DEPARTAMENTO DE ENGENHARIA MECÂNICA

ESCOLA POLITÉCNICA

UNIVERSIDADE DE SÃO PAULO

PMC 581

PROJETO MECÂNICO II

TRANSFERÊNCIA DE DADOS ENTRE O ROBÔ E O PC

ORIENTADOR: **PROF. DR. LUCAS ANTÔNIO MOSCATO**

JUSSARA KOGA

MARCELO MASSAKI KAWAGUCHI

São Paulo

1998

8,2 (oito e dois)

AGRADECIMENTOS

Ao Professor Doutor Lucas Antônio Moscato,

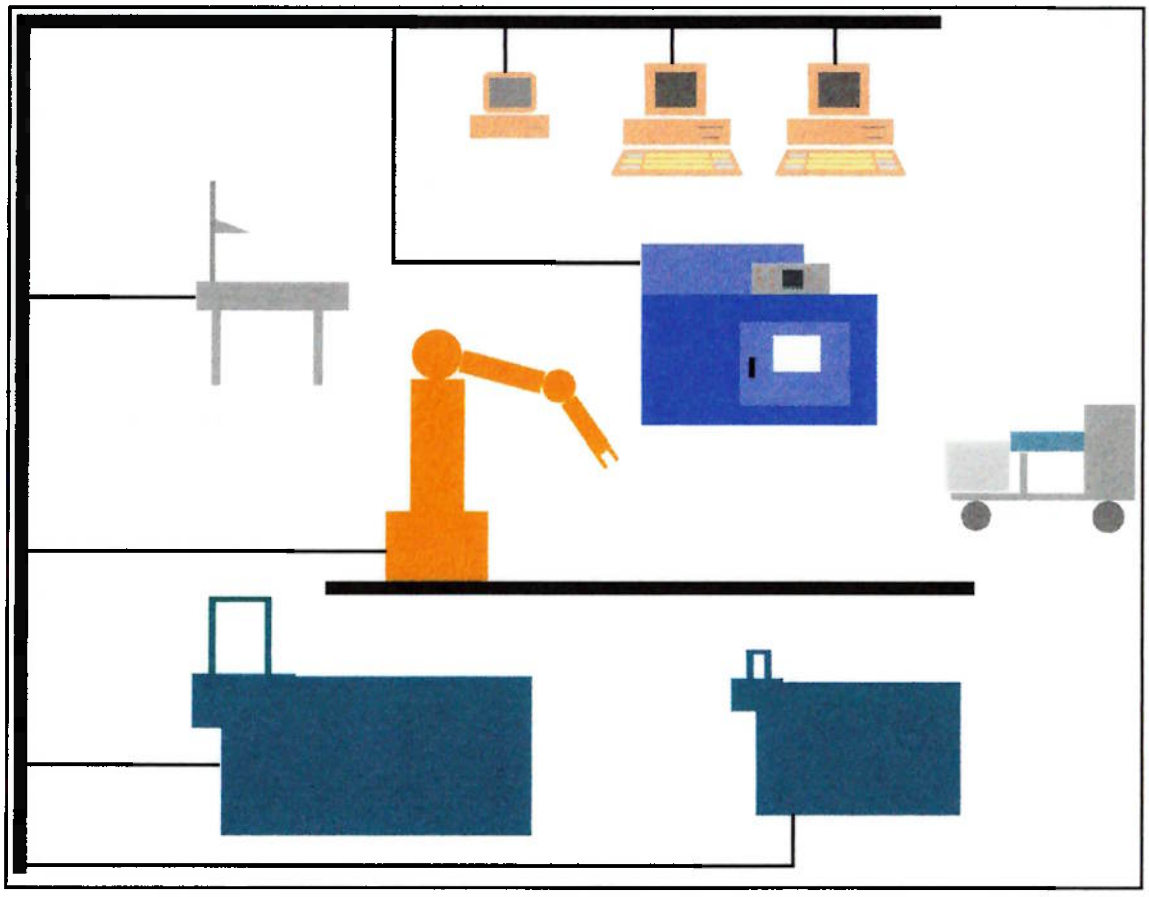
Ao Cássio, funcionário do departamento de Engenharia Mecatrônica,

As nossas famílias,

Agradecemos a todos pela orientação, apoio, paciência, ajuda e tudo o mais que me foi oferecido para que este trabalho de formatura fosse concluído.

1. DEFINIÇÃO DO TEMA DE TRABALHO

Baseando-se na Célula de Manufatura existente no Departamento de Engenharia Mecatrônica (cujo esquema está esquematizado a seguir), nos dispusemos a adicionar ao sistema ferramentas de controle e de monitoração ainda não existentes.



A célula esquematizada acima consiste basicamente de duas *workstations* **RISK 6000** (IBM), um PC, um torno de **CNC Mazak**, uma fresa e um torno ambos **Traub**, um carrinho de movimentação, um robô **ASEA IRB 6**, e um trilho.

Nossa idéia inicial era de controlar e sincronizar duas máquinas - o CNC Mazak e o Robô ASEA - através de um **PLC** (modelo PS 3 da *Klockner Moeller*) existente no Departamento; o intuito sugerido pelo nosso orientador era programar a saída serial do PLC, de forma que este mandasse sinais de controle digitais para ambas as máquinas referidas acima. O problema nesse caso seria programar essa saída, pois o PLC

disponível não possui a saída serial desejada; a existente no PLC é utilizada somente para transmitir o programa a partir da "maleta de programação do PLC".

Visto isso, nossa idéia passou a utilizar somente as saídas digitais geradas pelo PLC, as quais seriam transmitidas às duas máquinas através de suas entradas digitais.

Paralelamente a isso, o outro grupo também orientado pelo Prof. Dr. Lucas Moscato, formado pelos alunos Eric Preuss e Júlio César Roggero estavam estudando a viabilidade de utilizar o PROFIBUS da Siemens na interface entre o robô e um PC. Após algum tempo de pesquisa, chegaram à conclusão que a utilização do PROFIBUS seria inviável para o problema.

Assim, com o decorrer dos fatos, decidiu-se dividir o nosso trabalho que havia sido definido anteriormente em duas partes: nós ficaríamos com o controle do CNC pelo PLC, e o outro grupo, com o controle do robô ASEA também pelo PLC, e ambos utilizando as entradas e saídas digitais dos três equipamentos.

Entretanto, ao pesquisarmos detalhadamente o esquema e os manuais do torno CNC da Mazak, concluímos que mesmo sendo possível o controle do mesmo pelo PLC, havia a falta de documentação técnica necessária para a continuidade do projeto, inviabilizando a atividade de execução a qual precisa necessariamente ser realizada e concluída no próximo semestre, na matéria de PMC581. Além disso, segundo a opinião do Prof. Marcos Barreto (o qual é bastante familiarizado com o torno em questão), o trabalho seria muito extenso, e que, somado à ausência de documentação técnica, reforça a desistência do projeto proposto na Célula de Manufatura.

Portanto, para continuarmos focados na Célula de Manufatura, escolhemos de acordo com o nosso orientador, o projeto de carregar dados no robô ASEA via entrada serial (segundo o protocolo RS 232), através de um PC (transmissão "*off-line*"). Com essa decisão, retomaremos o trabalho iniciado mas não concluído do aluno Ériko Roberto

Bagatim. E para enfatizar a integração de sistemas, iremos interagir com o projeto do outro grupo; segundo o estipulado nas várias reuniões de grupo, o outro grupo fará um programa o qual será carregado diretamente no robô, segundo sua programação específica, mas incluirá no mesmo, um registrador que espera um sinal de controle proveniente da entrada serial do robô. E é exatamente esse sinal de controle via entrada serial do robô que o nosso grupo gerará, através da utilização dos protocolos específicos do robô de comunicação e aplicação. Assim, para que se estabeleça uma comunicação serial entre duas máquinas (PC/PC, PC/Torno, PC/Robô), é necessário que ambas possuam portas seriais e que "falem a mesma língua", ou seja, o mesmo protocolo de transferência de dados Particularmente no nosso caso, este protocolo já está implementado no robô; de forma que será necessário implementar o mesmo protocolo no PC, através da linguagem de programação C.

É de extrema importância enfatizar que todo esse processo de viabilidade do projeto, nos adicionou vários dados e características as quais desconhecíamos da Célula de Manufatura e também dos PLC's.

2. INTRODUÇÃO

2.1. Automação e Controle

Com o quadro mundial atual de competição e globalização, toda empresa que queira ter sucesso e rentabilidade, deve ser eficiente e flexível. Nas indústrias de manufaturas e processos, este fato tem como principal conseqüência o aumento de demanda por sistemas de controle industriais, de forma a otimizar as operações da fábrica em termos de velocidade, confiabilidade, versatilidade e qualidade do produto produzido.

Basicamente em todas as formas de indústria, o caminho mais seguro para a melhora de produtividade corresponde ao aumento da automação e integração de sistemas dos processos e das máquinas em si. Este tipo de automação e integração pode ser necessária para, de um modo direto, aumentar as quantidades produzidas, ou para melhorar a qualidade do produto e da precisão. De qualquer maneira, a automação e integração de sistemas requer a substituição de algumas ou de basicamente todas as entradas de dados realizadas pela "mão humana".

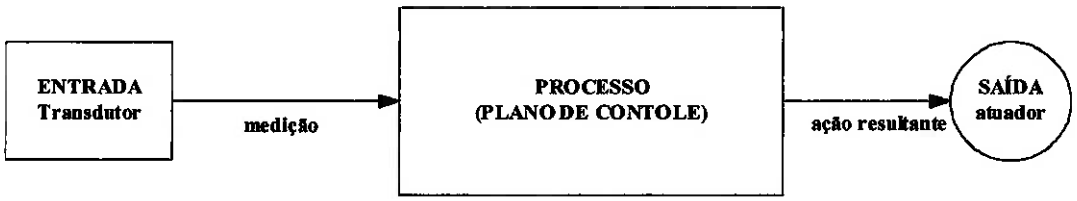
Muitas indústrias e plantas posicionam os trabalhadores no controle de máquinas e equipamentos, ao invés de precisarem de esforços propriamente físicos de forma a realizarem as suas tarefas. Este tipo de controle citado e largamente utilizado possui como principal requisito o conhecimento por parte do operário, de como o processo em particular funciona, e quais as entradas necessárias para alcançar e manter a saída do sistema desejada.

Para implementar e adquirir a automação e integração de sistemas do processo, deve-se substituir o operador por algum tipo de sistema automático que seja capaz de controlar o processo com alguma ou talvez nenhuma intervenção humana. Para tanto, é preciso desenvolver um sistema que possui a autonomia de iniciar, regular e interromper o processo, em resposta às variáveis medidas ou monitoradas dentro do processo, de

forma a se obter a saída do sistema desejada. Portanto, o sistema que possui todas essas características é chamado de "*sistema de controle*".

2.2. Controle Automático

Qualquer sistema de controle pode ser dividido em três seções constituintes: entrada (*input*), saída (*output*) e processamento (*processing*).



O modelo esquematizado acima pode também descrever em termos de ações, constituídas de medições de entrada, processo de controle baseado nessas entradas, e as ações resultantes da saída produzidas. O papel da seção de processo corresponde a produzir respostas pré determinadas (na forma de saídas) como resultado das informações fornecidas pelos sinais de entrada medidos. Existem vários métodos diferentes disponíveis para implementar esta função, mas todos eles utilizam entradas e saídas bastante similares.

Este modelo também representa o controle através de um operador humano na seção de processamento. Este operador deve saber qual é a saída desejada do sistema, monitorando visualmente as variáveis relevantes ao processo - i.e. entradas. Em resposta a essas leituras, o operador irá alterar os ajustes dos controles apropriados (válvulas, aquecedores...) de forma a obter a saída do processo desejada.

2.2.1. Entradas

Os sinais de entrada são normalmente fornecidos por vários transdutores os quais convertem grandezas físicas em sinais elétricos. Estes transdutores podem ser simples "push-buttons", chaves, termostatos, "strain gages",...Todos eles transmitem informações a respeito da "quantidade" que está sendo medida e analisada. Dependendo do transdutor utilizado, esta informação pode ser descontínua on/off (binária) ou contínua (analógica).

TRANSDUTOR	*INPUT*	*OUTPUT*
Chave	Movimento/posição	Voltagem binária (on/off)
Termostato	Temperatura	Voltagem binária
Strain Gage	Pressão/movimento	Variação de resistência
Fotocélula	Luz	Variação de voltagem
Proxímetro	Presença de Objetos	Variação de resistência

Tabela 1 - Tipos de transdutores de entrada

2.2.2. Saídas

O sistema de controle deve ser capaz de alterar elementos específicos ou quantidades durante o processo. Isto é realizado utilizando periféricos de saídas como bombas, motores, pistões, relês... os quais convertem sinais provenientes do sistema de controle em grandezas outras necessárias.

PERIFÉRICO (SAÍDA)	*GRANDEZA PRODUZIDA*	*ENTRADA*
Motor	Movimento rotacional	Elétrica
Bomba	Movimento rotacional/ deslocamento do produto	Elétrica

Pistão	Pressão/movimento linear	Hidráulica / pneumática
Solenóide	Pressão/movimento linear	Elétrica
Aquecedor	Calor	Elétrica
Válvula	Variação de orifício	Elétrica / hidráulica / pneumática
Relê	Chaveamento elétrico	Elétrica

Tabela 2 - Tipos de periféricos de saída

2.2.3. Área de Processamento

Corresponde ao conhecimento do operador das operações que são necessárias de forma a manter o processo "em controle". O operador utiliza adicionalmente ao seus conhecimentos, informações obtidas a partir de leituras de entradas, as quais produzem as ações de saída.

