FÁBIO PAVAN DA SILVA

ROBINSON YAMAGUTI MATSUKUMA

# AUTOMAÇÃO DE UMA CÉLULA DE MANUFATURA

Trabalho de conclusão de curso apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do título de graduação em Engenharia.

São Paulo

1999

FÁBIO PAVAN DA SILVA
ROBINSON YAMAGUTI MATSUKUMA

# AUTOMAÇÃO DE UMA CÉLULA DE MANUFATURA

Trabalho de conclusão de curso apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do título de graduação em Engenharia.

Área de concentração:
Engenharia Mecânica – Automação e Sistemas

Orientador:
Lucas A. Moscato

São Paulo
1999

Aos nossos pais, amigos, professores e em especial à Rosana pelo apoio e compreensão.

# AGRADECIMENTOS

Ao Prof. Lucas A. Moscato, nosso orientador, pelo auxílio no desenvolvimento da solução do projeto.

Ao Marcos Hunold, aluno de pós-graduação da Escola Politécnica, pelo tempo despendido e, em especial, pelo auxílio com o hardware do trilho.

Aos colaboradores Fernando, Éric Preuss, Júlio César, Aleno, Breno e Professor Marcos Barreto pela ajuda fornecida nas etapas do projeto.

# SUMÁRIO

# LISTA DE FIGURAS

# LISTA DE TELAS

# RESUMO

Automação de Células de Manufatura constituídas por equipamentos CNC têm sido largamente utilizadas nas indústrias modernas com o objetivo de otimizar o processo de produção e gerenciamento das informações.

Com o objetivo de estabelecer o funcionamento da Célula de Manufatura presente no laboratório da Escola Politécnica foram desenvolvidas separadamente as diversas partes. Foram estudados os componentes robô, trilho, PLC, Supervisório, fresadora, torno, alimentador e rack de peças prontas.

A integração de todos os componentes resultou num programa completo de simulação da Célula de Manufatura.

# ABSTRACT

Automation of cells of manufacture using CNC equipment has been largely utilized in modern industries, with the objective of optimizing the production process and information management.

In order to set up the cell of manufacture's working process present in the Escola Politecnica's laboratory, several of the parts were developed separately. Robo components, track, PLC, supervisor, milling machines, lathes, metal feeder, and finished items rack were studied.

The integration of all the components resulted in a complete program simulating a manufacturing cell.

# 1. MOTIVAÇÃO

Atualmente diversas indústrias vêm utilizando células de manufatura como forma de aumentar a produtividade, a qualidade dos produtos e reduzir espaços ociosos. Para cumprir tais objetivos, o conjunto de máquinas é posicionado de forma a garantir a otimização da seqüência de operações necessárias à produção da peça. Com isso, consegue-se minimizar ao máximo o tempo de "set-up" das máquinas, o transporte de materiais, os estoques intermediários, o espaço ocupado e os tempos de espera, requisitos importantes para uma resposta rápida às novas necessidades de mercado.

As células de manufatura são destinadas à fabricação de famílias ou grupo de peças que possuem processos de usinagem semelhantes apesar de apresentarem características diferentes entre si. Também está relacionado à Tecnologia de Grupos que baseia-se na criação de códigos para as características e dimensões dos produtos de fabricação similar. A avaliação desse código permite definir um grupo para o qual uma célula é aplicável. As operações para cada codificação são por fim definidas em uma matriz de processos.

A Automação da Célula de Manufatura confere à mesma o máximo desempenho pois elimina erros dos operadores; proporciona repetibilidade e garante um tempo preciso de fabricação que permite o planejamento da produção. Para isso é necessário utilização de equipamentos CNC que podem fornecer um grande número de informações a respeito de cada etapa de

usinagem, estas serem tratadas por um PLC e a seguir monitoradas numa interface homem-máquina por um software SCADA (Supervisório). Com isso obtemos um gerenciamento eficiente de todo o processo.

Esse tema foi escolhido por se tratar de um interesse comum entre os integrantes estando relacionado à área de atuação no estágio realizado ao longo do ano na Siemens. O estágio possibilitou o aprendizado de conceitos de PLC e Supervisório que são utilizados em projetos de Automação Industrial atuais. Tais conceitos foram associados ao problema proposto a fim de que a solução seja próxima a realidade. A empresa nos incentivou através dos equipamentos fornecidos (PLC e software Supervisório) e orientações ao longo do projeto.

# 2. DESCRIÇÃO DO PROBLEMA

Nosso trabalho de formatura está baseado na célula de manufatura existente no departamento de engenharia mecatrônica.

A célula é composta por um torno de CNC Mazak, uma fresadora, um robô ABB IRB-6 que se movimenta sobre um trilho.

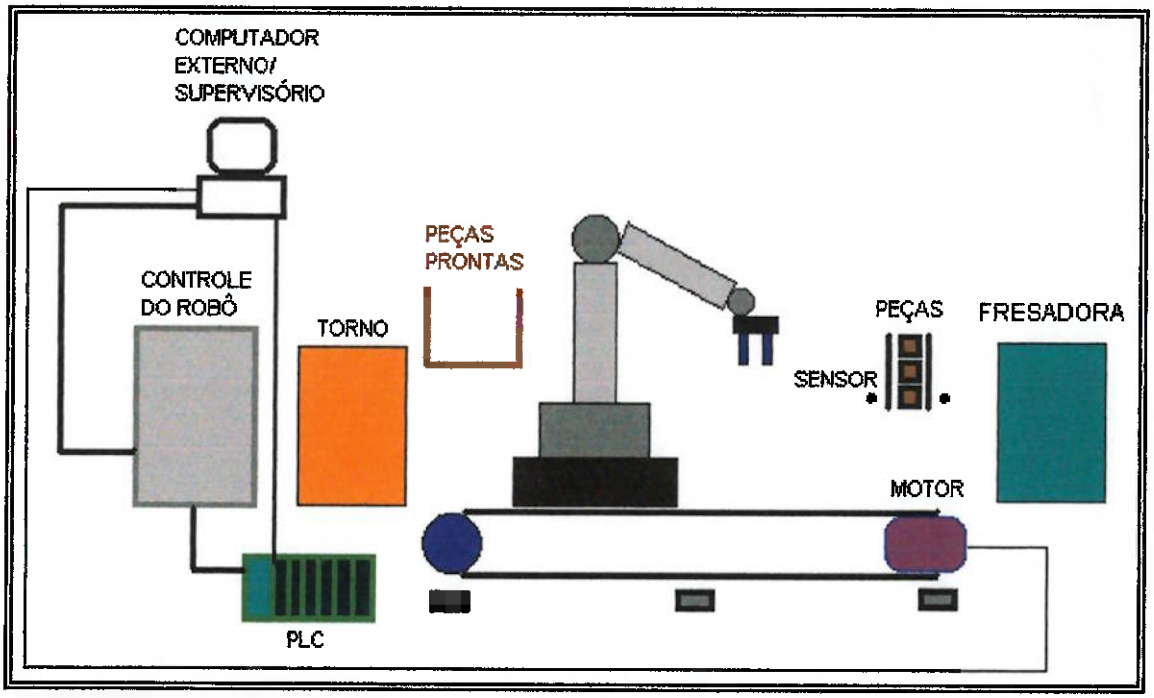



**Figura 1 - Célula de Manufatura**

A simulação da célula de manufatura prevê o seguinte funcionamento:

1. O robô recebe sinal que há peça no alimentador e a pega;
2. Fica em uma posição intermediária aguardando sinal de liberação da fresadora;

3. O robô coloca a peça na fresadora;
4. Vai para uma posição de espera, durante a usinagem;
5. O robô recebe o sinal de fim de usinagem e retira a peça da fresadora;
6. Fica em uma posição intermediária aguardando sinal de liberação do torno;
7. O robô coloca a peça no torno;
8. Vai para uma posição de espera, durante o torneamento;
9. O robô recebe o sinal de fim de usinagem e retira a peça do torno;
10. Coloca a peça no rack de peças prontas, e fica pronto para reiniciar o ciclo.

Para garantir essa seqüência de operações os sinais provenientes dos diversos dispositivos são enviados ao PLC, que faz o tratamento desses dados através de uma lógica programada e envia sinais de comando ao robô. Todo o processo é monitorado em tempo real pelo Programa Supervisório WinCC.

O Supervisório será utilizado para permitir a interface homem-máquina, ou seja, possibilitar ao usuário do sistema o acompanhamento do processo, em tempo real, podendo até mesmo atuar sobre o processo.

