = “” (texto em branco), retorna uma copia da string1.

Se trechoprocurado = “” (texto em branco), retorna uma copia da string1.

Se trechodesubstituição = “” (texto em branco), retorna uma copia da string 1 em que todas as ocorrências de trechoprocurado foram apagadas.

Se começo > tamanho da string1, retorna uma string em branco

Se contador = 0, retorna uma copia da string1.

Caso contrário, retorna a string1 com as ocorrências de trechoprocurado substituídas por techodesubstituição.

Exemplo 35 – Utilização da função REPLACE

```
DEF var1 = (S)
DEF var2 = (S)
LOAD
        var1 = “string original”
        var2 = REPLACE(var1,”original”,”com substituição”,1,15)
                                        ;Resultando em var2 = “string com substituição”
```

Implementando arquivos de configuração

Concluída a programação dos arquivos de configuração que serão utilizados na implementação das telas gráficas, devemos seguir alguns passos simples para que as telas estejam acessíveis ao usuário.

Neste projeto foi utilizada uma máquina que possuía hard disk e acesso via ethernet, portanto para implementar os arquivos de configuração é apenas necessário copiar o arquivo para a pasta !!!. Além disso devemos renomear os arquivos de maneira a indicar em qual tela basica (Tela padrão da máquina CNC) a sua “start key” estará visível e para cada tela devemos utilizar uma tecla específica como “start key”. Confira a tabela abaixo:

Tela Basica	Tecla Horizontal	Nome do arquivo
Tela machine e em modo JOG	1	MA_JOG.COM

Tela machine e em modo MDA	1	MA_MDA.COM
Tela machine e em modo AUTO	2	MA_AUTO.COM
Tela parameters	7	PARAM.COM
Tela program	8	PROG.COM
Tela services	7	SERVICE.COM
Tela Diagnostics	7	DIAG.COM
Tela Startup Menu expandido	7 6,7	STARUP.COM
Tela de editor Menu expandido	6 6,7	AEDITOR.COM

Tabela 4 - Relação entre nome de arquivo e tecla associada

Após estes passos devemos reiniciar a máquina CNC para que os arquivos de configuração sejam carregados e as telas gráficas estejam acessíveis. Observe que devemos utilizar como start key as teclas horizontais disponíveis para cada tela básica.

Exemplo 36 - Nomeando um arquivo de configuração

Caso o arquivo seja nomeado como MA_JOG.COM

Ao ligar a máquina se entrarmos na tela “Machine” pressionando a tecla “Machine”, e passarmos para o modo “JOG” pressionando a tecla “JOG”, a “start key” HS1 será a tecla que irá abrir o formulário de tela criado no arquivo de configuração.

APÊNDICE C – Linguagem G para comandos FANUC

A linguagem utilizada para a programação dos CNC's Fanuc é a uma linguagem G de fácil programação e pode ser divididos em Códigos G, Códigos M, Códigos T e ainda as programações de mais alto nível que são denominadas Custom Macro B.

Sistema de Coordenadas

Quando se trata de máquinas CNC pode-se utilizar sistemas de coordenadas diferentes, o que facilita muitos processos que requerem um centro de coordenadas diferente do padrão estabelecido na máquina no momento de sua fabricação.

Costuma-se dividir os sistemas de coordenadas de uma máquina CNC em dois tipos chamados "Zero Máquina" e "Zero Peça". Sendo o primeiro destes o sistema de coordenadas padrão e o segundo um sistema de coordenadas que pode ser definido pelo usuário segundo sua necessidade.

Os comandos Fanuc dispõem de vários sistemas de coordenadas personalizáveis dependendo da versão do software instalado e/ou dos adicionais habilitados. Para utilizar um "Zero Peça" deve-se sinalizar qual será utilizado, pois por definição o sistema de coordenadas utilizado é o "Zero Máquina". A tabela abaixo demonstra alguns exemplos de códigos G para sinalizar qual "Zero Peça" será empregado no posicionamento.

Código G	**Sistema de Coordenada**
G53	Zero Máquina (Padrão)
G54	Zero Peça 1
G55	Zero Peça 2
G56	Zero Peça 3
G57	Zero Peça 4
G58	Zero Peça 5
G59	Zero Peça 6

Tabela 5 - Códigos G e os "Zero Peça" correspondentes

Estes sistemas de coordenadas são configurados utilizando a interface de usuário presente no centro CNC, com exceção do "Zero Máquina" que é fixo, mas existe em comandos Fanuc uma terceira possibilidade que é utilizar o comando G50 para definir um "Zero Peça" através da posição atual do spindle.