A partir das informações de entrada, o sistema de controle automático deve produzir os sinais de saída necessários, em resposta ao *plano de controle* embutido na seção de processamento. Este plano de controle pode ser implementado de duas formas distintas, utilizando **controle via hardware** ("*hard-wired control*") ou **controle programável.**

Os controles via hardware possuem as funções de controle fixas permanentemente, no momento em que os elementos do sistema são conectados (i.e. eletricamente), ao passo que nos sistemas de controle programável a função de controle é programada (e armazenada) dentro da unidade de memória, e pode ser portanto alterada através de reprogramação quando necessário.

A tabela abaixo descreve exemplos de sistemas de controle via hardware e programáveis, juntamente com o tipo de controle que eles são capazes de realizar - digital (discreto) ou analógico (contínuo).

HARD-WIRED	*FORMA DE CONTROLE*	*PROGRAMÁVEIS*	*FORMA DE CONTROLE*
Relês	Digital	Computadores	Digital/analógico
Lógica eletrônica	Digital	Microcomputadores	Digital/analógico
Lógica pneumática	Digital	PLC's	Digital
Lógica hidráulica	Digital		
Eletrônica analógica	Analógico		

Tabela 3 - Sistemas de Controle

2.3. Tipos de Processos Industriais

No quadro industrial atual, existe um vasto número de diferentes processos de manufatura; porém, todos os processos podem ser agrupados em três categorias principais com relação ao *tipo de operações* as quais existem dentro do processo:

- Produção contínua;
- Produção em lote;
- Produção de um item discreto.

Cada processo possui características e requisitos individuais e singulares, os quais devem ser considerados no projeto do sistema de controle.

2.3.1. Produção Contínua

Um processo contínuo tem como entrada a matéria prima e a processa continuamente, produzindo materiais e produtos finais como saída. O processo pode durar um longo período de tempo, como minutos, horas ou até semanas em certos casos. Um exemplo de produção contínua corresponde à manufatura de lâminas de aço; este processo em particular envolve uma grande quantidade de blocos de aço incandescentes, os quais são

submetidos, continuamente, à uma série de rolos compressores, reduzindo dessa forma a espessura do aço em estágios, e finalmente produzindo a lâmina de aço desejada ao final da linha de rolos compressores. Este processo pode levar vários minutos para ser completado, dependendo do comprimento do aço da matéria prima.

Como os rolos reduzem a espessura do aço, a velocidade da lâmina no sistema passa a ser crescente; considerando esse fato, o sistema de controle deve atuar sobre cada conjunto de rolos, de forma a manter constante a espessura da lâmina produzida na saída do processo. Portanto, nesse caso, acuracidade, precisão e controle rápido do primeiro conjunto de rolos são extremamente importantes para permitir a compensação na entrada dos blocos de aço em contraposição à variação da espessura dos mesmos.

2.3.2. Produção em Lotes

Este tipo de processo utiliza uma quantidade definida de matéria prima (input) e a processa, resultando numa quantidade específica de produto final (output) ou um produto que sofrerá estágios de processamento posteriores.

O exemplo mais comum corresponde ao processamento químico de polímeros; nesses processos, várias substâncias químicas são bombeadas em tanques distintos, e depois misturadas a uma dada temperatura, durante um período de tempo pré-definido; a mistura ainda é processada em vários estágios (processo de lotes), ao invés de um processamento contínuo. Dessa forma, diversos processos em lotes configuram o ciclo completo: medição de quantidades específicas de cada substância química, aquecimento, filtragem,...Consequentemente, cada atividade requer alguma forma de controle para assegurar a formação correta e exata de cada lote.

O tipo de controle utilizado em cada estágio do processo depende de cada processo individualmente; por essa razão, para cada tarefa é utilizado um sistema de controle discreto da forma on/off.

2.3.3. Produção de Partes Discretas

Nesse tipo de processo, um único item individualmente é submetido à várias operações, antes de ser propriamente produzida a peça final. Alternativamente, muitos componentes podem ser combinados ou manufaturados dentro do processo, de forma a originar um único item ou unidade.

Um exemplo dessa produção de partes discretas seria um processo onde estão envolvidas várias seqüências de operações, envolvendo várias máquinas (robôs, fresas, tornos,...) de alta precisão. A maioria dessas atividades à qual está submetida essa peça devem e são controladas baseadas em comandos binários on/off, incluindo conexões entre cada máquina, de forma a transferir as informações necessárias para garantir a finalização completa e com sucesso de cada passo do processo.

É exatamente nesse contexto em que se originou e se enquadra toda a nossa motivação desse trabalho de formatura. Ele se baseia em um trabalho previamente iniciado no Departamento de Engenharia Mecatrônica da Escola Politécnica da USP, com a participação de várias pessoas envolvidas na área.

Nele, foram integradas diversas máquinas, no sentido de estabelecer comunicação e controle entre si, comandos de sincronização,...

3. COMPUTER LINK

Atualmente, a troca de informações entre diferentes sistemas é fundamental na automação industrial. A funcionalidade ***Computer Link*** foi desenvolvida pela ASEA, para prover esta capacidade, de uma forma padronizada e segura para a sua linha de robôs.

O *Computer Link* permite a conexão entre o robô industrial, denominado **IRB** (*Industrial Robot*), e um computador externo, chamado **SC** (*Superior Computer*).

A funcionalidade *Computer Link* contida no controlador do robô necessita adicionalmente de um pacote tanto de hardware quanto de software de comunicação.

3.1. SC (Superior Computer)

Quanto à configuração do SC, esta pode ser de qualquer tipo, desde que suporte comunicação serial **RS 232-C/CCITT V24,28**, e utilize o mesmo protocolo que utilizado pelo robô. Apesar do software de controle e/ou supervisão para computadores padrão IBM-PC ser disponível, o mesmo será desenvolvido neste trabalho (como citado anteriormente).

Para tanto, serão apresentadas em tópicos posteriores o protocolo de comunicação **ADPL10** e o protocolo de aplicação **ARAP**.

As possíveis funções que o SC pode exercer no sistema são as seguintes:

- **Controle**: é possível alterar o modo de operação, iniciar e parar a execução, ler e escrever informações no robô. Consequentemente, o controle do robô de torna centralizado e simplificado;
- **Supervisão**: o SC recebe informações sobre possíveis eventos e erros que ocorreram no IRB. Desta maneira, torna-se possível avaliar o funcionamento do sistema;

- **Base de Dados**: o SC pode ser utilizado como uma fonte de reposição dos programas de aplicação do IRB. Fica assim possível armazenar todos os programas do robô em uma biblioteca do SC.

3.2. Computer Link Software

O *Computer Link Software* é descrito e baseado em dois protocolos distintos:

- **ADPL 10** (*Asea Data Link Protocol*): protocolo de comunicação;
- **ARAP** (*Asea Robot Application Protocol*): protocolo de aplicação.

3.2.1. Funcionalidades

- **Servidor de Arquivos:** permite carregar programas no IRB a partir do SC, e gravar programas no SC provenientes do IRB;
- **Manipulação de Informações:** permite leitura e escrita de informações sobre o robô e o processo;
- **Controle Remoto do Robô:** permite troca do modo de operação: selecionar, iniciar e finalizar o programa; movimentar o IRB em coordenadas XYZ e de punho;
- **Supervisão:** permite a supervisão do status do IRB e leitura de mensagens de erro;
- **Programação Off-Line:** permite a programação do robô via SC.

3.2.2. Princípios

De forma a garantir a integridade dos dados durante a transmissão entre o IRB e o SC, um protocolo de comunicação deve ser utilizado. O IRB utiliza o protocolo **ADPL 10**, o qual corresponde a um protocolo assíncrono similar ao ISO 1745.

A conexão é serial, ponto a ponto, com transmissão assíncrona, o que significa que um caracter é enviado por uma mesma via e que cada um deles possui *start* e *stop bits*, para sincronizar o destinatário.

O protocolo é totalmente transparente para todos os tipos de informações, incluindo funções para detecção e correção de erros. Na transmissão existe uma relação mestre-escravo entre duas partes (IRB/SC) em comunicação; isto significa que o mestre envia informação enquanto que o escravo a recebe. Tanto o IRB como o SC podem ser o mestre, dependendo de quem iniciou a transmissão.

O protocolo ARAP define as mensagens transmitidas em ambos os sentidos de comunicação. Três tipos deferentes de mensagens são usados:

- **Comandos:** enviados tanto pelo SC quanto pelo IRB;
- **Respostas:** enviadas como resultado de um comando; podem ser positivas ou negativas (associadas a um código de erro);
- **Mensagem espontânea:** enviadas apenas pelo IRB (exemplo: parada de emergência).

O operador seleciona na unidade de programação a identidade (ID) do robô, a qual será utilizada em todas as mensagens do IRB ao SC.

O modo de operação do robô - local ou remoto - é também selecionado na unidade de programação.

Comandos enviados pelo SC ao IRB somente serão executados se o **modo remoto** estiver selecionado.

3.3. Computer Link Functions

3.3.1. Comandos do SC ao IRB

Os comandos os quais podem ser enviados do SC ao IRB incluem:

- Carregar um bloco de programas ou um único programa do SC para o IRB;
- Iniciar um programa ou reiniciar o programa em execução, a partir do ponto no qual este havia parado;
- Interromper um programa;
- Carregar as seguintes informações:
  - *TCP register;*
  - *Location register;*
  - *Sensor register;*
  - *I/O data;*
  - *Configuration data;*
  - *Frame register;*
  - *Arc welding data.*
- Ler o status do IRB sobre:
  - Modo de operação (stand-by, em operação, programa em execução, parada de emergência);
  - Unidade de programação;
  - Interrupção de programa via entrada externa, permitida ou não;
  - Identificação do programa e linha de execução;
  - Posição do robô.
- Mudar o modo de operação:
  - *Stand-by;*
  - *Operation;*
  - *Synch;*
  - *Operation plus synch.*

- Determinar quais programas estão armazenados no robô, e qual a memória livre disponível;
- Apagar programas da memória do robô;
- Carregar um bloco de programas ou um único programa do *floppy disk* para o IRB;
- Movimentar o IRB a partir do SC;
- Transferir um bloco de programas ou um único programa do IRB para o SC.

3.3.2. Comandos do IRB ao SC

Os comandos enviados ao SC são gerados pelo operador da unidade de programação; dentre os quais se destacam:

- Salvar ou ler no SC um bloco de programas ou um único programa;
- Salvar ou ler no SC deferentes tipos de dados (informações de configuração e/ou processo);
- Enviar informações para uso na programação off-line.

3.3.3. SUCTRL

O operador pode adicionar uma instrução chamada *Superior Control* (SUCTRL) ao programa do robô. Quando o robô executar esta instrução, uma mensagem espontânea será enviada ao SC. É possível fazer o robô parar e esperar por novas instruções do SC, ou continuar a execução do programa após o envio desta mensagem. Um valor de registrador pode ser incluído nesta mensagem.

3.3.4. Mensagens Espontâneas

Mensagens espontâneas são enviadas do IRB ao SC, contendo informações sobre o estado do robô quando um dos eventos abaixo ocasionalmente ocorrer:

- Instrução SUCTRL é executada;
- Parada de emergência;
- *Search stop*;
- Início ou parada de programa via unidade de programação, ou painel de controle;
- Unidade de programação colocada ou removida de seu compartimento;
- Unidade de programação conectada ou desconectada;
- Mudança do modo de controle do robô (remoto ou local);
- Erro de sistema;
- Inicialização do robô;
- Mudança no modo de operação do robô (*stand-by* ou *operate*).

3.3.5. Movimentação do Robô

Este comando pode ser utilizado para controlar o robô em coordenadas absolutas ou relativas à posição atual. A movimentação do robô é feita através de três comandos: iniciar, movimentar e parar.

3.4. Computer Link Hardware

A interface do SC deve ser compatível com V24, V28/RS232. Para distâncias superiores a 15 metros, um modem específico deve ser utilizado. A velocidade de comunicação deve ser de **9600 bauds**. O tamanho do caracter é de **8 bits**, mais um bit de **paridade par**, um **bit de parada** (*stop bit*) e um **bit de inicio** (*start bit*).

3.4.1. Descrição Técnica

O hardware do *Computer Link* é constituído das seguintes unidades:

- Placa de Comunicação assíncrona **DSCA 114;**
- Cabos ("*wiring*") **DSTK 152;**
- **DSTC 120** - Terminal de Comunicação para o **DSCA 114.**

3.4.2. A Placa de Comunicação DSCA 114

Esta unidade é conectada ao sistema de bus paralelo, e pode ser utilizada para comunicar-se com unidades as quais possuam interfaces seriais assíncronas.

Algumas características físicas são apresentadas abaixo:

Número de canais	4
Interface de sinais, saídas	CCITT V24 e modem de baixo alcance
Velocidade de transmissão	300 ~ 9600 bits/s
Tamanho da palavra	5, 6, 7 ou 8 bits
Paridade	Par, ímpar ou sem paridade
Stop bits	1, 1.5 ou 2
Buffer	128 + 96 bytes/canal
Conexão	Via DSTC 120 com o cabo DSTK 152

Tabela 4 - Placa de comunicação DSCA 114

O **canal 0** é reservado para impressora, o **canal 1** para o *Computer Link* e os canais 2 e 3 não são utilizados.

3.4.3. Unidade de Conexão DSTC 120

A comunicação serial assíncrona com o IRB é implementada através da placa **DSTC120**, localizada no rack posterior do controlador do robô. Deve ser utilizado o plug padrão DB25 (maiores informações detalhadas estão contidas no tópico de Comunicação Serial).

Algumas características físicas são apresentadas abaixo:

Número de Canais	2
Interface de sinal	CCITT V24 (RS232C)
Peso	100 g
Dimensões	120 x 80 x 27mm

Tabela 5 - A unidade de conexão DSTC 120

4. ADPL 10 - PROTOCOLO DE COMUNICAÇÃO

Este decorrente capítulo descreve o uso do protocolo ADPL 10 entre o IRB e o SC.