Esse trabalho de formatura se propõe a fazer a integração dos trabalhos:

- Interligação Robô-PLC : Eric Preuss e Júlio César;
- JNC – Java Numerical Control : Aleno Braga e Breno Polli;
- Geração e Transferência de Dados para o Robô ASEA: Jussara Koga e Marcelo Kawaguchi.

# 3. ROBÔ – ASEA

O Robô IRB-6 é um robô com 6 graus de movimento, sendo que cada um é comandado por um dispositivo independente, conforme mostra a figura abaixo:

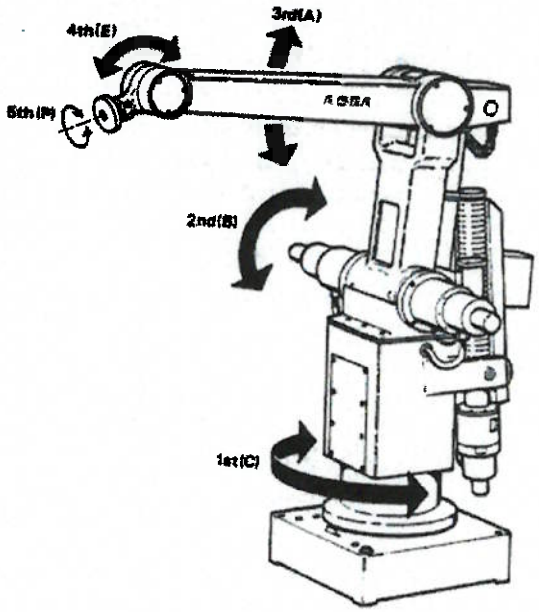
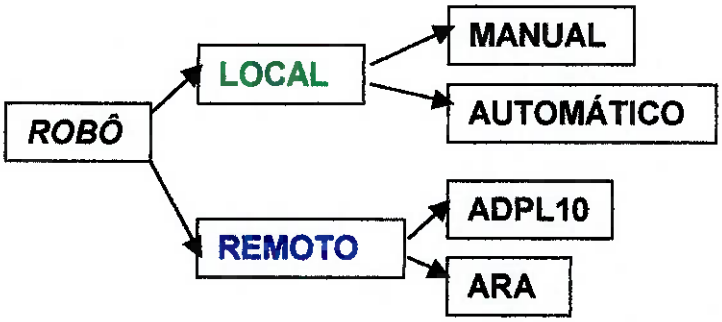



**Figura 2 - Robô ASEA IRB-6**

O robô pode ser programado localmente através de um painel de operação denominado Caixa de Aprendizado ou remotamente através de uma interface serial conectado à um Computador Externo (SC).

O funcionamento se dá de duas formas: manualmente (usando o joystick presente no painel) ou automaticamente (usando programas gravados previamente e armazenados na memória da CPU do robô ou em disco flexível 5 ¼".




O Robô da ASEA permite essa interligação com um Computador Externo via canal serial e também uma interligação com um PLC através de seus módulos de I/O digital.

## 3.1 Programação do Robô

A programação do Robô foi feita através da unidade de programação.


**Figura 3 - Unidade de Programação**

### 3.1.1 Posicionamento

Determinamos um ponto de posicionamento do braço do Robô para o qual este deve ser movido durante a execução do programa. Devemos especificar também a velocidade para cada instrução.

### 3.1.2 Teclas de Operação do Painel

A seguir temos o mapeamento dos comandos sem um estudo mais aprofundado:

f

1) GRIPPER
   a) Gripper wait 0,5s
   b) Change Time?
2) WAIT
   a) Time
   b) Input n°
3) OUTPUT
   a) Output n°

4) JUMP
   a) Instr n°
5) VELOC
   a) V(mm/s)
6) CALL
   a) Via reg?
      i) Sim: reg n°?
      ii) Não: program n°?
         - Pattern?
            * Sim: reg°?
            * Não: repeat?
               Sim?
               Não?
7) RETURN
   a) (deixa o programa em looping, ou seja, após a última instrução, ele retorna à primeira)
8) REG
   a) Reg n°?
9) TCP
   a) TCP n°?

10) INTER

a) Enable Interrupt?

i) Sim?

ii) Não?

11) COORD

a) Robot

b) Modrect

c) Rect

12) GETB

a) Block n°

13) COM

a) (*Comment*)

b) Program Stop?

i) Sim?

ii) Não?

14) SUCTRL

a) Continue?

i) Sim: register?

- Sim: reg n°?
- Não?

ii) Não?

15) FRAME

a) Frame n°?

16) EXTAX

a) Axis n°?

f-

1) V(%)

a) (velocidade)

2) INSTNO

a) Instr n°?

3) STEP

a) Instn°

b) Forwd

c) Bwd

d) Break

4) MODPOS

5) MODIFY

a) Displ

b) Cle.lo

c) Modarg

d) Modins

6) DELETE
   a) How many? (quantidade de instruções a serem apagadas a partir da primeira instrução)
7) INSERT
   a) Instr n°? (insere uma instrução entre duas outras)
8) PROGRAM
   a) Program n°? (carrega um programa)
9) RESEQ
10) COPY
   a) New Program n°? (copia o programa atual em um outro)
11) ERASE
   a) Program=? To?
12) PROGRNO
   a) Used Prog: _ _ _ (mostra os programas na memória)
13) MIRROR
   a) XY
   b) YZ

c) XY

i) Offset?

- Sim: value?

* Sim: Y?

* Não: St pos – End po

- Não?

d) Same

14) TIME

P

(POS V=0%)

1) WELD

2) WEND

3) CLEAN

4) EXTC

5) EXTPOS

1) CLEAR

2) FR DISK

3) TO DISK

4) IN/OUT

5) TOOL

6) PARAM

7) INITDIS

8) FRAME

9) LIST

10) FRSC

11) TOSC

12) RBMODE

13) LANG

14) ERRORS

15) TEACH

16) WDATA

17) READRES

18) DESYNC

Vai gravando as posições (coordenadas) do robô.

Mostra as coordenadas do robô.

### 3.1.3 Para executar um programa

- Carregar o Programa (f-g)
- Deixar a primeira instrução no display (STEP)
- Clicar em 
  - PROG ST (Program Start)
  - INS ST (Instruction Step)
  - BWD
  - SIM
  - DISPL
  - CORV ST

OBS: O teclado operacional deve estar no robô.

### 3.1.4 Exemplo de um Programa do Robô

10: ROBOT COORD

20: V=200MM/S MÁX=500MM/S

30: TCP 0

40: FRAME 0

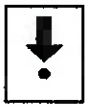
50: POS V=50%

60: POS V=25%

70: RETURN

Esse programa realiza o movimento do robô em duas posições definidas nas linhas 50: e 60: ciclicamente (devido à instrução RETURN na linha 70:).

As posições são gravadas da seguinte maneira:

- Posicionamos a garra através do joystick presente no painel de operação e gravamos sua posição através do botão 

A execução do programa é feita pressionando-se o botão  e PROG ST (ou INS ST se desejar executar linha a linha).

O programa pode ser interrompido a qualquer momento pelo botão  .

## 3.2 Movimentação do Robô

A movimentação do Robô – ASEA se dá em 3 tipos diferentes de Sistemas de Coordenadas:

- Sistema de Coordenadas do Robô;
- Sistema de Coordenadas Retangular;
- Sistema de Coordenadas Vetorial.

#### 3.2.1.1 Sistema de Coordenadas do Robô

Esse sistema permite a movimentação em coordenadas cilíndricas, permitindo a rotação em torno do eixo da base, movimentação na direção Z e outros movimentos.

O robô mexe uma articulação de cada vez, o que torna mais previsível o seu movimento.