Posicionamento Absoluto ou Incremental

Quando se deseja realizar um posicionamento existem duas possibilidades em máquinas com comando FANUC, os métodos incremental e absoluto. O posicionamento absoluto utiliza sempre como referencial a origem do sistema de coordenas definido no momento, ou seja, caso utiliza-se o comando X10 Y10 com posicionamento absoluto o spindle se deslocará para a posição X = 10, Y = 10 do sistema de coordenadas ativo.

Ao utilizar-se o posicionamento incremental a referencia passa a ser o ponto de partida, ou seja, utilizando o comando X10 Y10 com o posicionamento incremental o spindle se deslocará em 10 unidades nos eixos X e Y a partir da posição que se encontrava inicialmente.

Para definir qual sistema de posicionamento será utilizado podem-se empregar duas maneiras de sinalização como mostrado no quadro abaixo:

Método	**Sinalização A**	**Sinalização B**
Absoluto	X10 Y10	G90 X10 Y10
Incremental	U10 V10	G91 X10 Y10

Tabela 6 - Métodos de posicionamento

Deve-se destacar que caso as duas sinalizações sejam utilizadas em um bloco de programa em linguagem G a ultima especificada será considerada. Entretanto os modos, incremental e absoluto, podem ser programados simultaneamente em uma mesma linha de comando desde que se refiram a eixos diferentes.

Programação de Ponto Decimal

Existem dois modos de programação de ponto decimal o primeiro deles denotado tipo calculador que admite que o valor sempre seja entrado em milímetro, utilizando o ponto decimal ou não. No segundo, denominado modo padrão, quando se omite o ponto decimal é considerado que o valor foi entrado é a quantidade de incrementos, como no exemplo abaixo:

X1000:

No tipo calculadora corresponde a 1000 mm

No modo padrão corresponde a 1 mm (incremento de 0.001 mm), ou seja, 1000 incrementos de 0.001 mm

X1000.0

No tipo calculadora corresponde a 1000 mm

No modo padrão corresponde a 1000 mm (devido ao ponto decimal a unidade adotada é mm)

Controle do Spindle

Quanto ao controle do spindle deve-se destacar dois comandos que são empregados com mais freqüência. Caso deseja-se determinar a velocidade de rotação do Spindle utiliza-se o comando S seguido de cinco dígitos correspondentes à velocidade de rotação dada em rpm. Por exemplo, o comando S3500 inicia a rotação do spindle com 3500 rpm.

Outra função relacionada com o spindle é a de posicionamento angular, que permite que o eixo seja posicionado em um ângulo arbitrário ou semi-fixo. Para utilizar ângulos semi-fixos faz-se uso do comando M seguido de 2 dígitos numéricos que possibilita o uso desde Mα à M(α+5), onde α deve ser definido no parâmetro #4962 previamente. Os ângulos correspondentes à cada comando M são mostrados no quadro abaixo.

Código M	**Ângulo de Posicionamento**	**Ex para β = 30**

Mα	β	30
M(α+1)	2 β	60
M(α+2)	3 β	90
M(α+3)	4 β	120
M(α+4)	5 β	150
M(α+5)	6 β	180

Tabela 7 - Ângulos e comandos M correspondentes

O valor de β deve ser configurado no parâmetro #4963, a manipulação de parâmetros de sistema será discutida em tópicos a seguir.

Quando se deseja trabalhar com ângulos arbitrários utiliza-se os comando C e H seguidos do ângulo de posicionamento, em graus. O comando C, por exemplo C180, é utilizado para posicionar o spindle em um ângulo com referencia na origem. O comando H por sua vez rotaciona o spindle de forma a percorrer este ângulo, a partir do ponto que se encontra inicialmente. Como ilustrado abaixo:

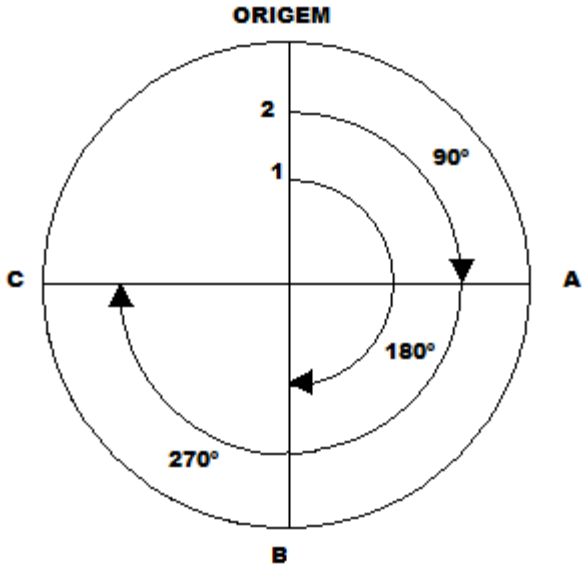

Figura 27 - Posicionamento angular absoluto e incremental do spindle

Na Figura 27 o trajeto 1 demonstra o resultado da aplicação da seqüência de comandos C90 C180, no modo absoluto ele dirigi-se para a posição absoluta de 90° (Ponto A) e no segundo comando para a posição absoluta de 180° (Ponto B). O trajeto 2 é desenvolvido ao se aplicar o código H90 H180 que caminha partindo da origem 90° e no segundo comando caminha partindo do Ponto A 180° atingindo assim a posição absoluta de 270° (Ponto C).

Comandos G

Pode-se classificar os comandos G em dois grupos básicos, os modais e os não modais. Os modais são aqueles que uma vez utilizados permanecem ativos até serem cancelados e os não modais são ativos apenas na linha em que são utilizados. Por exemplo, G01 e G00 são modais com abaixo.

G01 X_;
Z_;
X_;
G00Z_;

Os comandos G são utilizados para preparação da máquina CNC e para movimentação do spindle entre outras funções.

Comandos de posicionamento

Para posicionamento do spindle deve-se utilizar os comandos G00 e G01 que diferem apenas na velocidade de movimentação, ambas são funções globais. O comando G00 seguido das coordenadas absolutas ou incrementais realiza a movimentação em velocidade máxima. O comando G01 permite que a velocidade de avanço seja determinada pelo comando F seguido da velocidade de avanço desejada. Observe o exemplo abaixo.

```
G00 X30.0
G01 Y90.0 F2000
```

1.1.1.1 Comandos preparatórios

Os comandos preparatórios permitem que seja estabelecida uma linguagem de comunicação padrão entre o programador e a máquina CNC, por exemplo, quais unidades serão utilizadas para avanços, qual sistema de coordenadas utilizar, etc. Observa-se que estes comandos são comandos modais,ou seja, uma vez estabelecidos eles permanecem ativos até que um comandos altere-os. Abaixo são mostrados alguns comandos preparatórios mais utilizados, além dos comandos de “Zero Peça” mostrados anteriormente.

Comando G	Significado
G94	Unidade de avanço por minuto
G95	Unidade de avanço pó revolução
G36	Compensação automática de ferramenta em X
G37	Compensação automática de ferramenta em Z
G38	Compensação automática de ferramenta em Y

Tabela 8 - Comandos preparatórios básicos

Comandos T

O conjunto de comandos T é composto por funções que se destinam a manipulação das ferramentas disponíveis. O comando mais relevante deste conjunto é o de seleção de ferramenta que deve ser utilizado, por exemplo, como T1 que indica que q ferramenta 1 será selecionada. O número de ferramentas disponível e o número correspondente a cada ferramenta é especifico de cada máquina. Este comando é utilizado também com compensação de ferramenta. Deste modo utiliza-se o comando T seguido de quatro dígitos,

onde os dois primeiros indicam o número da ferramenta selecionada e os restantes indicam o corretor de geometria que será aplicado, o conceito de corretor será apresentado mais adiante.

Comandos M

As máquinas ferramenta CNC possuem um conjunto de funções auxiliares, conhecidas como comandos M, as quais são utilizadas para comandar alguns processos como ligar a rotação do spindle, fim de programa, entre outras. As principais são citadas abaixo.