ADPL 10 corresponde a um protocolo para comunicação hierárquica assíncrona, entre duas estações. É caracterizado por:

- Alto nível de segurança;
- Código transparente (sem restrições), constituído de 8 bits + bit de paridade par + bit de parada;
- Transmissão serial assíncrona via hardware normalmente disponível;
- Possibilidade da transmissão ser iniciada por qualquer uma das estações.

Este protocolo somente é aplicável à comunicação ponto a ponto entre duas estações, sendo uma delas mestre e a outra consequentemente escrava.

4.1. Termos Aplicados

Os seguintes termos são normalmente utilizados:

- **HS**: corresponde ao SC (*host station*) na comunicação;
- **SS**: corresponde ao IRB (*subordinate station*) na comunicação;
- **Mestre**: é a estação que em determinado instante transmite um telegrama. Tanto o HS quanto o SS podem ser mestres;
- **Escravo**: é a estação que em determinado instante recebe um telegrama. . Tanto o HS quanto o SS podem ser escravos;
- ***Half Duplex***: procedimento que possibilita a comunicação nos dois sentidos. De forma que a transmissão nunca esteja ocorrendo em ambas as direções simultaneamente;

- **Procedimento de Recuperação**: procedimento através do qual uma estação tenta solucionar um conflito ou corrigir um erro ocorrido no processo de comunicação;
- **Caracteres de Identificação**: caracteres utilizados para definir a natureza da operação subsequente;
- **Texto**: um string de caracteres numa comunicação, que contenha informações a serem transmitidas;
- **Telegrama**: toda informação transmitida entre o computador e o robô, dividida em unidades menores; no protocolo ADPL 10, telegramas são iniciados com o caracter STX e finalizados com o caractere ETX.

4.2. Caracteres de Controle

Os seguintes caracteres de controle são utilizados na transmissão:

- **ENQ (Enquiry)**: utilizado como um pedido de resposta à outra estação; utilizado para estabelecer a conexão;
- **ACK (Acknowledge)**: utilizado pelo escravo para a confirmação da última informação enviada pela outra estação;
- **WACK (Wait and Acknowledge)**: utilizado pelo escravo para a confirmação da última transmissão; entretanto este não está ainda preparado para receber nova informação;
- **RVI (Reverse Interrupt)**: utilizado pelo escravo para a confirmação da última transmissão, e para pedir permissão para assumir a posição de mestre. O IRB sempre envia como confirmação um RVI caso ele possua informações a transmitir;
- **NAK (Negative Acknowledgement)**: utilizado pelo escravo para indicar o não entendimento da última transmissão;

- **DLE (Data Link Escape)**: utilizado pelo mestre para mudar num telegrama o significado do próximo caracter de dado para caracter de controle;
- **STX (Start of Text)**: utilizado para indicar um início de um telegrama;
- **ETX (End of Text)**: utilizado para indicar o final de um telegrama;
- **EOT (End of Transmission)**: utilizado para indicar o fim da transmissão; transmissões subsequentes somente poderão ser realizadas mediante nova conexão;
- **BCS (Block Check Sum)**: utilizado para verificação das informações transmitidas anteriormente.

4.2.1. Códigos dos Caracteres de Controle

Código	Octal	Hexadecimal	Decimal
ENQ	005	05	05
ACK	006	06	06
WACK	016	0E	14
RVI	017	0F	15
NAK	025	15	21
DLE	020	10	16
STX	002	02	02
ETX	003	03	03
EOT	004	04	04

Tabela 6 - Códigos dos Caracteres de Controle

4.3. Procedimentos de Comunicação

Este procedimento é utilizado para comunicação entre duas estações. A transmissão é assíncrona, serial e *half-duplex*. A transmissão pode ser iniciada por qualquer uma das estações. Cada caracter consiste em 8 bits + 1 bit de paridade par + 1 bit de parada.

4.3.1. Fases da Comunicação

A troca de informações entre as estações pode ser divididas e diferenciadas nas seguintes fases:

1. Estabelecimento do contato;
2. Transferência da informação;
3. Conclusão.

A informação é transmitida em apenas um sentido durante a fase 2. Para a mudança no sentido de transmissão de informação, devem ser realizadas necessariamente a fase 3 e uma nova fase 1.

4.3.2. Relação Mestre - Escravo

A estação que toma a iniciativa ao estabelecer o contato torna-se o mestre durante a subsequente troca de informação, e consequentemente, a outra se torna escravo. O mestre controla a troca de fases e também é responsável pela continuidade da comunicação.

4.3.3. Fase 1: Estabelecimento do Contato

A estação que deseja transmitir informações à outra deve necessariamente tentar estabelecer contato, enviando para isso o sinal de ENQ. Caso a outra estação esteja preparada para receber o telegrama, ela responde com um sinal de ACK; senão, com um sinal de WACK ou RVI.

- Caso ACK seja a resposta a um ENQ, o contato foi estabelecido.
- Caso WACK seja a resposta a um ENQ, a outra estação ainda não está preparada para receber.

- Caso RVI seja a resposta para um ENQ, a outra estação também deseja transmitir.
- Se o HS e o SS transmitam um sinal de ENQ simultaneamente, o IRB dá prioridade ao SC. O IRB responde com ACK, enquanto que o SC espera pela resposta.

4.3.4. Fase 2: Transmissão da Informação

O mestre inicia a transmissão da informação quando o contato é corretamente estabelecido. Cada telegrama é precedido por STX e concluído por ETX.

É importante notar que neste contexto, telegrama significa uma quantidade independente de informações. Vários telegramas podem ser transmitidos durante uma mesma fase, cada um deles concluído com ETX. Logo, ETX não significa necessariamente o fim desta fase.

Para garantir que os caracteres de controle STX e ETX não sejam confundidos com dados, eles são precedidos por DLE. Entretanto, caso o texto contenha algum caractere cujo código seja o mesmo que DLE, este deve ser transmitido duas vezes (duplicado).

Exemplo:

- Na seqüência XXX/DLE/YYY (com XXX e YYY diferentes de DLE), XXX é interpretado como dado e YYY é interpretado como caractere de controle (STX ou ETX);
- A seqüência DLE/DLE é interpretada como um único caractere de dado, cujo código é o mesmo que DLE.

Um caractere de verificação de soma de bloco (BCS) deve ser transmitido após cada telegrama. Este caracter é enviado imediatamente após ETX. O BCS é calculado pela estação emissora e é checado pela estação receptora.

Caso o telegrama recebido pela outra estação seja considerado correto, a resposta deste é um ACK, WACK ou RVI. Caso o escravo detecte alguma anomalia no telegrama, a resposta será certamente NAK.

- Recebendo um AKC, o mestre é capaz de transmitir a próxima mensagem ou iniciar a fase 3 (conclusão).
- Recebendo um WACK, o mestre deve esperar ou iniciar a fase 3.
- Recebendo um RVI, o mestre deve iniciar a fase 3;
- Recebendo um NAK, o mestre deve repetir a última mensagem.

4.3.5. Fase 3: Conclusão

A terceira fase é iniciada pelo mestre quando não há mais informações a serem transmitidas, ou quando uma das condições apresentadas anteriormente ocorrer. O mestre encerra a conexão enviando EOT. Esta ação não necessita de confirmação correspondente do escravo.

4.3.6. Reverse Interrupt (RVI)

Este comando é utilizado pela estação a qual em determinado momento encontra-se como escravo, de forma a confirmar a última transmissão do mestre e, simultaneamente, pedir permissão para assumir a função de mestre.

O RVI é sempre enviado pelo robô como uma confirmação positiva e quando o mesmo possui informações a serem transmitidas. O HS decide caso ele continue sendo mestre, ou caso o robô se torne o mesmo. O robô tem concedida a permissão para tornar-se mestre se receber EOT após o envio do RVI.

As seguintes regras são corretamente aplicáveis:

- RVI somente pode ser enviado após a conclusão de um telegrama ou como resposta a um ENQ;
- O HS pode aceitar ou rejeitar o RVI, de acordo com a sua situação;
- Caso o mestre aceitar o RVI, isto é feito através do envio de um EOT. Uma nova conexão é estabelecida pela estação que fazia o papel de escravo;
- Caso o mestre rejeitar o RVI, isto é feito através do envio da seqüência DLE/STX, a qual inicia o próximo telegrama. O escravo deve estar preparado para continuar recebendo informações após enviar esse RVI.

4.4. Verificação

A comunicação é verificada por paridade horizontal (caracter) e vertical (bloco), e também através de meios de marcação de seqüência de frases dentro de uma fase de comunicação.

4.4.1. Paridade Horizontal

Cada símbolo consiste de 8 bits + 1 bit de paridade. O bit de paridade é calculado como soma módulo 2 dos 8 bits do caracter de paridade par e de um stop bit.

4.4.2. Paridade Vertical

Entende-se como caractere BCS o resultado do cálculo bloco por bloco. Isto é feito através do cálculo bit a bit módulo 2 dos caracteres envolvidos.

As seguintes regras se aplicam:

1. O cálculo se inicia com a seqüência DLE/STX;
2. A seqüência DLE/STX não é incluída na soma;

3. O primeiro DLE em cada seqüência DLE/XXX (DLE/DLE ou DLE/ETX) também não é incluído na soma;
4. Todos os outros caracteres (incluindo ETX) são considerados no cálculo.

O valor calculado (BCS) é então enviado como um caractere adicional, após cada telegrama e cada ETX.

4.4.3. Marcação de Seqüência

É utilizada para permitir ao receptor determinar caso o telegrama consiste numa repetição do último, ou caso se trata de um novo. Isto corresponde a uma marcação par ou ímpar em cada telegrama. É utilizado para tanto o bit mais significativo do código STX:

B7	B6	B5	B4	B3	B2	B1	B0	
0	0	0	0	0	0	0	1	STX par
1	0	0	0	0	0	0	1	STX ímpar

Tabela 7 - Marcação de Seqüência

B7 deve ser "0" na fase 1. O primeiro telegrama durante a fase 2 deve ser par. O próximo telegrama deve ser ímpar. Caso um telegrama deva ser repetido, sua condição de par ou ímpar não irá mudar, sendo possível ao receptor diferenciá-lo.

4.5. Recuperação de Erros

Quando a comunicação diverge dos procedimentos definidos anteriormente, o mestre é responsável pelo retorno à rotina correta. O procedimento para este fato é denominado de Recuperação de Erros.

4.5.1. Falha na Transmissão

Um telegrama recebido de forma incorreta é respondido pelo escravo com um NAK. O mestre deve então reenviar o mesmo telegrama, até que uma confirmação positiva seja obtida. Quando houver repetição de mensagem, o bit de seqüência (par ou ímpar) não deve ser alterado.

Caso a transferência da informação não seja bem sucedida em um número máximo de tentativas, o mestre deve finalizar a comunicação, passando à fase 3. O telegrama deve ser considerado como não transmitido. O robô termina a comunicação quando recebe quatro sucessivos NAK para um mesmo telegrama.

Após tal interrupção, resultante de repetidos erros de transmissão, uma nova tentativa não pode ser feita durante um tempo de recuperação. No caso do robô, este tempo eqüivale a 0,1 segundo.

4.5.2. Time-Out

Os tempos para *time-out* são de 1 segundo para o mestre, e 3 segundos para o escravo. A razão 1:3 entre estes tempos significa basicamente que o mestre detecta interrupções na comunicação mais rapidamente que o escravo, o que permite a correção antecipada do erro. As seguintes regras se aplicam:

- Caso o mestre detecte *time-out* na fase 1 (após ENQ), ou na fase 2 (após o término do telegrama), ele assume que o escravo não entendeu o telegrama emitido. O mestre então deve repetir o telegrama, de acordo com as regras as quais são aplicada quando do recebimento de um NAK;
- Caso o escravo detecte *time-out* na fase 1(após responder ao ENQ), ou na fase 2 (durante a recepção de um telegrama ou após responder ao mesmo), a conexão será

considerada interrompida. Uma nova transferência de informação deve ser feita através de uma nova fase 1. A transmissão do telegrama é considerada sem sucesso.

Para garantir que a rotina esteja correta quando o mestre repetir um telegrama após o time-out, o escravo deve interpretar a seqüência DLE/STX segundo as seguintes regras:

- Caso a seqüência DLE/STX seja encontrada durante a recepção de um texto em um telegrama, o escravo deve considerar perdido o fim do telegrama. DLE/STX é portanto considerado como o início da repetição deste último telegrama. O escravo deve verificar através da não alteração do bit de seqüência. Se este tiver sido alterado, será considerado que um erro de transmissão ocorreu;
- Caso a seqüência DLE/STX seja encontrada após a recepção de um telegrama, o bit de marcação de seqüência deve ser verificado. Se este não se alterou, deve ser considerada uma repetição do telegrama. Caso este tenha-se alterado, somente é considerada uma transmissão correta caso uma repetição não tenha sido requisitada; em caso afirmativo, é considerado então a existência de erro na transmissão, e o escravo deve responder com um NAK ao fim deste telegrama.

4.5.3. Sobrecarga do Receptor

Caso a capacidade de recepção do escravo esteja temporariamente sobrecarregada, este responderá com um WACK. Trata-se de uma confirmação positiva, combinada com o pedidos para o mestre atrasar a próxima transmissão.

Caso o WACK seja recebido como confirmação, resta ao mestre duas alternativas distintas:

- Se o mestre não tiver mais informações a serem transmitidas, a conexão é encerrada com o envio de EOT;

- Se o mestre tiver mais informações a serem transmitidas, ENQ é enviado. O escravo responderá com WACK até que sua capacidade permita a recepção de novas informações. Quando o mestre receber um ACK, a comunicação deverá retornar do ponto em que foi interrompida.