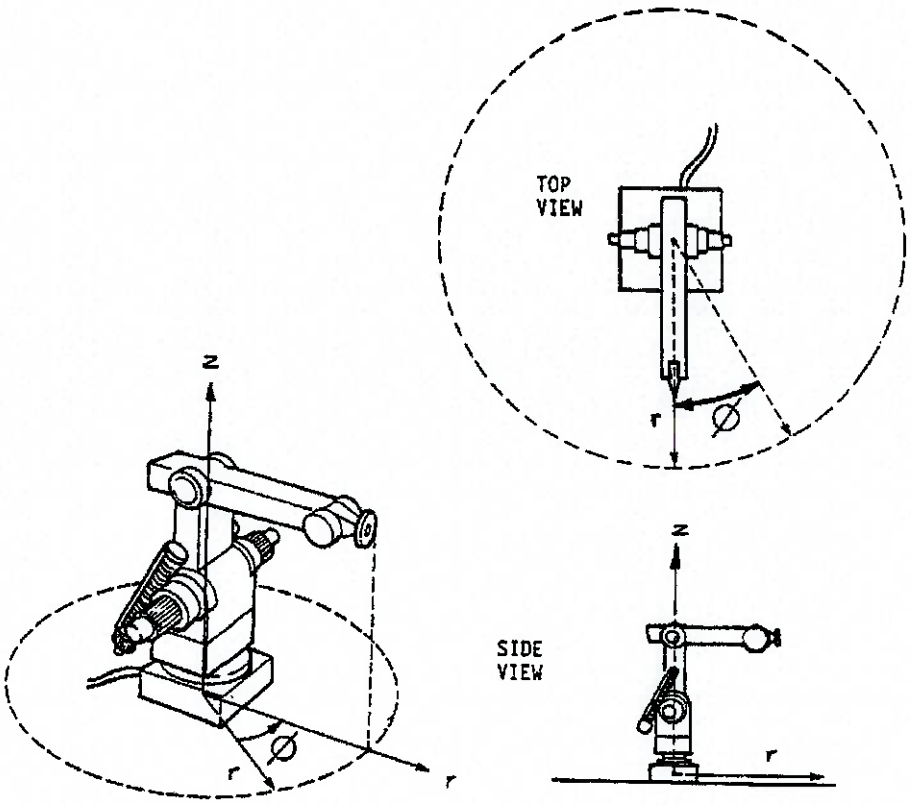



**Figura 4 - Sistema de Coordenadas do Robô**

### 3.2.1.2 Sistema de Coordenadas Retangular

Esse sistema permite movimento nos eixos de coordenadas cartesianas (X, Y e Z) e a rotação em torno dos mesmos. O robô pode movimentar mais de uma articulação ao mesmo tempo.

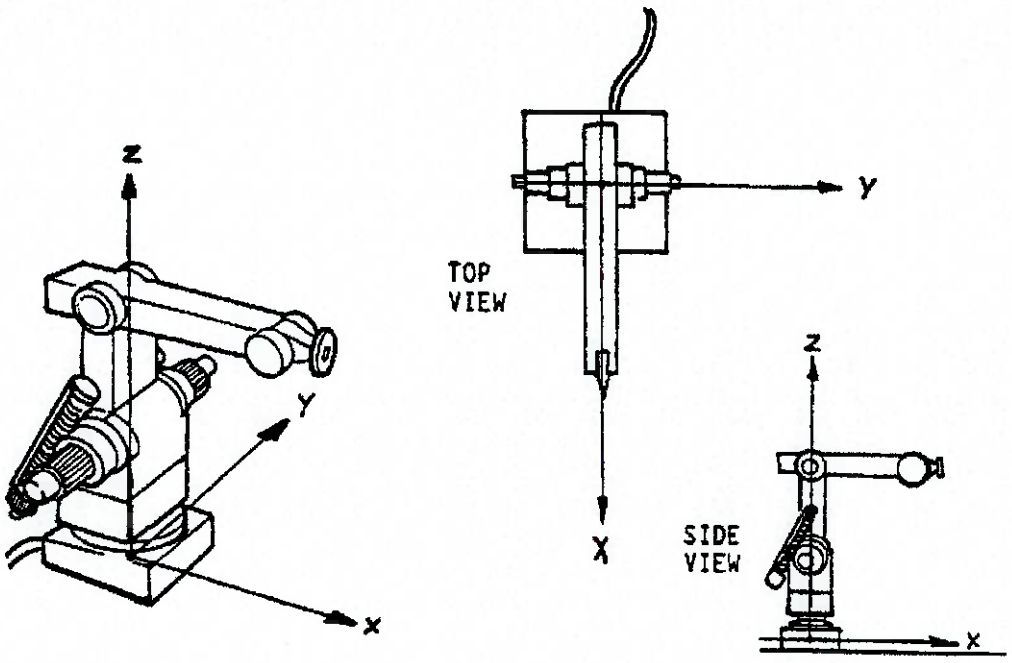



**Figura 5 - Sistema de Coordenadas Retangular**

### 3.2.1.3 Sistema de Coordenadas Vetorial

O robô executa movimentos de translação e rotação em torno de um sistema de eixos presente na garra.

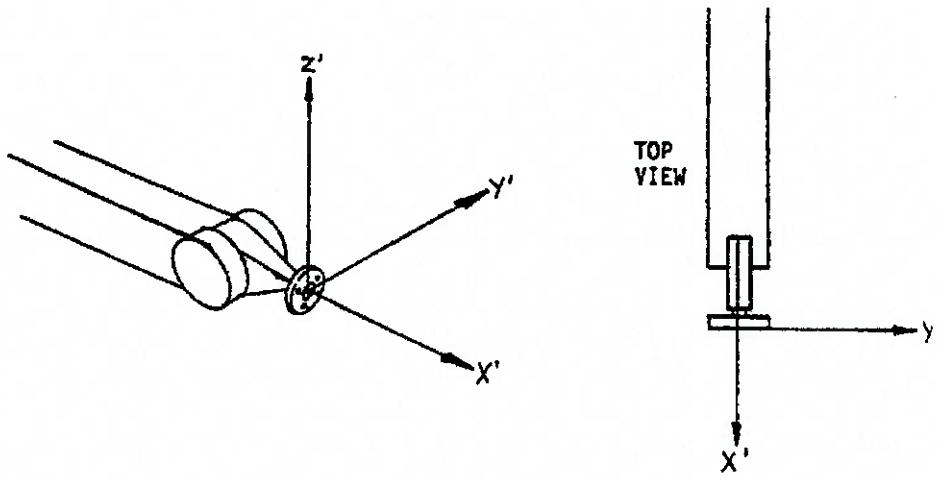



**Figura 6 - Sistema de Coordenadas Vetorial**

## 3.3 Entradas e Saídas Digitais

A comunicação entre o robô e o CLP será feita através das entradas e saídas digitais presentes no robô. O robô possui:

- 7 entradas digitais;
- 6 saídas digitais;
- 2 saídas digitais para controle da garra.

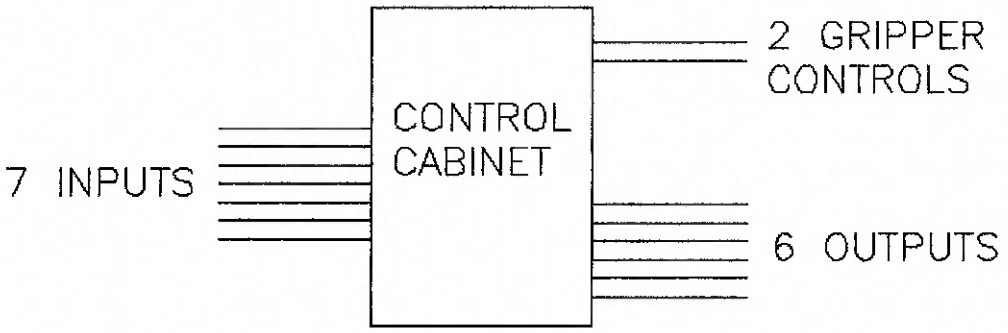



**Figura 7 - Entradas e Saídas Digitais**

### 3.3.1 Dados Técnicos

As entradas e saídas digitais estão numa placa conectada ao controle do robô.

1) Tensão Nominal: 24V DC
2) Faixa do sinal de entrada:
   a) Lógica "1": de 18 a 35V
   b) Lógica "0": de –21 a +2V
3) Corrente de entrada na tensão nominal: 6.8mA
4) Impedância de Entrada: 3.5kΩ
5) Sinais de Saída:
   a) Fonte externa: 24V nominal (20-35V)
   b) Capacidade de carga: no máximo 150mA

O Robô disponível apresenta como entradas e saídas digitais 2 placas do modelo DSDX 110. Uma placa sendo a primária, que realiza algumas funções básicas internas do Robô (Motor ON/OFF, Movimentação da Garra, etc.) e uma secundária que também possui alguns bits ocupados para interrupções internas, porém tem-se boa parte livre para uso.

Essas placas se comunicam com o mundo externo através de 2 outras placas (que possuem bornes para conexões de fios com o mundo externo) que estão posicionadas na parte traseira da cabine de controle do Robô, sendo

respectivamente as placas DSQC 124 (ligada à placa primária) e DSQC 122 (ligada à placa secundária). Um esquema da placa DSQC 122 é fornecido abaixo.

