Pontifícia Universidade Católica do Rio Grande do Sul

Faculdade de Engenharia

Curso de Engenharia Mecânica
&
Curso de Engenharia de Controle e Automação

LABORATÓRIO CIM

WWW.EM.PUCRS.BR/~LABCIM

DISCIPLINAS

PROGRAMAÇÃO DE ROBÔS
SISTEMAS ROBOTIZADOS

PROFESSORES

PROF. JÚLIO CÉSAR MARQUES DE LIMA
PROF. ANDRÉ LUIZ TIETBÖHL RAMOS

TÉCNICOS
ENG. TIAGO LEONARDO BROILO
TÉC. LIANGRID LUTIANI DA SILVA
TÉC. RODRIGO KRUG

REGULAMENTO DO USO DO LABORATÓRIO DE ROBÓTICA

Art. 1º O Laboratório CIM da Universidade Católica do Rio Grande do Sul – PUCRS destina-se aos alunos e professores com prioridade de utilização aos alunos.

Art. 2º As atividades desempenhadas no Laboratório devem ser restritas ao ambiente acadêmico, orientadas às disciplinas dos respectivos cursos.

Art. 3º O Laboratório poderá ser utilizado de forma individual, para pesquisa e elaboração de trabalhos, ou de forma coletiva, para aulas regulares.

Art. 4º Ao início de cada ano letivo será elaborado um planejamento para o uso coletivo do Laboratório para cada disciplina, com salas e horários estipulados.

Parágrafo único. Uma vez definida a programação, não é permitida a mudança ou troca de qualquer horário.

Art. 5º Em aulas coletivas, é de responsabilidade do professor da disciplina orientar os trabalhos e zelar pela ordem e utilização dos equipamentos.

Art. 6º O professor responsável deve solicitar os materiais necessários à condução de seus trabalhos ao responsável pelo Laboratório CIM, com antecedência.

Art. 7º As aulas coletivas a serem ministradas no Laboratório são preparadas com antecedência pelo professor, com a preocupação de verificar a compatibilidade dos equipamentos às necessidades previstas.

Art. 8º Cabe ao professor responsável orientar a preparação e a utilização dos programas e equipamentos. A requisição de programas deve ser feita com antecedência ao Coordenador do Curso.

Art. 9º Ao término dos trabalhos, o professor responsável deve solicitar aos alunos que recoloquem as cadeiras em seus devidos lugares, desliguem os equipamentos corretamente, retornando-os à posição de origem, e que mantenham limpo o ambiente.

Art. 10. A utilização de forma individual do Laboratório é permitida fora dos horários de aulas regulares, com a autorização do responsável pelo Laboratório CIM.

Parágrafo único. Para fazer uso dos equipamentos do Laboratório, o aluno deverá identificar-se ao responsável pelo Laboratório com o respectivo crachá.

Art. 11. Para a utilização dos equipamentos, os alunos deverão observar os procedimentos e recomendações afixados no Laboratório para a utilização e o manuseio dos equipamentos.

Art. 12. Para a preservação do meio ambiente acadêmico necessário às atividades do Laboratório é importante:

I - não fumar;

II - manter silêncio;

III - preservar a limpeza do ambiente;

IV - não escrever nas mesas;

V - não colocar os dedos ou as mãos sobre a tela nem objetos sobre o monitor;

VI - não comer ou beber no recinto;

VII - entrar e sair do Laboratório de forma tranqüila, sem arrastar os móveis;

VIII - utilizar as instalações e os equipamentos do Laboratório da forma recomendada pelos procedimentos da sala (em caso de dúvida, informar-se com os responsáveis);

IX - não levar equipamentos pessoais ou de terceiros ao Laboratório;

X - identificar-se sempre que solicitado

XI - observar o horário de funcionamento fixado.

Art. 13. Ao fazer uso dos equipamentos o aluno deve:

I - verificar se a máquina apresenta as condições necessárias para uso;

II - reportar qualquer problema ao responsável, caso constate alguma irregularidade;

III - no caso de não observância dos incisos anteriores, a responsabilidade pela utilização passa a ser do próprio aluno.

Art. 14. Sempre haverá no recinto um técnico responsável para auxiliar em qualquer dúvida, ou problema, que venha a ocorrer. O responsável sempre levará em conta o artigo anterior.

Art. 15. Ao fazer uso da máquina o aluno não deve:

I - utilizar o equipamento com o intuito de alterá-lo, mudá-lo de posição, retirar ou conectá-lo a qualquer outro equipamento

II - causar danos nos equipamentos.

Art. 16. O uso de equipamentos, acessórios, *softwares* entre outros deve ser objeto de requisição pelo professor da disciplina ao responsável pelo Laboratório CIM.

Art. 17. Fica expressamente proibida a instalação de *softwares* e o acesso a salas de chat, sites pornográficos ou jogos.

Art. 18. Por questões legais referentes a Direitos Autorais, não é permitida a gravação, reprodução ou a utilização de quaisquer programas sem a autorização ou permissão por escrito do responsável pelo Laboratório.

Parágrafo único. As impressoras devem ser usadas de forma ordenada entre os alunos que se encontram no Laboratório. Não será permitida a impressão de trabalhos extensos como monografias, teses, etc durante os períodos de aula.

Art. 19. O descumprimento de qualquer artigo deste regulamento será considerado falta grave, sujeita a responsabilidade administrativa, se o caso assim o requerer.

Art. 20. É de competência do responsável pelo Laboratório CIM estabelecer as demais normas e procedimentos para o bom andamento dos trabalhos no Laboratório e se manifestar nos casos omissos do presente Regulamento.

Art. 21. É expressamente proibido o uso do Laboratório por pessoas estranhas ao meio acadêmico da PUCRS sem prévia autorização.

PROCEDIMENTOS DO USO DO LABORATÓRIO DE ROBÓTICA

1. Não levar alimentos e bebidas ao laboratório

2. Não fumar nos laboratórios e nos corredores próximos.

3. Não apoiar os pés nos assentos.

4. Não jogar lixo no chão.

5. No acesso a sites ou aplicativos que contenham som, utilizar fones de ouvido evitando desordem e poluição sonora.

6. Desligar aparelhos celulares, ou deixá-los em modo silencioso. Somente atender ligações fora do laboratório. (no período de aula o uso será proibido)

7. Utilizar com cautela os equipamentos do laboratório.

8. Recolher material de estudo, não deixá-los espalhado sob e/ou sobre a bancada dos computadores.

9. Respeitar os horários de utilização extra-aula.

10. A utilização do laboratório em horários extra-aula fica restrita ao aluno, não sendo permitido a entrada de acompanhantes que não sejam alunos da Instituição.

11. Não utilizar a Internet para acessar sites pornográficos ou com conteúdos violentos, racistas, etc. A monitoria reserva-se o direito de bloquear o acesso a sites que sejam inadequados ao uso acadêmico sem aviso prévio.

12. Não será permitido trazer e instalar programas de casa, ou obtidos através de download. Caso sejam necessários para apresentação de trabalhos, deve-se pedir autorização ao monitor responsável.

13. Para utilização dos equipamentos, serão OBRIGATÓRIOS o uso do login e senha pessoal.

14. As informações de Login e Senha são pessoais e intransferíveis, sendo assim, é de inteira responsabilidade do aluno zelar pela segurança destes dados. Estas informações também serão necessárias no acesso aos serviços on-line encontrados no site.

15. Em caso de perda ou esquecimento do login e senha de acesso, o aluno deverá solicitar um novo login ao departamento responsável.

16. Não serão informados por telefone ou a terceiros dados de login e senha.

17. O Login do aluno possui uma pasta na rede para o armazenamento dos seus trabalhos e atividades acadêmicas. Ao final do ano letivo e/ou em caso de trancamento ou cancelamento da matrícula, o login e o conteúdo da pasta serão apagados. Caso o aluno deseje realizar uma cópia da sua pasta, o mesmo deverá realizá-la até o ultimo dia letivo constante no Calendário Oficial da Instituição.

18. Todo sistema robotizado é completo por hardware e software, para um perfeito funcionamento os dois precisam estar funcionando perfeitamente. Se isso não acontecer, não cabe ao aluno a solução do problema. O mesmo deve se reportar de imediato ao técnico.

19. O Laboratório CIM é composto por vários tipos de robôs, cada robô com a característica especifica daquela estação de trabalho. Logo cada robô tem sua particularidade e modo de operação.

20. Este laboratório é utilizado por vários usuários. Por este motivo será totalmente padronizado o nome dos vetores de posição e nomes de programas. Sendo que ao final de cada aula os alunos deverão utilizar o comando REMOVE + "nome do programa" para remover o programa. Nomes de vetores de posição devem que estar no padrão, caso contrário serão apagados da memória.

21. Após o término das aulas os grupos deverão organizar a sua estação de trabalho, isto implica em desligar corretamente o robô, colocar as cadeiras nos seus devidos lugares e deixar a mesa limpa. O não cumprimento deste procedimento será reportado ao professor responsável da turma e caberá ao mesmo solicitar aos alunos o cumprimento deste procedimento, ou, uma avaliação do professor.

22. Não será permitido o uso indevido do robô. Sejamos cautelosos e cuidadosos no manuseio do robô. O uso não é individual, o uso indevido pode causar problemas nos trabalhos dos outros grupos.

23. O robô, como qualquer outra máquina, não foi implementada para trabalhar sobre esforços indevidos. É importante gravar as posições de "*get*" e "*put*" sem esforços inadequados. Pois, desta forma você estará evitando impactos e perda de posições.

24. Um esboço da movimentação do robô e a programação em outros computadores podem aumentar o rendimento do grupo, evitando assim atividades extraclasse. Procure sempre que possível antecipar as experiências, para evitar transtornos e superlotação do laboratório no final semestre letivo.

O descumprimento deste Regulamento pelo corpo discente, implica na aplicação das sanções previstas no Regimento Geral da Faculdade, capítulo dos direitos e deveres dos alunos. Eventuais danos causados aos equipamentos implicam o ressarcimento à Faculdade das despesas decorrentes dos reparos necessários.

PROCEDIMENTOS PARA USO EXTRA-CLASSE

1. Os horários livres para utilização dos alunos durante o período letivo serão divulgados no início de cada semestre e divulgados no mural do laboratório. Os horários e laboratórios reservados às aulas não poderão ser utilizados para realização de trabalhos sob hipótese alguma.
2. No período matutino recomenda-se ao aluno verificar o horário de funcionamento do laboratório e a reserva da estação para a realização do trabalho.
3. No período vespertino fica disponível aos alunos, no horário das 13h30 às 17h15.
4. No período noturno o laboratório fica disponível aos alunos nos horários que não haverá aula.
5. Aos sábados, o laboratório estará disponível para utilização dos alunos no horário existente no mural do laboratório. As datas de feriado, manutenção e/ou qualquer outra eventualidade que causar a necessidade de fechamento do laboratório serão divulgadas antecipadamente.
6. Os horários livres para utilização dos laboratórios poderão sofrer mudanças de acordo com o horário de aulas do semestre, sendo assim, eventuais indisponibilidades serão comunicadas com antecedência.

II. Agendamento do Laboratório

1. O Agendamento deve ser feito no dia previsto para uso e somente serão liberados se não houver reserva por professores. O agendamento vai ser realizado através de uma lista que estará no mural do laboratório
2. Responsáveis pelo agendamento:
Manhã: Estagiários que estarão no laboratório.
Tarde: Tiago Leonardo Broilo, Liangrid Lutiani da Silva e Rodrigo Krug
Noite: Tiago Leonardo Broilo, Liangrid Lutiani da Silva e Rodrigo Krug
Sábado: Técnico que estará no laboratório.
3. O local de Agendamento será o Laboratório CIM, sala 104, bloco F, prédio 30, Ramal 4393.

Capítulo 1
USANDO O TEACH PENDANT (TP)

Objetivos:

Conhecer os equipamentos do Laboratório
Ligar os equipamentos do Laboratório
Uso do Teach Pendant (TP)
Referenciar os robôs

1. INTRODUÇÃO

Neste capítulo, você aprenderá a usar o Teach Pendant (TP) para movimentar o robô. No nosso Laboratório existem dois tipos de TP's:

- **TP para o controlador *A***
- **TP para os controladores *B* e *AC***

O TP para o controlador *A* é usado com os robôs ER-V e ER-VII.

O TP para os controladores *B* e *AC* é usado com os robôs ER-9 (controlador B), Scara ER-14 (controlador B), Cartesiano do ASRS (controlador B) e Performer MK-3 (controlador *AC*).

Os robôs estão dispostos no laboratório da seguinte maneira:

ESTAÇÃO	ROBÔ
Estação 1	Robô ASRS
Estação 2	Robô ER-9
Estação 3	Robô ER-7
Estação 4	Robô MK-3
Estação 5	Robô ER-14

2. PRIMEIRO CONTATO

Estudo do Teach Pedant (TP) para o controlador *A* e, após, o TP para os controladores B e *AC*.

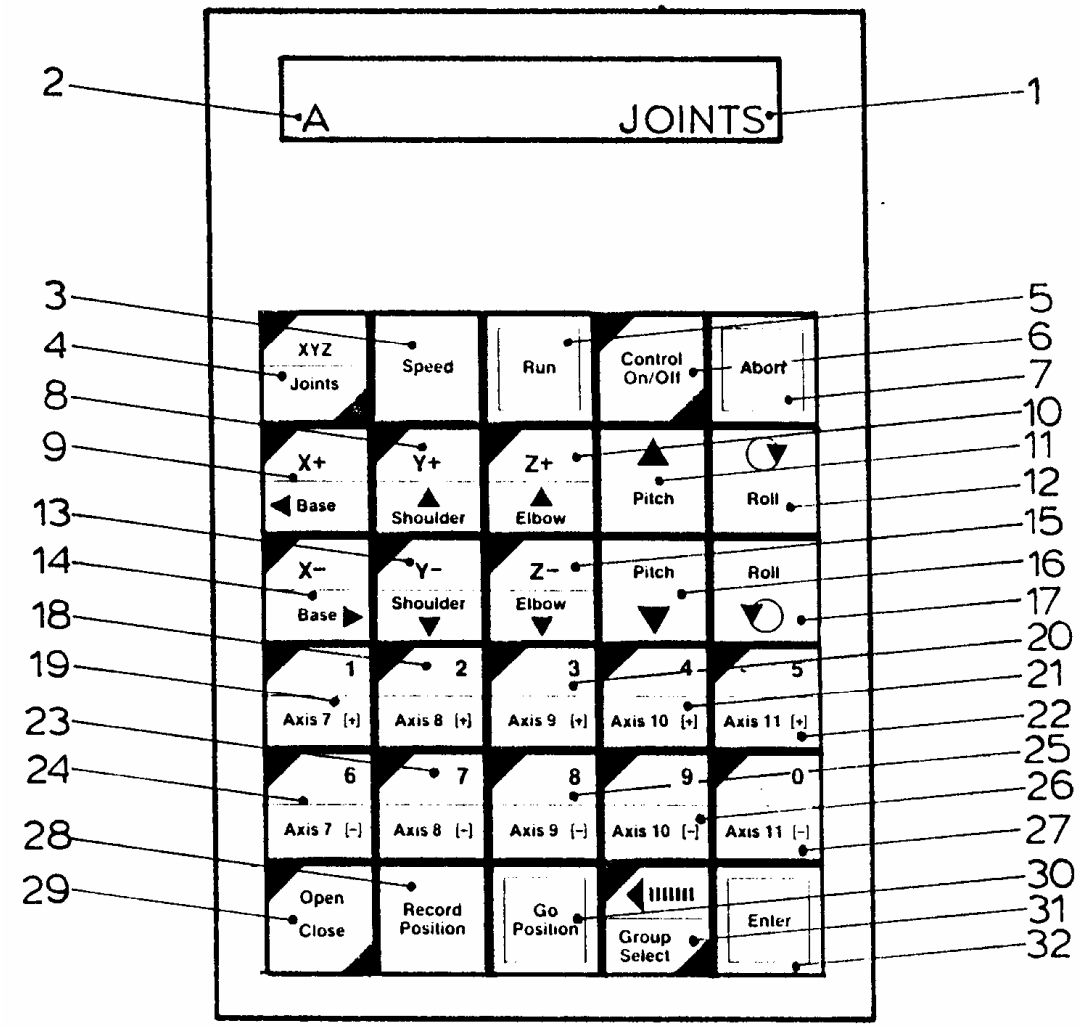
2.1. O TEACH PENDANT PARA O CONTROLADOR A

Antes de começar a aprender a usar o TP, você deve primeiramente ligar o controlador do robô. Por motivo de segurança, verifique se o robô está disposto *paralelamente* à máquina que ele serve. Após ligar o controlador, aparecerá no TP a mensagem.

A JOINTS

onde A significa robô e JOINTS significa o sistema de coordenadas que está sendo usado (são os *default* do sistema).