Comando M	Breve Descrição
M00	Para o programa que estava sendo executado e reinicia ao pressionar cycle start
M02, M03	Indica o fim do programa
M98	Chamada de subprograma
M99	Fim de subprograma
M198	Chamada de subprograma em arquivo externo

Tabela 9 - Comandos M básicos

Observa-se que existem outros comandos M menos relevantes, para obter informações deve-se consultar o manual da máquina ferramenta

Programa em Linguagem G

A estrutura padrão de um programa em linguagem G é composta por um título, uma secção de comentários, uma seqüência de instruções e a sinalização de termino de programa. Pode-se ilustrar esta estrutura padrão como na Figura 28.

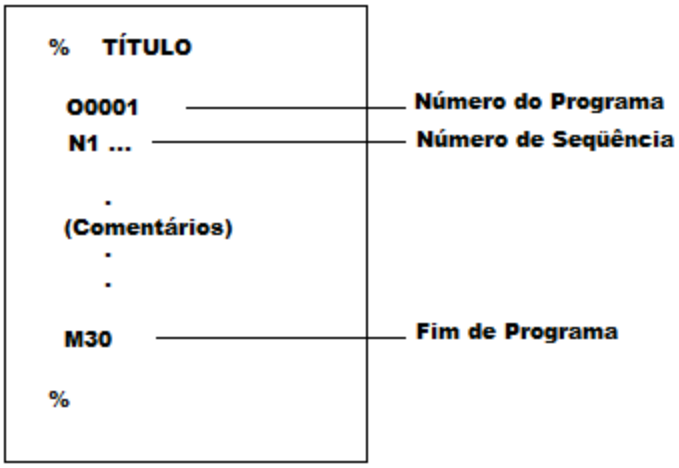

Figura 28 - Estrutura de programa padrão

Observa-se que toda informação contida no programa que estiver entre parêntesis é considerada comentário e assim não será compreendida como código e não será executada.

O número de programa é indicado pelo comando “O” seguido de quatro dígitos que indicam o número escolhido para identificar o programa. Caso seja especificado um número de programa com mais de quatro dígitos os quatro menos significativos serão considerados.

Observa-se que os números de programa de 8000 a 9999 são utilizados pelo fabricante da máquina ferramenta, portanto não se deve utilizá-los.

Cada instrução é denominada bloco e cada bloco é separado por um sinal de fim de bloco que é representado por ";". O número de seqüência é composto pelo comando N seguido por cinco dígitos que pode variar de 1 até 99999. Os números de seqüência podem ser especificados em ordem qualquer e qualquer numero pode ser omitido, assim como podem estar presentes em todo bloco ou apenas em alguns, mas em geral utiliza-se uma seqüência crescente e contínua para os números de seqüência. Por exemplo.

N299 T1;

N300 X200.0 Z300.0 ;

Cada bloco pode conter várias funções de comandos diferentes apresenta-se abaixo um quadro com algumas funções possíveis de serem utilizadas.

Função	Comando	Significado
Número de Programa	O	Número de Programa
Número de Seqüência	N	Número de Seqüência
Função Preparatória	G	Especifica o modo de movimentação
Palavra de Dimensão	X, Y, Z, U, V, W, A, B, C	Movimentação dos Eixos Coordenados
	I,J,K	Coordenada do Centro de Arco
	R	Raio do Arco
Função de Avanço	F	Taxa de Avanço por Minuto Taxa de Avanço por Revolução
Função de Velocidade do Spindle	S	Velocidade do Spindle
Função de Ferramenta	T	Seleciona a Ferramenta
Função Auxiliar	M	Controle da Máquina CNC
	B	Indexação de Tabela, etc
Designação de Número de Programa	P	Numero de Subprograma
Número de Repetições	P	Número de Repetições do Subprograma
Parâmetros	P, Q	Parâmetros de Ciclos

Tabela 10 - Parâmetros da estrutura padrão de um bloco

A estrutura padrão de um bloco pode ser apresentada como na Figura 29.

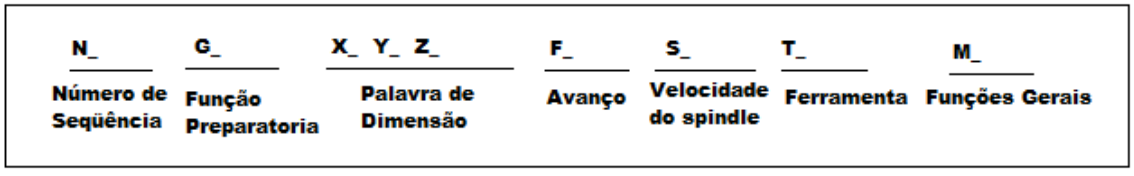

Figura 29 - Estrutura básica de um bloco em linguagem G

Por fim no final do programa sinaliza-se com os comandos M30 ou M02 para programas principais ou M99 para subprogramas.