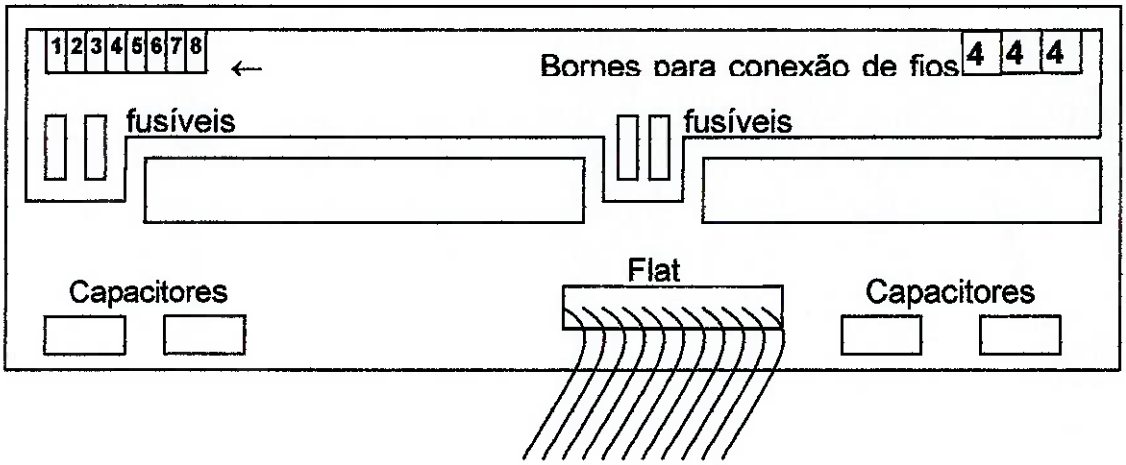



**Figura 8 - Placa DSQC 122**

As entradas digitais utlizadas foram as da placa secundária. Os quatro primeiros bits dessas entradas digitais são endereçados pelo PORT nº 12 do Robô. Para as saídas digitais, utilizou-se a placa primária (as saídas digitais da placa secundária não funcionam). Seus bits são endereçadas a partir do PORT nº 1 do Robô. Tabelas fornecendo os números dos bornes, e bits de entradas e saídas digitais são fornecidas a seguir.

*Entradas Digitais – placa secundária*

| Bit | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| INPUT | 13 | 14 | 15 | 16 |
| Borne | 43 | 44 | 45 | 46 |

*Saídas Digitais – placa primária*

| Bit | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| OUTPUT | 3 | 4 | 5 | 6 |
| Borne | 8 | 9 | 10 | 11 |

As indicações INPUT e OUTPUT referem-se aos LEDs que acendem no painel da cabine do Robô para um controle nas Entradas e Saídas ativas (HIGH) no momento.

A placa secundária necessita de alimentação externa ou de uma outra fonte interna. No caso, utilizou-se a outra placa primária DSDX 110 como fonte.

O esquema de ligações com alimentação entre as placa fica conforme mostrado a seguir.

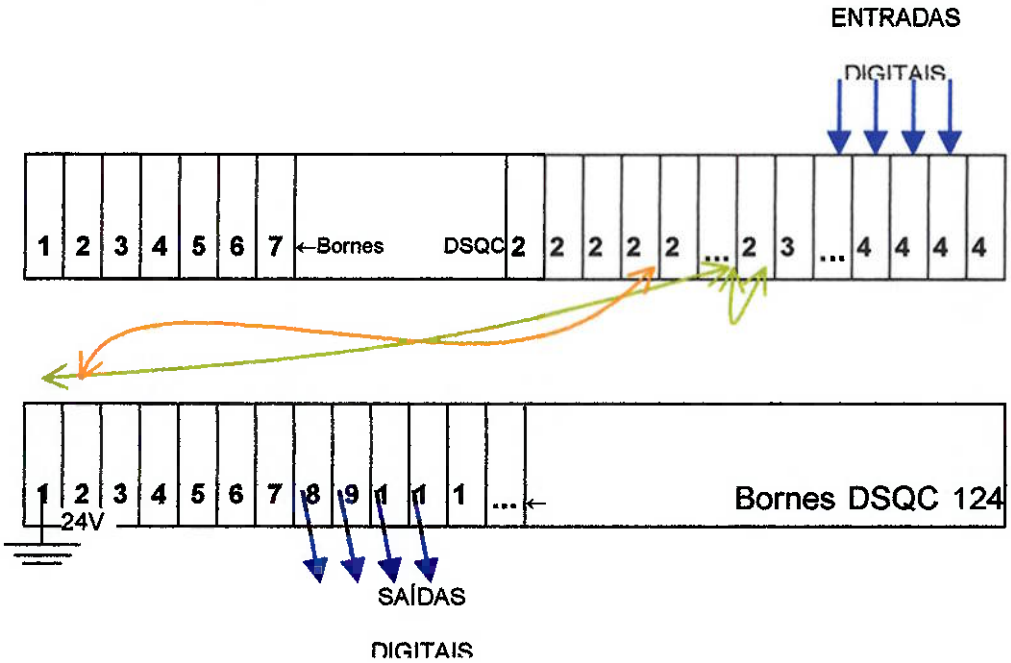



**Figura 9 - Esquema de Ligação das Placas**

# 4. TRILHO

O objetivo desta parte do projeto é propiciar ao Robô o sétimo grau de liberdade. Estará baseada no trabalho JNC – Java Numerical Control (Aleno Chaves Braga e Breno Tordini Polli).

A movimentação do Robô sobre o trilho tem a função de transportar a peça em processo de usinagem entre o torno e a fresa.

Apesar de ser um movimento simples, a operação como um todo acaba sendo complicada devido à necessidade de um posicionamento preciso do Robô para o funcionamento perfeito dos processos de retirada e colocação da peça nas máquinas operatrizes. Por esse motivo torna-se necessário a implementação de um controle de posição com alta precisão, além de um bom controle de velocidade e aceleração para garantir o transporte rápido das peças e sincronia de tempo entre as execuções do robô, do torno e da fresadora.

Para tal foi escolhido um controle do tipo PID digital para que possa ser implementado através de equações de diferenças rodando em um computador. O controlador foi determinado após a modelagem do sistema trilho, motor, redutor e sensor. A função de transferência foi obtida de maneira experimental, analisando-se a resposta do sistema para uma dada referência de entrada (degrau).

Foi definido como parâmetro de projeto um sobressinal de zero, para garantir que a plataforma vá para a posição desejada sem oscilar em torno da

mesma. Além disso foi projetado um tempo de assentamento pequeno, para que o sistema tenha uma resposta rápida.

Também escolheu-se o pólo de operação de forma a tornar o controlador imune a possíveis variações de massa na plataforma. Essas variações ocorrem dependendo da peça que o robô estiver carregando.

A integração do trilho à célula de manufatura consistiu em três atividades:

## 4.1 Arquitetura de Hardware

O sistema referente ao trilho utilizou os seguintes equipamentos:

1) Trilho de 259mm projetado e construído com metal e madeira na própria faculdade, com uma guia para permitir deslocamento da plataforma;
2) Plataforma quadrada de madeira e base de metal de 52mm onde será fixado o robô e que se movimenta sobre o trilho através de rolamentos que são fixados na parte superior e inferior da guia do trilho, permitindo somente o movimento em uma única direção. A plataforma possui uma guia para ser percebida pelos sensores fim-de-curso, e, além disso, é amarrada a um cabo de aço que se enrola a um tambor com 9 voltas;

3) Tambor de aço com 120mm de diâmetro no qual é enrolado o cabo de aço que é ligado à plataforma. De um lado desse tambor é ligado o redutor e do outro o gerador de pulsos (encoder);
4) Redutor com relação 1:10 com entrada e saída perpendiculares de marca Cestari tipo K-40, com potência de 1,08CV a 1750rpm;
5) Gerador de Pulsos Óptico (encoder) de 5000 raias, marca Diadur, com 2 discos graduados que permitem discernir se este está girando no sentido horário ou anti-horário;
6) Servo-motor de corrente contínua da marca Varimot S.A. modelo SD-2535 com as seguintes características:
   a) Tensão máxima: 130Vcc
   b) Torque: 3Nm
   c) Rotação de Saída em vazio: 3000rpm
   d) Tacogerador: 9,5V / 1000rpm
7) Servo-amplificador (parametrizável) de marca Transaxe, série 700, modelo 710 (conversor estático para motores CC), com uma malha controlada por um circuito proporcional integral, realimentado por um tacogerador e que trabalha com pulsos (PWM). Tem saída para motor e entrada no micro que irá fornecer o sinal de referência;
8) Placa de Interface Microcomputador-Hardware de marca Sensoray, modelo 421, contador de pulsos que serão gerados pelo encoder, conversor D/A para gerar sinal para o servo-amplificador e leitora de entradas digitais vindas das chaves fim-de-curso;

9) Um microcomputador.