O TP portátil fornece controle direto dos robôs ER-V e ER-VII. As especificações deste TP são as seguintes:



Características do TP:

- Tela de Status:	Sistema de Coordenadas, Grupo de controle ativo e mensagens. LCD, 2 linhas, 32 caracteres.
- Teclado:	30 teclas multi-functionais, codificadas por cores.
- Sistemas de Coordenadas (número 1 da figura):	XYZ e JOINTS.
- Número de eixos controlados	11
- Movimentos retilíneos	X+, X-; Y+, Y-; Z+, Z-
- Controle de Pitch:	Dois modos: 1. O Órgão Terminal (garra) move-se para cima ou para baixo enquanto os demais eixos permanecem estacionários; 2. O Órgão Terminal mantém sua posição e troca de orientação enquanto os demais eixos se movimentam.
- Definição de velocidade (número 3 da figura)::	Valores percentuais variando de 0 a 100%.
- Gravação de posições (número 28 da figura):	Para cada grupo de controle, separadamente.
Movimentos tipo "Go position" (número 30 da figura):	Para posições previamente gravadas, ou em movimento ponto a ponto, ou ao longo de uma trajetória linear.
- Execução de programas (número 5 da figura):	Executa programas do usuário e as rotinas HOME e TEST.
- Abort (número 7 da figura)::	Parada de emergência: aborta programas do usuário e pára imediatamente qualquer movimento.
- ON/OFF (número 6 da figura)::	Habilita/desabilita o controle dos eixos.

O TP está dividido em duas áreas, a **tela** e o **teclado**:

A **tela** mostra o status corrente do controlador, o comando corrente do usuário e mensagens do sistema. Existem duas notas residentes na figura acima.

O **teclado** contém 30 teclas, a maioria das quais são multi-funcionais. Algumas teclas incluem o comando de acionamento do eixo e também função. A função numérica estará ativa somente se você teclou anteriormente uma das funções SPEED, RUN, RECORD POSITION, GO POSITION ou GROUP SELECT. Caso contrário, o comando de acionamento do eixo é que estará ativo.

Nº	COMANDO	DESCRIÇÃO
1	JOINTS/XYZ	Mostra o sistema de coordenadas ativo.
2	A/B/AXIS	Mostra o grupo ativo (ver tecla de seleção,31): A, B ou um dos eixos independentes não pertencentes aos grupos A ou B.
3	SPEED + número + ENTER	Define a velocidade do grupo corrente como um percentual (0 a 100%) da velocidade máxima do Órgão Terminal.
4	JOINTS/XYZ	Seleciona um dos sistemas de coordenadas: JOINTS ou XYZ.
5	RUN + número + ENTER	Executa um programa existente válido. A cada programa é automaticamente atribuído um número pelo controlador. O comando DIR da linguagem ACL (Advanced Control Language), que será estudada mais adiante, lista os programas e respectivos números de identidade.
6	CONTROL ON/OFF	Ativa/desativa o sistema de controle, mostrando as seguintes mensagens: CONTROL ENABLED CONTROL DISABLED
7	ABORT	Parada de emergência. Aborta imediatamente todos os programas em execução e pára todos os eixos em movimento.
9 14	BASE ou X+	Em JOINTS, as teclas movem o eixo da base no sentido horário (SH) ou anti-horário (SAH). Em XYZ, as teclas movem o Órgão Terminal ao longo de X+ e X- (Y e Z não mudam).
8 13	SHOULDER ou Y	Em JOINTS, as teclas movem o eixo do ombro no sentido horário (SH) ou anti-horário (SAH). Em XYZ, as teclas movem o Órgão Terminal ao longo de Y+ e Y- (X e Z não mudam).
10 14	ELBOW ou Z	Em JOINTS, as teclas movem o eixo do cotovelo no sentido horário (SH) ou anti- horário (SAH). Em XYZ, as teclas movem o Órgão Terminal ao longo de Z+ e Z- (X e Y não mudam).
11 16	PITCH	Em JOINTS, as teclas movem o pitch para cima ou para baixo. Os outros eixos não movem. Em XYZ, as teclas movem 3 eixos (ombro, cotovelo e pitch) para manter o Órgão Terminal enquanto mudam os ângulos de pitch.
12 17	ROLL	Movimenta o eixo de roll SH ou SAH.
18 a 27	EIXOS 7-11 ou teclas numéricas	Cada tecla move, respectivamente, os eixos 27 Teclas Numéricas 7 a 11 SH ou SAH. Se uma das funções SPEED, RUN, RECORD POSITION, GO POSITION ou GROUP SELECT tiver sido ativada anteriormente, a interpretação numérica é que terá efeito.
28	OPEN/CLOSE	Abre/fecha a garra.
29	RECORD POSITION + ENTER	Grava a localização corrente do grupo ativo número como uma posição específica numerada.
30	GO POSITION + número + ENTER	Manda o robô ou o dispositivo periférico para uma posição específica numerada. O movimento do robô (grupo A) será ou ponto a ponto (em JOINTS) ou ao longo de uma reta (em XYZ). GO POSITION 0 mandará todos os eixos do grupo A para as suas posições de HOME. GO POSITION 00 mandará todos os eixos do grupo B para as suas posições de HOME.
31	GROUP SELECT	Seleciona o grupo ativo em seqüência, isto é: A → B → AXIS → A → B → AXIS → A → ... Ao selecionar AXIS, você deve também teclar o número do eixo e após dar ENTER. As funções RECORD POSITION e SPEED afetam somente o grupo corrente selecionado. Quando ativada seguindo uma função numérica, essa tecla atua como função backspace, isto é, ela cancela a última entrada numérica e move o cursor uma casa para a esquerda.
32	ENTER	Confirma a última função ativada.

2.2. COMANDOS USANDO O TEACH PENDANT PARA O CONTROLADOR A

Vejamos, a seguir, alguns comandos que podem ser feitos usando o TP:

2.2.1. Programa Homes – referenciamento do robô.

Pressione "Run" + Digite "1": **Run 1 ↵**

onde o símbolo ↵ significa a tecla **ENTER**. O robô começa a se mover, procurando por uma posição de referência denominada HOME. Qualquer programa do usuário que esteja sendo executado será abortado. É mostrada uma mensagem na tela do TP:

HOMING

Concluída a tarefa com sucesso, o robô se encontrará numa posição aproximadamente vertical e a seguinte mensagem é mostrada:

HOMING COMPLETE

Se o procedimento não for completado, uma mensagem de erro identificando a falha será mostrada. Por exemplo:

*****HOME FAILURE AXIS 2**

2.2.2. Control On/Off

Pressione "Control On/Off": **Control On/Off**

Quando for ativado Control On/Off, as mensagens CONTROL ENABLED/DISABLED, respectivamente, aparecem:

CONTROL ENABLED

OU

CONTROL DISABLED

Com o controlador em Control On, você pode operar o robô. Com o controlador em Control Off, você não pode operar o robô. Certos erros ativam automaticamente o estado Control Off. Quando isso acontecer, Control On deve ser ativado para que o movimento possa continuar.

2.2.3. Abort

Pressione "Abort" ou Digite no teclado do computador "Crtl"+"a" **Abort ou Crtl + a**

Esse comando pára a execução de todos os programas e de todos os movimentos do robô e demais eixos ligados ao ocontrolador.

Os procedimentos seguintes assumem que o HOMES tenha sido executado com sucesso e que JOINTS apareça na tela do TP.

Pressione por 3 segundos "Base": **Base**
Pressione por 3 segundos "Shouder": **Shouder**
Pressione por 3 segundos "Elbow": **Elbow**
Pressione por 3 segundos "Roll": **Roll**

Movem os eixos do robô em ambas as direções. Somente o eixo referido movimentar-se-á.

Pressione "Speed" + Digite "número"+Pressione "Enter":

Speed <número> ↵ (1 <= número <= 100)

Varia a velocidade (default = 50% da velocidade máxima).

Pressione "Go Position" + Digite "0" + Pressione "Enter":

Go Position 0 ↵

O robô volta à posição HOME.

Até agora, você operou o robô no sistema de coordenadas JOINTS. Agora, você vai usar o sistema de coordenadas XYZ.

Pressione "XYZ/Joints": **XYZ/Joints**

Note agora que aparece na tela do TP: **A** **XYZ**

Pressione por 3 segundos "+X": **X+**
Pressione por 3 segundos "-X": **X-**
Pressione por 3 segundos "+Y": **Y+**
Pressione por 3 segundos "-Y": **Y-**
Pressione por 3 segundos "+Z": **Z+**
Pressione por 3 segundos "-Z": **Z-**

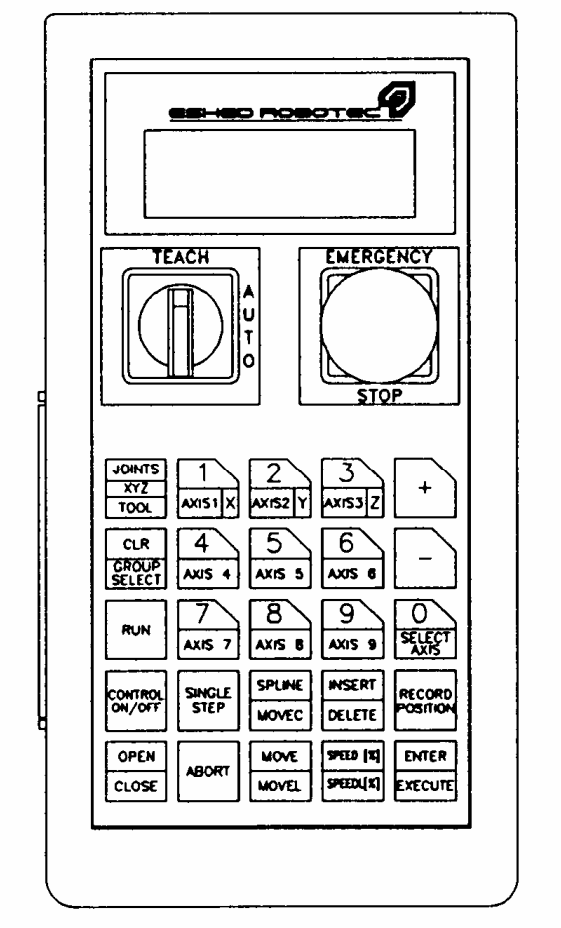
Os comandos acima movem o robô nas direções especificadas. Nesse caso, mais do que um eixo se movimentará, para que seja possível o Órgão Terminal seguir uma trajetória retilínea, segundo a direção especificada. Em XYZ, não movimente o robô para posições localizadas no alcance máximo do braço, a fim de evitar movimentos bruscos. Também as velocidades variarão com o movimento do braço.

Pressione "Pitch": **Pitch**

Move o eixo correspondente do robô em ambas as direções. Em JOINTS, somente o eixo de pitch se movimenta. Em XYZ, o centro do Órgão terminal permanece em sua posição, enquanto que os eixos do ombro (shoulder), cotovelo (elbow) e pitch se movimentam.

2.3. O TEACH PENDANT PARA OS CONTROLADORES B e AC

Os TPs para os controladores B e AC operam e controlam os eixos conectados a ambos os tipos de controladores. Ele está equipado com um botão de emergência EMERGENCY/STOP, uma chave seletora AUTO/TEACH e um botão lateral que é acionado pela mão do operador quando ele empunha o TP. O TP pode ser usado **empunhado** ou **montado sobre um dispositivo especial** localizado fora do volume de trabalho do robô, neste caso uma pequena caixa. A operação do TP varia de acordo com essas duas situações de uso.



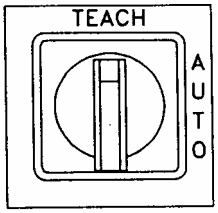
2.3.1. TP Montado ou TP empunhado

Quando o TP está montado sobre o dispositivo especial (verifique que o TP é colocado dentro de uma caixa), duas chaves magnéticas são ativadas por meio de tiras magnéticas existentes no dispositivo, permitindo que programas possam ser executados a partir do TP montado. Usar o TP

montado também oferece segurança ao operador, já que ele se encontra fora do volume de trabalho do robô quando o programa inicia.

Neste caso, o botão lateral deve permanecer acionado (apertado) durante todas as operações, ou, o TP deve estar dentro do seu dispositivo. Se o botão for solto, ou, se estiver fora do seu dispositivo, o TP se torna inoperante. Por motivos de segurança, não se pode executar programas com o TP empunhado.

2.3.1.1. Chave Seletora Auto/Teach TP Montado



2.3.1.1.1. Posição Teach

O TP tem controle total dos eixos.
Programas podem ser executados a partir do TP montado.
Comandos que causam movimento de eixos (tais como CON, MOVE, RUN) não podem entrar a partir do teclado; todos os demais comandos ACL (tais como COFF, DIR, STAT, EDIT, DIMP, ATTACH) permanecem disponíveis.

2.3.1.1.2. Posição Auto

O TP está desabilitado e o teclado tem controle total dos eixos.

2.3.1.1.3. Transição de Auto para Teach

Programas em execução continuam a ser executados;
O controle é transferido para o TP.

2.3.1.1.4. Transição de Teach para Auto

Programas em execução continuam a ser executados;
O controle é transferido para o teclado, porém somente após ter sido digitado o comando AUTO no teclado.

2.3.1.2. Chave Seletora Auto/Teach TP Empunhado

2.3.1.2.1. Posição Teach

O TP tem controle total dos eixos, desde que o botão lateral esteja comprimido;

Programas não podem ser executados a partir do TP empunhado;

Comandos que causam movimento de eixos (tais como CON, MOVE, RUN) não podem entrar a partir do teclado; todos os demais comandos ACL (tais como COFF, DIR, STAT, EDIT) permanecem disponíveis.

2.3.1.2.2. Posição Auto

O TP está desabilitado e o teclado tem controle total dos eixos.

2.3.1.2.3. Transição de Auto para Teach

Todos os programas que estiverem sendo executados são abortados.

2.3.1.2.4. Transição de Teach para Auto

O controle é transferido para o teclado, porém somente após ter sido digitado o comando AUTO no teclado.

2.3.1.3. Mudança de posição do TP

Se o TP for colocado ou removido do dispositivo especial com a chave seletora na posição Auto, a operação do sistema não é afetada;

Se o TP for removido do dispositivo especial com a chave seletora na posição Teach, os programas em execução são abortados; se o botão lateral for solto durante a remoção do TP do dispositivo especial, todos os eixos pararão e o TP tornar-se-á inoperante;

Se o TP for colocado no dispositivo especial com a chave seletora na posição Teach, a operação do sistema não é afetada; uma vez colocado o TP no dispositivo especial, programas podem ser executados a partir do TP.

2.3.1.4. Controle Manual do Teclado

Os comandos < Alt > + M ou ~ são usados para obter o controle manual a partir do teclado e estão disponíveis somente quando o TP está em modo Auto;

Se a chave seletora for movida da posição Auto para a posição Teach quando o Controle Manual do Teclado estiver ativo, a operação a partir do teclado é abortada; se a chave seletora for então movida de novo para a posição Auto e o comando AUTO digitado com a finalidade de retornar o controle ao teclado, este último é resetado para o modo DIRECT, mas não para o modo Manual.

2.3.2. Botão Lateral

O botão lateral é uma chave interruptora localizada na lateral esquerda do TP. Com o TP empunhado, esse botão deve permanecer sempre comprimido para que o controle dos eixos seja

possível a partir do TP. Se o botão lateral for solto com o TP em modo Teach, todos os eixos pararão e a maioria das teclas do TP tornar-se-ão inoperantes. As teclas numéricas, Record e Select Axis podem ser usadas para gravar posições e mudar o modo de movimento Direct Teach.

2.3.3. Botão de Emergência

É um botão vermelho, situado na parte superior direita do TP. Ele tem o mesmo efeito do botão de emergência existente no controlador. O botão de emergência pode ser acionado (pressionando-o) independentemente da localização do TP (montado ou empunhado) ou modo do TP (Teach ou Auto). Para retirar a situação de parada de emergência, o botão de emergência deve ser desacionado (puxando-o) e acionada a tecla CONTROL ON/OFF.