Subprogramas

Se um programa tem uma sequência fixa ou é utilizado várias vezes torna-se interessante convertê-lo para um subprograma. Um subprograma pode ser chamado dentro de um programa principal ou dentro de outro subprograma. Para realizar a chamada destes subprogramas utiliza-se o código M98 como mostrado abaixo.

M98 P____ _____

Onde no primeiro campo deve-se informar o número de repetições que devem ser executadas do subprograma e no segundo campo o número do subprograma que se deseja executar. Observa-se que se o primeiro campo por omitido o subprograma será executado apenas uma vez.

Ao se realizar uma chamada de subprograma pode-se utilizar além de o número de programa um número de sequência presente no programa atual. Deve-se atentar que o limite de encadeamento de subprogramas é de no máximo quatro chamadas aninhadas como mostrado na Figura 30.

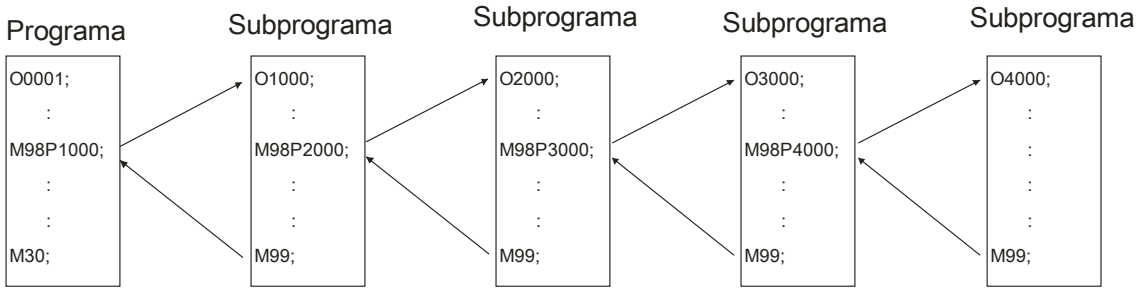

Figura 30 - Encadeamento de Subprogramas

Caso o número do subprograma chamado não seja encontrado um alarme será mostrado no visor da máquina CNC.

Função de Compensação de Ferramenta

Esta função permite informar para a máquina CNC a geometria da ferramenta para que desta maneira o centro CNC possa compensar as dimensões da ferramenta no momento de posicionamento. O sistema de compensação é bastante completo e permite a compensação de vários parâmetros inclusive o desgaste de ferramenta. A seleção de um corretor é realizada quando se seleciona uma ferramenta como no exemplo abaixo.

T0101

No exemplo acima a ferramenta 01 é selecionada e o corretor de geometria 01 para ser aplicado durante o posicionamento com a ferramenta 01. Para cancelar o corretor para uma ferramenta basta carregar o corretor 00 como no exemplo abaixo.

Para configurar os corretores e seus parâmetros é utilizada a interface com o usuário presente na máquina CNC.

Custom Macro

Assim como os subprogramas as custom macros são versáteis e podem ser utilizadas várias vezes, mas as custom macros têm a capacidade de utilizar variáveis.

Variáveis de uso geral

Quando se realiza a programação com variáveis pode-se desenvolver um programa mais flexível e estruturado. As variáveis podem ser entendidas como envelopes que contém valores e podem ser utilizados em vários trechos do programa.

Em custom macros representam suas variáveis utilizando o símbolo "**#**" seguido pelo número de variável, por exemplo, #1. Os valores das variáveis que podem ser utilizados são demonstrados abaixo.

Número de Variável	Tipo de Variável	Função
#0	Sempre nulo	Esta variável é sempre nula e nenhum valor pode ser estabelecido para ele.
#1 - #33	Variáveis locais	Variáveis locais só podem ser acessadas dentro da macro em que são estabelecidas. Quando a máquina CNC for desligada os valores são perdidos. Quando uma macro recebe parâmetros eles são armazenados em variáveis locais
#100 - #149 #500 - #531	Variáveis globais	São variáveis que podem ser compartilhadas entre as macros. A faixa de #100 - #149 são setadas para nulo quando a máquina é desligada. A faixa de #500 - #531 mantêm os valores mesmo com o desligamento da máquina.
#1000 -	Variáveis de sistema	São utilizadas para ler e escrever valores correspondentes aos parâmetros da máquina como por exemplo a compensação de ferramenta.