Estudou-se o trabalho JNC, o manual da placa de I/O (Senoray 421) e o manual de acionamento (Transaxe) com a finalidade de testar os equipamentos individualmente. Verificou-se assim que todos os equipamentos estão funcionando de forma adequada.

## 4.2 Arquitetura de Software

Na solução proposta no trabalho JNC, a estação de trabalho que coordena a célula enxerga o sistema como um comando numérico e envia comandos em linguagem G, reconhecida pelo controlador da célula.

Para isto, foi desenvolvido um compilador de linguagem G com a finalidade de controlar o trilho. A interface homem-máquina foi desenvolvida em Java, utilizando o aplicativo Visual Age for Java (IBM), e as funções que realizam a troca de sinais com a placa de interface colocada no micro (placa de entradas digitais, conversora digital-analógica e contadora de pulsos do encoder), em linguagem C/C++.

O software é ainda responsável pelo acionamento de um motor que movimenta a plataforma sobre o trilho e pelo controle de posição da plataforma. Sua implementação deve seguir os critérios do projeto de controle, com o PID sendo realizado em malha semi-fechada, realimentado por um encoder.

O microcomputador utilizado, um K6-350Mhz dispõe de apenas 32Mb de RAM, o que inviabiliza o uso do programa Visual Age for Java. Decidiu-se portanto pela utilização do software JDK para operar com o software dedicado (JNC).

O software JNC permite a operação do trilho tanto em modo manual como em modo automático utilizando programação em linguagem G. Na simulação da célula de manufatura, a operação será feita em modo automático.

No microcomputador, onde está o controlador da planta e a interface homem-máquina, o sinal de controle é gerado e convertido pela placa de I/O, que possui função de conversora D/A, chegando ao PWM, responsável pelo acionamento do motor e , consequentemente, da plataforma onde se encontra o robô. Este acionamento possui um controle proporcional integrativo (PI) para tensão e corrente em seu interior, necessitando da realimentação do tacogerador para um perfeito funcionamento.

A leitura da posição da plataforma é feita através da posição do motor, utilizando-se um encoder, que gera dois sinais defasados de 90°, permitindo a placa de I/O que possui função contadora de pulsos, saber em qual sentido está sendo o giro do motor. O valor de contagem é utilizado pelo computador para gerar um novo valor de controle. O sinal de reset proveniente da chave fim-de- curso localizada no centro do trilho é utilizado para zerar a contagem.

Além disso , o computador recebe os sinais das chaves de fim-de-curso também através I/O, que possui função de leitura de entradas digitais, para que

possa parar a movimentação da plataforma e informar ao operador a anormalidade no posicionamento.

## 4.3 Integração Hardware e Software

Para colocar o trilho em funcionamento foram efetuadas todas as conexões dos sinais de hardware. A arquitetura de hardware utilizada pode ser vista a seguir:

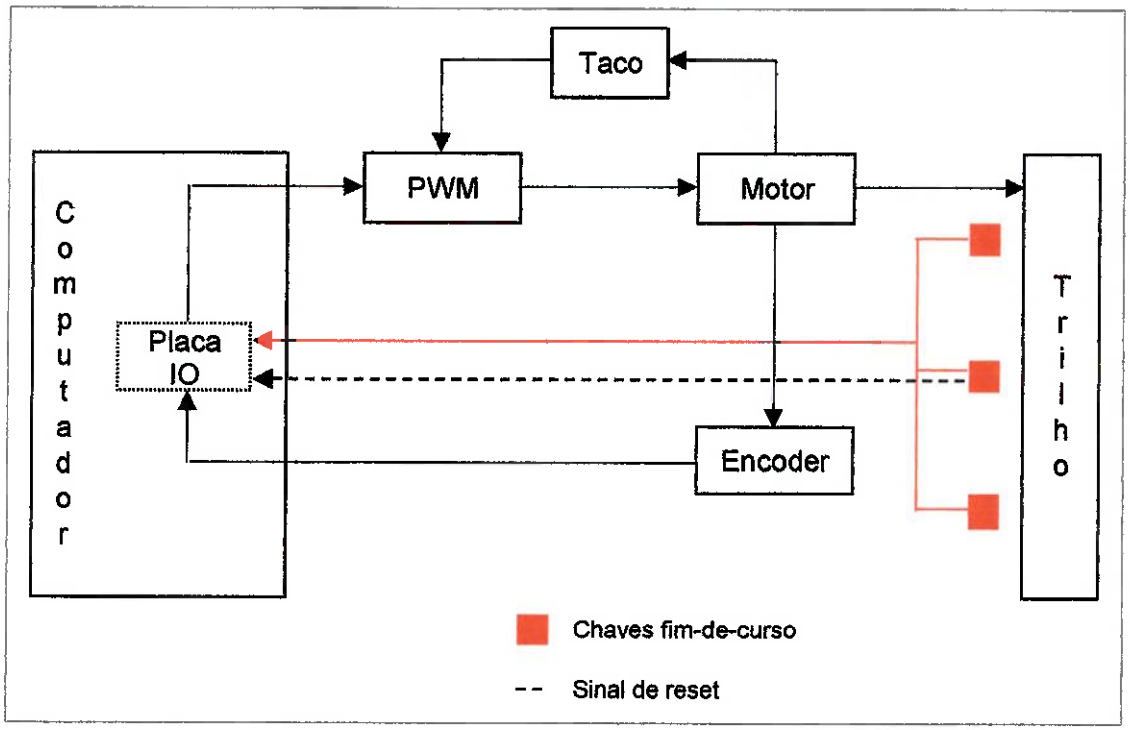
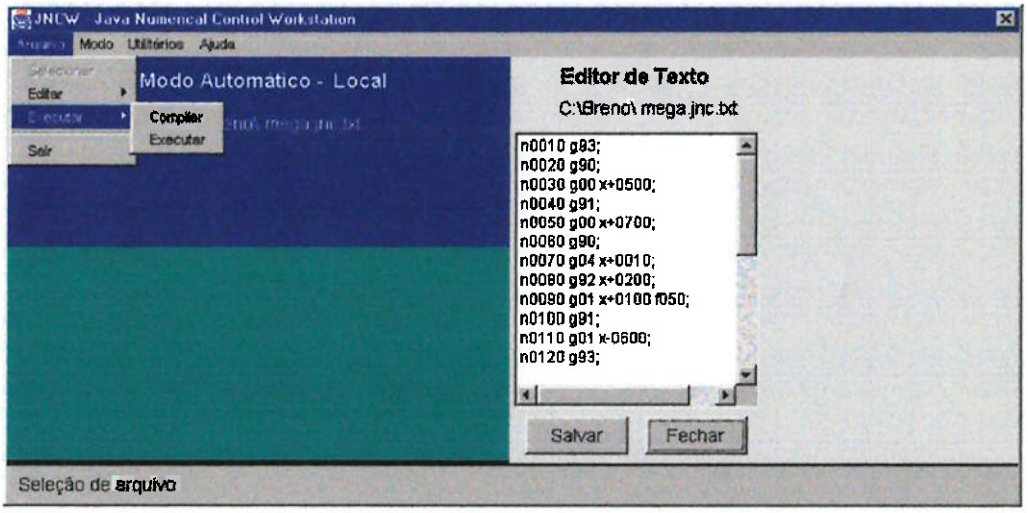



**Figura 10 - Arquitetura de Hardware**

Foi desenvolvido um programa em Linguagem G destinado à Simulação da nossa Célula de Manufatura. Este programa pode ser visto a seguir:

```
n 0010    g93;                { zera trilho }
n 0020    g90;                { sistema de coordenadas absolutas }
n 0030    g04 x+0010          { tempo para pegar peça }
n 0040    g01 x+1000 f050     { posiciona o robô próximo à fresadora }
n 0050    g04 x+0010          { tempo de usinagem da fresadora }
n 0060    g01 x-1000          { posiciona o robô próximo ao torno }
n 0070    g04 x+0010          { tempo de usinagem do torno }
n 0080    g01 x+0000          { posiciona próx. ao alim./rack de peças prontas }
n 0090    g04 x+0010          { tempo para depositar peças }
n 0100    m03 x+0030          { jump para a linhe n 0030 }
n 0110    m02                 { desliga }
```




**Tela 1 - Modo Automático JNC**

Para a simulação atual o programa G foi feito prevendo os tempos gastos em cada etapa do processo. No entanto é preciso ressaltar que para trabalhos futuros que disponham de equipamentos CNC reais deverá ser feita uma alteração. O programa JNC deverá ser alterado (o programa em Java e a Interface de Hardware *.dll*) de forma a disponibilizar um comando que envia um sinal +5V para ser utilizado pelo PLC e outro comando que permita aguardar um sinal externo (vindo do PLC). Só assim seria possível a integração perfeita do programa do trilho com o resto do sistema.