2.3.4. Funções do Teclado

BOTÃO	FUNÇÕES
ENTER/EXECUTE	• executa e/ou aceita o comando digitado anteriormente; • executa um programa
JOINTS **XYZ** **TOOL**	• seleciona o sistema de coordenadas, a seleção é realizada de maneira seqüencial, depende do sistema que estiver ativo: • os movimentos manuais dos eixos são executados coerentemente com o sistema ativo; • o comando RECORD POSITION grava posições executando os comandos ACL: HERE e HERER (modo JOINTS) HEREC e HERERC (modo XYZ) HEREC e HERERT (modo TOOL)
CLR **GROUP SELECT**	• anula um comando digitado parcialmente (**não** é o mesmo comando CLR da linguagem ACL) • habilita o controle, via TP, dos eixos periféricos (grupo B, grupo C), ou, uma garra servo elétrica (grupo G). O default do robô é o grupo A. A seleção é seqüencial.
+	• modo JOINTS: move o eixo selecionado na direção positiva. • modo XYZ e TOOL: move o TCP (Tool Center Point) na direção positiva. • os movimentos acima continuarão enquanto a tecla estiver sendo pressionada, ou, até que o limite do eixo seja atingido.
-	• modo JOINTS: move o eixo selecionado na direção negativa. • modo XYZ e TOOL: move o TCP (Tool Center Point) na direção negativa. • os movimentos acima continuarão enquanto a tecla estiver sendo pressionada, ou, até que o limite do eixo seja atingido.
0 **SELECT AXIS**	• tecla numérica 0. • muda o modo de movimento: pressionar para mudar para Free Axes, Fixed Pitch ou Fixed Pitch e modo Z.
1 **AXIS 1** **X**	• tecla numérica 1. • modo JOINTS: eixo 1 • modos XYZ e TOOL: eixo X
2 **AXIS 2** **Y**	• tecla numérica 2. • modo JOINTS: eixo 2 • modos XYZ e TOOL: eixo Y
3 **AXIS 3** **Z**	• tecla numérica 3. • modo JOINTS: eixo 3 • modos XYZ e TOOL: eixo Z
4 **AXIS 4**	• tecla numérica 4. • modo JOINTS: eixo 4 • modos XYZ e TOOL: eixo pitch
5 **AXIS 5**	• tecla numérica 5. • modo JOINTS: eixo 5 • modos XYZ e TOOL: eixo Roll
6 **AXIS 6**	• tecla numérica 6 • modo JOINTS : eixo 6 (somente se instalado e configurado).
7 **AXIS 7**	• tecla numérica 7 • modo JOINTS: eixo 7 (somente se instalado e configurado).

8 **AXIS 8**	• tecla numérica 8. • modo JOINTS: eixo 8 (somente se instalado e configurado).
9 **AXIS 9**	• tecla numérica 9 • modo JOINTS: eixo 9 (somente se instalado e configurado).
CONTROL ON/OFF	• habilita e desabilita o controle do grupo selecionado ou de todos os grupos. • se pressionada uma vez, alterna entre CON e COFF para o grupo selecionado. • se pressionada duas vezes, muda CON e COFF para todos os grupos; se ao menos um grupo está em modo CON, COFF é aplicado a todos os grupos; se todos os grupos estão em modo COFF, CON é aplicado a todos os grupos.
RECORD POSITION	• define e grava uma posição. • somente valores numéricos com até 5 dígitos podem entrar pelo TP. • a posição é definida para o grupo ativo corrente. • as coordenadas são gravadas no sistema de coordenadas corrente; dependendo do sistema de coordenadas corrente, são gravadas posições absolutas e relativas (sufixo R): • modo JOINTS: HERE e HERER • modo XYZ: HEREC e HERERC • modo TOOL: HEREC e HERERT • pressione 1 vez, digite o valor numérico da posição e “ENTER” para gravar uma posição absoluta. • pressione 2 vezes, digite o valor numérico da posição e “ENTER para gravar uma posição relativa. • se você mover os eixos e pressionar novamente ENTER, o TP repetirá o comando RECORD POSITION e automaticamente incrementará em 1 a última posição gravada. • se você gravar numa posição que já tenha sido definido anteriormente, as novas coordenadas sobrescreverão as existentes. • se você quiser gravar posições em um vetor (o que é muito usual), o vetor deve ser criado e atachado através dos comandos via terminal do computador: “DIMP”+”NOME VETOR[tamanho do vetor]”+”ENTER” “ATTACH”+”NOME VETOR” + “ENTER”
INSERT **DELETE**	• grava/remove posições de um vetor previamente atachado. • INSERT grava uma posição em um vetor e desloca em uma unidade para cima todas as posições de nome numérico mais alto pré-existentes. • DELETE remove uma posição de um vetor e desloca de uma unidade para baixo todas as posições de nome numérico mais alto pré-existentes. • INSERT e DELETE estão disponíveis apenas para vetores que tenham sido definidos com o prefixo &. • para inserir uma posição, pressione INSERT/DELETE uma vez, entre com o número (índice) da posição a ser inserida e dê ENTER. • para deletar uma posição, pressione INSERT/DELETE duas vezes, entre com o número (índice) da posição a ser deletada e dê ENTER; uma mensagem "ARE YOU SURE?" ("VOCÊ TEM CERTEZA?") aparecerá na tela do TP; pressione + para sim e dê ENTER.
SPEED(%) **SPEEDL(%)**	• define o percentual da velocidade máxima para a junta ativa • SPEED(%) é usado em conjunto com o modo JOINTS. • SPEEDL(%) é usado em conjunto com os modos XYZ e TOOL. • pressione SPEED(%)/SPEEDL(%); a velocidade corrente é mostrada; dê ENTER para aceitar ou use as teclas numéricas para modificar e depois dê ENTER.
OPEN **CLOSE**	• abre/fecha a garra; funciona para garras elétricas e pneumáticas.
MOVE **MOVEL**	• Move os eixos para uma posição de destino • MOVEL aplica-se apenas para os eixos do robô (grupo A). • MOVE está associado ao modo JOINTS. • MOVEL está associado aos modos XYZ e TOOL (nesses dois casos, o movimento é linear). • pressione MOVE/MOVEL; após, entre o número da posição de destino e pressione a tecla ENTER/EXECUTE (mantendo-a pressionada até que a posição de destino seja atingida); se a tecla ENTER/EXECUTE for solta, o movimento pára imediatamente e o comando é abortado.
SPLINE **MOVEC**	• movem os eixos para uma posição de destino • MOVEC aplica-se somente aos eixos do robô (grupo A) • SPLINE aplica-se somente aos dispositivos multi-eixos do grupo A ou do grupo B. • pressionando 1 vez, um movimento SPLINE será executado. • pressionando 2 vezes, um movimento MOVEC será executado. • digite o número da primeira posição e dê ENTER/EXECUTE; • digite o número da segunda posição e dê ENTER/EXECUTE; • pressione a tecla ENTER/EXECUTE (mantendo-a pressionada até que a posição de destino seja atingida); se a tecla ENTER/EXECUTE for solta, o movimento pára imediatamente e o comando é abortado.

RUN	• executa um programa; somente disponível com o TP montado. • pressione RUN, o número de identidade do programa e pressione ENTER/EXECUTE. • o comando DIR do ACL lista os programas e os respectivos números de identidade que foram atribuídos pelo controlador. • o comando RUN 1 executa o homing (= comando HOME do ACL). • o comando RUN 999 executa o teste do sistema (= comando TEST do ACL).
ABORT	• aborta todos os programas em execução. • pára o robô e todos os eixos periféricos.

Capítulo 2
USANDO A LINGUAGEM ACL PARA CONTROLAR O ROBÔ

Objetivos:

Conhecer o software envolvido
Referenciar os robôs
Editar programas
Comando PRINT e PRINTLN
Controle de loops com os comandos LABEL e GOTO
Comentários nos programas
Comandos ACL para gerenciamento de programas

1. INTRODUÇÃO

No capítulo anterior, você aprendeu a utilizar o TP, referenciar o robô (fazer o "homing" do robô), mover o robô para diversas posições, entre outras coisas que o TP permite. Neste capítulo, você aprenderá os comandos da linguagem ACL equivalentes para as tarefas de referenciamento do robô, a partir do teclado. Tais comandos, combinados com o uso de variáveis e de vetores de posição, que serão estudados mais tarde, permitirão que você execute muitas aplicações robóticas interessantes, usando programas relativamente curtos. Logo, neste capítulo você aprenderá a escrever um programa na linguagem ACL.

Os comando ACL podem ser divididos em dois tipos:

- **modo direto (direct mode)**
- **modo edição (edit mode)**

No modo direto, os comandos são executados imediatamente após a digitação do comando e do "ENTER" no hiperterminal do software. No modo edição, os comandos são escritos dentro de um aplicativo (editor de texto), uma vez editado ele é escrito na memória do controlador, é necessário fazer o "download" do programa, e após pode ser executado no hiperterminal através do comando RUN + "NOME DO PROGRAMA".

2. O PROGRAMA HOMES

Conforme já visto anteriormente, o robô deve ser referenciado cada vez que o controlador é ligado.

Lembrando, se o robô foi ligado e referenciado **não se pode referenciar o robô novamente**, se o controlador desligou os drivers do robô é necessário somente ligar os drivers pressionando o botão "CONTROL On/Off" do TP ,ou, digitando o comando "CON ALL" via terminal do computador. Se o problema persistir chame o professor, ou, o técnico.

Para referenciar o robô a linguagem ACL tem comandos específicos para isso. Mas, nós utilizaremos o programa "HOMES", pois este irá referenciar o robô e demais eixos.

DIGITE NO TERMINAL DO COMPUTADOR:

RUN HOMES ↵ (Enter)

NO TERMINAL APARECERÁ A SEGUINTE MENSAGEM:

Wait !! homing...

QUANDO O ROBÔ PARAR DE SE MOVER, VOCÊ RECEBERÁ A MENSAGEM:

Homing complete (robot).

VOCÊ PODE RECEBER UMA DAS SEGUINTES MENSAGENS NA TELA, ENQUANTO ESTIVER MOVIMENTANDO O ROBÔ:

***** UPPER LIMIT AXIS n**

***** LOWER LIMIT AXIS n**

onde n é o número do eixo. Isso significa que o robô não pode mais se movimentar naquela direção, por já ter atingido o seu limite mecânico.

***** IMPACT PROTECTION AXIS n**

***** THERMIC PROTECTION AXIS n**

Nesse caso, o robô não pode mais se movimentar naquela direção, tendo o controle sido automaticamente desconectado. Você precisará conectar o controle novamente a fim de mover o robô, através do comando CON.

3. COMANDOS DIRETOS

A linguagem ACL tem vários comandos, alguns que operam somente no modo direto (direct), outros apenas no modo edição (edit) e alguns que operam em ambos os modos. No Anexo desta Apostila você encontra, em ordem alfabética, os principais comandos com os respectivos modos de operação. Veremos, a seguir, alguns comandos diretos.

No modo direct, você emite os comandos a partir do Terminal do computador. Para isso, abra o aplicativo "Robosoft" no seu computador ("Start"+"Programs"+"Robsoft" ou clique duas vezes sobre o ícone na área de trabalho do computador). Irá abrir uma tela conforme a figura abaixo. Entretanto, antes de começar, você deve preparar o robô. Verifique a posição da chave de seleção Auto/Teach dos controladores tipo B e AC, no tipo A não existe a chave, coloque em "Auto". Após no terminal digite "Auto", este comando habilita o uso do terminal. No terminal do controlador tipo A isso não é necessário. Em ambos os controladores faça o "Homing" (referenciamento do robô):

DIGITE: RUN HOMES ↵ (enter)

3.1.1. COMANDO "PRINT"

O comando PRINT tem a seguinte forma:

PRINT "texto"

Esse comando escreve no terminal do computador, na posição corrente do cursor, o conteúdo entre as aspas. No exemplo acima iria mostrar no terminal a palavra texto.

Faça o seguinte exemplo:

DIGITE: PRINT "CIM LAB" ↵

Observe que o comando que você digitou e a resposta do controlador aparecem uma após a outra na tela, na mesma linha. Isso acontece porque o comando PRINT escreve o texto entre aspas na posição corrente do cursor.

Se você cometer um erro, o controlador responderá com uma mensagem de erro. Por exemplo, repita o comando anterior, porém, desta vez não coloque as aspas. Veja o que acontece.

Além disso, tente escrever algo que não é um comando ACL, por exemplo, a palavra aluno.

Observe a resposta do controlador.

Os comandos CON e COFF ligam e desligam, respectivamente, o servo controle para todos os eixos ou para um grupo de eixos especificados.

```
Exemplos:    CON        ; liga o servo controle para todos os eixos
             COFFA      ; desliga o servo controle para todos os
                        ; eixos do grupo A
             CON 10     ; liga o servo controle para o eixo 10
             COFF       ; desliga o servo controle para todos os
                        ; eixos
```

4. ESCREVENDO E EXECUTANDO PROGRAMAS EM ACL

Um programa é um conjunto ordenado de comandos que recebe um nome pelo qual ele pode ser referido. Quando você executa (comando "run") o programa, o conjunto de comandos que foi armazenado no controlador será executado em seqüência pelo mesmo. Um programa em ACL sempre começa pelo comando PROGRAM <NOME> e termina pelo comando END:

PROGRAM PROG1

.
. comandos
.

END

Você vai agora escrever um programa muito simples em ACL que imprime a mensagem "Hello Mundo" na tela. Inicialmente, você deve dar um nome ao programa, usando até 5 caracteres alfanuméricos. Dê o NOME do programa já no formato padrão solicitado, neste exemplo chamaremos este seu programa de Prog1. Você utilizará o comando PRINTLN, o qual é semelhante ao PRINT, mas move o cursor para uma nova linha antes de imprimir a mensagem. **PRINTLN** é um comando que só pode ser usado no modo **edição**.

DIGITE E RODE O PROGRAMA PROG1

Suponhamos, agora, que você queira modificar o programa Prog1, colocando nele uma nova linha tal como "Bem-vindo ao LABCIM!". Insira a nova linha no programa, salve-o e faça um novo download. Em seguida, execute o programa.

A CADA MODIFICAÇÃO QUE VOCÊ FIZER EM UM PROGRAMA, UM NOVO DOWNLOAD DEVE SER FEITO!

5. ESTRUTURAS DE PROGRAMAÇÃO: "LOOPS"

5.1. Comandos GOTO and LABEL

Suponhamos que seja necessário um laço de repetição , uma vez alcançado um certo ponto do programa, você gostaria de saltar para um outro ponto do programa (anterior ou posterior) e executar os comandos existentes após aquele ponto. Você pode fazer isso com os comandos GOTO e LABEL, criando o que se chama de "*loop*".

O comando LABEL <n> marca uma localização onde o programa irá saltar e continuar o programa, onde n é um número inteiro. O comando GOTO <n> faz o programa saltar para a linha onde se encontra o respectivo LABEL <n>.

Exemplo:

```
PROGRAM PROG2
LABEL 2
PRINTLN "DEUS AJUDA"
PRINTLN "A QUEM CEDO MADRUGA"
PRINTLN
GOTO 2
END
```

No programa acima, o comando PRINTLN imprime numa nova linha (lembre que PRINT imprime aonde estiver o cursor). O último comando PRINTLN, o qual não tem argumento, coloca uma linha em branco na tela após o último comando de imprimir.

Se fossemos fazer o teste de mesa :

- Início do programa
- passo 1: Label 2
- passo 2: Imprime na nova linha – DEUS AJUDA
- passo 3: Imprime na nova linha – A QUEM CEDO MADRUGA
- passo 4: GOTO 2- Salta para o LABEL 1 (passo 1)

O programa imprimirá a mensagem a primeira vez. Quando ele alcançar o comando GOTO 2, ele saltará para LABEL 2 e repetirá a impressão na tela eternamente (ou até que, de algum modo, o programa pare). Esse programa vai realizar um "loop" infinito, nunca terá fim.

5.2. O comando ABORT

No programa PROG2 você necessita de uma maneira de parar o loop infinito. Isso pode ser feito através do comando CTRL + <a>.

PRESSIONE : CRTL + a

Esse importante comando pára imediatamente o programa. Quando você começar a usar comandos que movimentam o robô, é essencial que se tenha um modo de parar o seu movimento rapidamente, em caso de emergência, o "CTRL + a" pode ser uma delas.

6. COMANDOS ACL PARA GERENCIAMENTO DE PROGRAMAS

Esses comandos, emitidos no modo direto, ou, "*direct*", operam sobre o programa como um todo. Usando esses comandos, você poderá executar operações tais como copiar ou deletar um programa. Você também aprenderá como ilustrar os seus programas com *comentários*, os quais ajudam outros usuários a entender os seus programas.

6.1. O Comando DIR

O comando DIR fornece uma lista de todos os programas existentes na memória do controlador.

DIGITE: **DIR**

Para cada programa, são informados: **name**, **validity**, **identity** e **priority**.

Name é o nome que você atribuiu ao seu programa.

Validity informa se a sintaxe do programa está correta. Se "not valid" aparece na coluna validity, o programa não está sintaticamente válido e não poderá ser executado.

Identity é o número do programa, o qual é atribuído pelo controlador. O número é usado para acessar programas a partir do TP, conforme será explicado mais tarde.

Priority nos informa sobre a preferência que cada programa tem na alocação do tempo de computação, ou seja, programas com prioridade mais alta têm preferência quando vários programas são executados concorrentemente. As prioridades vão de 1 a 10, onde 10 é a prioridade mais alta. A prioridade default para programas dos usuários é 5. Quando um programa é "not valid", a sua prioridade é 0.

6.2. O Comando RENAME

RENAME troca o nome do programa sem afetar o seu conteúdo. Por exemplo, o comando abaixo troca o nome do programa de PROG3 para PROG4.

DIGITE: **RENAME PROG3 PROG4**

6.3. O Comando COPY

O comando COPY faz uma cópia do programa com um novo nome. Por exemplo, o comando abaixo cria um novo programa denominado PROG6, com conteúdo idêntico ao programa PROG5, sem destruir esse último.