Tabela 11 - Faixa de Variáveis

As variáveis podem assumir os seguintes valores:

-10^{47} à -10^{-29}

0

10^{-29} à 10^{47}

Observa-se que as variáveis não podem ser utilizadas em números de programa, em números de sequência.

Variáveis de Sistema

As variáveis de sistema são utilizadas para ler e escrever dados internos do CNC assim como a posição atual do spindle entre outros parâmetros.

A faixa de variáveis que são classificadas como variáveis de sistema é descrita na Tabela 11 e dentro desta faixa existem outras subdivisões que classificam as variáveis de sistema em grupos menores.

Quando se deseja exibir um alarme, por exemplo, deve-se utilizar a variável #3000 da maneira mostrada abaixo.

```
#3000 = 1(Ferramenta não encontrada!);
```

Onde a tela do CNC apresenta a mensagem “3001 Ferramenta não encontrada!” para conhecer outras variáveis de sistemas deve-se consultar o manual da máquina CNC pois as faixas são específicas de cada comando.

Operações Aritméticas e Lógicas

A programação utilizando custom macros pode realizar as operações aritméticas básicas como soma, subtração, divisão e multiplicação. Além das operações aritméticas pode-se realizar operações lógicas básicas como a função E, função OU, etc. O sintaxe das principais operações é apresentado abaixo.

Teste condicional IF

Quando se utiliza um teste condicional IF pode-se permitir ao custom macro tomar algumas decisões baseadas em um segmento condicional. A sintaxe utilizada é como mostrado abaixo.

```
IF[<expressão condicional>]THEN
```

A expressão condicional deve estar contida entre colchetes ([,]) e tem que ser a comparação entre uma constante e uma variável ou entre duas variáveis. Cada operador utilizado nas comparações é composto por duas letras e as possibilidades são apresentadas abaixo.

Operador	Significado
EQ	Igual à
NE	Não igual à
GT	Maior que
GE	Maior ou igual à
LT	Menor que
LE	Menor ou igual que

Tabela 12 - Operadores de expressões lógicas

Abaixo se apresenta um programa exemplo utilizando custom macro e uma estrutura condicional IF

```
O9500;
#2 = 0;
```

```
N1 IF[#2 GT 10] GOTO 2;
#2 = #2+1;
GOTO 1;
N2 M30;
```

Este programa inicia com o valor 0 na variável #2 e incrementa de um em um esta variável até que a estrutura condicional seja satisfeita e a função “GOTO 2” seja executada. Observa-se que a função GOTO recebe um numero como parâmetro que representa o número de seqüência para o qual a função GOTO deve saltar.

Laço condicional WHILE

Quando se realiza um programa em custom macro pode ser necessária a utilização de um laço de repetição, para realizar um trecho do custom macro várias vezes ou até que uma condição não seja mais satisfeita. A sintaxe padrão é como mostrada abaixo.

```
WHILE [<expressão condicional>] DO m;
END m;
```

Onde os trecho entre DO m e END m é repetido até que a condição estabelecida pela expressão condicional não seja mais satisfeita. O parâmetro m é utilizado para identificar os pares DO e END correspondentes (mesmos parâmetros m). Observa-se que se a expressão condicional for omitida este comando será compreendido como um laço infinito. Abaixo apresentamos o mesmo exemplo demonstrado para o teste condicional IF, mas construído utilizando laço condicional WHILE

```
O9501;
#2 = 0;
N1 WHILE[#2 LE 10] DO 1;
#2 = #2+1;
END 1;
N2 M30
```

Nota-se que utilizando o laço condicional a função GOTO não se faz necessária e o custom macro fica mais estruturada.

Chamada de Macro

Uma chamada de macro pode ser realizada utilizando os seguintes métodos.

1. Chamada simples;
2. Chamada modal;
3. Chamada com código G;
4. Chamada com código M;

A chamada de subprograma (M98) e a chamada de macro (G65) diferem principalmente pelo fato de que quando se realiza chamada de uma macro pode-se passar parâmetros para ela, o que não é possível com subprogramas.