# 5. PLC – CONTROLADOR LÓGICO PROGRAMÁVEL

O PLC – Controlador Lógico Programável – teve inicialmente seu principal uso associado à execução de lógicas de intertravamento, ou seja, verificava-se as condições de segurança (sensores e contatos) estavam satisfeitas de modo a habilitar a energização.

Com o aumento da capacidade de processamento dos PLC's eles passaram a executar não só as operações binárias (associadas ao intertravamento), mas também a manipular banco de dados (por exemplo de receitas de produção), tratar de valores analógicos (associados a variáveis de processo) e executar malhas de regulação.

Deste modo esses equipamentos passaram a ser encontrados não somente como componentes internos de máquinas, mas também controlando conjuntos de máquinas em instalações de maior porte.

No problema proposto, *Automação da Célula de Manufatura*, estaremos trabalhando com diferentes sinais provenientes de equipamentos diferentes (alguns desses sinais serão apenas simulados). Nesse contexto, o PLC tem a função de receber esses sinais, tratá-los de forma adequada e enviar sinais de comando de forma a garantir a execução das operações na seqüência correta.

No trabalho Interligação Robô – PLC (Eric Preuss / Júlio César) foi utilizado um PLC da marca *Klockner Moeller* tipo *SUCOS PS 3 – AC* com as seguintes características técnicas:

- Alimentação: 230/120V AC;
- 16 Entradas Digitais de 24V DC;
- 8 Saídas Digitais;
- 4 Entradas Analógicas (0 – 10V DC);
- 1 Saída Analógica (0 – 10V DC);
- 1 Entrada para Contador (10KHz)

As saídas desse PLC são do tipo relé (contato seco) , necessitando portanto de alimentação para que ao ser fechada, injete 24V na saída.

Esse PLC pode ser programado através de 3 "linguagens" diferentes:

- LDI – Utiliza comandos do tipo Instruction List;
- DDC – Utiliza diagramas de relé;
- DDB – Utiliza diagramas com blocos lógicos, tipo E , OU , etc.

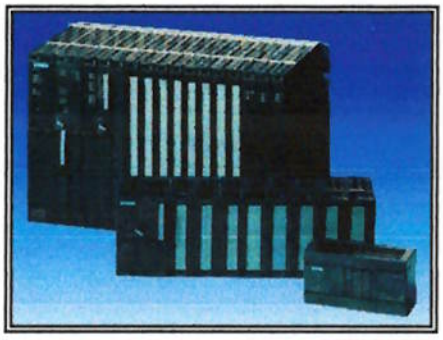
Essa programação pode ser feita de 3 formas:

- Através de um microcomputador;
- Por meio de maleta de programação;
- Através de caneta óptica.

No desenvolvimento do projeto seus autores utilizaram a linguagem LDI, programando-a através da maleta programadora.

Com relação as características técnicas poderíamos continuar utilizando esse mesmo PLC. No entanto, para o nosso projeto foi decidida a troca desse PLC por um PLC Siemens. Os motivos que levaram a essa troca foram:

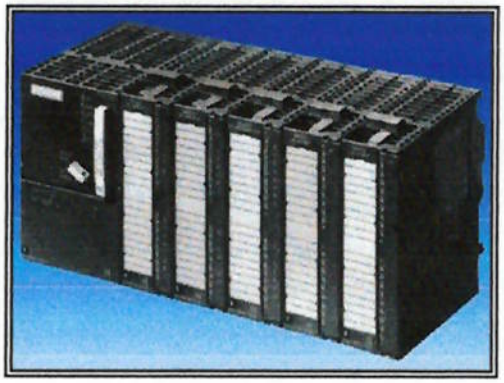
- Por estarmos fazendo estágio na Siemens temos grande familiaridade com esses PLC's e sua programação;
- A utilização do Supervisório da Siemens torna necessária a utilização de um PLC compatível. Neste caso os mais indicados são os PLC's da linha Simatic S5 ou Simatic S7.



O Estudo de Viabilidade deste projeto previa a utilização de um PLC Siemens da linha S5 (CPU 135U). Por fim, decidiu-se pela utilização de um PLC da linha S7 (S7- 300, CPU 315) contendo 8 entradas / 8 saídas digitais.

Essa mudança foi motivada pelo fato da linha S7 ser mais moderna, apresentar interface com o usuário mais amigável e melhores recursos de programação (software S7 Manager).

Foram realizados testes no laboratório da Siemens utilizando um Kit didático do S7-300 contendo uma CPU 315, 8 entradas / 8 saídas digitais, uma botoeira e um display. Os sinais do robô foram simulados através de botões e como resultado final pôde-se definir um programa no PLC capaz de realizar a seqüência de operações prevista para a célula. O PLC foi interligado ao Supervisório e foram definidas todas as funções deste com relação a monitoração e controle (maiores detalhes no item SUPERVISÓRIO).



O Programa pode ser feito em:

- STL – Instruction List (correspondente ao LDI do Klockner Moeller);
- LAD – Diagrama de relé (correspondente ao DDC);
- CSF – Diagrama de blocos de funções lógicas (correspondente ao DDB).

# 6. SUPERVISÓRIO WINCC

O WinCC (Windows Control Center) é um sistema de software de interface homem-máquina que integra o controle e monitoramento da planta ao processo automatizado.

Ele combina a arquitetura dos aplicativos do Windows NT com um programa de interface gráfica (objetos).

## 6.1 Módulos

O WinCC possui os seguintes módulos para construção de um projeto:

**Graphics** – para criar um desenho representativo do processo industrial e outras telas de monitoração conforme as necessidades.

**Archiving** – para armazenar dados de processo (por exemplo alarmes, parâmetros, etc) num banco de dados SQL.

**Reports** – para gerar relatórios sobre os dados do processo.

**Data Management** – para definir, selecionar e coletar dados do processo.

## 6.2 Hardware

O WinCC suporta plataformas padrão Intel.

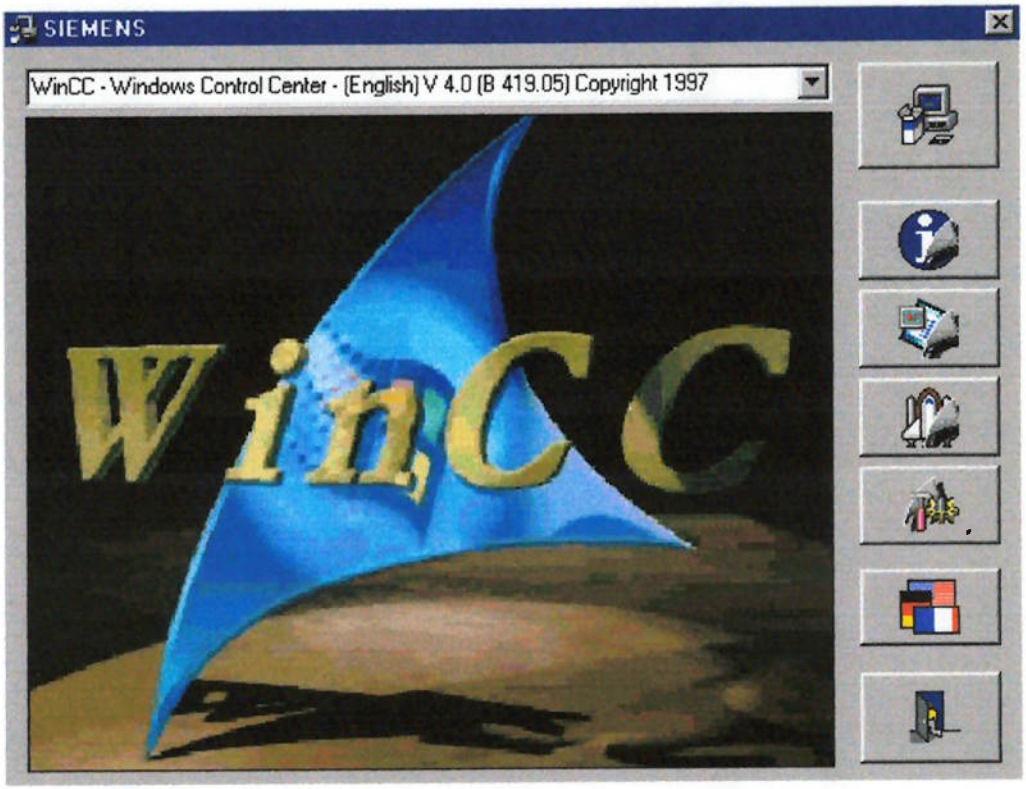
| | Mínimo | Recomendado (ou superior) |
|---|---|---|
| **Processador** | Intel Pentium 133 | Intel Pentium 166 |
| **Memória RAM** | 32 MB | 64 MB |
| **Vídeo** | VGA | SVGA (2MB) |
| **Resolução** | 640x480 | 1024x768 |
| **Disco Rígido** | 200MB | 500MB ou mais |
| **CD ROM** | 2x | 8x |