O mestre aceitará um número máximo de sucessivos caracteres WACK. Caso este número seja superado, o mestre deve assumir que houve erro , e encerrar a conexão enviando um EOT. Este número máximo corresponde à 100 para o robô. O tempo entre cada ENQ sucessivo é de 0,1 segundo para o robô. O robô **nunca** transmite um WACK. Após a interrupção causada pelo envio de sucessivos WACK's, deve ocorrer um certo tempo de recuperação até que um novo contato seja estabelecido; este tempo é de 1 segundo para o robô.

4.6. Exemplos

4.6.1. Comunicação Normal

HS (SC)	SS (IRB)	OBS.
ENQ		
	ACK	
DLE		
STX (par)		
...		
DLE		
ETX		
BCS		
	ACK	
DLE		
STX (ímpar)		
...		
DLE		
DLE		Caractere DLE
...		
ETX		
BCS		
	ACK	
EOT		
	ENQ	
ACK		
	DLE	
	STX (par)	
	...	
	DLE	
	ETX	
	BCS	
ACK		
	EOT	

Tabela 8 - Comunicação Normal

4.6.2. Recuperação de Erro de Transmissão

HS (SC)	SS (IRB)	OBS.
ENQ		
	ACK	
DLE		
STX (par)		
...		Interferência
DLE		
ETX		
BCS		
	NAK	
DLE		
STX (par)		
...		Repetição
DLE		
ETX		
BCS		
	ACK	
DLE		
STX (ímpar)		
...		
DLE		
ETX		
BCS		
	ACK	
EOT		

Tabela 9 - Recuperação de Erros na Transmissão

4.6.3. Recuperação de Time-Out

HS (SC)	SS (IRB)	OBS.
	ENQ	
ACK		
	DLE	
	STX (par)	
	...	
	DLE	
	ETX	
	BCS	
		Sem resposta
	DLE	
	STX (par)	
	...	Repetição
	DLE	
	ETX	
	BCS	
ACK		
	EOT	
ENQ		
	ACK	
DLE		
STX (par)		
...		
DLE		
ETX		
BCS		
	NAK	Resposta ignorada
DLE		
STX (par)		
...		Repetição
DLE		
ETX		
BCS		
	ACK	
EOT		

Tabela 10 - Recuperação de Time-Out

4.6.4. Recuperação de Sobrecarga

HS (SC)	SS (IRB)	OBS.
	ENQ	
ACK		
	DLE	
	STX (par)	
	...	
	DLE	
	ETX	
	BCS	
WACK		
	ENQ	Mais informações
WACK		
	ENQ	
ACK		
	DLE	
	STX (par) *	
	...	
	DLE	
	ETX	
	BCS	
WACK		
	EOT	Término de informações

Tabela 11 - Recuperação de Sobrecarga

- Este telegrama é transmitido com bit de seqüência par, pois o comando ENQ que o precede reinicializa a seqüência de marcação.

4.6.5. Reverse Interrupt (RVI)

HS (SC)	SS (IRB)	OBS.
	ENQ	
ACK		
	DLE	
	STX (par)	
	...	
	DLE	
	ETX	
	BCS	
RVI		
	DLE	RVI rejeitado
	STX (ímpar)	
	...	
	DLE	
	ETX	
	BCS	
RVI		
	EOT	RVI aceito
ENQ		
	ACK	
DLE		
STX (par)		
...		
DLE		
ETX		
BCS		
	ACK	
EOT		

Tabela 12 - Reverse Interrupt

5. ARAP - PROTOCOLO DE APLICAÇÃO

As regras apresentadas a seguir constituem o protocolo de aplicação para controle de robôs ASEA. É descrito também a troca de mensagens e o formato das mesmas.

5.1. Definições

5.1.1. Caracter

Um caracter consiste de 8 bits de dados, mais um bit de paridade e um stop bit. É sempre codificado em binário.

5.1.2. Telegrama

A série de caracteres transmitida utilizando-se o protocolo de transmissão ADPL10, o qual é delimitado pelos caracteres de controle DLE/STX e DLE/ETX.

5.1.3. Mensagem

Uma dada quantidade de informação que está logicamente relacionada e constitui uma unidade. Cada telegrama pode somente conter uma mensagem.

Mas como o comprimento máximo de um telegrama é de 128 bytes, uma mensagem pode consistir de um ou mais telegramas.

5.2. Estrutura do Telegrama

Um telegrama sempre consiste de duas partes: cabeçalho (*header*) e campo de dados (*data field*). O cabeçalho sempre possui o mesmo tamanho e é composto dos mesmos campos.

Todos os caracteres de um telegrama, independentemente de pertencerem ao cabeçalho ou ao campo de dados, são codificados em binário. Quando um campo consiste de mais de um byte (8 bits), o byte mais significativo é enviado primeiro.

O conteúdo e tamanho do campo de dados é variável, dependendo do tipo de mensagem. Em alguns casos, ele simplesmente não é necessário.

5.2.1. Cabeçalho

7	6	5	4	3	2	1	0	Byte
NOB								0
								1
DESTINATION ADDRES								2
SOURCE ADDRESS								3
FUNCTION CODE								4
NOT USED				MLI	RS	TT		5
FUNCTION SUFFIX								6
								7

- **NOB**: número de bytes do telegrama, incluindo o cabeçalho;
- **DESTINATION ADDRESS**: endereço que identifica o receptor do telegrama. Deve estar de acordo com a identidade do robô;
- **SOURCE ADDRESS**: endereço do transmissor do telegrama;
- **FUNCTION CODE**: código que especifica a função da mensagem;
- **MLI**: (*Message Length Indicator*) Indicador do comprimento da mensagem. "0" para uma mensagem contida em um só telegrama ou último telegrama da mensagem, e "1" para mensagem contínua no próximo telegrama;

- **RS**: (*Response Status*) Status da Resposta. "0" para reconhecimento positivo do último comando (*Positive Acknowledge*), e "1" para o caso contrário (*Negative Acknowledge*).
- **TT**: (*Telegram Type*) Tipo de Telegrama. "01" para comando, "10" para resposta e "11" para mensagem espontânea;
- **FUNCTION SUFIX**: Informação adicional ao *function code*, a qual é necessária esporadicamente.

5.2.2. Campo de Dados

DATA FIELD	**8**
(ou código de erro - ERROR CODE - na resposta)	**9**
DATA FIELD	
	NOB-1

- **DATA FIELD**: corresponde aos dados, ou seja, valores ou programas. O seu comprimento máximo é de 120 bytes.
- **ERROR CODE**: código de erro em caso de **RS=1** (*Negative Acknowledge*). Quando isto ocorre, o código é sempre enviado nos bytes 8 e 9.

5.2.3. Tipos de Telegrama

Existem 3 tipos pré - estabelecidos de telegrama:

- **Comando**: pode ser enviado tanto pelo SC quanto pelo IRB, significando que o transmissor está pedindo que o receptor execute alguma função. O comando pode possuir um campo de dados ou não. No caso do comando ser maior do que 128

bytes, este será enviado em vários telegramas sucessivos. Um comando deve sempre receber uma resposta positiva ou negativa;

- **Resposta**: somente é enviada como feedback à comandos. O formato da resposta é semelhante ao do comando. Quando ocorre um NAK, o código de erro de 16 bits é enviado nos bytes 8 e 9 do telegrama de resposta, que consequentemente tem RS setado para 1;
- **Mensagem Espontânea**: enviada pelo IRB ao SC, para informar a ocorrência de eventos particulares, como a parada de emergência.

5.3. Comandos do SC ao IRB

Dentre uma longa lista de comandos, foram selecionados aqueles que serão utilizados pelo nosso trabalho em questão:

- Início da execução de um programa específico do robô;
- Parada de execução de um programa específico do robô;
- Leitura de um registrador específico através do SC;
- Escrita de um valor em um registrador específico através do SC;
- Requisição de status através do SC.

5.4. Formato dos telegramas

5.4.1. Início da execução de um programa do robô

- **Comando**

7	6	5	4	3	2	1	0	Byte
NOB=10 (decimal)								0
								1
DESTINATION ADDRES								2
SOURCE ADDRESS								3
FUNCTION CODE=2								4
0	0	0	0	0	0	0	1	5
FUNCTION SUFFIX								6
(0 para iniciar o programa do início / 1 para iniciar do ponto onde parou)								7
PROGRAM NUMBER								8
(de 0 a 9999)								9

- **Resposta**

7	6	5	4	3	2	1	0	Byte
NOB=10 (decimal)								0
								1
DESTINATION ADDRES								2
SOURCE ADDRESS								3
FUNCTION CODE=2								4
0	0	0	0	0	0/1	0	1	5
FUNCTION SUFFIX								6
(0 para iniciar o programa do início / 1 para iniciar do ponto onde parou)								7
PROGRAM NUMBER								8
(ou ERROR CODE)								9

5.4.2. Parada de execução de programa do robô

- **Comando**

7	6	5	4	3	2	1	0	Byte
NOB=8 (decimal)								0
								1
DESTINATION ADDRES								2
SOURCE ADDRESS								3
FUNCTION CODE=3								4
0	0	0	0	0	0	0	1	5
FUNCTION SUFFIX								6
								7

- **Resposta**

7	6	5	4	3	2	1	0	Byte
NOB=8 (decimal)								0
(ou 10 em caso de erro)								1
DESTINATION ADDRES								2
SOURCE ADDRESS								3
FUNCTION CODE=3								4
0	0	0	0	0	0/1	0	1	5
FUNCTION SUFFIX								6
								7
ERROR CODE								8
								9

5.4.3. Leitura do registrador através do SC

- **Comando**

7	6	5	4	3	2	1	0	Byte
NOB=8 (decimal)								0
								1
DESTINATION ADDRES								2
SOURCE ADDRESS								3
FUNCTION CODE=6								4
0	0	0	0	0	0	0	1	5
FUNCTION SUFFIX								6
(0 para o registrador 0,..., 119 para o registrador 119)								7

- **Resposta**

7	6	5	4	3	2	1	0	Byte
NOB=8 (decimal)								0
(ou 10 em caso de erro)								1
DESTINATION ADDRES								2
SOURCE ADDRESS								3
FUNCTION CODE=6								4
0	0	0	0	0	0/1	0	1	5
FUNCTION SUFFIX								6
								7
REGISTER VALUE (2 bytes)								8
(ou ERROR CODE)								9

5.4.4. Escrita do registrador através do SC

- **Comando**

7	6	5	4	3	2	1	0	Byte
NOB=10 (decimal)								0
								1
DESTINATION ADDRES								2
SOURCE ADDRESS								3
FUNCTION CODE=14								4
0	0	0	0	0	0	0	1	5
FUNCTION SUFFIX								6
(0 para o registrador 0,..., 119 para o registrador 119)								7
REGISTER VALUE (2 bytes)								8
								9

- **Resposta**

7	6	5	4	3	2	1	0	Byte
NOB=8 (decimal)								0
(ou 10 em caso de erro)								1
DESTINATION ADDRES								2
SOURCE ADDRESS								3
FUNCTION CODE=14								4
0	0	0	0	0	0/1	0	1	5
FUNCTION SUFFIX								6
								7
ERROR CODE								8
								9

5.4.5. Requisição de status através do SC

- **Comando**

7	6	5	4	3	2	1	0	Byte
NOB=8 (decimal)								0
								1
DESTINATION ADDRES								2
SOURCE ADDRESS								3
FUNCTION CODE=19								4
0	0	0	0	0	0	0	1	5
FUNCTION SUFFIX								6
(Irrelevante)								7

- **Resposta**

7	6	5	4	3	2	1	0	Byte
NOB=48 (decimal)								0
								1
DESTINATION ADDRES								2
SOURCE ADDRESS								3
FUNCTION CODE=19								4
0	0	0	ORT	0	0	0	1	5
FUNCTION SUFFIX								6
								7
DUMMY								8
								9
DUMMY								10
								11
PROGRAM NUMBER								12
								13
INSTRUCTION NUMBER								14
								15

DUMMY	**16**
LR(1) - IR(1) - PU(2) - MODE (4)	**17**
POSITION OF ROBOT	**18**
	...
	47

- **ORT**:(orientation type) "0" para orientação de punho, e "1" para orientação de ferramenta;
- **DUMMY**: irrelevante;
- **PROGRAM NUMBER**: identificação do programa ativo;
- **INSTRUCTION NUMBER**: identificação da instrução ativa;
- **LR**: "0" para modo local, e "1" para modo remoto;
- **IR**: "0" para interrupção, e "1" para não permissão de interrupção;
- **PU**: "00" para unidade de programação no compartimento de controle, "01" para unidade de programação fora do compartimento de controle, e "11" para unidade de programação desconectada;
- **MODE**: bit 0 para *stand-by*, bit 1 para operação, bit 2 para programa em execução, bit 3 para parada de emergência.