Chamada simples

Ao se realizar uma chamada simples deve-se utilizar a seguinte estrutura básica.

G65 P_ L_ <especificação de argumentos>

Onde G65 é o comando utilizado para chamar uma custom macro, P seguido de quatro dígitos informa o número de programa da macro que será chamada, L seguido de um número entre 1 e 9999 indica a quantidade de repetições que serão realizadas. No campo de especificação de argumentos pode-se determinar valores que serão utilizados dentro da macro. Para realizar esta especificação utiliza-se a seguinte relação.

Endereço de chamada	Variável dentro da macro	Endereço de chamada	Variável dentro da macro	Endereço de chamada	Variável dentro da macro
A	#1	I	#4	T	#20
B	#2	J	#5	U	#21
C	#3	K	#6	V	#22
D	#7	M	#13	W	#23
E	#8	Q	#17	X	#24
F	#9	R	#18	Y	#25
H	#11	S	#19	Z	#26

Tabela 13 - Parâmetros e variáveis correspondentes em chamada de macros

A utilização da especificação de parâmetros pode ser entendida a partir do exemplo demonstrado abaixo.

```
O0001;
G65 P9010 L1 A1.0 B2.0 Y4.5;
M30;
```

Neste exemplo quando a macro 9010 for chamada as variáveis #1, #2 e #25 receberão os valores 1, 2 e 4.5 respectivamente. Observa-se que a especificação de parâmetros não precisa ser realizada em ordem com exceção das variáveis I, J e K, ou seja, estas três podem surgir em qualquer posição da seqüência de especificação, mas apenas mantendo ordem alfabética entre elas.

Chamada modal

Quando se realiza uma chamada modal de macro, utilizando o comando G66, esta macro passa a ser executada após cada bloco de movimentação de eixo e permanece assim até ser cancelada pelo comando G67. A sua estrutura é igual à da chamada simples de macro com exceção que se deve iniciar chamada modal com G66 e cancelar a chamada modal com o comando G67.

Chamada de macro usando código G

Este método de chamada permite que um comando G seja reservado para chamada de uma macro. Para realizar este procedimento deve-se armazenar um número de dois dígitos em uma variável de sistema, o comando G acompanhado deste número executará a chamada da macro com número de programa correspondente conforme a tabela mostrada abaixo.

Número de programa	Número da variável de sistema
O9010	6050
O9011	6051
O9012	6052
O9013	6053
O9014	6054
O9015	6055
O9016	6056
O9017	6057
O9018	6058
O9019	6059

Tabela 14 - Correspondência entre número de programa e macro com código G

Por exemplo, armazena-se 81 no parâmetro 6050 e em seguida utiliza-se o comando G81 em um programa, é realizada chamada da macro com número de programa 9010.

Chamada de macro utilizando código M

Este método do é similar ao anterior, mas utilizam-se códigos M para fazer a chamada e seus parâmetros devem ser configurados segundo a seguinte tabela.

Número de programa	Número da variável de sistema
O9020	6080
O9021	6081
O9022	6082
O9023	6083
O9024	6084
O9025	6085
O9026	6086
O9027	6087
O9028	6088
O9029	6089

Tabela 15 - Correspondência entre número de programa e macro com código M

O método de chamada de macro com código M é ilustrado no exemplo mostrado abaixo.

Admitindo a variável 6080 = 50

```
O0001;
M50 A1.0 B2.0;
M30;
```

O comando M50 irá chamar a macro seguinte:

```
O9020;
    :
    :
M99;
```

“High speed skip” (G31)

A linguagem G presente em comandos FANUC possui um comando opcional que permite que seja comandada uma movimentação condicional (utilização similar ao G01 visto anteriormente, mas utilizando G31), esta será executada até o final ou até que um sinal externo seja enviado ao CNC. Caso o sinal externo chegue ao CNC antes do termino do trajeto programado pela função G31 o trajeto é imediatamente interrompido e as coordenadas em que ocorreu esta interrupção ficam armazenadas no sistema de variáveis de macros como mostrado abaixo.

Variável	Conteúdo após a interrupção
#5061	Coordenada do eixo X
#5062	Coordenada do eixo Z
#5063	Coordenada do eixo Y

Tabela 16 - Variáveis da função "Skip"