O número de sinais (tags) provenientes do PLC que o Supervisório pode tratar está relacionado com a quantidade de memória RAM que o computador possui. No geral, cada PLC contém 100 ou mais tags e adicionamos 1MB de memória RAM para cada PLC adicional.

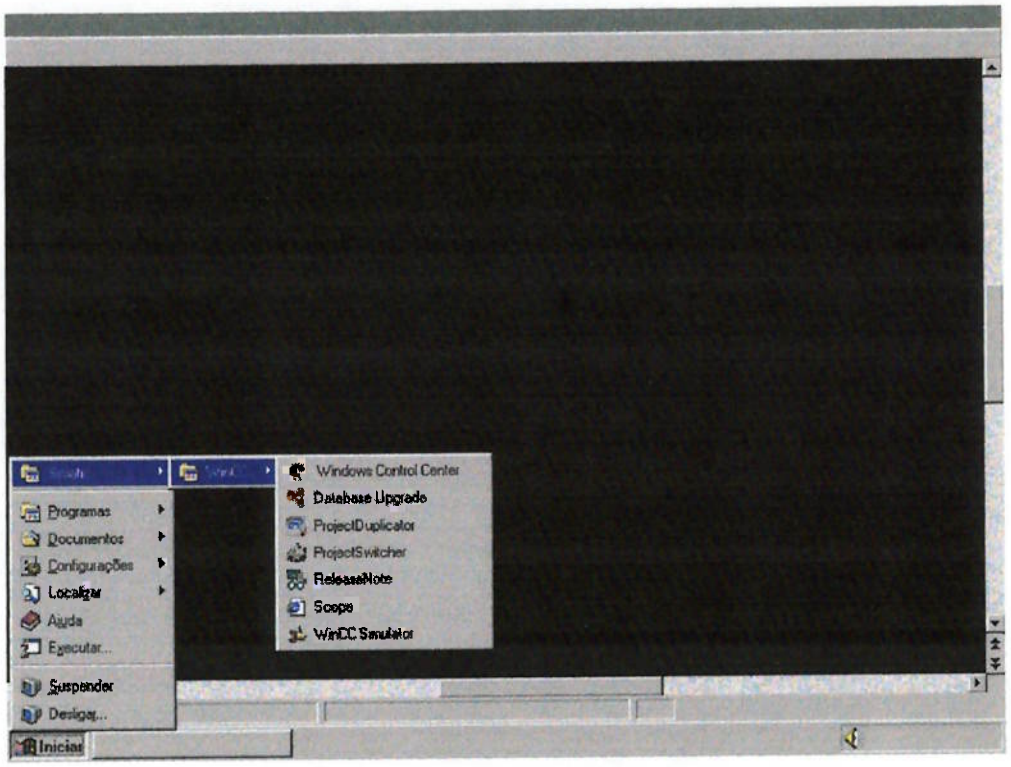
## 6.3 Software

O WinCC precisa de uma plataforma 32bits e um sistema Windows (Microsoft). Pode-se utilizar o Windows 95 ou o Windows NT 4.0 (recomendado para melhores resultados).

## 6.4 O Programa

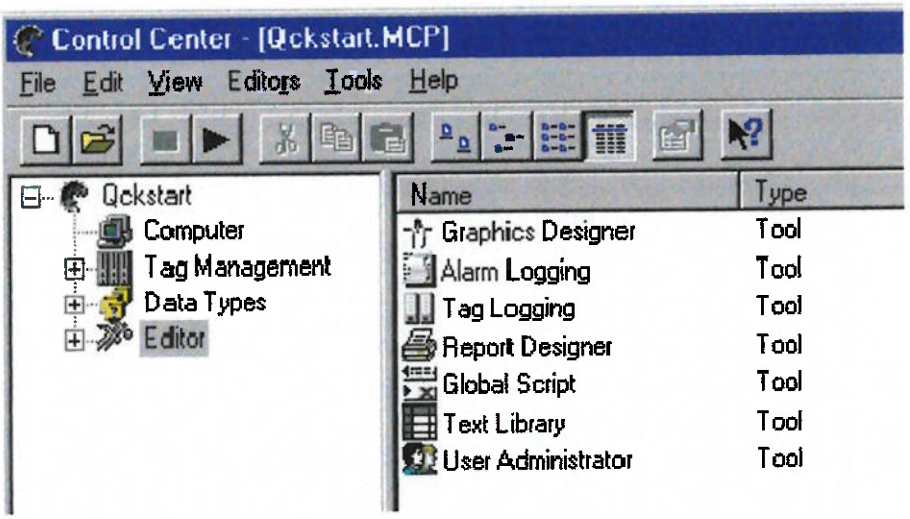



**Tela 2 – Tela de inicialização do WinCC**




**Tela 3 - Execução do software**

Os módulos do Supervisório estão dispostos em no Control Center conforme uma árvore de categorias de gerenciamento:




**Tela 4 - Gerenciadores do WinCC**

### 6.4.1 Computer

Permite configurar uma rede de supervisórios que compartilharão as informações vindas da planta de processo através dos PLCs. Pode-se definir quem será o mestre (disponibiliza as informações na rede – geralmente TCP/IP – e comanda a sincronia dos dados) e quem será o escravo.

A arquitetura de rede é bastante utilizada quando se tem vários processos e as informações necessitam estar disponíveis como no caso de subestações de geração e distribuição de energia elétrica ou várias unidades de abastecimento de água numa cidade ou numa usina que possui vários processos tais como laminação, fiação, rolos, etc.

### 6.4.2 Tag Management

Define os tags, que representam as informações vindas em forma de sinais (bits ou palavras de bytes) e cria mnemônicos associados à estes sinais. De maneira geral, mapeia a memória do PLC onde estão contidas as informações a respeito dos sinais do processo e associa a strings de caracter (mnemônicos) que são utilizados no programa Supervisório para efetuar o monitoramento e controle de todo o processo.

### 6.4.3 Editor

O editor contém os módulos:

1) Graphics Designer;
2) Alarm Logging;
3) Tag Logging;
4) Report Designer;
5) Global Script;
6) Text Library;
7) User Administrator.

#### 6.4.3.1 Graphics Designer

Possui ferramentas de desenho para criar a tela representativa dos processos, bem como objetos que permitirão o monitoramento dos sinais.

#### 6.4.3.2 Alarm Logging

Gerenciamento dos alarmes. Os alarmes são informações a respeito do funcionamento do processo. Por exemplo, podemos criar um alarme para indicar que uma determinada operação ocorreu satisfatoriamente, ou que uma

determinada válvula está em determinada posição, ou que um determinado modo de operação está selecionado ou que ocorreu algum defeito, etc.

Para cada alarme gerado, pode-se exigir do usuário (pessoa que vai estar no comando do Supervisório) que tome ciência do fato ocorrido, ou execute alguma rotina de tratamento de um possível erro (ou falha) e se disposto em forma de relatório para posterior análise.

#### 6.4.3.3 Report Designer

Define o formato dos relatórios e quais informações devem ser dispostas e relatadas para análise ou arquivo.

#### 6.4.3.4 User Administrator

Permite configurar um certo nível de segurança para aqueles que utilizarão o sistema após elaborado. Por exemplo, a utilização do Supervisório pelo gerente de processo de laminação é diferente daquela pelo gerente de processo de formação das bobinas pois as informações importantes para um podem não ser importantes para o outro.