5.4.6. Transferência de um Programa/Bloco para o IRB a partir do SC

- **Comando**

7	6	5	4	3	2	1	0	Byte
NOB								0
								1
DESTINATION ADDRES								2
SOURCE ADDRESS								3
FUNCTION CODE=1								4
0	0	0	0	0/1	0	0	1	5
FUNCTION SUFFIX								6
(0 - Bloco de Programas / 1 - Programas)								7
PROGRAM NUMBER								8
(de 0 a 9999)								9
BLOCK NUMBER								10
(de 0 a 9999)								11
ROBOT PROGRAM								12
								NOB-1

- **Resposta**

7	6	5	4	3	2	1	0	Byte
NOB=12								0
(10 se for ERROR CODE)								1
DESTINATION ADDRESS								2
SOURCE ADDRESS								3
FUNCTION CODE=1								4
0	0	0	0	0	0/1	1	0	5
FUNCTION SUFFIX=0/1								6
								7
PROGRAM NUMBER								8

(ou ERROR CODE)	9
BLOCK NUMBER	10
	11

5.4.7. Transferência de um programa/bloco de programa da memória do robô paro SC, inicializado a partir da Unidade de Programação

- **Comando**

7	6	5	4	3	2	1	0	Byte
NOB								0
								1
DESTINATION ADDRES								2
SOURCE ADDRESS								3
FUNCTION CODE=65								4
0	0	0	0	0/1	0	0	1	5
FUNCTION SUFFIX								6
(0 - Bloco de Programas / 1 - Programas)								7
PROGRAM NUMBER								8
(de 0 a 9999)								9
BLOCK NUMBER								10
(de 0 a 9999)								11
ROBOT PROGRAM								12
								NOB-1

- **Resposta**

7	6	5	4	3	2	1	0	Byte
NOB=12								0
(10 se for ERROR CODE)								1
DESTINATION ADDRESS								2
SOURCE ADDRESS								3
FUNCTION CODE=65								4
0	0	0	0	0	0/1	1	0	5
FUNCTION SUFFIX								6
								7
PROGRAM NUMBER								8
(ou ERROR CODE)								9
BLOCK NUMBER								10
								11

5.4.8. Mensagem de Status espontânea do IRB para o SC

7	6	5	4	3	2	1	0	Byte
NOB=48 (S2)								0
								1
DESTINATION ADDRES								2
SOURCE ADDRESS								3
FUNCTION CODE=127								4
0	0	0	0	ORT	0	0	3	5
FUNCTION SUFFIX								6
								7
ERROR CODE / REGISTER NUMBER								8
								9
ERROR SUB NUMBER / REGISTER								10
								11
PROGRAM NUMBER								12
								13
INSTRUCTION NUMBER								14
								15
GOFLAG / ACTUAL TCP (S3)								16
LR	IR	PU		M	O	D	E	17
COORDINATES for actual ROBOT								18
POSITION								19
(42 bytes em S2)								...
								47

ORT	*Tipo de Orientação:* • 0 = Orientação da Garra • 1 = Orientação da Ferramenta
Function Suffix	• 0 = SUCTRL (controle pertencente ao SC). • 1 = Parada de Emergência no sistema do robô. • 2 = Search Stop • 3 = Parada a partir da Unidade de Programação ou da caixa de controle. • 4 = Ao conectar, a Unidade de Programação estava ou não conectada à caixa de controle. • 5 = Unidade de Programação conectada/desconectada. • 6 = Mudança do Modo do Robô (LOCAL / REMOTE). • 7 = Erros do Sistema. • 8 = Startup do sistema do robô. • 9 = Erro de inicialização do sistema de visão. • 10 = Erro do programa de teste do sistema de visão. • 11 = Início de programa a partir da Unidade de Programação, Painel de Controle ou Entrada Digital. • 12 = Mudança do Modo de Operação para OPERATE. • 13 = Mudança do Modo de Operação para STAND BY.
Error Code	Erro do Sistema do Robô (somente quando FS=7).
Error sub number	Sub número para o Erro do Sistema do Robô (somente quando FS=7).
Register number	Número do Registrador de 0...119 (somente quando FS=0 - SUCTRL).
Register	Conteúdo do Registrador apontado pelo Número do Registrador

	(somente quando FS=0 - SUCTRL).
Program Number	Programa ativo atual.
Instruction Number	Número de instrução atual ativa dentro do programa.
GOFLAG	• 0 = Parada. • 1 = Contínuo (Somente para FS=0)
Actual TCP (S3)	TCP atual no sistema do robô.
MODE (bit 0-3)	***Modo de Operação:*** • Bit 0 - Stand-by. • Bit 1 - Operação. • Bit 2 - Programa em execução. • Bit 3 - Parada de Emergência.
PU (bit 4-5)	***Unidade de Programação:*** • 0 - In. • 1 - Out. • 2 - Desconectada.
IR (bit 6)	***Interrupção:*** • 0 - Permitida. • 1 - Não permitida.
LR (bit 7)	• 0 - Modo Local. • 1 - Modo Remoto
Coordinates	• X, Y, Z • Q1 - Q4 • R6 - R9

5.5. ERROR CODE - Códigos de Erro

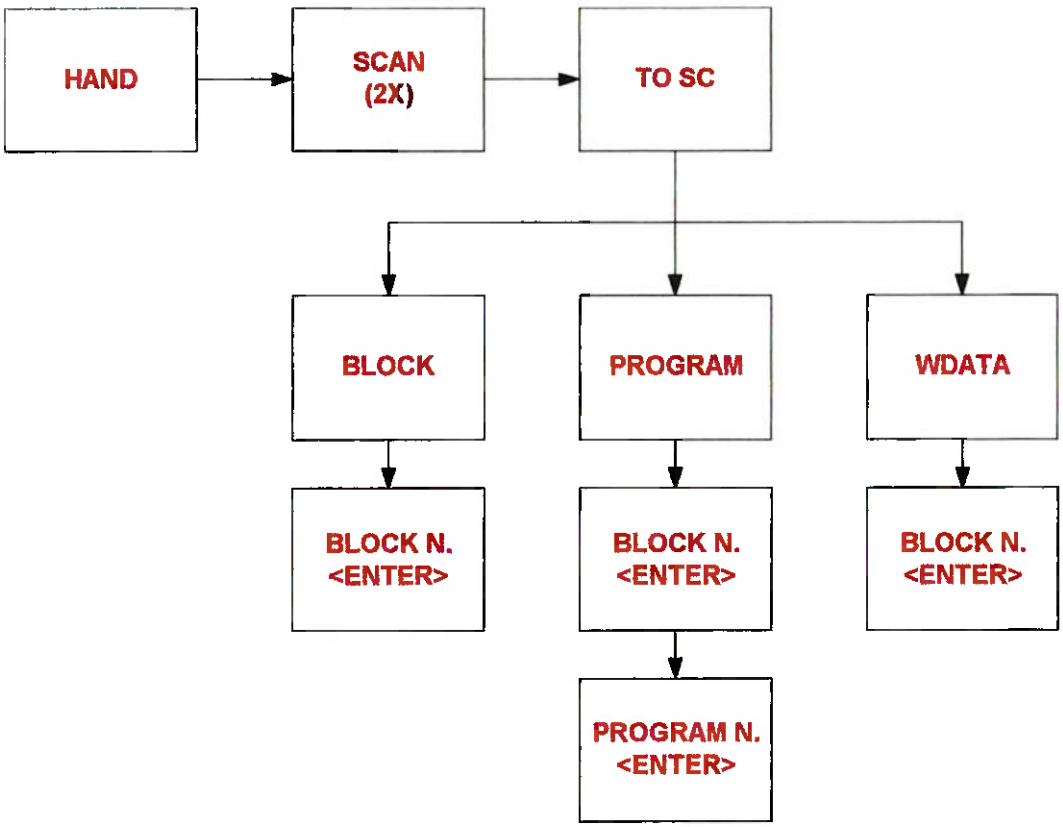
Toda vez que **RS** estiver "setado" em 1 numa resposta, esta inclui um código de erro.

Dos comandos definidos acima, os erros mais comuns podem ser:

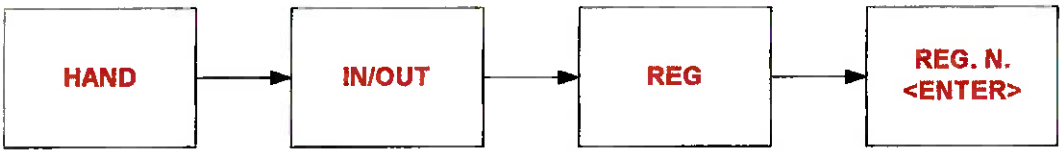
Código de Erro	Descrição
1	Programa está sendo executado.
2	*Function Suffix* incorreto.
3	Número de Programa incorreto.
4	Número de Bloco incorreto.
5	Unidade de Programação fora da unidade de controle.
6	Modo de operação do robô incorreta.
7	Programa inexistente.
13	Parada, Parada de Emergência
31	Continuação de programa não permitida.

Tabela 13 - Códigos de Erro

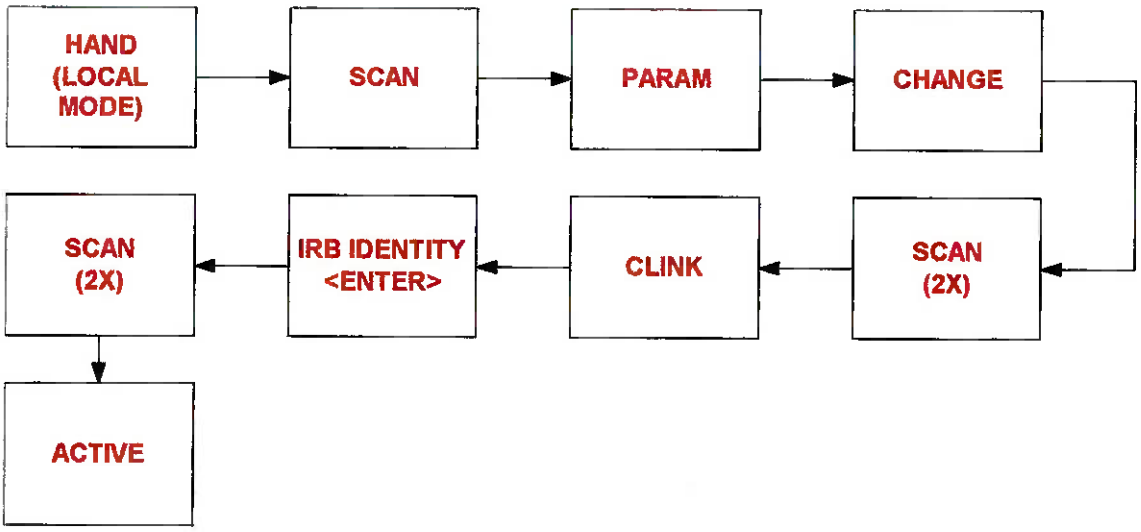
5.6. Como Transferir um Programa ao SC a partir da Unidade de Programação do IRB



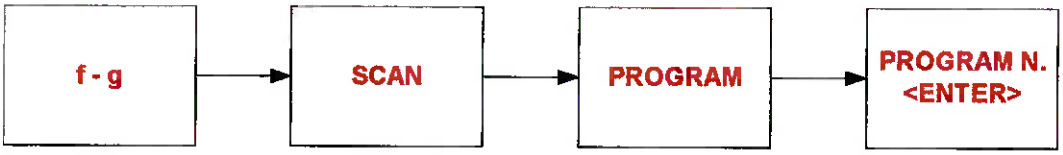
5.7. Como Verificar o Conteúdo de um Registrador Específico a partir da Unidade de Programação do Robô



5.8. Como Atribuir uma Identidade para o Robô a partir da Unidade de Programação do Robô



5.9. Como Iniciar um Programa Específico a partir da Unidade de Programação do Robô



6. COMUNICAÇÃO SERIAL

6.1. Introdução

Para que se estabeleça uma comunicação serial entre duas máquinas (PC/PC, PC/Torno, PC/Robô), é necessário que ambas possuam portas seriais e que "falem a mesma língua", ou seja, o mesmo protocolo de transferência de dados Particularmente no nosso caso, este protocolo já está implementado no robô; de forma que será necessário implementar o mesmo protocolo no PC.

Para isso, é necessário controlar três diferentes elementos de hardware:

- a porta serial RS232;
- transmissor /receptor assíncrono universal INS 8250;
- controlador de interrupção 8259.

Esse controle é realizado através de um software, que pode ser programado através de linguagens de programação, como por exemplo C ou Pascal.

Além disso, deve-se controlar as variações de velocidade de transmissão, paridade, bits de parada, bits de dados e a saída serial, entre outras coisas. Esses controles também são estabelecidos pelo software.

Assim, neste capítulo, será apresentada e exemplificada a norma **RS232**, devido ao fato desta ser um meio de comunicação amplamente utilizado entre máquinas industriais. Serão apresentados alguns conceitos básicos dos protocolos orientados a caracter, devido ao fato de um destes constituir o protocolo utilizado pelo presente robô ASEA.

6.2. Interrupções

A interrupção é uma ferramenta utilizada pelo sistema, de forma que o programa interrompa por um instante o seu fluxo normal de atividades, para executar uma certa

ação. Terminada a execução desta ação de interrupção, o programa automaticamente continua a execução a partir do ponto de onde parou.

Existem dois tipos de interrupção: **interrupção de hardware** e **interrupção de software**.

As **interrupções de hardware** ocorrem através de vários processos, como por exemplo: aperto de teclas, "tique" do relógio do sistema, entrada de dados por uma porta serial,... Elas originam-se no circuito do computador, e são controladas por um chip especial, o **controlador de interrupção 8259**.

As **interrupções de software** são geradas por programas que solicitam serviços e atividades especiais da BIOS e do DOS.

Na parte mais baixa da memória do PC existe uma tabela de vetores de interrupção, que consiste num array de endereços de memória. Esse array tem 1.024 bytes e contém endereços de todas as rotinas acionadas pelas interrupções.

Quando inicializada, uma interrupção obtém um endereço de memória da tabela de vetores de interrupção, passa direto para essa posição de memória, e executa a rotina localizada nessa posição. Cada endereço da tabela de vetores de interrupção é usado exclusivamente por uma única interrupção.

Interrupções		Deslocamento	*Funções*
Decimal	Hexadecimal		
8	8	0020h	Tique do relógio do sistema
9	9	0024h	Interrupção do Teclado
10	A	0028h	Sem uso
11	B	002Ch	Segunda saída serial (COM 2)
12	C	0030h	Primeira saída serial (COM 1)
13	D	0034h	Interrupção da unidade de disco rígido
14	E	0038h	Interrupção da unidade de disquete
15	F	003Ch	Interrupção da impressora

Tabela 14 - Algumas posições das interrupções de hardware na tabela de vetores de interrupção

6.3. A norma RS232

A norma RS232-C foi estabelecida pelo EIA - *Eletronic Industries Association* - em Washington DC, no ano de 1969. A RS232-C corresponde à descrição física de uma interface serial para comunicação de dados. Esta interface permite comunicação síncrona ou assíncrona, e também a comunicação *simplex*, *half-duplex* ou *full-duplex*.