DIGITE: **COPY PROG5 PROG6**

6.4. O Comando REMOVE

REMOVE apaga programas da memória do controlador. É um comando poderoso que deve ser usado com extremo cuidado. Para apagar efetivamente um programa, você deverá responder YES à pergunta **Are you sure (YES/NO)?** Por exemplo:

DIGITE: **REMOVE PROG6**
Are you sure (YES/NO)?
YES

PROG6 será removido da memória do controlador.

6.5. O Comando LIST

LIST <PROG> mostra todas as linhas do programa PROG.

LIST, apenas, mostra todas as linhas de todos os programas que estão na memória do controlador.

6.6. O Comando STAT

STAT mostra o estado dos programas ativos. A lista inclui prioridades e o estado de operação (DELAY, PEND, SUSPENDED). O estado PEND É INFORMADO QUANDO O PROGRAMA ESTIVER AGUARDANDO UM MOVIMENTO SER COMPLETADO.

7. COMENTÁRIOS DE PROGRAMAS

A maioria dos programas que você escreveu até agora foram curtos e de fácil entendimento. Entretanto, à medida que você progride, eles se tornarão mais longos e mais complicados, podendo se tornar de difícil compreensão para outros usuários.

Uma maneira de contornar essa dificuldade é usar **comentários** nos seus programas. Um comentário é um texto explicativo inserido em um programa. Não é um comando executável e não afeta a execução do programa. Qualquer linha começando com um ";" é interpretada como um comentário. São permitidos até 40 caracteres em uma linha de comentário.

COLOQUE, SEMPRE, *COMENTÁRIOS* EM SEUS PROGRAMAS.
ELES SERÃO MUITO ÚTEIS PARA O RÁPIDO ENTENDIMENTO DO PROGRAMA.

Capítulo 3
USANDO A LINGUAGEM ACL PARA CONTROLAR O ROBÔ

Objetivos:

Conhecer o software envolvido
Referenciar os robôs
Editar e compilar programas
Comando PRINT e PRINTLN
Controle de loops com os comandos LABEL e GOTO
Comentários nos programas
Comandos ACL para gerenciamento de programas

1. INTRODUÇÃO

Apresentar os diversos tipos de variáveis que podem ser empregadas na programação ACL:

- **Variáveis Locais e Globais**
- **Controle de loops**
- **Comandos – Global, Define, Set, Listvar, Delvar, Read, Set, For -Endfor**

2. VARIÁVEIS

Para se definir uma variável, deve-se atribuir um nome a mesma. Este nome tem no máximo 5 caracteres, e deve ser iniciado com um caracter alfanumérico. Existem dois tipos de variáveis: as Globais - aquelas vistas por qualquer programa e as Locais -aquela que é vista somente pelo programa que a criou. Em ambos os casos, elas são do tipo inteiro de 32 bits.

2.1. VARIÁVEL GLOBAL

A variável GLOBAL pode ser definida através de comando **direto** ou **indireto**. O nome da variável não deve ultrapassar 5 caracteres alfanuméricos.

É definida através do comando:

GLOBAL <nome>

Exemplo:
Os comandos realizados no modo direto.

Crie uma variável global MyVar e imprima seu resultado. Qual é o valor?

GLOBAL MyVar
PRINT MyVar

Exemplo:
Comandos realizados no modo indireto, dentro do programa:

GLOBAL A
GLOBAL VAR AUX B

2.2. VARIÁVEL LOCAL

Toda variável LOCAL é definida no modo **indireto** e só pode ser utilizada **pelo programa que definiu a mesma**. `

DEFINE <var>

Em uma linha de comando, pode-se definir até **8 variáveis**, separadas por um caractere de espaço.

DEFINE <var1> <var2> <var3> <var4> <var5> <var6> <var7> <var8>

Exemplos:

DEFINE R
DEFINE S T

2.3. COMANDO SET

Para definir o valor de uma variável você deve utilizar o comando SET. Comando pode ser definido no modo **direto e indireto**. O comando SET possui a seguinte sintaxe:

SET <nome variável> = <valor numérico>

Exemplos:

SET A=1
SET VAR=8

2.4. COMANDO READ

Este comando aguarda uma entrada de dado numérico e atribui a uma variável. Deste modo é possível escrever um programa interativo, pois com o mesmo é possível atribuir um valor a uma variável através do prompt do computador. Este comando só pode ser definido no modo **indireto**. O comando READ possui a seguinte sintaxe:

READ <nome da variável>
Ou
READ "texto" <nome da variável>

2.5. CONTROLE DE LOOP - FOR

Às vezes é necessário fazer o controle de execução de um laço de repetição. Este comando é realizado normalmente pelo comando FOR. Este comando só pode ser definido no modo **indireto**. O comando FOR possui a seguinte sintaxe:

FOR <NOME DA VARIÁVEL> = <INICIO> TO <FIM>
.
.
comandos
.
.
ENDFOR

3. EXERCÍCIOS

a) Teste o comando PRINT na forma direta e indireta. Defina uma variável global e uma variável local e teste na foma direta e indireta. Imprima nestes casos o valor da variável. Como a variável é do tipo global, tente imprimir o valor da mesma, dentro de um outro programa. Qual o erro que será sinalizado se a variável local pertencer a outro programa?

b) Crie um programa que defina as variáveis locais A, B e C. Atribua a estas, os valores 5, -2 e 8. Imprima os resultados. O que se observa?

c) Crie um programa com os seguintes comandos, rode este programa e faça comentários sobre cada comando:

```
PROGRAM EX1G'_' (ultimo digito é o número do grupo)
        DEFINE CONT A AUX
        PRINTLN "TESTE DO COMANDO FOR"
        PRINTLN "ENTRE COM O NUMERO DE LOOPS "
        READ AUX
        PRINTLN
        SET AUX=AUX+'_'  ( ' _' substituir pelo número do grupo)
        FOR CONT=0 TO AUX
                FOR A=0 TO '_'  ( ' _' substituir pelo número do grupo)
                        PRINT ">"
                ENDFOR
                PRINT "LOOP NUMERO = "
                PRINT CONT
                PRINTLN
        ENDFOR
END
```

d) Crie um programa, que entrando com QUALQUER valor através do promp, realize a seguinte a saída:

Se o número digitado for ímpar, por exemplo, o nº 3, o mesmo deve imprimir da seguinte forma:

```
 > 3
***
 **
  *
```

Se o número digitado for par, por exemplo, o nº 4, o mesmo deve imprimir da seguinte forma:

```
> 4
*
**
***
****
```

e) Crie um programa, que entrando com dois valores QUAISQUER através do promp, realize a seguinte saída de dados:
Exemplo de uma matriz 4 por 4 e uma 3 por 2

```
>
>RUN EX2G1
PROGRAMA MATRIZ
ENTRE COM NUMERO DE LINHAS = 4
ENTRE COM NUMERO DE COLUNAS = 4
[1,1] [1,2][1,3] [1,4]
[2,1][2,2][2,3][2,4]
[3,1][3,2][3,3][3,4]
[4,1][4,2][4,3][4,4]
```

```
>
>RUN EX2G1
PROGRAMA MATRIZ
ENTRE COM NUMERO DE LINHAS = 3
ENTRE COM NUMERO DE COLUNAS = 2
[1,1][1,2]
[2,1][2,2]
[3,1][3,2]
```

Capítulo 4
PROGRAMAÇÃO UTILIZANDO VETORES

Objetivos:

Apresentar os diversos tipos de variáveis que podem ser empregadas na programação ACL. AS variáveis do ACL são sempre do tipo inteiro de 32bits. Isto deve ser considerado. Principalmente quando estamos realizando operações de divisão
Vetores locais e globais
Comandos de decisão

1. INTRODUÇÃO

Apresentar os diversos tipos de variáveis que podem ser empregadas na programação ACL. As variáveis do ACL são sempre do tipo inteiro de 32bits. Isto deve ser considerado. Principalmente quando estamos realizando operações de divisão. Além disso, será mostrada a utilização de vetores locais e globais.

Neste capítulo será apresentado o comando de decisão IF. Com este comando será possível criar lógicas mais complexas de decisão.

2. Comandos

2.1. Vetores - Comando DIM

Como visto anteriormente, uma variável ACL pode ser do tipo local ou global. As variáveis locais só podem ser definidas dentro de um programa, **comando indireto,** através do comando DEFINE. Uma variável do tipo global pode ser definida dentro ou fora de um programa e é visível por qualquer programa que esteja rodando no controlador. O ACL permite também que se definam vetores que podem ser do tipo local ou global.

O vetor do tipo local é declarado através da seguinte sintaxe:

DIM <nome do vetor>[tamanho do vetor]

DIM var [n], var é o nome do vetor, n é o tamanho do vetor. Este comando é do tipo indireto. A variável é composta dos elementos var[1], var[2], ...var[n].

2.2. Vetores - Comando DIMG

O vetor do tipo global é declarado através da seguinte sintaxe:

DIMG <nome do vetor>[tamanho do vetor]

DIMG glob[n], onde n é a dimensão do vetor. Este comando é do tipo direto ou indireto e declara uma variável global de nome glob. A variável glob é composta dos elementos glob[1], glob[2], ...glob[n].

2.3. Comando SET

O valor dos vetores é definido também pelo comando SET, neste caso, cada elemento do vetor deve ser carregado independentemente. Exemplo:

SET vet[1]=10
SET vet[2]=3
.....
SET vet[n]=40

O comando SET é utilizado para realizar operações lógicas e aritméticas com todas as variáveis da linguagem ACL. O comando SET pode ter as seguintes combinações de operandos e variáveis :

SET var1 = var2
SET var1 = oper var2
SET var1 = var2 oper var3

onde oper pode ser as operações aritméticas de soma, subtração, multiplicação e divisão (respectivamente +,-,* e /), operações algébricas de resto da divisão, módulo, exponencial e logarítmo (respectivamente MOD, ABS, EXP e LOG), operações trigonométricas seno, cosseno, tangente e arcotangente (respectivamente SIN, COS, TAN e ATAN), e as operações lógicas AND, OR e NOT.

Uma vez que o controlador trabalha apenas como representação de números inteiros, operações que resultem valores em módulo menor do que 1, devem ser escalonados para obter a precisão necessária. É o caso, por exemplo, das funções trigonométricas. Observe os exemplos a seguir:

SET A = 1000 COS 60. A recebe 1000*cos(60). O argumento var3 está em graus.

SET A = 100 EXP 200. A recebe 100*exp(200/1000). O argumento var3 é primeiro dividido por 1000 para então ser aplicado a função.

SET A = 10 LOG 10000. A recebe 10*LOG(10000/1000). O argumento var3 é primeiro dividido por 1000 para então ser aplicado a função.

2.4. Operações em PONTO FIXO

Uma vez que o ACL representa somente números inteiros, o resultado de muitas operações como, por exemplo, o da divisão, é truncado. Uma forma de contornar este problema é escalar um

dos operandos de modo a obter um resultado mais preciso. Por exemplo, se desejarmos dividir os operandos A e B, respectivamente de valores 30 e 4 obteríamos o valor 7. Podemos obter uma maior precisão, digamos três dígitos, multiplicando o operando A por 1000. Desta forma obteríamos o resultado 7500.

2.5. Comando de decisão IF/ELSE

Em muitas situações é necessário tomar decisões em resposta a uma entrada de dados ou aos valores dos dados fornecidos pelo usuário. No ACL podemos tomar alguma decisão através de valores numéricos de variáveis, condições das entradas e saídas gerenciadas pelo controlador. Estas tomadas de decisão são realizadas pelo comando **IF/ELSE**. A sintaxe deste comando é dada por:

IF <VAR> COND <VAR>
.
comandos
.
ELSE
.
comandos
.
ENDIF

Onde COND pode ser as seguintes relações: > (maior que), = (igual), < (menor que), >=(maior que ou igual), <>(diferente) e <= (menor que ou igual). O comando de IF pode testar apenas duas variáveis. Se houver necessidade de combinar outras variáveis, é possível empregar os comandos ANDIF e ORIF. A sintaxe destes comandos são dados conforme os exemplos abaixo:

IF A = B
ANDIF D=9
.
comandos
.
ELSE
.
comandos
.
ENDIF
	IF A<>B
	ORIF E>8
	.
	comandos
	.
	ELSE
	.
	comandos
	.
	ENDIF

3. EXERCÍCIOS:

a) Faça um programa que cria um vetor local e um vetor global com 10 posições. Como é possível listar estas variáveis? Qual seria a resposta esperada a este comando?

b) Altere o programa anterior de modo a carregar os valores 1, 2, 3, ..., 10 aos respectivos elementos dos dois vetores. Execute o programa. Altere novamente o programa de modo a inserir valores aleatórios diretamente do teclado. Neste caso, altere o valor dos vetores indicando a posição e o valor que o mesmo possuía. Tente agora remover um dos vetores. Qual comando deve ser empregado ? É possível remover estas variáveis ?

c) Escreva um programa que utilizando o algoritmo anterior, calcule o seno de 10 arcos igualmente espaçados. A precisão desejada é de quatro dígitos.

d) Escreva um programa que converta a temperatura expressa em °C para °F para todos os valores a cada 10° de temperatura no intervalo de 10° a 100°. Utilize uma precisão de três dígitos.

e) Escreva um programa que converta °C para °F. O programa encerra se o usuário digitar um número menor que zero ou maior que 300.

f) Para o conjunto de pontos (4, 1, -3), (2, 9, 1), (3, -3, 6). Escreva um programa que indique qual é o maior dos elementos e qual é o menor deles.

g) Escreva um programa que faça o produto escalar de dois vetores. O usuário deverá fornecer também o tamanho destes vetores. Crie um menu inicial onde o usuário deverá digitar "0" para sair, ou, "1" para multiplicar os vetores. Se o valor estiver fora destes limites, o programa deverá sinalizar com uma mensagem de erro, e, voltar ao menu. O programa só será encerrado quando digitado o valor "0".

h) Faça um programa. Neste você deve declarar dois vetores de dimensão 5, entrar com os valores através do teclado e imprimir o conteúdo destes vetores. O usuário deverá ser informado qual é o nome da variável e a posição que o mesmo vai inserir o dado. O mesmo deve acontecer na impressão do conteúdo dos vetores. A inserção e a impressão dos dados devem ser dispostas em forma de lista.

Ex.: var[1]=12
var[2]=52
var[3]=10 ...

i) Acrescente no programa anterior as operações de soma, subtração, multiplicação e divisão. Estas operações serão realizadas entre as mesmas posições dos vetores.

Exemplo: O elemento 1 do primeiro vetor deve ser somado com o elemento 1 do segundo vetor, 2 com 2, ... O programa armazena os resultados em outro vetor. Teste o seu programa com os seguintes operandos :

A[1]=10, A[2]=15, A[3]=3, A[4]=30, A[5]=25 e B[1]=2, B[2]=7, B[3]=-7, B[4]= 9, B[5]=4.

j) Modifique o programa para que este determine o resto da divisão e a média de cada um dos operandos do primeiro vetor e do segundo vetor.

k) Modifique o exercício anterior de forma a separar a parte inteira e a parte fracionária, de modo a imprimir o resultado da operação como: INTEIRO "," FRACIONÁRIO.

l) Desenvolva um programa que utilizando o conjunto de pontos dados no exercício anterior, calcule a média dos operandos do primeiro vetor e do segundo vetor com três dígitos de precisão. Modifique o programa de forma a permitir ajustar a precisão conforme a necessidade do usuário.

m) Desenvolva um programa que teste os comandos IF, ANDIF, ORIF. O programa deve indicar se um determinado valor digitado pelo usuário é igual a "3", "4" ou "5". Se for deve sinalizar a mensagem "Pertence ao conjunto", caso contrário deve sinalizar com "Não pertence".

Capítulo 5
COMANDOS DE MOVIMENTAÇÃO DO ROBÔ

Objetivos:

Apresentar os comandos básicos para a movimentação do robô.
Variáveis de posição.
Comandos de movimentação.

1. INTRODUÇÃO

Apresentar os comandos básicos para movimentação do robô. Para isso será necessário definir e manipular as variáveis de posição. Será apresentada a utilização do TEACH PEDANT (TP) para a movimentação dos eixos, para a troca de sistemas de coordenadas, para a gravação de coordenadas e para a execução de movimentos previamente gravados, entre outros comandos.

Neste capítulo serão apresentados os seguintes comandos:
DEFP, DEFP, DEFPB, DEFPC
LISTP
DELP
DIMP
ATTACH

Comandos de movimentação:
MOVE
MOVED
OPEN
CLOSE
SPEED

2. MANIPULAÇÃO DAS VARIÁVEIS DE POSIÇÃO

A variável de posição é um tipo especial de dado capaz de manipular as informações de localização do robô, ou seja, armazena a posição de cada junta do robô. Diferentemente das variáveis locais ou globais, que manipulam um único valor inteiro, cada variável de posição manipula as informações de: Base (eixo 1), Shouder (eixo 2), Elbow (eixo 3), Pitch (eixo 4), Roll (eixo 5) e Gripper (eixo 6), para o grupo A ou (eixo 7) e (eixo 8) para o grupo B. As variáveis de posição podem ser denominadas de forma numérica ou de forma literal. Por exemplo :

Denominação Numérica: 15, 2, 24, .
Denominação Literal: Posit, Posl, X25,

Cada variável pode ter um nome com no máximo 5 caracteres. A denominação numérica permite com que o Teach Pendant associe a posição atual do robô a variável numérica apenas, pois o mesmo não é capaz de manipular caracteres alfanuméricos.

2.1. Comando DEFP

Uma variável do tipo posição é criada com os comandos DEFP ou DEFPA, DEFPB ou DEFPC, os quais tem a seguinte sintaxe :

DEFP <varl> (padrão grupo A) ou DEFPA <var2>
DEFPB <var3>
DEFPC <var4> n

O comando DEFP <var1> cria uma variável de nome var1, do tipo posição e associa a mesma ao grupo A. Lembrando que os controladores dos robôs que estão no laboratório manipulam 8 eixos, sendo 6 eixos do robô os quais são definidos como grupo A e 2 eixos de acessórios, definidos como grupo B. Os comandos DEFPA, DEFPB e DEFPC, criam variáveis de posição e associam estas variáveis aos grupos A, B e C respectivamente. O comando DEFPC tem como argumento além do nome da variável, o número do eixo que a mesma irá ser associada.

2.2. Comando LISTP

Crie as variáveis de posição pos1 e pos2. Assim como, as variáveis de manipulação numérica, as variáveis de posição podem ser listadas e removidas quando necessário. O comando para listar as variáveis é o comando LISTP, comando **direto**. Assim como o comando LISTVAR, comando **direto**, este comando não tem argumento. Ao ser executado este lista todas as variáveis de posição que foram definidas serão msotrados. Ao executar, é esperado o seguinte resultado:

DEFINE POINTS
****

point name :	group	axis
P[10]	:A	
PPCP	:A	
BI	:B	
ALFA	:B	
BRAS	:C	:10
NEVE	:C	:11

2.3. Comando LISTPV

O ACL permite também visualizar o conteúdo de uma variável do tipo posição de modo a observar nos dois sistemas de coordenadas, juntas ou cartesianas, o valor desta variável ou a posição atual do robô. O comando para esta tarefa é LISTPV, cuja sintaxe é:

LISTPV <var>
LISTPV POSITION

onde **POSITION** é uma palavra reservada e quando utilizada, indica a posição atual do robô. LISTPV <var> indica em valores de encoder o conteúdo de uma determinada variável. Se a variável

for do grupo A, o comando também irá indicar o conteúdo da variável em coordenadas cartesianas, conforme exemplo a seguir:

```
LISTPV pos1
Position pos 1
1:0          2:5          3:13926      4:0          5:0
X: 5.425     Y: 0.000     z: 29.963    P: 12.640    R: -3.00
```

onde as informações XYZ são dadas em milímetros, P e R são expressas em graus, com precisão de 0.002°.

2.4. Comando DELP

Como pode-se observar, ao listar as variáveis de posição, aparece o nome da variável, o grupo que a mesma está associada e se for o caso, qual eixo que a mesma irá controlar. Para remover uma variável do tipo posição, o ACL dispõe do comando DELP. A sintaxe deste comando é:

DELP <var>

Para remover qualquer variável, você deve digitar "YES" para confirmar sua intenção.

2.5. Comando DIMP

As variáveis de posição podem também ser declaradas como vetores com o comando DIMP[A/B] ou DIMPC, cuja sintaxe é apresentada a seguir:

```
DIMP     pvec1[n]
DIMPA    pvec2[n]
DIMPB    pvec3[n]
DIMPC    pvec4[n] m
```

O comando DIMP <pvec1[n]> cria uma variável de nome pvec1, do tipo posição de n elementos (pvec1[1], pvec1[2], ...pvec1[n]) e associa a mesma ao grupo A. Os comandos DIMPA, DIMPB e DIMPC criam variável do tipo posição de n elementos e associam estas variáveis aos grupos A, B e C respectivamente. O Comando DIMPC tem como argumento além do nome da variável e o número de elementos, o número do eixo que a mesma irá ser associada.

2.6. Comando ATTACH

Para indexar ao vetor de posições à posição selecionada, existem dois modos: via terminal do computador ou pelo Teach Pedant (TP). Pelo terminal seria necessário utilizar o comando HERE <var1[n]>, comando **direto/indireto,** onde var1 é o nome da variável e n é a posição da variável. Pelo Teach Pedant (TP) seria necessário habilitar a indexação do vetor de posições pelo comando ATTACH via terminal. Este comando é um comando **direto** e não depende da chave seletora do TP dos controladores B/AC, ou seja, a chave pode estar em AUTO, ou, em TEACH. Uma vez selecionado uma posição do robô, a mesma poderá ser gravada pressionando a tecla RECORD POSITION e após digitando a POSIÇÃO do vetor de posição. Este vetor de posição poderá ser empregado dentro de um programa, no qual será definido a trajetória do robô.

Logo, o comando ATTACH possui as seguinte variações de sintaxe:

```
ATTACH      VET
ATTACH      OFFA
ATTACH      OFFB
ATTACH      OFFC
ATTACH      ?
```

onde ATTACH VET associa a variável VET do tipo vetor de posição de n elementos ao grupo que esta variável foi criada (A, B ou C).

Os comandos ATTACH OFFA, ATTACH OFFB e ATTACH OFFC, retiram as associações feitas aos grupos A, B ou C respectivamente.

O comando ATTACH ? mostra no terminal quais as associações feitas até o momento.

Para testar se uma determinada posição está correta, ou, se necessita ser corrigida, o TP possui a tecla "GO POSITION n" onde n é o enésimo elemento de uma variável tipo vetor de posições. É possível associar também variáveis do tipo posições definidas numericamente. Neste caso, cada dígito é uma posição isolada.

3. COMANDOS ACL PARA MOVIMENTAÇÃO DO ROBÔ

O ACL possui um repertório bastante amplo para a execução de movimentos. Vejamos um conjunto básico, com os quais podemos escrever programas que executem tarefas simples como movimentar por uma determinada trajetória, em velocidades preestabelecidas, aguardando um determinado tempo em cada posição, pegando determinados objetos e colocando-os em outras posições. Estes comandos são :

3.1. Comando SPEED

O parâmetro velocidade pode ser ajustado através dos seguintes comandos :

SPEED vel
SPEEDA vel
SPEEDB vel
SPEEDC vel axis
SHOW SPEED

SPEED e SPEEDA ajustam a velocidade em percentual de 0 até 100 dos eixos do grupo A através do parâmetro vel.
SPEEDR ajusta a velocidade do grupo R
SPEEDC ajusta a velocidade de um eixo do grupo C especificado no parâmetro axis.

O comando SHOW SPEED apresenta no terminal, os ajustes feitos pelo usuário no parâmetro velocidade. O valor de vel deve estar entre 1 e 100.

3.2. Comando MOVE (DIRETO/EDIÇÃO)

`

O comando MOVE, modo **direto/indireto,** possui a seguinte sintaxe:

MOVE pos [duração]

move o Tool Center Point (TCP) do robô para a posição 'pos' em um tempo duração (opcional), especificado em centésimos de segundo. O comando é depositado no buffer de movimento e o comando seguinte do programa não espera a conclusão do comando MOVE, não garantindo seqüencialidade ou precisão. o comando MOVE é executado com a velocidade definida pelo último comando SPEED

EXEMPLO:

MOVE p[23] 500

3.3. Comando MOVED (EDIÇÃO)

O comando MOVED, modo **direto/indireto,** possui a seguinte sintaxe:

MOVED pos [duração]

move o TCP do robô para a posição 'POS' em um tempo duração (OPCIONAL), especificado em centésimos de segundo. O comando só é depositado no buffer de movimento quando o comando MOVED anterior tiver sido totalmente executado se o MOVED for precedido do comando EXACT, que é o default, garante-se seqüencialidade e precisão, mas não é garantida a duração. Se o MOVED for precedido do comando EXACT OFF, garante-se seqüencialidade e duração, mas não se garante a precisão.

o comando MOVED é executado com a velocidade definida pelo último comando SPEED

EXEMPLO:

MOVED p[58]

3.4. Comandos OPEN/CLOSE

Os comandos OPEN/CLOSE possuem a função de abrir e fechar a garra, modo **direto/indireto**, possui a seguinte sintaxe:

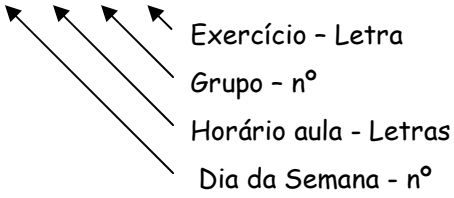
OPEN
CLOSE

Estes comandos OPEN/CLOSE podem ser substituídos pelos respectivos programas: OGRIP e CGRIP. Muitas vezes os comandos OPEN/CLOSE não acionam as válvulas, por este motivo é importante rodar estes programas, pois eles acabam zerando os parâmetros e garantindo o acionamento das válvulas.

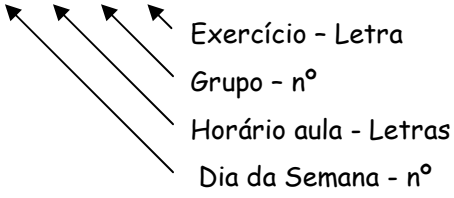
4. EXERCÍCIOS:

OBS.: A partir desta aula as variáveis de posição e os nomes dos programas deverão ser criados de acordo com as seguintes características, qualquer outro nome poderá acarretar em perda da variável e do programa.

NOME PROGRAMAS: __ __ __ __ __



NOME VARIÁVEL: __ __ __ __ __



Exemplo.: 2LM1A

2 – Segunda-feira (os dias teriam os seguintes números: 2-segunda, 3-terça, 4-quarta, 5-quinta, 6-sexta, 7-sabado)
LM – as respectivas letras de horário da grade da PUCRS
1 – grupo 1
A - exercício A

a) Crie algumas variáveis de posição com o comando DEFP. Como exercício, liste o conteúdo das variáveis criadas com o comando LISTP e comente o resultado. Determine também a posição atual do robô.
b) Defina um vetor de 10 elementos do tipo posição. Faça a indexação deste vetor ao TP. Movimente o robô em 10 diferentes e grave estas posições nas respectivas posições

Capítulo 6
PROGRAMAÇÃO UTILIZANDO OS COMANDOS DE MOVIMENTAÇÃO DO ROBÔ

Objetivos:

Apresentar os comandos avançados para a movimentação do robô.

1. INTRODUÇÃO

Apresentar os comandos avançados para movimentação do robô. Para isso será necessário definir variáveis de posição.

Neste capítulo serão apresentados os seguintes comandos:

MOVEC/MOVECD
MOVEL/MOVELD
MOVES/MOVESD

2. COMANDOS DE MOVIMENTAÇÃO COM CONTROLE DE TRAJETÓRIA

2.1. Comando MOVEC/MOVECD

O comando MOVEC/MOVECD possui a seguinte sintaxe:

MOVEC pos1 pos2 (modo de edição/direto)
MOVECD pos1 pos2 (modo de edição)

Este comando move o robô ao longo de uma trajetória circular, a partir da localização atual para <pos1>, através da <pos2>. A localização da <pos2> e <pos1> determina o comprimento da trajetória, a qual em giro determina o tempo necessário para completar o movimento.

O tempo requerido para completar o movimento pode ser definido pelo comando SPEED precedente. A velocidade do movimento circular determina a precisão da trajetória circular.

Todos os outros aspectos dos comandos MOVEC/MOVECD são similares aos dos comandos MOVE/MOVED.

Atenção! Cuidado quando gravar posições para comandos MOVEC. Limitações mecânicas ou obstáculos, tais como o próprio robô, podem tornar o arco resultante inválido.

Exemplos:

MOVEC 1 2
Move ao longo da trajetória circular a partir da atual posição para a posição1 via posição2.

SPEED 20
MOVEC 2 1

Move ao longo da trajetória circular a partir da atual posição para a posição 2 via posição 1, a velocidade 20.

Nota:
Este comando é válido apenas para posições do robo (grupo A).

2.2. Comando MOVEL/MOVELD

O comando MOVEL/MOVELD possui a seguinte sintaxe:

MOVEL pos1 [duração] (modo de edição/direto)
MOVELD pos1 [duração] (modo de edição)

Move o robô ao longo de uma trajetória linear (linha reta) a partir da localização atual para a <pos1>.
A velocidade do movimento linear determina a precisão da trajetória.

Todos os outros aspectos deste comando são similares aos dos comandos MOVE/MOVED.

Atenção! Cuidado quando gravar posições para os comando MOVEL. Limitações mecânicas ou obstáculos, tais como o próprio robô, podem tornar o resultado do arco inválido.

Exemplos:

MOVELD TR
Move-se ao longo de uma linha reta para a posição TR.

Nota:
Este comando é válido apenas para posições do robo (grupo A).

2.3. Comando MOVES/MOVESD

O comando MOVES/MOVESD possui a seguinte sintaxe:

MOVES < pvect > < firstpos> <lastpos> <tempo> (modo direto/edição)
MOVESD < pvect > < firstpos> < lastpos> < tempo> (modo edição)
Onde:
<firstpos> é a primeira posição a ser alcançada, e
<lastpos > é a última posição alcançada.

Este comando move os eixos do robô através de qualquer número de vetores de posição consecutivos, ou seja, primeira posição (<firstpos>) para última posição (<lastpos>) desejada, sem pausa.

Um perfil de movimento é aplicado ao movimento introduzido. Trajetórias de aceleração e desaceleração ocorrem apenas no início e fim do movimento total. Uma vez que o tempo distribuido para cada segmento é igual, a velocidade do movimento é determinado pela distância entre os vetores de posição.

Todos os outros aspectos dos comandos MOVES/MOVESD são similares aos dos comandos MOVE/MOVED.

Exemplos:

MOVED PATH [1]
MOVESD PATH 2 20
MOVESD PATH 19 1

Move para a posição de partida 1. Após ocorre uma trajetória contínua através das posições 2 para 20. Então executa a mesma trajetória na direção oposta.

Nota:

E recomendável que os vetores de posição sejam igualmente espaçados, para permitir uma velocidade de curva suave.

3. EXERCÍCIOS

MANIPULAÇÃO DE OBJETO

a) Escreva um programa em ACL que faça as seguintes operações:

→ Inicialmente, é dada uma pilha de quatro paralelepípedos, estando o maior embaixo e o menor em cima, em ordem decrescente de tamanho;

→ O robô deverá desmontar a pilha, colocando os objetos nos vértices de um quadrado cujo centro é a posição da pilha, em ordem crescente de tamanho e no sentido horário;

→ Acima da pilha de paralelepípedos existirá uma posição de espera. O robô antes de começar a operação deverá ir a está posição e imprimir na tela a mensagem "Estou na posição de espera";

→ A cada posição ocupada pelo robô, deverá ser impressa na tela a mensagem "Cheguei na posição XX" (completar com o nome e o índice do vetor de posições);

→ Todos os movimentos de uma posição de alinhamento até a outra deverá usar comando MOVECD passando pela posição de espera;

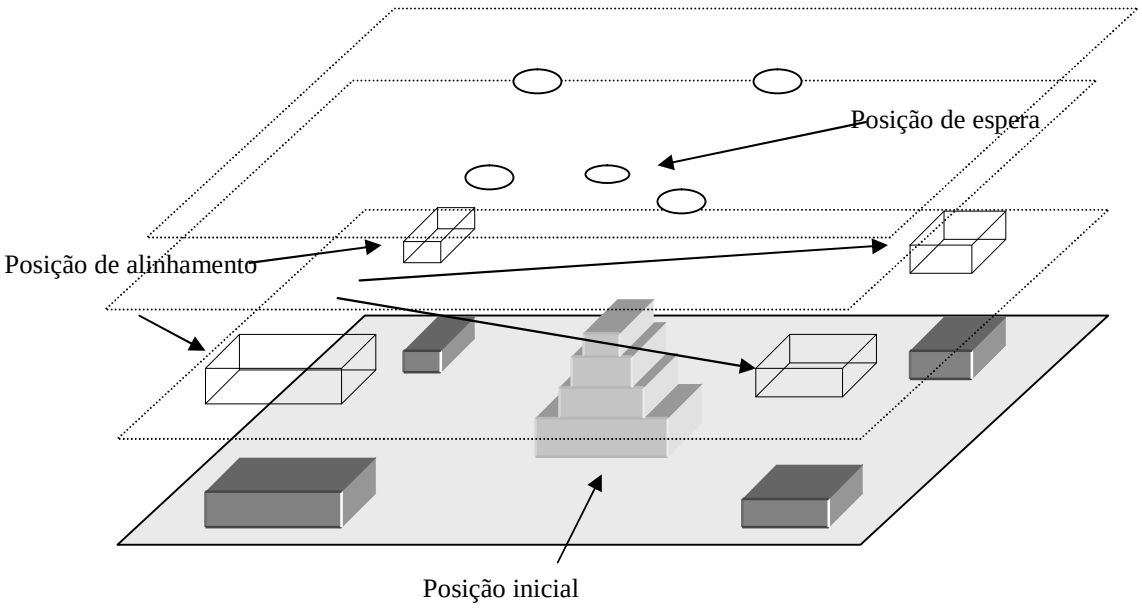
→ Após realizada a distribuição dos paralelepípedos, o robô deverá voltar à posição de espera e "bater palmas" três vez;

→ Por fim, o robô deverá remontar a pilha original, voltar a posição de espera e novamente "bater palmas" três vezes;

→ O programa deverá ter um menu com as seguintes opções:

- nº. 1 – repetir o movimento
- nº. 0 – sair do programa

→ Para a operação de montar e desmontar deve ser utilizado o comando FOR.