Atualmente, a interface RS232 pode ser utilizada para comunicação do computador com outro computador, ou com quase todas as máquinas industriais existentes (robôs, CLP's, máquinas CNS's, ...). Os grandes motivos desta ampla gama de aplicações desta norma correspondem à sua grande versatilidade e ao seu custo relativamente reduzido.

A interface RS232 define, além do canal de comunicação de dados, outros canais para controle de fluxo de informação, na forma de intertravamento, para que não ocorra perda de dados importantes.

6.3.1. Comunicação Síncrona

Neste tipo de comunicação, são utilizados os circuitos de *clock* definidos na interface. Seu funcionamento não nos interessará neste caso, pois ela praticamente não é utilizada na comunicação entre máquinas industriais, através da interface RS232. Maiores detalhes sobre comunicação síncrona via interface RS232 podem ser encontradas detalhadamente nos textos apresentados na bibliografia, posteriormente.

6.3.2. Comunicação Assíncrona

Neste tipo de comunicação, os circuitos de *clock* não são utilizados. É apenas definida uma velocidade de transmissão a ser adotada pelas suas estações em comunicação.

Existe a possibilidade de escolha remota entre as duas taxas pré-definidas através do pino 23, mas isto também praticamente não é utilizado em comunicação entre duas máquinas industriais.

Os seguintes pinos são extremamente importantes na transmissão assíncrona:

- **Pino 1** Terra de proteção; deve ser a malha do cabo.
- **Pino 2** Dados sendo enviados desta interface.
- **Pino 3** Dados sendo enviados para esta interface.
- **Pino 4** Interface envia sinal, pedindo permissão para enviar dados.
- **Pino 5** Interface recebe sinal dando permissão para enviar dados.
- **Pino 6** Interface recebe sinal sobre *status* da outra ponta em comunicação - ativa/inativa.
- **Pino 7** Interface recebe sinal sobre *status* da outra ponta em comunicação, informando se o meio físico de transmissão está disponível ou não.
- **Pino 20** Interface envia sinal sobre seu *status* (ativo/inativo), para a outra ponta em comunicação.

O formato de transmissão dos dados devem possuir a seguinte estrutura:

- 1 *start-bit* (lógica 0);
- bits de dados (em geral de 5 a 8 bits);
- 1 bit de paridade (pode ou não existir; em caso afirmativo, pode ser par ou ímpar);
- 1, 1.5 ou 2 *stops-bit* (lógica 1).

A inserção dos dados neste formato é conhecida como encapsulamento.

Após a transmissão (último *stop-bit*), a linha de dados entra em condição inativa (lógica 1), até o início da nova transmissão, indicada por um *start-bit* (lógica 0).

A comunicação assíncrona é utilizada na maioria das máquinas industriais, geralmente com velocidade de transmissão (taxa) de 9.600 bauds (bits de informação/segundo).

6.3.3. Características Físicas

A conexão padrão para a norma RS232 é a tomada de 25 pinos DB25. Como nem todos os pinos precisam ser implementados para uma comunicação por esta interface, é também utilizado o formato de tomada com 9 pinos DB9.

Todos os sinais transmitidos por esta interface são binários, podendo assumir níveis de voltagem positivos ou negativos, relativos a uma terra comum. Nos **canais de transmissão de dados**, níveis positivos representam a *lógica "0"*, enquanto que os níveis negativos, a *lógica "1"*. Nos **canais para controle de fluxo de informação**, são considerados *ativos* os níveis positivos, e *inativos* os níveis negativos.

EMISSOR

15V ~ 5V	**"0" ATIVO**
5V ~ -5V	**INVÁLIDO**
-5V ~-15V	**"1" INATIVO**

RECEPTOR

25V ~3V	**"0" ATIVO**
3V ~-3V	**INVÁLIDO**
-3V ~25V	**"1" INATIVO**

Para reduzir os problemas causados pela presença de ruído na transmissão, é definido um maior intervalo válido para a recepção do sinal.

Assim, devido ao surgimento do ruído na transmissão, são sugeridos:

- comprimento máximo do cabo de transmissão de 15 metros;
- velocidade máxima de transmissão de 20.000 bauds.

6.3.4. Implementação Física

Para cada um dos pinos do circuito (cabo) de ligação, é definida uma função específica. Os circuitos secundários representam um canal extra independente do primário, mas com a mesma função.

PINO	SÍMBOLO	SENTIDO	DESCRIÇÃO
1	PG	-	Terra de proteção
2	TxD	→	Saída de dados
3	RxD	←	Entrada de dados
4	RTS	→	*Request to Send*
5	CTS	←	*Clear to Send*
6	DSR	←	*Data Set Ready*
7	SG	-	Terra dos sinais
8	DCD	←	*Data Carrier Detected*
9	-	-	Não definido
10	-	-	Não definido
11	-	-	Não definido
12	SDCD	←	*Data Carrier Detected* secundário
13	SCTS	←	*Clear to Send* secundário
14	STxD	→	Saída de dados secundário
15	-	←	Clock em que se deve transmitir
16	SRxD	←	Entrada de dados secundário
17	-	←	Clock em que se deve receber
18	-	-	Não definido
19	SRTS	→	*Request to Send* secundário

20	**DTR**	**→**	***Data Terminal Ready***
21	**SQD**	**←**	**Detetor da qualidade do sinal**
22	**RI**	**←**	**Indicador de chamada**
23	**-**	**← / →**	**Indicar qual *baud rate***
24	**-**	**→**	**Clock de transmissão síncrona**
25	**-**	**-**	**Não definido**

Tabela 15 - Função dos Pinos do Circuito

Para uma *transmissão* a partir desta interface, os seguintes pinos devem estar ativos:

- RST (4);
- CTS (5);
- DSR (6);
- DTR (20).

Para a recepção, os seguintes pinos devem estar ativos:

- DSR (6);
- DCD (8);
- DTR (20).

Em ambientes half-duplex, os circuitos RTS e DCD não podem ficar simultaneamente ativos.

6.3.5. Divisão Das Linhas

LINHAS	DEFINIÇÃO	DESCRIÇÃO
2	TXD – Transmitted Data	Transmite Dados
3	RXD – Received Data	Recebe Dados
4	RQT – Request to Send	Handshaking
5	CTS – Clear to Send	Handshaking
6	DSR – Data Set Ready	Handshaking
20	DTR – Data Terminal Ready	Handshaking
7	SG – Signal Ground	Sinal Terra
8	CD – Carrier Detect	Presença de Portadora
22	RI – Ring Indicator	Telefone Chamando

Tabela 16 - Divisão das Linhas

6.3.6. Comunicação Simples

Quando um periférico somente transmite e outro somente recebe, a comunicação necessita apenas de dois sinais em cada periférico: TXD e SG (linhas 2 e 7) no periférico que transmite (por exemplo, o PC), e RXD e SG (linhas 3 e 7) no periférico que recebe (por exemplo, modem).

6.4. Handshaking

Handshaking corresponde ao método pelo qual periféricos que se comunicam possam controlar o fluxo de transmissão de dados entre eles.

Podemos imaginar, por exemplo, que um periférico recebe os dados mais rápido do que tem capacidade de processá-los. Nesse caso, ele deve ser capaz de avisar ao transmissor para parar a transmissão, até que tenha processado os caracteres já recebidos.

Isso ocorre, por exemplo, quando mudamos dados de um computador para outro que não possui uma velocidade de processamento de dados suficiente para processar os dados os quais estão sendo recebidos.

Outro exemplo ocorre quando enviamos dados a uma impressora serial a qual não imprime na velocidade de envio.

Nesses casos, o periférico pode transmitir um ou dois sinais de handshaking, indicando que a sua capacidade de recepção se esgotou, solicitando assim uma parada temporária do envio de dados. A linha 20 e/ou 4 é usada pelo periférico terminal (PC) e a linha 6 e/ou 5 é usada pelo periférico modem para essa finalidade.

6.4.1. Comunicação Nas Duas Direções

Caso cada um dos dois periféricos interligados tanto transmita como receba, o número mínimo de linhas necessárias em cada periférico passa a ser 3: TXD, RXD e SG. Quando se deseja controlar o fluxo de transmissão de dados (handshaking), devem ser adicionadas as linhas DSR e DTR em cada periférico. Quando uma Segunda linha de handshaking for desejada, devem ser adicionadas as linhas RQS e CTS em cada periférico. Assim, o número total de linhas sobe para 7.

Duas outras linhas adicionais são geralmente usadas. A linha 8 (CD), a qual indica a presença de portadora, isto é, indica que o modem da outra ponta está pronto. E a linha

22 (RI), que indica que o telefone está chamando, ou seja, que o modem está sendo chamado pelo periférico remoto.

6.4.2. Comunicação Micro A Micro

A conexão de dois periféricos terminais (micro e micro) ou de dois periféricos modems é chamada NULL MODEM. Esse tipo de ligação requer uma atenção especial pelo fato de os dois periféricos transmitirem pela mesma linha e receberem também pela mesma linha. Ou seja, o problema surge porque o primeiro periférico transmite pela linha 2 e recebe pela linha 3 e o segundo periférico também transmite pela linha 2 e recebe pela linha 3.

Portanto, é necessário que o cabo de ligação tenha essas linhas cruzadas para permitir que a linha ligada ao pino 2 do primeiro periférico seja ligada ao pino 3 do segundo periférico, e a linha ligada ao pino 2 do segundo periférico esteja ligada ao pino 3 do primeiro.

As linhas 2 e 3 cruzadas permitem que um periférico transmita pela linha 2 e os dados transmitidos sejam recebidos pela linha 3 de outro periférico.

Caso se ligue o pino 2 com o 2, o 3 com o 3,... estará sendo ligada saída com saída e entrada com entrada, e assim, a conexão certamente não será bem sucedida.

Para conectar os dois micros. Pode-se utilizar um cabo serial com as linhas cruzadas, ou colocar um conector que conecte os dois cabos e faça o cruzamento necessário internamente. Em qualquer dos casos, o cabo ou conector é chamado de null modem, pois fazem o papel de dois modems (terminal: modem : modem : terminal).

O cabo de ligação deve seguir as seguintes especificações mínimas:

1. O pino 2 do primeiro micro deve ser ligado ao pino 3 do segundo.
2. O pino 4 do primeiro micro deve ser ligado ao pino 5 do segundo.

3. O pino 20 do primeiro micro deve ser ligado ao pino 6 do segundo.

6.4.3. As Linhas De Um Cabo Null Modem

Pode-se fazer um cabo para interligar dois micros pela placa serial, soldando os fios nos conectores, seguindo o seguinte esquema:

Primeira Ponta		Outra Ponta
2	→	3
3	←	2
4	→	5
5	←	4
6	←	20
7	↔	7
20	→	6

6.5. UART

O coração da porta serial é o chip chamado UART (*Universal Asynchronous Receiver Transmmitter*). O trabalho do UART corresponde tanto a transformar cada byte de informação paralela numa corrente de bits seriais, como também executar a operação inversa.

O UART deve ser configurado conforme as necessidades da nossa comunicação, antes que as informações possam ser transmitidas ou recebidas. Os elementos de configuração que devem ser considerados são os seguintes:

6.5.1. Taxa De Transmissão

A velocidade de transmissão deve ser escolhida considerando-se a velocidade de recepção do equipamento com o qual vamos nos comunicar.

Podemos escolher uma entre várias taxas de transmissão suportadas pelo UART para o envio dos dados. A escolha comum com modems é de 300 bits por segundo (bps), 1.200 bps ou 2.400 bps. O UART pode enviar e receber dados em várias outras velocidades, chegando até a 115.200 bps.

6.5.2. Número De Bits De Dados

O número de bits de dados pode ser 7 caso os dados a serem transmitidos estiverem no formato texto e 8 no caso dos dados estiverem no formato binário.

6.5.3. Paridade

Caso forem transmitidos 7 bits de dados, o oitavo bit pode ser usado para indicar paridade.

A paridade pode ser par, ímpar ou nenhuma. Se for escolhida paridade par, os bits de dados transmitidos mais o bit de paridade formarão um número par. Se for escolhida paridade ímpar, os bits de dados transmitidos mais o bits de paridade formarão um número ímpar.

O bit de paridade age como um simples esquema de detecção de erro. Se um dos 7 bits de dados é perdido na transmissão, o bit de paridade não estará correto e pode ser detectado pelo receptor.

Muitas vezes o bit de paridade não é utilizado. Nesse caso, deve-se avisar o UART desse fato.

6.5.4. Start Bit e Stop Bit

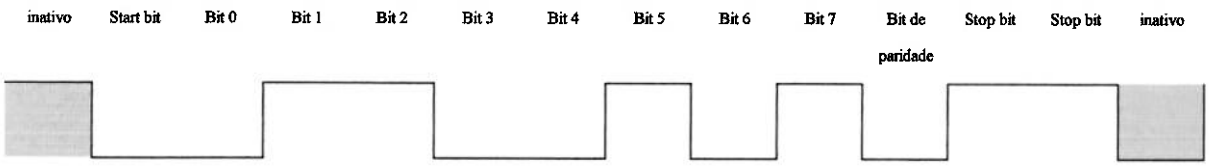
Quando a porta serial recebe um caractere, ela deve ser capaz de identificar quando o envio do caractere começa e quando termina. Isso é feito por meio do bit de início (***start bit***) e por um ou dois bits de finalização (***stop bits***).

Quando não há caracteres sendo mandados, o estado dos bits da porta será sempre 1. Para sinalizar a chegada de um caractere, enviamos um bit de início com o valor zero. Em seguida, são enviados os 7 ou 8 bits de dados, o bit de paridade e finalmente um ou dois bits de finalização com o valor um.

6.5.5. Seqüência De Sinais Para A Transmissão De Um Byte

Quando enviamos um bit ligado (1) para a porta serial, estamos colocando o estado da linha alto, e quando enviamos um bit desligado (0), estamos abaixando a linha.

O esquema do envio dos bits para a transmissão de um byte é representado abaixo:



6.6. Configuração Da Porta Serial – Uart

O UART contém registradores internos que devem ser programados conforme os elementos citados anteriormente, para especificar as opções desejadas.

Os endereços desses registradores estão relacionados com as portas COM01 e COM02, conforme a tabela a seguir:

COM01	COM02	Descrição
0x3F8	0x2F8	Buffer de Transmissão
0x3F8	0x2F8	Buffer de Recepção
0x3F8	0x2F8	Byte mais baixo da taxa de transmissão
0x3F9	0x3F9	Byte mais alto da taxa de transmissão
0x3F9	0x2F9	Permite habilitar interrupções
0x3FA	0x2FA	Identifica última interrupção ocorrida
0x3FB	0x2FB	Controla características da comunicação
0x3FC	0x2FC	Controla sinais de modem
0x3FD	0x2FD	Status
0x3FE	0x2FE	Status (Modem)

Tabela 17 - Registradores da COM1 e COM2

6.6.1. Registrador De Buffer De Transmissão

O registrador de buffer de transmissão armazena o próximo caractere a ser transmitido. O caractere é colocado nesse registrador pelo programa a ser implementado, e quando é transmitido, o registrador de status indica o fato.

6.6.2. Registrador De Buffer De Transmissão

O registrador do buffer de recepção armazena o último caractere recebido. Quando o caractere for lido pelo programa a ser implementado, o registrador de status indicará que o buffer de recepção está vazio até que um outro caractere seja recebido.

Caso um segundo caractere for recebido antes que o primeiro seja lido, um erro de ***overrun*** ocorrerá.

6.6.3. Registradores Da Taxa De Transmissão

O número a ser enviado para o UART para informar a taxa de transmissão desejada não corresponde ao número de bps (bits por segundo) o qual desejamos. Esse valor é chamado divisor e possui a seguinte relação com o número de bps:

Visto que esse valor é um número de 16 bits e cada registrador do UART é de 8 bits, precisamos enviar nosso valor em dois registradores: o byte menos significativo para o registrador de endereço 0x03F8 e o byte mais significativo para o registrador de endereço 0x03F9 (para o caso da COM02, os endereços correspondentes são 0x02F8 e 0x02F9).

6.6.4. Registrador Que Permite Habilitar Interrupções

É possível instruir o UART para gerar um sinal de interrupção sempre que certos eventos ocorrerem. Esse registrador é utilizado para informar ao UART quais eventos causarão a interrupção.

O método o qual utilizaremos aqui **não necessita habilitar tais interrupções**, pois o programa examinará continuamente o registrador de status para verificar o que está ocorrendo.

Os bits desse registrador correspondem às seguintes solicitações de interrupções:

Bit	Interrupção Selecionada
0	Dado presente
1	Registrador de transmissão vazio

2	Linha de Status
3	Status do Modem
4 - 7	Sempre nulos.

Tabela 18 - Registradores de Habilitação das Interrupções

6.6.5. Registrador Que Identifica A Última Interrupção Ocorrida

O registrador de identificação de interrupções informa sobre o status de interrupções pendentes. Caso não haja interrupções pendentes, o bit 0 estará ligado (1). Caso este bit estiver desligado (0), os bits 1 e 2 indicarão qual interrupção está pendente, conforme a tabela a seguir:

Bit 1	Bit 2	Interrupção Selecionada
0	1	Dado presente
1	0	Registrador de transmissão vazio
1	1	Linha de Status
0	0	Status do modem

Tabela 19 - Registrador que identifica as últimas interrupções

Os bits de 3 a 7 estarão sempre desligados.

6.6.6. Registrador De Controle Das Características De Comunicação

O registrador de controle é utilizado para selecionar os parâmetros de comunicação. O significado de cada bit do registrador é mostrado na tabela a seguir:

Bit	Objetivo
0 e 1	***Número de bits de dados:*** • 00 – 5 bits; • 01 – 6 bits; • 10 – 7 bits; • 11 – 8 bits.
2	***Stop bits:*** • 0 – Um stop bit; • 1 – Dois stop bits.
3	***Paridade:*** • 0 – Sem paridade; • 1 – Permite paridade.
4	***Seleciona Paridade (ignorado se bit 3=0):*** • 0 – Paridade ímpar; • 1 – Paridade par.
5	***Paridade um:*** • 0 – 0 bit de paridade será 1lógico; • 1 – 0 bit de paridade será 0 lógico.
6	***Usado para gerar um comando Break:*** • 1 – Força o envio de um 0 lógico até que esse bit seja desligado.
7	***Taxa de Transmissão:*** • 0 – Acessa os registradores de velocidade nas operações de leitura e envio. • 1 – Acessa os registradores de interrupção.

Tabela 20 - Registrador de Controle das Características de Comunicação

6.6.7. Registrador De Controle De Sinais Do Modem

O registrador de controle de sinais do modem é utilizado para selecionar os parâmetros de comunicação do modem. O significado de cada bit do registrador é mostrado na tabela a seguir:

Bit	Objetivo
0	Data Terminal Ready
1	Request to Send
2	Carrier Detect
3	Ring Indicator
4	Permite teste de diagnóstico
5 - 7	Sempre zero.

Tabela 21 - Registrador dos sinais de controle do modem

6.6.8. Registrador De Status

O Registrador de Status é utilizado para obter informações que dizem respeito aos dados recebidos ou transmitidos. O significado de cada bit do registrador é descriminado na tabela a seguir:

Bit	Objetivo se o bit estiver ligado (1)
0	**Dados Prontos (Data Ready)** Um dado foi recebido; esse bit permanecerá ligado até que o caractere seja lido.
1	**Erro de Overrun** Outro dado foi recebido antes que o anterior fosse removido do buffer; isso indica que os dados forma recebidos mais rápido que o tempo necessário para processá-los.
2	**Erro de Paridade** Um erro de paridade ocorreu no dado recebido.
3	**Erro de Framing** O caractere recebido não possui o número de stop bits selecionados.

4	**Interrupção por Break** Um break foi recebido; isto significa que os bits recebidos possuem o valor zero para mais bits que o tamanho selecionado.
5	**Registrador de Transmissão Vazio** O UART está pronto para receber um novo caractere para ser transmitido.
6	**Registrador de Conversão Vazio** O Registrador de conversão paralelo para a serial está pronto para receber outro dado.
7	Permanentemente zero.

Tabela 22 - Registradores de Status

6.6.9. Registrador De Status Do Modem

O registrador de Status do modem é utilizado para obter informações que dizem respeito às linhas de handshaking. O significado de cada bit do registrador é mostrado na tabela a seguir:

Bit	Objetivo se o bit estiver ligado (1)
0	Mudança na linha "Clear to Send"
1	Mudança na linha "Data Set Ready"
2	Mudança na linha "Ring Indicator"
3	Mudança na linha "Signal Detect"
4	Linha "Clear to Send" OK
5	Linha "Data Set Ready" OK
6	Linha "Ring Indicator" OK
7	Linha "Signal Detect" OK

Tabela 23 - Registradores de Status do modem

6.7. Erros na Camada Física

Alguns erros de comunicação podem ser identificados ainda na camada física da transmissão, tais como: erros de paridade, erros de sobreposição e erros de enquadramento. Normalmente, esses tipos de erro são tratados pelo próprio software de controle da interface RS232.

6.8. Protocolos de Ligação Baseados em Caractere

Estes protocolos baseiam-se em caracteres especiais, os quais se diferenciam dos dados. Estes caracteres possuem funções específicas, que são interpretadas e executadas.

Alguns dos principais objetivos destes protocolos são citados a seguir:

- Proporcionar segurança na transmissão de dados;
- Criar métodos para iniciar e finalizar a transmissão;
- Permitir a retransmissão de dados que foram transmitidos previamente com problemas.

As funções básicas dos protocolos de ligação são as seguintes:

1. Controle de quadro: associado à divisão dos dados em blocos chamados telegramas, com a introdução de um caractere de início e fim;
2. Controle de erro: associado à detecção de erros e à rotinas que permitam a retransmissão de telegramas transmitidos com problemas;
3. Controle de inicialização: associado ao estabelecimento da conexão que estava inativa;
4. Controle de fluxo: associado à capacidade ou incapacidade de transmissão;
5. Gerenciamento da ligação: associado à definição do sentido de comunicação e à definição de qual terminal está enviando e qual é o que deve receber dados;

6. Transparência: associada à possibilidade de transmissão de qualquer tipo de mensagem;
7. Reconhecimento de anomalias: outro nível de detecção de erros, associado principalmente à detecção de seqüências ilegais e *time-outs*.

Em geral, os caracteres utilizados nestes protocolos são os 32 caracteres iniciais da tabela ASCII. O protocolo pode fazer uso diferenciado destes caracteres, de acordo com as necessidades de quem o implementou.

Um aspecto interessante corresponde ao método utilizado para prover a transparência em muitos protocolos orientados à caractere. Este método baseia-se no caracter **DLE** (*Data Link Escape*). O método consiste em enviar antes do caracter de controle um caracter DLE; desta forma, o protocolo "entende" que o caracter após o DLE não é um dado, mas sim um caracter de controle. Caso um byte de dado coincidir com o caracter DLE, são enviados 2 caracteres DLE's em seqüência (como é o caso do protocolo do robô em estudo).

Os caracteres padrões de controle da tabela ASCII são os seguintes:

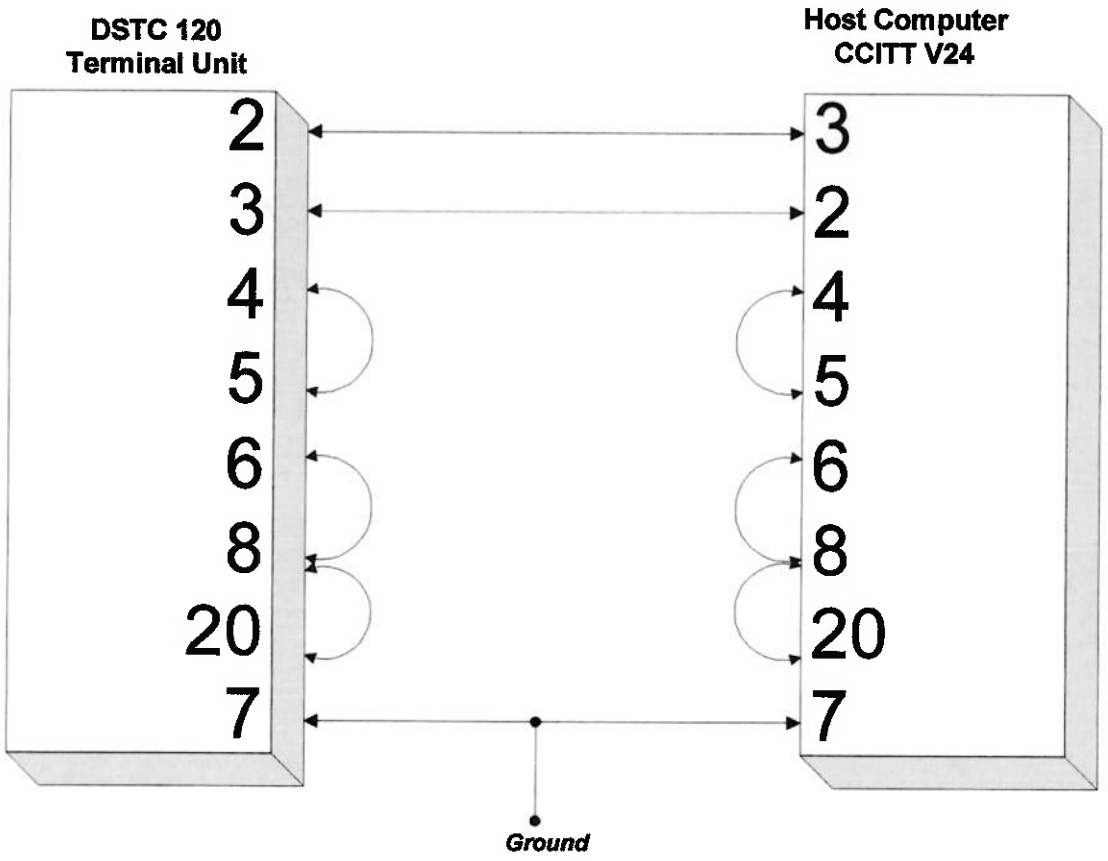
ASCII	DECIMAL	DESCRIÇÃO
NUL	0	NULL
SOH	1	START HEADING
STX	2	START TEXT
ETX	3	END TEXT
EOT	4	END OF TRANSMISSION
ENQ	5	ENQUIRE
ACK	6	POSITIVE ACKNOWLEDGE
BEL	7	BELL
BS	8	BACK SPACE
HT	9	HORIZONTAL TAB
LF	10	LINE FEED
VT	11	VERTICAL TAB

FF	12	FORM FEED
CR	13	CARRIAGE RETURN
SO	14	SHIFT OUT
SI	15	SHIFT IN
DLE	16	DATA LINK ESCAPE
DC1	17	DEVICE CONTROL 1
DC2	18	DEVICE CONTROL 2
DC3	19	DEVICE CONTROL 3
DC4	20	DEVICE CONTROL 4
NAK	21	NEGATIVE ACKNOWLEDGE
SYN	22	SYNCHRONOUS IDLE
ETB	23	END TRANS BLOCK
CAN	24	CANCEL
EM	25	END MEDIUM
SUB	26	SUBSTITUTE
ESC	27	ESCAPE
FS	28	FILE SEPARATOR
GS	29	GROUND SEPARATOR
RS	30	RECORD SEPARATOR
US	31	UNIT SEPARATOR
SP	32	SPACE
DEL	132	DELETE

Tabela 24 - Caracteres padrões da tabela ASCII

6.9. Solução Indicada para o Handshaking do Sistema

Segundo consta no manual do Robô IR B6da ASEA, as ligações das vias do cabo de comunicação sugerida corresponde ao esquema demonstrado a seguir:



7. TESTES

7.1. Procedimentos básicos para os testes

- Ligar o Notebook;
- Ligar o robô e sincronizá-lo (através da sua unidade de programação), conforme consta em seu manual;
- Conectar os dois equipamentos por meio de um cabo de confecção própria, conectado às saídas seriais dos respectivos equipamentos (notebook: COM01 e robô: canal DSTC120);
- Executar nas duas estações o programa de comunicação serial implementado.