Capítulo 6
COMANDOS BÁSICOS DE PROGRAMAÇÃO E INTEGRAÇÃO COM A CÉLULA

Objetivos:

Apresentar os comandos básicos de programação.
Apresentar os conceitos de entradas (INPUTS) e saída (OUTPUTS).
Apresentar a diferença entre rodar um programa normal (RUN) e chamar uma subrotina (GOSUB)
Comandos de espera
Comandos de leitura

1. INTRODUÇÃO

Apresentar os comandos básicos de programação. Neste capítulo serão apresentados os seguintes comandos:

INPUT
OUTPUT
DELAY
WAIT
GOSUB
GET

2. COMANDOS BÁSICOS DE PROGRAMAÇÃO E DE INTEGRAÇÃO

2.1. INPUT/OUTPUT

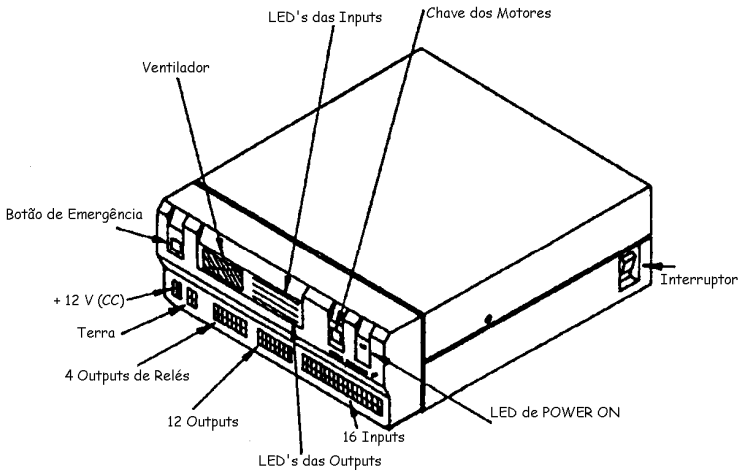
As ENTRADAS (INPUTS) e SAÍDAS (OUTPUTS) são canais de comunicação através dos quais o controlador se comunica com o mundo exterior. O meio deve estar pronto para receber comandos e enviar alguns sinais de controle.

Até neste momento, nos comunicamos com o controlador através do teclado e do teach pendant para acionarmos os eixos do robô. Entretanto, em um sistema automatizado, o robô deve ser capaz de receber sinais do seu ambiente, sem participação do usuário.

Exemplos:

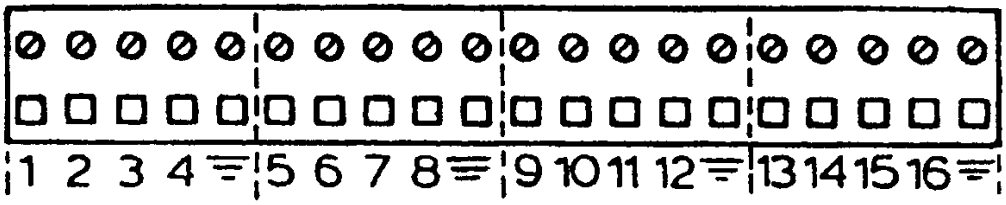
- Um robô deve colocar uma peça em um torno CNC para ser usinada. O torno manda um sinal para o controlador, indicando se ele está ou não usinando uma peça naquele momento.

- Dois robôs trabalham em conjunto em uma mesma tarefa. Quando um robô termina uma operação, ele envia um sinal para o controlador do outro robô, indicando que ele pode começar a próxima parte da tarefa.



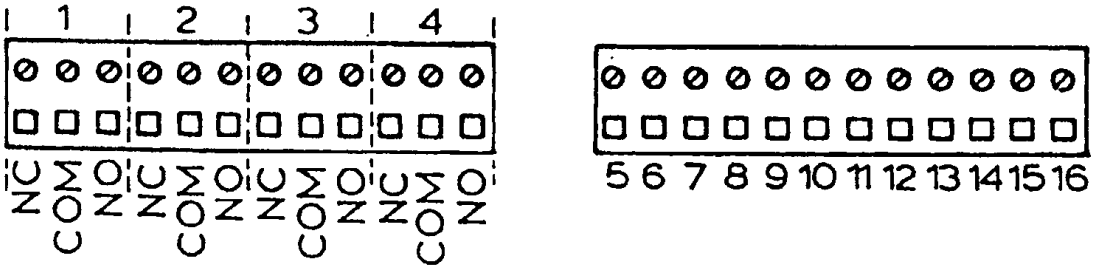
Nos controladores dos robôs temos 16 ENTRADAS (INPUTS) e 16 SAÍDAS (OUTPUTS).

Nas ENTRADAS (INPUTS) podemos conectar alguns tipos de sensores: sensores indutivos, sensores capacitivos, sensores magnéticos, sensores de toque, entre outros.



Os sensores são conectadas as ENTRADAS (INPUTS) através de fios que são inseridos nas portas de entrada. Podemos simular uma entrada (INPUT) externa, conectando uma entrada a um dos 4 "grounds" existentes no painel. O LED (Light Emitter Diode) correspondente aquela entrada deve acender.

Nos controladores existem 16 SAÍDAS (OUTPUTS) e os LED's correspondentes. As portas de saída consistem de 4 saídas a relés (números 1 a 4) e de 12 portas abertas (números 5 a 16). Normalmente as saídas são ligadas as válvulas pneumáticas ou outros dispositivos que poderão ser acionados na célula de manufatura.



2.1.1. Variável IN

As ENTRADAS (INPUTS) é o canal de comunicação através do qual o controlador recebe sinais do mundo exterior. Porém, necessitamos utilizar está informação recebida. Isto pode ser feito através do vetor IN [<n>], onde n é a posição do vetor. O conteúdo de cada posição deste vetor é do tipo binário. Ou seja, o valor do vetor pode ser 1(um) ou 0 (zero).

Podemos utilizar o conteúdo deste vetor das ENTRADAS para lógicas de decisão, repetição, controle, etc...

Exemplos:

DECISÃO

```
PROGRAM LUZ
IF IN[5]=1
        PRINTLN "ESTÁ LIGADO"
ENDIF
PRINTLN
END
```

REPETIÇÃO

```
PROGRAM ON
FOR I=1 TO 16
IF IN[I]=1
        PRINTLN "INPUT[" I "] ESTÁ ON"
ENDIF
ENDFOR
END
```

CONTROLE

```
PROGRAM PINO
IF IN[11]=1
SET OUT[12]=1
ENDIF
```

```
END
```

2.1.2. Variável OUT

As SAÍDAS (OUPUTS) é o canal de comunicação através do qual o controlador poderá acionar algum dispositivo do mundo exterior. Isto pode ser feito através do vetor OUT [<n>], onde n é a posição do vetor. O conteúdo de cada posição deste vetor é do tipo binário. Ou seja, o valor do vetor pode ser 1(um) ou 0 (zero). As SAÍDAS precisam receber um conteúdo, este deve ser binário, para que possam interagir com o meio. Isso deve ser feito pelo comando SET. Logo a sintaxe deve ser a seguinte:

```
SET OUT[n]=1  //neste caso a saída estaria no estado ON
SET OUT[n]=0  //neste caso a saída estaria no estado OFF
```

Podemos utilizar o conteúdo deste vetor das SAÍDAS para lógicas de decisão, repetição, controle, etc... Porém, isso não é muito usual.

Exemplo:

```
PROGRAM OUT01
LABEL 1
FOR I=1 TO 16
        SET OUT[I]=1
ENDFOR
PRINTLN
GOTO 1
END
```

2.2. Comando DELAY

Muitas vezes num programa necessitamos utilizar o TEMPO para aguardar algum acontecimento ou para elaborar uma lógica.

Na linguagem ACL existe o comando chamado DELAY que faz o programa parar por um determinado tempo em centésimos de segundo. O comando possui a seguinte sintaxe:

```
DELAY <n>
```

onde n é o valor do tempo em centésimos de segundo. Ou seja, se n for igual a 300, o programa pára por 3 segundos.

Exemplo:

```
DELAY 800
```

O programa pára durante 800 centésimos de segundo, ou, 8 segundos.

2.3. Comando WAIT

Muitas vezes num programa necessitamos aguardar que alguma coisa aconteça para dar prosseguimento ao programa. Isso é aplicado principalmente através do uso das ENTRADAS E SAÍDAS do controlador, ou, pela utilização de variáveis. Este comando é o comando WAIT. A sintaxe deste comando segue abaixo:

WAIT <VAR1> <COND> <VAR2>

onde <COND> pode ser operadores relacionais (<, >, >=, <=, =)

Exemplo:

WAIT IN[5] = 1

O programa aguarda até que a entrada 5 seja igual a 1, ou, a entrada estaja no estado ON.

2.4. Comando GOSUB

Para rodar um programa é utilizado o comando RUN. Entretanto, depedendo da lógica podemos ter um programa com várias linhas. Todos sabemos que é complicado de debugar (encontrar um erro) um programa muito extenso, ou, adicionar mais linhas a este programa. Por este motivo a existência deste comando GOSUB.

Com o comando GOSUB você pode dividir o seu programa em pequenos programas. Estes programas são chamados agora de subrotinas. Estes programas precisam ser armazenados no controlador antes do programa principal, ou seja, é necessário fazer o download para o controlador, antes de chamá-los no programa principal.

O programa principal vai executando normalmente linha após linha, quando encontra o comando GOSUB, o programa principal pára e neste momento começa a executar o programa chamado pelo comando GOSUB. Logo, a sintaxe para este comando é o seguinte:

GOSUB <nome do programa>

Exemplo:

```
.
.
SET Z = 10
GOSUB PROG1
MOVED P[3]
.
.
```

No programa acima podemos ver que após atribuir o valor a variável Z, o próximo comando é o GOSUB. Neste momento o programa principal pára de executar o programa principal e executa o

programa chamado pelo comando GOSUB, neste exemplo o PROG1. O programa PROG1 é inteiramente executado e quando o seu END é atingido, o controle retorna para o programa principal, o qual continua com a execução do comando MOVED e seguintes.

2.5. Comando GET

Um novo comando para a entrada de dados. Quando o programa encontra um comando GET, ele para e aguarda um caracter ser pressionado no teclado. A variável é atribuída o valor ASCII do caracter que é pressionado. O comando get possui a seguinte sintaxe:

GET <var>
GET "<mensagem ao usuário>"<var>

onde <var> é uma variável definida. O comando GET pode ser precedido por uma mensagem ao usuário. A mensagem deve vir entre aspas, indicando por exemplo ao usuário a inserção de um caracter via teclado.

Exemplos:

```
PROGRAM PROG1
DEFINE VP
PRINTLN "SELECT PROGRAM: P Q R"
GET VP      (VP é a variável)
IF VP=80     (80 é o ASCII para P)
        RUN P
ENDIF
IF VP=81     (81 é o ASCII para Q)
        RUN Q
ENDIF
IF VP=82     (82 é o ASCII para R)
        RUN R
ENDIF
END
```

3. EXERCÍCIOS

a) ENCONTRAR PALET (Apresentar e Entregar)

Algumas definições:

Templates são bandejas plásticas que podem comportar vários tipos de *peças*. Eles permitem que *as peças* sejam transportadas sobre a esteira num *pallet*. Um *template* contém uma matriz de furos nos quais pinos são colocados conforme as dimensões das peças. Cada arranjo de pinos define um tipo único de *template*. Cada *peça* pode somente ser comportado pelo *template* que tenha sido determinado para ele. Esta configuração dos *templates* permite um fácil manuseio do robô de cima ou de frente. Um código de barra opcional é colado sobre um dos lados do templates mostrando os códigos de identificação (ID) do produto ou matéria-prima. Há um leitor de códigos de barras próximo da esteira verificando a identidade de cada *template* inserido ou removido no ASRS.

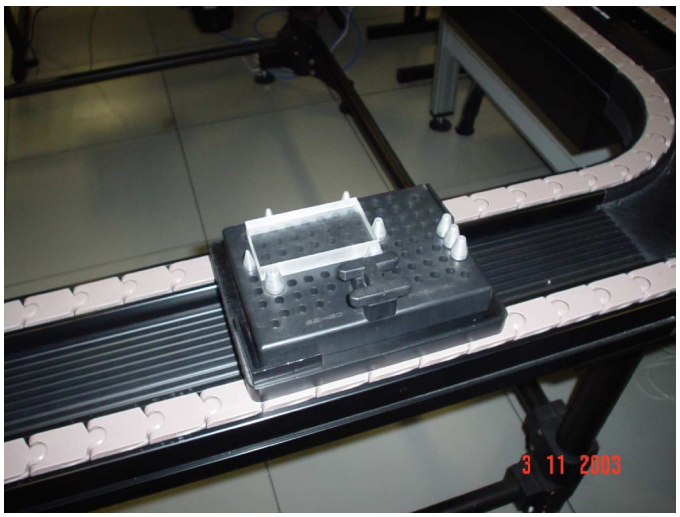


Figura 1. Imagem do template sobre o pallet.

Um *pallet* é uma bandeja que viaja sobre a esteira e é feito para carregar um *template*. Para transportar uma peça para outra estação, um *template* carrega a peça que é colocada pelo robô sobre um *pallet* da esteira. A esteira transporta os *pallets* de estação por estação. Cada *pallet* tem um ID # no qual é magneticamente decodificado em uma barra embaixo do *pallet* por meios de sensores postos em cada estação da célula.

Decodificação dos pallets é feita através da conversão do valor binário lido pelo decodificador em número decimal. Cada estação possui 5 sensores magnéticos. Sendo que um é para verificar se o pallet está no local correto, este sensor é o INPLACE. Os outros quatro sensores são para identificar o número do pallet, estes números são binários e são compostos da seguinte maneira: ID3, ID2, ID1 e ID0. Para converter para decimal pasta multiplicar o valor de ID3 por 8, ID2 por 4, ID1 por 2 e ID0 por 1 e somar todos estes valores para saber o respectivo valor em decimal.

A esteira "*conveyor*" CIM-ACL carrega *pallets* em circuito contínuo de estação para estação. Em normal, na célula de operação, cada pallet é parado brevemente quando ele chega numa estação somente para a leitura do código magnético. Se a estação determina que o *pallet* é necessário nesta estação, este pallet permanece nesta estação até que o robô retire o template. Enquanto um *pallet* é parado, a esteira continua a transportar outros *pallets*, os quais são movimentados entre as estações. Cada estação CIM-ACL tem sua própria estação de esteira no qual o *pallet* é parado. A parada dos pallets é realizado por pistões pneumáticos de parada, usando entrada dos sensores de detecção de *pallets* localizados na esteira da estação. Se o *peça* é carregada por um *pallet* e não requer processamento na estação, o *pallet* permanecera sobre a esteira. Por outro lado, se processado, o robô remove o template contendo o *part* do *pallet* e o coloca sobre o buffer da estação. O *pallet* vazio é então liberado e pode continuar sobre a esteira, pronto para colocar outro *template*.

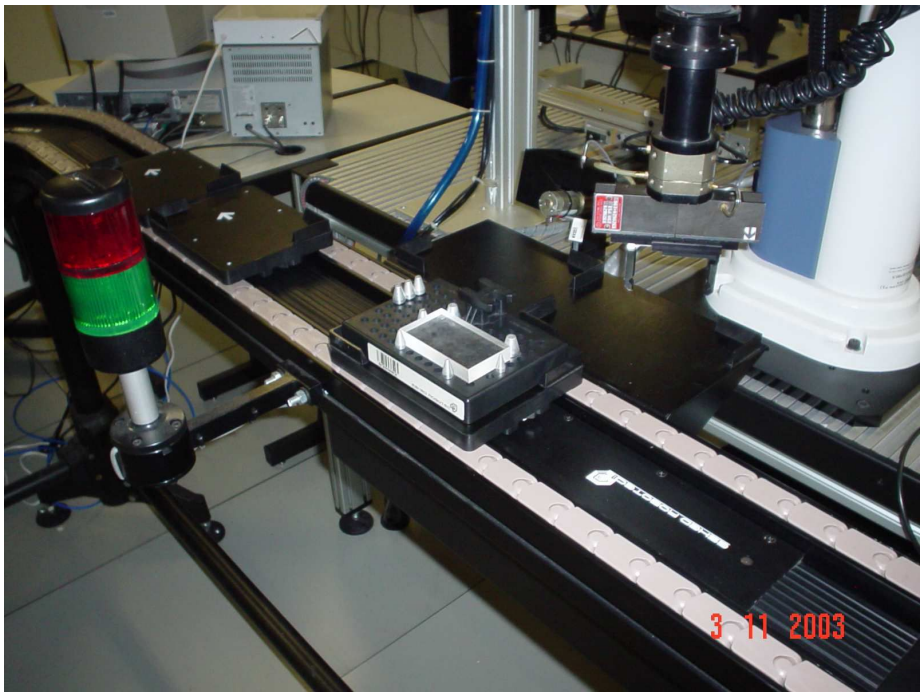


Figura 2. Imagem da estação 5 com um pallet parado no local de leitura. Local este onde é acionado um pistão para parar o pallet e onde o robô vai pegar o template.

Desenvolver 5 pequenos programas:

- Programa que serve de interface com o usuário, ou seja, um programa MENU. Este deverá possuir como opções apenas letras, e não números. As opções são: Sair, Encontrar pallets, imprimir os estados das entradas, qualquer outro valor deverá acusar erro e voltar para o menu.
- Programa que encontrará o pallet que o usuário pedir.
- Programa que irá retirar o template do pallet, colocar no buffer.
- Programa que irá retirar o template do buffer e colocará sobre o pallet indicado pelo usuário através do programa 2.
- Programa irá imprimir na Tela de 1 em 1 min o tempo de utilização do programa.

Apêndice I
MENSAGENS DE ERRO

Objetivos:

Conhecer os erros mais comuns nos controladores ACL.
Permitir que os alunos identifiquem os erros.

1. MENSAGENS DO TEACH PEDANT

A seguir estão listadas as principais mensagens que aparecem no visor de LCD do teach pendant durante a operação em modo manual causado por alguma instrução incorreta. Esta listado também as possíveis soluções para cada erro

ALL PROGRAMS ABORTED

(205) All programs aborted

O Botão de Emergência está precionado ou o TP está fora da casa sem o botão do modo empunhado pressionado.

Libere o botão de emergência, no caso de movimentação volte a habilitar os eixos do robô (CON).

ARITHMETIC OVERFLOW

(302) Arithmetic overflow (or division by zero)

O resultado de uma operação matemática está for a do range ou é invalido.

AXIS DISABLED

(304) Axis disabled.

Os eixos do robô estão desabilitados (COFF).

Volte a habilitar os eixos do robô (CON).

BAD AXIS

(310) Invalid axis.

O eixo não é do grupo selecionado ou o mesmo não está configurado.

Selecione outro eixo ou outro grupo.

BAD GROUP

(321) MOVES/SPLINE not allowed for a single axis group.

O commando (MOVES) não pode ser executado com um único eixo no grupo.

BAD POS n

(312) Invalid position coordinate values.

Você tentou usar uma posição inválida.

BAD XYZ POSITION

(317) Invalid cartesian position pos.

A posição não pode ser gravada ou alcançado porque suas coordenadas de XYZ estão fora do sistema XYZ

CONTROL DISABLED

(72) CONTROL DISABLED.

Os eixos estão desabilitados ou motores não foram ligados. Verificar o botão branco com o Led verde no painel frontal do controlador e habilitar os eixos (CON)

CONTROL ENABLED

(73) CONTROL ENABLED.

Os motores estão habilitados para funcionar.

EMERGENCY

(148) To perform action - release emergency button.

O Botão de emergência está pressionado.
Libere o Botão e volte a habilitar os eixos do robô (CON).

HANDLE PUSHED!

(167) Pressure was applied on handle before switch activation

(168) or payload is too heavy for Direct Teach

Ao usar o modo direto de movimentação, você deve primeiramente passar o TP para o modo AUTO. Somente após você pode começar a pressionar as teclas a fim mover o robô.

HOME FAIL n

(51) *** HOME FAILURE AXIS n.

O processo de referenciamento do robô falou no eixo "n".

HOME NOT DONE

(301) Group/axis has not been homed.

Você tentou mover o braço para posições gravadas, ou gravar uma posição, antes de referenciar os eixos (homes) do robô.

Referencie o robô e tente novamente executar o comando atual.

HOMING

(31) WAIT !! homing...

Aguarde o robô está sendo referenciado.

HOMING COMPLETE

(32) Homing complete

O robô já terminou o seu referenciamento.

IMPACT AX n

(53) *** IMPACT PROTECTION axis n

O controlador detectou um impacto. O sistema abortou todos os movimentos desse grupo de eixos, e incapacitou todos os movimentos desse grupo.

Volte a habilitar os eixos deste grupo (CON).

INDEX RANGE

(309) Index value - out of range.

O valor do índice esta fora da range.

LIMIT AX n

(33) *** UPPER LIMIT AXIS n

(34) *** LOWER LIMIT AXIS n

O encoder do eixo alcançou o seu valor máximo ou mínimo permitido.

Habilite novamente as juntas e movimente no sentido contrário o eixo em questão.

I SERVO ERROR

(212) Error must be reset.

O botão dos motores está desligado.

Ligue o botão no painel frontal do controlador.

MOTORS OFF

(319) Motor power switch is off.

O botão dos motores está desligado.

Ligue o botão no painel frontal do controlador.

POS NOT DEF/ARRAY

(145) position is not defined or is not a vector.

A posição não foi definida ou este não é o nome do vetor de posição

Verifique se sua variável de posição foi definida corretamente.

POS NOT FOUND

(110) Position undefined, not recorded, or for other group.

A posição que você tentou usar não foi definida, não foi gravada, ou não foi definida para um outro grupo de eixos do robô.

POS NOT RECORDED

(74) (303) No coordinate values for position.

A posição não foi gravada.

2. MENSAGENS DO CONTROLADOR NO TERMINAL

A variável ERROR é uma variável do sistema ACL. Quando um erro ocorre no sistema durante uma execução do programa, a variável ERROR é atribuída ao correspondente valor e ao tipo específico de erro, e uma mensagem de erro será mostrada. A seguir, uma descrição dos erros do ACL que você pode encontrar.

53 IMPACT PROTECTION AXIS [n]

Eixo [n] no está conectado, ou falhou, ou colidiu contra um obstáculo.

54 OUT OF RANGE AXIS [n]

Eixo [n] foi além de seu limite e está fora de alcance.

55 THERMIC OVERLOAD AXIS [n]

Eixo [n] está sobrecarregado e/ou sobreaquecido; ou o parametro térmico do eixo está setado errado.

302 ARITHMETIC OVERFLOW AT LINE [n]

O resultado de uma operação aritmética está além da capacidade do controlador.

303 NO POSITION ASSIGNED TO POINT [pos]

O atual comando no pode ser executado uma vez que nenhum valor de localização foi atribuído a posição especificada.

304 AXIS DISABLED

Você tentou mover um eixo o qual no estava desabilitado. Use o comando CON para reativar o servo controle do eixo.

305 TOO DEEP NESTING

Muitas sub-rotinas - comandos GOSUB - estão "aninhados" dentro de outras.

306 INVALID PROGRAM

O programa no pode ser executado, devido a uma lógica ou sintaxe errada. Por exemplo, o programa contém um comando IF sem o correspondente comando ENDIF.

309 INDEX OUT OF RANGE

Você tentou usar um índice de valor o qual está além da gama definida de variáveis ou vetores de posição.

310 BAD AXIS

Muitas causas possíveis. Por exemplo, o eixo no está no grupo definido, o eixo no está definido, o eixo está fora de limite, etc.

311 LOOPING RELATIVE CHAIN OR DEPTH EXCEEDED 32 POINTS

Você pode ter ligado uma posição relativa em um loop inválido ou infinito. Ou você usou mais de 32 pontos relativos ligados.

312 BAD POINT COORDINATE [pos]

Você tentou usar uma posição inválida. Por exemplo, a posição relativa que você definiu está fora do limite do eixo.

315 INCOMPATIBLE POINTS

Você tentou copiar um ponto (SETP) a partir de grupo de eixo para outro grupo de eixo.

316 NO GRIPPER CONFIGURATION

O comando gripper no pode ser executado uma vez que o gripper no foi configurado.

317 BAD CARTESIAN POINT [pos]

O atual comando no pode ser executado devido a um valor Cartesiano inválido da posição especificada

Apêndice II
LISTA DE INPUTS/OUTPUTS DE CADA ESTAÇÃO

Objetivos:

Conhecer todas as entradas e saídas conectadas aos robôs do Laboratório CIM.

1. ESTAÇÃO 1 – ASRS – ALMOXARIFADO

ENTRADAS		
NOME DA ENTRADA	NÚMERO DA ENTRADA	FUNÇÃO
I_SafetyFence	9	Sinal de segurança (ligado = ativo)
IN PLACE	10	Sensor IN PLACE da esteira
ID0	11	Sensor ID0 da esteira
ID1	12	Sensor ID1 da esteira
ID2	13	Sensor ID2 da esteira
ID3	14	Sensor ID3 da esteira
SAÍDAS		
NOME DA SAÍDA	NÚMERO DA SAÍDA	FUNÇÃO
Gripper	3	Saída da garra
O_Light	4	Saída da luz
T1	11	Trava o pallet na estação
L1	12	Liga a lâmpada verde
L2	13	Liga a lâmpada vermelha

Funcionamento do leitor de lódigo de barras:

i. Posicionar o template na posição correta.
ii. Executar o programa READ (GOSUB READ)
iii. Valor do tipo do template será armazenado na posição do vetor REAL[3]
iv. Valor do número do template será armazenado nas posições do vetor REAL[4 .. 7]

2. ESTAÇÃO 2 – ER9

ENTRADAS		
NOME DA ENTRADA	NÚMERO DA ENTRADA	FUNÇÃO
I_CycleStatus	1	Status de ciclo (set = máquina esperando)
I_ChuckClosed	2	Castanha Fechada
I_ChuckOpened	3	Castanha Aberta
I_DoorClosed	4	Porta Fechada
I_DoorOpened	5	Porta Aberta
I_MachAlarm	6	Alarme setado na máquina
I_NoPartChuck	7	Castanha fechada, mas sem peça
I_MachReady	8	Máquina pronta para operar ***
I_RefAuto	9	Referência de status automática (Set=Ref)
ID0	10	Sensor ID0 da esteira
ID1	11	Sensor ID1 da esteira
ID2	12	Sensor ID2 da esteira
ID3	13	Sensor ID3 da esteira
IN PLACE	14	Sensor IN PLACE da esteira
SC	15	Sensor auxiliar capacitivo
SI	16	Sensor auxiliar indutivo
SAÍDAS		
NOME DA SAÍDA	NÚMERO DA SAÍDA	FUNÇÃO
Gripper	1	Saída da garra
O_CycleStart	4	Inicia a máquina
O_CloseChuck	5	Fecha castanha
O_OpenChuck	6	Abre castanha
O_CloseDoor	7	Fecha a porta
O_OpenDoor	8	Abre a porta
O_RefAxes	9	Eixos de referencia
O_RefAuto	10	Troca entre referencia automatica
O_AuxOn	11	Auxiliares ligados
T1	12	Trava o pallet na estação
L1	13	Liga a lâmpada verde
L2	14	Liga a lâmpada vermelha

*** Aux ligado, porta aberta, ponto de referencia está ativo, castanha aberta, espera aberta, modo automático

3. ESTAÇÃO 3 – ER7

ENTRADAS		
NOME DA ENTRADA	**NÚMERO DA ENTRADA**	**FUNÇÃO**
I_DoorClosed	1	**Porta fechada (se setado)**
I_ViceClosed	2	**Morsa fechada (se setado)**
I_CycleStatus	3	**Status de ciclo (set = máquina ocupada)**
ID0	**8**	**Sensor ID0 da esteira**
ID1	**9**	**Sensor ID1 da esteira**
ID2	**10**	**Sensor ID2 da esteira**
ID3	**11**	**Sensor ID3 da esteira**
IN PLACE	**12**	**Sensor IN PLACE da esteira**
SI	**15**	**Sensor auxiliar indutivo**
SC	**16**	**Sensor auxiliar capacitivo**
SAÍDAS		
NOME DA SAÍDA	**NÚMERO DA SAÍDA**	**FUNÇÃO**
O_SwitchDoor	5	**Estado da porta**
O_CloseVice	6	**Fecha a morsa**
O_OpenVice	7	**Abre a morsa**
O_LoadMode	8	**Coloca a máquina no modo de carga**
O_CycleStar	9	**Inicia a máquina**
O_AutoMode	10	**Coloca a máquina no modo automático**
T1	11	**Trava o pallet na estação**
L1	--	**Não implementado**
L2	--	**Não implementado**

Observações:

O ER7 tem uma garra elétrica, sendo possível controlar o quanto se quer abrir ou fechar. O comando para abrir e fechar a garra possui a seguinte sintaxe:

JAW <valor>

onde <valor> é o percentual que a garra deverá abrir.

Exemplo:
JAW 30
Neste exemplo a garra irá abrir 30%.

4. ESTAÇÃO 4 – Performer MK3

ENTRADAS		
NOME DA ENTRADA	**NÚMERO DA ENTRADA**	**FUNÇÃO**
ID0	9	**Sensor ID0 da esteira**
ID1	10	**Sensor ID1 da esteira**
ID2	11	**Sensor ID2 da esteira**
ID3	12	**Sensor ID3 da esteira**
IN PLACE	13	**Sensor IN PLACE da esteira**
SC	14	**Sensor auxiliar capacitivo**
SI	15	**Sensor auxiliar indutivo**
SAÍDAS		
NOME DA SAÍDA	**NÚMERO DA SAÍDA**	**FUNÇÃO**
Caliper	2	**1 = Abre paquimetro**
		0 = Fecha paquimetro
T1	11	**Trava o pallet na estação**
L1	12	**Liga a lâmpada verde**
L2	13	**Liga a lâmpada vermelha**

Funcionamento Paquímetro e altímetro

Execute o programa INITC uma única vez, fora do seu programa, **RUN INITC**

ALTÍMETRO

i. Execute o programa ***HEIGH*** para baixar o medidor.
ii. Espere o medidor baixar até a peça.
iii. Verifique o valor da medida na variável ***HGLNR***.
iv. Execute o programa ***UP*** para subir o medidor.

PAQUÍMETRO

i. Coloque o paquímetro no local da medida
ii. Execute o programa chamado ***CALP.***
iii. O valor da medida estará armazenado na variável ***CLNUM***

OBSERVAÇÕES:

Executar os programas com o comando ***GOSUB***

5. ESTAÇÃO 5 – SCARA ER14

ENTRADAS		
NOME DA ENTRADA	**NÚMERO DA ENTRADA**	**FUNÇÃO**
I_BallSensor	1	**Sensor do alimentador de esferas**
ID0	9	**Sensor ID0 da esteira**
ID1	10	**Sensor ID1 da esteira**
ID2	11	**Sensor ID2 da esteira**
ID3	12	**Sensor ID3 da esteira**
IN PLACE	13	**Sensor IN PLACE da esteira**
SC	7	**Sensor auxiliar capacitivo**
SI	8	**Sensor auxiliar indutivo**
SAÍDAS		
NOME DA SAÍDA	**NÚMERO DA SAÍDA**	**FUNÇÃO**
O_BallFeeder	1	**Alimentador de bolinhas**
O_LIGHT	2	**Saída da Luz**
O_ToolChange	3	**Troca de ferramenta**
O_ScrewDriver	4	**Parafusadeira**
O_OpenGripper	5	**Abre a garra**
O_CloseGripper	6	**Fecha a garra**
O_Vacum	6	**Aciona o vácuo**
T1	11	**Trava o pallet na estação**
L1	--	**Não implementado**
L2	--	**Não implementado**

Observações:

ABRIR GARRA
RUN OGRIP **(no terminal)**
GOSUB OGRIP **(no programa)**

FECHAR GARRA
RUN CGRIP **(no terminal)**
GOSUB CGRIP **(no programa)**

VÁCUO
ACIONAR: *SET OUT[6]=1*
DESACIONAR: *SET OUT[6]=0*

PARAFUSADEIRA
ACIONAR: *SET OUT[4]=1*
DESACIONAR: *SET OUT[4]=0*

ESFERAS
ACIONAR: *SET OUT[1]=1*
DESACIONAR: *SET OUT[1]=0*

Apêndice III
DADOS ADICIONAIS

Objetivos:

Conhecer os valores que o comando GET recebe ao digitar um dígito.