7.2. Testes Preliminares

Os primeiros testes realizados foram os relacionados à comunicação serial entre dois micros PC's, sem utilizar os protocolos particulares do robô ASEA. Implementamos então um programa simples, de forma a verificarmos como funcionava a interface serial, bem como a comunicação.

Assim, após testarmos os programas implementados em linguagem C através da comunicação serial entre dois microcomputadores, passamos a testá-los no Laboratório da Escola Politécnica da USP, no qual se encontra o robô em estudo.

Para tanto, solicitamos ao Prof. Dr. Lucas Moscato um notebook de forma a facilitar nossa locomoção ao laboratório; os primeiros testes realizados com o robô foram somente de recepção de dados seriais no notebook.

Com este segundo teste, pudemos concluir que os dados enviados pelo robô estavam bastante coerentes com o que era esperado, e então pudemos passar para o passo

seguinte, o qual corresponde à implementação no micro dos protocolos particulares do robô.

7.3. Testes dos Protocolos do Robô

Nesta segunda fase de testes no laboratório, após implementados os protocolos no micro, nos propusemos a tentar receber dados e programas da saída serial do robô, encontrando vários problemas os quais serão listados abaixo.

7.3.1. Problemas iniciais encontrados

- Incoerência dos dados recebidos;
- Ambigüidade dos programas recebidos;
- Erros de overrun.

7.4. Soluções Adotadas

A primeira solução adotada pelo grupo foi a de adicionar os sinais de handshaking (os quais foram acima exemplificados) ao software, e também confeccionar um outro cabo com as devidas ligações dos pinos referentes ao sinais de controle desejados.

Ao testarmos o novo programa modificado, chegamos à conclusão que os sinais de handshaking não foram suficientes nem tampouco eficazes para a solução dos problemas acima explicitados.

Ao consultar nosso orientador e também o manual das ligações elétricas das placas do robô, constatamos enfim que o mesmo não utilizava os sinais de handshaking que estávamos implementando, e então confeccionamos um novo cabo com as ligações dos pinos sugeridas pelo fabricante do robô.

Outro fato bastante relevante ao processo de testes, foi o de verificar através de um osciloscópio digital existente nos laboratórios da faculdade, a forma dos pulsos dos sinais de saída tanto do robô quanto do micro.

A conclusão que chegamos foi o de que os sinais correspondiam exatamente ao que estávamos programando na saída do micro, e ao que estávamos esperando receber do terminal do robô.

Feitas todas as modificações necessárias tanto no cabo quanto no programa implementado, partimos para uma nova bateria de testes no laboratório.

7.5. Testes Finais Realizados

Como especificado no início do projeto, nossos principais objetivos neste trabalho são:

- Envio de um valor desejado a um registrador específico do robô;
- Início de um programa específico contido na memória do robô;
- Parada de um programa em execução no robô (e possibilidade de reiniciá-lo após a sua parada).
- Transferência de Programas entre os dois terminais;

Com relação aos três primeiros itens propostos, nosso programa de comunicação obteve êxito, sendo possível visualizar as ações geradas e os valores transmitidos.

Mas com relação ao último item – transferência de programas entre os dois terminais – não foi possível concluí-lo, pois durante os testes, o robô apresentou problemas os quais foram impossíveis de serem solucionados.

8. ESPECIFICAÇÃO DO PROGRAMA (DIAGRAMAS NS)

8.1. Main

<table>
<tr><td colspan="8">Enquanto não quiser sair</td></tr>
<tr><td></td><td colspan="7">Monta a Tela</td></tr>
<tr><td></td><td colspan="7">Configura a Porta Serial</td></tr>
<tr><td></td><td colspan="7">Enquanto nenhuma tecla for pressionada</td></tr>
<tr><td></td><td></td><td colspan="6">Verifica o Status da Serial</td></tr>
<tr><td></td><td></td><td colspan="6">Existem dados novos?</td></tr>
<tr><td></td><td></td><td colspan="3">S</td><td colspan="3">N</td></tr>
<tr><td></td><td></td><td colspan="3">Receber_da_Serial</td><td colspan="3"></td></tr>
<tr><td></td><td colspan="7">Verifica qual tecla foi pressionada</td></tr>
<tr><td></td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>Outro</td></tr>
<tr><td></td><td>Comunica</td><td>Comunica</td><td>Comunica</td><td>Comunica</td><td>Comunica</td><td>Saída</td><td></td></tr>
<tr><td></td><td>Envia Programa</td><td>Inicia Programa</td><td>Para Programa</td><td>Valor para Registrador</td><td>Recebe Programa</td><td>Sair</td><td></td></tr>
</table>

8.2. Configura a Porta Serial

Pergunta qual a porta serial a ser utilizada na comunicação
Envia para o registrador de velocidade o valor da taxa de transmissão
Envia para o registrador de configuração de dados a configuração padrão do robô (paridade, n° de stop bits, n° de bits de dados)

8.3. Receber_da_Serial

Recebe dado da porta serial					
Verifica se o sinal corresponde ao **ENQ**					
N	**S**				
	Envia **ACK**				
	Espera				
	Recebe dado da porta serial				
	Enquanto OK = 0				
	Verifica se o sinal corresponde ao STX				
	N	**S**			
	Se for DLE	Para I=1 ate 128			
	Espera		**Espera**		
	Recebe dado da serial		Recebe o dado da porta serial		
	É EOT		Joga o dado no vetor		
	OK = 1		O dado corresponde ao caracter DLE?		
			S	**N**	
			DLE=1	O dado corresponde ao ETX?	
				N	**S**
				DLE=0	I=128
	Espera				
	Recebe dado da serial (BCS)				
	Calcula o BCS dos dados enviados				
	BCS recebido = BCS calculado?				
	N	**S**			
	Envia NACK	Joga os dados no arquivo			
		Envia ACK			
Finaliza a transmissão					

8.4. Envia Programa

Monta a Tela
Envia o programa através da porta serial escolhida

8.5. Iniciar Programa

<table>
<tr><td colspan="3">OK=0</td></tr>
<tr><td colspan="3">Enquanto OK=0</td></tr>
<tr><td></td><td colspan="2">Monta a tela</td></tr>
<tr><td></td><td colspan="2">Dá a opção de escolha entre reiniciar ou iniciar um programa</td></tr>
<tr><td></td><td colspan="2">A opção escolhida foi a de reiniciar o programa do robô?</td></tr>
<tr><td></td><td>S</td><td>N</td></tr>
<tr><td></td><td>Ajusta adequadamente o sufixo da função</td><td>Requisita o número do programa a ser iniciado</td></tr>
<tr><td></td><td></td><td>Ajusta o sufixo e o número da função</td></tr>
<tr><td></td><td colspan="2">Comunica</td></tr>
<tr><td></td><td colspan="2">Envia o número de bytes do telegrama</td></tr>
<tr><td></td><td colspan="2">Espera</td></tr>
<tr><td></td><td colspan="2">Envia o endereço de destino</td></tr>
<tr><td></td><td colspan="2">Espera</td></tr>
<tr><td></td><td colspan="2">Envia o endereço fonte</td></tr>
<tr><td></td><td colspan="2">Espera</td></tr>
<tr><td></td><td colspan="2">Envia o código da função correspondente</td></tr>
<tr><td></td><td colspan="2">Espera</td></tr>
<tr><td></td><td colspan="2">Envia o sufixo da função</td></tr>
<tr><td></td><td colspan="2">Espera</td></tr>
<tr><td></td><td colspan="2">Envia o número do programa</td></tr>
<tr><td></td><td colspan="2">Espera</td></tr>
<tr><td></td><td colspan="2">Envia DLE</td></tr>
<tr><td></td><td colspan="2">Espera</td></tr>
<tr><td></td><td colspan="2">Envia ETX</td></tr>
<tr><td></td><td colspan="2">Espera</td></tr>
<tr><td></td><td colspan="2">Envia o BCS</td></tr>
<tr><td></td><td colspan="2">Espera</td></tr>
<tr><td></td><td colspan="2">Envia DLE</td></tr>
<tr><td></td><td colspan="2">Espera</td></tr>
<tr><td></td><td colspan="2">Envia EOT</td></tr>
<tr><td></td><td colspan="2">Espera</td></tr>
<tr><td></td><td colspan="2">OK=1</td></tr>
<tr><td colspan="3">Verifica se os dados foram enviados com sucesso</td></tr>
<tr><td colspan="2">S</td><td>N</td></tr>
<tr><td colspan="2">Exibe na tela a mensagem de “Comando executado com sucesso!”</td><td>Exibe na tela a mensagem “Ocorreu um erro!”</td></tr>
<tr><td colspan="2"></td><td>Fornece ao usuário o número do ERROR CODE</td></tr>
</table>

8.6. Parar um Programa

OK=0	
Enquanto OK=0	
	Monta a tela
	Ajusta adequadamente o sufixo da função correspondente
	Comunica
	Envia o número de bytes do telegrama
	Espera
	Envia o endereço de destino
	Espera
	Envia o endereço fonte
	Espera
	Envia o código da função correspondente
	Espera
	Envia o sufixo da função
	Espera
	Envia DLE
	Espera
	Envia ETX
	Espera
	Envia o BCS
	Espera
	Envia DLE
	Espera
	Envia EOT
	Espera
	OK=1
Verifica se os dados foram enviados com sucesso	
S	**N**
Exibe na tela a mensagem de "Comando executado com sucesso!"	Exibe na tela a mensagem "Ocorreu um erro!"
	Fornece ao usuário o número do ERROR CODE

8.7. Enviar Valor Para um Registrador

OK=0	
Enquanto OK=0	
	Monta a tela
	Pergunta ao usuário o número do registrador a ser referido o valor
	Param[0]=número do registrador definido pelo usuário
	Pergunta ao usuário o valor a ser enviado ao registrador definido anteriormente
	Param[3]=valor do registrador
	Comunica
	Envia o número de bytes do telegrama
	Espera
	Envia o endereço de destino
	Espera
	Envia o endereço fonte
	Espera
	Envia o código da função correspondente
	Espera
	Envia o sufixo da função (Param[0])
	Espera
	Envia o valor do registrador (Param[1])
	Espera
	Envia DLE
	Espera
	Envia ETX
	Espera
	Envia o BCS
	Espera
	Envia DLE
	Espera
	Envia EOT
	Espera
	OK=1
Verifica se os dados foram enviados com sucesso	
S	**N**
Exibe na tela a mensagem de "Comando executado com sucesso!"	Exibe na tela a mensagem "Ocorreu um erro!"
	Fornece ao usuário o número do ERROR CODE

8.8. Espera

Enquanto o registrador da UART não liberar o status que determina se os dados estão prontos	
	A porta serial continua a receber pela serial o dado correspondente ao status do registrador

8.9. Comunica

Monta a Tela				
Imprime na tela "Efetuando a comunicação"				
OK=0				
Enquanto OK=0				
	Envia para o registrador de status que os dados ainda não estão disponíveis			
	Envia ENK para o robô			
	Espera			
	Verifica o status do registrador da UART			
	Os dados estão disponíveis na porta serial?			
	S			**N**
	Verifica o sinal enviado pelo robô			
	É ACK	**É WACK**	**É NACK**	
	Envia DLE	OK=0	Imprime na tela "Falha na Comunicação"	
	Espera	break	OK=1	
	Envia STX		break	
	Espera			
	OK=1			
	break			

9. BIBLIOGRAFIA

- **ASEA Robotics**

  "Asea Robot Application Protocol for Computer Link - ARAP"

  Product Manual.

- **ASEA Robotics**

  "Asynchronous Communication for Computer Link - ADPL 10"

  Product Manual.

- **ASEA Robotics**

  "Computer Link"

  Product Manual.

- **Collins, Denis e Lane, Eamonn.**

  "Programmable Controllers - A Practical Guide"

  McGraw - Hill Book Company.

- **Warnock, Ian G.**

  "Programmable Controllers - Operation and Application"

  Prentice Hall.

- **Conard, J. W.**

  "Characters Oriented Data Link Control Protocols"

  IEEE Trans. on Communications.

- **Cunha, V. L. C.**

  "Tópicos sobre Comunicação em Máquinas Industriais"

  PMC 815 - Fundamentos sobre Manufatura Automatizada.

- **Seyer, M. D.**

  "RS 232 Made Easy"

- **Stone, H. S.**

  "Interfacing Microcomputer"

- **Tompkins, W. J.; Webster, J. G.**

  "Interfacing Sensors to the IBM PC"

- **O'Brien, Stephen**

  "Turbo Pascal 6 - Completo e Total"

  Makron Books

- **Schildt, Herbert**

  "C - Power User's Guide"

  McGraw Hill