1. TABELA ASCII

Dec	Char	Dec	Char	Dec	Char	Dec	Char	Dec	Char	Dec	Char
33	!	49	1	65	A	81	Q	97	a	113	q
34	"	50	2	66	B	82	R	98	b	114	r
35	#	51	3	67	C	83	S	99	c	115	s
36	$	52	4	68	D	84	T	100	d	116	t
37	%	53	5	69	E	85	U	101	e	117	u
38	&	54	6	70	F	86	V	102	f	118	v
39	'	55	7	71	G	87	W	103	g	119	w
40	(	56	8	72	H	88	X	104	h	120	x
41	)	57	9	73	I	89	Y	105	i	121	y
42	*	58	:	74	J	90	Z	106	j	122	z
43	+	59	;	75	K	91	[	107	k	123	{
44	,	60	<	76	L	92	\	108	l	124	|
45	-	61	=	77	M	93	]	109	m	125	}
46	.	62	>	78	N	94	^	110	n	126	~
47	/	63	?	79	O	95	_	111	o	127	_
48	0	64	@	80	P	96	`	112	p		

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
128	80	Ç	160	A0	á	192	C0	└	224	E0	α
129	81	ü	161	A1	í	193	C1	┴	225	E1	ß
130	82	é	162	A2	ó	194	C2	┬	226	E2	Γ
131	83	â	163	A3	ú	195	C3	├	227	E3	π
132	84	ä	164	A4	ñ	196	C4	─	228	E4	Σ
133	85	à	165	A5	Ñ	197	C5	┼	229	E5	σ
134	86	å	166	A6	ª	198	C6	╞	230	E6	µ
135	87	ç	167	A7	º	199	C7	╟	231	E7	τ
136	88	ê	168	A8	¿	200	C8	╚	232	E8	Φ
137	89	ë	169	A9	⌐	201	C9	╔	233	E9	Θ
138	8A	è	170	AA	¬	202	CA	╩	234	EA	Ω
139	8B	ï	171	AB	½	203	CB	╦	235	EB	δ
140	8C	î	172	AC	¼	204	CC	╠	236	EC	∞
141	8D	ì	173	AD	¡	205	CD	═	237	ED	ø
142	8E	Ä	174	AE	«	206	CE	╬	238	EE	ε
143	8F	Å	175	AF	»	207	CF	╧	239	EF	∩
144	90	É	176	B0	░	208	D0	╨	240	F0	≡
145	91	æ	177	B1	▒	209	D1	╤	241	F1	±
146	92	Æ	178	B2	▓	210	D2	╥	242	F2	≥
147	93	ô	179	B3	│	211	D3	╙	243	F3	≤
148	94	ö	180	B4	┤	212	D4	╘	244	F4	⌠
149	95	ò	181	B5	╡	213	D5	╒	245	F5	⌡
150	96	û	182	B6	╢	214	D6	╓	246	F6	÷
151	97	ù	183	B7	╖	215	D7	╫	247	F7	≈
152	98	ÿ	184	B8	╕	216	D8	╪	248	F8	°
153	99	Ö	185	B9	╣	217	D9	┘	249	F9	∙
154	9A	Ü	186	BA	║	218	DA	┌	250	FA	·
155	9B	¢	187	BB	╗	219	DB	█	251	FB	√
156	9C	£	188	BC	╝	220	DC	▄	252	FC	ⁿ
157	9D	¥	189	BD	╜	221	DD	▌	253	FD	²
158	9E	₧	190	BE	╛	222	DE	▐	254	FE	■
159	9F	ƒ	191	BF	┐	223	DF	▀	255	FF	

Apêndice IV
COMUNICAÇÃO ENTRE AS ESTAÇÕES

Objetivos:

Apresentar a comunicação entre duas estações.
Enviar e receber mensagens numa estação

1. COMUNICAÇÃO ENTRE AS ESTAÇÕES

1.1. Como enviar uma mensagem para outra estação?

Para enviar uma mensagem são necessárias três operações:

i. Dar um valor para variável **PALET** (SET PALET= *número*). **Está variável é global, logo, não é necessário definir está variável.** Esse valor indicará qual o valor do pallet deverá ser parado na estação que receberá essa mensagem.
ii. Dar um valor para variável **EST** (SET EST = *número*). **Está variável é global, logo, não é necessário definir está variável.** Esse valor indicará para qual estação será enviada a mensagem.
iii. Executar o programa **ENVIO** (GOSUB ENVIO). Envia a mensagem com o número do pallet para a estação desejada.

1.2. Como receber a mensagem em uma estação?

Para receber uma mensagem é necessário verificar em que momento o programa principal poderá parar para receber a mensagem.

Nesse local, executar o programa **RECEB** (GOSUB RECEB). O valor da variável **PALLET** será armazenado na variável **PLT. A variável PLT é global, logo, não é necessário definir está variável.**

1.3. Observações importantes

i. Ambos os programas (ENVIO e RECEB), farão parte de um programa principal que irá executá-lo.
ii. **NUNCA DECLARE** as variáveis **PLT, PALET e EST**, SOMENTE **USE.**

Exemplo:

O exemplo a seguir mostra como enviar uma mensagem da estação 1 e receber na estação 5.

Na estação 1:

PROGRAM **nome_prog**
.....

```
.....
PRINTLN
READ "DIGITE O NUMERO DO PALLET: " PALET
PRINTLN
READ "DIGITE O NUMERO DA ESTACAO: " EST
PRINTLN
GOSUB ENVIO  // ENVIA OS VALORES
.....
.....
END
```

Na estação 5:

```
PROGRAM nome_prog
.....
.....
PRINTLN
GOSUB RECEB // ESPERA RECEBER OS VALORES
PRINTLN "O VALOR DO PALLET EH" PLT
IF PLT=8
SET OUT[11]=1
.....
.....
END
```

Apêndice V
OUTROS COMANDOS

Objetivos:

Apresentar outros comandos existentes na linguagem ACL.

1. MANUAL MODE

Existem comandos ACL equivalentes aos do TP para movimentar o robô. Entretanto, se existe um TP disponível, é mais conveniente usá-lo para movimentar o robô.

Para ativar o controle manual do robô, seja no modo **DIRECT** ou no modo **MANUAL MODE**. Você deve seguir os seguintes comandos:

PRESSIONE **~ (ou Alt M)**

Na tela do seu computador irá aparecer a mensagem "MANUAL MODE" e o sistema de coordenadas corrente, JOINT ou XYZ, dependendo em que modo ele estava:

MANUAL MODE!
>
JOINT MODE

Podemos trocar para JOINT da seguinte maneira:

PRESSIONE **J**

Aparecerá na tela:

JOINT MODE

Para trocar para XYZ:

PRESSIONE **X**

Aparecerá na tela:

XYZ MODE

Você pode usar algumas teclas para movimentar o robô:

JOINTS		
1/Q	move a base (eixo 1)	direita/esquerda
2/W	move o ombro (eixo 2)	acima/abaixo
3/E	move o cotovelo (eixo 3)	acima/abaixo
4/R	move o punho-pitch (eixo 4)	acima/abaixo
5/T	move o punho-roll (eixo 5)	direita/esquerda

6/Y	controla a garra (eixo 6)	abre/fecha

Notas:

1. As teclas 6 e Y funcionam de maneira diferente das demais. Pressionando Y uma vez abre a garra completamente. Pressionando 6 uma vez a garra fecha completamente.

2. As funções das teclas 5/T e 6/Y são as mesmas, tanto em JOINT como em XYZ.

XYZ		
1/Q	move na direção X	direita/esquerda
2/W	move na direção Y	acima/abaixo
3/E	move na direção Z	acima/abaixo
4/R	muda o valor do pitch mantendo a extremidade da garra fixa no espaço	acima/abaixo

Os comandos em JOINTS, dados abaixo, movem os eixos 7 a 11. Tais eixos controlam acessórios periféricos, tais como uma mesa linear (slide base) ou uma mesa giratória.

ACESSÓRIOS PERIFÉRICOS		
7/U	move o eixo 7	+/-
8/I	move o eixo 8	+/-
9/O	move o eixo 9	+/-
0/P	move o eixo 10	+/-
-/[	move o eixo 11	+/-

OUTROS COMANDOS:

S <n> define a velocidade do robô speed (n = % da velocidade máxima)
C/F conecta/desconecta o controle

Para sair do modo MANUAL:

PRESSIONE **~ (Alt + M)**

Aparecerá na tela:

EXIT MANUAL MODE ...

e você retornará exatamente onde estava quando entrou no modo Manual.

2. COMANDO CONTINUE/SUSPEND

Os comandos **CONTINUE** e **SUSPEND** permitem que você possa, respectivamente, continuar ou suspender a execução de um programa. Assim, você pode continuar o programa PROG, que havia sido previamente suspenso usando o comando. A sintaxe deste comando segue abaixo:

CONTINUE PROG

Analogamente, você pode suspender a execução de um programa através da seguinte sintaxe:

SUSPEND PROG

Tanto CONTINUE como SUSPEND podem ser usados nos modos direto ou edição.

3. COMANDOS DE EDIÇÃO DE PROGRAMAS EM ACL

O controlador possui um editor de texto que permite ao usuário editar programas utilizando comandos do ACL. Para chamar o editor, execute o seguinte comando: Edit nome <ENTER>. Caso nenhum programa tenha sido criado anteriormente com este nome, deverá aparecer a mensagem do tipo :

"NOME NEW PROGRAM"
"DO Y OU WANT TO CREATE THAT PROGRAM (Y /N) ?"

Responda "Y" para criar o programa. Após este comando o usuário entra no modo de edição e todos os comandos são do tipo indireto. Se o arquivo já existir, o editor abre o mesmo, posicionando o cursor sobre a primeira linha. Em qualquer um dos casos, surge a mensagem dada a seguir :

PROGRAM NOME
****
30 ? //onde "30" é o número da primeira linha do programa. Esse número depende dos programas já criados.

O editor implementado dentro do controlador é um editor de linha, que possui poucos recursos de edição. Os comandos estão descritos na tabela a seguir:

COMANDO	FUNÇÃO
<ENTER>	MOSTRA PRÓXIMA LINHA
P <ENTER>	MOVE PARA LINHA ANTERIOR
S <ENTER>	MOVE PARA O INÍCIO DO PROGRAMA
S N <ENTER>	MOVE PARA A LINHA N
DEI <ENTER>	DELETA A LINHA ANTERIOR
L N1 N2	LISTA AS LINHAS N1 ATÉ N2
*	COMENTÁRIO

Crie um programa simples utilizando os comandos descritos anteriormente. Como exemplo, edite o programa abaixo, lembrando que todos os comandos são realizados no prompt do software:

EDIT FIRST <ENTER>

PROGRAM FIRST
****
***30** PRINTLN "PRIMEIRO PROGRAMA EM ACL "*
***31** PRINTLN "UTILIZANDO O COMANDO PRINTLN"*

```
32 PRINTLN "LABORATORIO DE ROBOTICA"
33 EXIT
```

O comando exit encerra a edição do programa ACL. Ao encerrar o modo de edição o ACL verifica a validade do programa, emitindo uma mensagem do tipo "**NOME IS VALID**" se o programa estiver correto. Após isto, o ACL é novamente colocado no modo de comando direto.

4. COMANDO TRIGGER

Ideal para sensoriamento de sinais cujo "TIMING" é indefinido ou imprevisível. Se uma aplicação exige interrupções repetidas dos sensores, o TRIGGER deve surgir antes de cada interrupção esperada.

SINTAXE: TRIGGER <PROG> BY {IN/OUT} <n> [0/1]

Exemplos:

TRIGGER FURA BY OUT 8 1//EXECUTA O PROGRAMA FURA QUANDO A SAÍDA 8 ESTIVER ON

TRIGGER PEGA BY IN 5 //EXECUTA O PROGRAMA PEGA QUANDO A ENTRADA 5 MUDAR DE estado.

5. COMANDO ENABLE/DISABLE

Os comandos ENABLE e DISABLE são comandos de modo direto que, respectivamente, reconectam e desconectam a I/O física (hardware) do I/O lógica (software).

Exemplos:

ENABLE IN 8 //RECONECTA A ENTRADA 8 AO SOFTWARE
ENABLE OUT 12 //RECONECTA A SAÍDA 12 DO SOFTWARE
DISABLE IN 8 //DESCONECTA A ENTRADA 8 DO SOFTWARE
DISABLE OUT 12 //DESCONECTA A SAÍDA 12 DO SOFTWARE

Apêndice VI
SIMULADOR DA LINGUAGEM ACL – ROBOSOFT SIMULATOR

Objetivos:

Utilizar está ferramenta para a AUXILIAR a programação de robôs

1. SIMULADOR DA LINGUAGEM ACL

Esse programa tem como objetivo simular os programas na linguagem ACL

1.1. Simulador da Linguagem Acl

Antes de começarmos a simular necessitamos fazer o download do software e a sua instalação. O software está disponível para download no site: www.ee.pucrs.br/~labcim.

1.1.1. Instalando:

i. Faça o download do arquivo RoboSoft_Simulator.exe
ii. Dê um dublo clique sobre o ícone RoboSoft_Simulator.exe
iii. Siga as instruções do instalador.

1.1.2. Como utilizar

O simulador tem o mesmo sistema de funcionamento do Robosoft. Logo, para executar uma simulação de um programa, temos que fazer os mesmos passos feitos na utilização do Robosoft, que são:

i. Executar o Robosoft Simulator
ii. Criar um programa no editor.
iii. Fazer o download do programa
iv. Execução do programa, comando run.

A interface é muito parecida com o Robosoft, o terminal de um tem a mesma importância que o terminal do outro, como por exemplo, criação de vetores e variáveis de posição, criação de vetores globais.

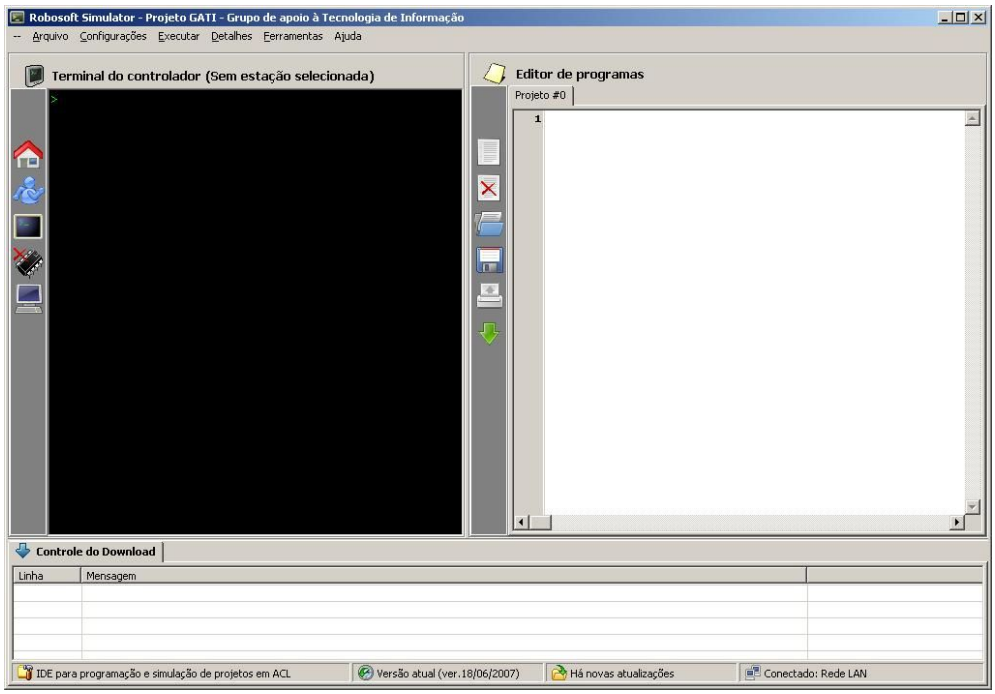


Figura 3. Robosoft Simulator.

1.1.3. Observações

O simulador tem o mesmo sistema de funcionamento do Robosoft. Logo, para executar uma simulação de um programa, temos que fazer os mesmos passos feitos na utilização do Robosoft, que são:

i. O usuário só vai utilizar o arquivo .dnl, pois nele está o conteúdo do programa. Os demais arquivos com os .acl são utilizados internamente no software.
ii. O comando GET só funciona na versão do WINDOWS NT, 2K e XP.
iii. O operador <> do comando de lógica relacional if, orif, andif e wait só funciona na versão do WINDOWS NT, 2K e XP.
iv. Não foram implementados os seguintes operadores para o comando SET: **COS, SIN, TAN, ATAN, EXP, LOG, AND e OR.**

1.1.4. Características

i. Manual eletrônico da linguagem ACL.
ii. Editor de programação.
iii. Simulação da lógica
iv. Simulação do terminal de comunicação com o controlador do robô
v. Alguns exemplos

1.1.5. Informações extras e download

O retorno de erros está em inglês e não está totalmente implementado. Alguns dos comandos implementados:

IF	RUN	DIM	PRINTLN
ANDIF	GOSUB	DIMG	PRINT

ORIF		DEFINE	READ
ELSE		GLOBAL	SET
LABEL			
GOTO			
FOR			
ENDFOR			
DELAY			

Você encontra mais informações no site: http://www.ee.pucrs.br/~labcim/.

BIBLIOGRAFIA

Manual ACL
Teach Pendant for Controller AC – User´s Manual
Teach Pendant for Controller A – User´s Manual
Teach Pendant for Controller B – User´s Manual
ACL – Linguagem de Controle Avançada – Guia de Referência