## 6.5 Compatibilidade

É importante notar que o WinCC possui a flexibilidade de integrar PLCs de outros fabricantes além dos PLCs da SIEMENS. Isso é muito importante pois permite uma gama de opções de sistemas e soluções para os mais diversos processos industriais além do que é sabido que hoje há várias empresas que utilizam um misto de plataformas de PLCs de vários fabricantes (SIEMENS, ABB, GE, etc) e tendo o WinCC a possibilidade de integração, pode-se facilmente montar um sistema de controle e monitoramento ágil, moderno e compatível.

# 7. COMUNICAÇÃO ROBÔ-COMPUTADOR

## 7.1 Computer Link

O Computer Link permite a troca de informações entre um robô industrial denominado IRB (Industrial Robot) e um computador externo denominado SC (Superior Computer). Ele está contido no controlador do robô e necessita de um pacote de software e um pacote de hardware.

## 7.2 Superior Computer (SC)

O computador externo deve possuir uma porta de comunicação serial RS 232-C V24, V28 utilizando os protocolos do robô: protocolo de comunicação ADPL10 (Asea Data Link Protocol) e protocolo de aplicação ARAP (Asea Robot Application Protocol).

As funções do SC no sistema são:

### 7.2.1 Controle

Relacionado ao modo de operação, início e término da execução, leitura e escrita de informações no robô, centralizando e simplificando o controle do mesmo.

### 7.2.2 Supervisão

Através da transmissão de eventos e erros que ocorrem no IRB é possível avaliar o funcionamento do sistema.

### 7.2.3 Base de Dados

O Computador pode armazenar os programas do robô numa biblioteca interna, dispensando o uso da memória (restrita) interna ao robô.

## 7.3 Computer Link Software

Como descrito anteriormente, o SC necessita de dois protocolos, sendo um de comunicação e outro de aplicação.

As funções do pacote de software do Computer Link são:

- Carregar programas no IRB e gravar programas do IRB;
- Ler e Escrever informações sobre o robô e o processo;
- Selecionar, Iniciar e Finalizar programas e movimentar o IRB;
- Supervisionar o status do IRB e ler mensagens de erro;
- Permitir programar o robô remotamente, num computador externo.

A conexão é serial, ponto a ponto com relação mestre-escravo (enquanto o mestre envia uma informação, o escravo recebe). Note que tanto o SC quanto o IRB pode ser mestre.

Podem ser enviados: comandos, respostas e mensagens espontâneas.

A instrução SUCTRL (Superior Control) quando lida pelo IRB gera uma mensagem espontânea e envia ao SC.

## 7.4 Computer Link Hardware

A interface deve ser compatível com RS232 / V24,V28. Permite uma distância menor que 15 metros, a velocidade de comunicação é de 9600 bauds, o tamanho do caracter é de 8 bits com 1 bit de paridade e 1 bit de parada.

As unidades que compõe o hardware são:

- Placa de Comunicação assíncrona DSCA 114;
- Cabos DSTK 152;
- Terminal de Comunicação para DSCA 114 – DSTC 120.

A placa de comunicação DSTC 120 está localizada no rack posterior do controlador do robô e o conector a ser utilizado é o DB25 (25 pinos).

## 7.5 Protocolo de Comunicação - ADPL 10

### 7.5.1 Características

- Alto nível de segurança;
- Possibilidade da transmissão ser iniciada por qualquer uma das estações (quando uma é mestre, outra é automaticamente escravo).

### 7.5.2 Termos Aplicados na Comunicação

- HS (host station) = SC;
- SS (subordinade station) = IRB;

- Telegrama = toda informação transmitida entre o computador e o robô, dividida em unidades menores.

## 7.5.3 Caracteres de Controle

### 7.5.3.1 ENQ (Enquiry)

Utilizado como um pedido de resposta à outra estação; utilizado par estabelecer conexão.

### 7.5.3.2 ACK (Acknowladge)

Utilizado pelo escravo par a confirmação da última informação enviada pela outra estação.

### 7.5.3.3 WACK (Wait and Acknowladge)

Utilizado pelo escravo par a confirmação da última transmissão; entretanto este não está ainda preparado par receber nova informação.

#### 7.5.3.4 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.

#### 7.5.3.5 NAK (Negative Acknowladgement)

Utilizado pelo escravo para indicar o não entendimento da última transmissão.

#### 7.5.3.6 DLE (Data Link Escape)

Utilizado pelo mestre para mudar num telegrama o significado do próximo caracter de dado para caracter de controle.

#### 7.5.3.7 STX (Start of Text)

Utilizado para indicar um início de um telegrama.

#### 7.5.3.8 ETX (End of Text)

Utilizado para indicar o final de um telegrama.

#### 7.5.3.9 EOT (End of Transmission)

Utilizado para indicar o fim da transmissão; transmissões subsequentes somente poderão ser realizadas mediante nova conexão.

#### 7.5.3.10 BCS (Block Check Sum)

Utilizado para verificação das informações transmitidas anteriormente.

### 7.5.4 Comunicação

#### 7.5.4.1 Fases

- Estabelecimento de Contato;
- Transferência de Informação;
- Conclusão.

O mestre controla a troca de fases e também é responsável pela continuidade da comunicação.

***Contato***

O mestre envia um ENQ, se a estação estiver preparada ela responde com um ACK, senão pode responder com um WACK ou RVI. Se ambos enviarem ENQ, a prioridade é do SC.

***Transmissão***

Para que os caracteres de controle não seja confundidos com dados (STX ≠ 'STX') colocamos o caracter DLE antes.

Se quisermos escrever DLE, devemos colocar o código DLE/DLE.

Um caracter BCS (verifica a soma de bloco) deve ser transmitido após cada telegrama (após ETX).

Se houver alguma anomalia, a estação responde com um NAK e repete-se a última mensagem.

### 7.5.5 Seqüência

Utilizada para permitir ao receptor determinar caso o telegrama consiste numa repetição do último, ou caso se trata de um novo.

Se o escravo receber um STX durante o texto, verifica-se o bit de seqüência.

## 7.6 Protocolo de Aplicação – ARAP

Consiste de 8 bits+1 bit de paridade. Cada telegrama contém 128 bytes no máximo. Cada mensagem contém 1 ou mais telegramas. A estrutura é composta por um cabeçalho e um Campo de Dados.

### 7.6.1 Cabeçalho

| Byte | Caracter Function | Observação |
|---|---|---|
| 0 e 1 | NOB | N° de bytes do telegrama (conta o cabeçalho |
| 2 | DESTINATION ADDRESS | Identificação do robô |
| 3 | SOURCE ADDRESS | Identificação do transmissor |
| 4 | FUNCTION CODE | Especifica a função da mensagem |

| | | |
|---|---|---|
| 5 | (NOTE USED) (MLI) (RS) (TT) | |
| 6 e 7 | FUNCTION SUFFIX | Informação adicional |

| | |
|---|---|
| MLI | 0: mensagem em 1 só telegrama<br>1: continua no próximo |
| RS | 0: reconhecimento positivo do último comando<br>1: reconhecimento negativo |
| TT | 01: comando<br>11: mensagem espontânea<br>10: resposta |

## 7.6.2 Campo de Dados

### 7.6.2.1 Comandos do SC ao IRB

- Início da execução do programa do robô;
- Parada de execução do programa do robô;
- Leitura do registrador através do SC;
- Escrita do registrador através do SC;
- Requisição de status através do SC;
- Troca de modo de operação através do SC.

# 8. COMUNICAÇÃO SERIAL

No PC é necessário uma porta serial RS232 (25 pinos); um transmissor/receptor assíncrono universal INS 8250 e um controlador de interrupção 8259 que por sua vez é um software programado em linguagem C ou Pascal.

Deve-se controlar as variações de velocidade de transmissão, paridade, bits de parada, bits de dados e a saída serial.

## 8.1 Função dos Pinos

| 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 | ← | Detector da Qualidade do Sinal |
| 22 | RI | ← | Indicador de Chamada |
| 23 | - | ←/→ | Indicar qual Baud Rate |
| 24 | - | → | Clock de Transmissão Síncrona |
| 25 | - | - | Não Definido |

# 9. SIMULAÇÃO

A proposta do Trabalho de Formatura é a simulação da Automação de uma Célula de Manufatura conforme a Seqüência de Operações para a fabricação de uma determinada peça descrita anteriormente:

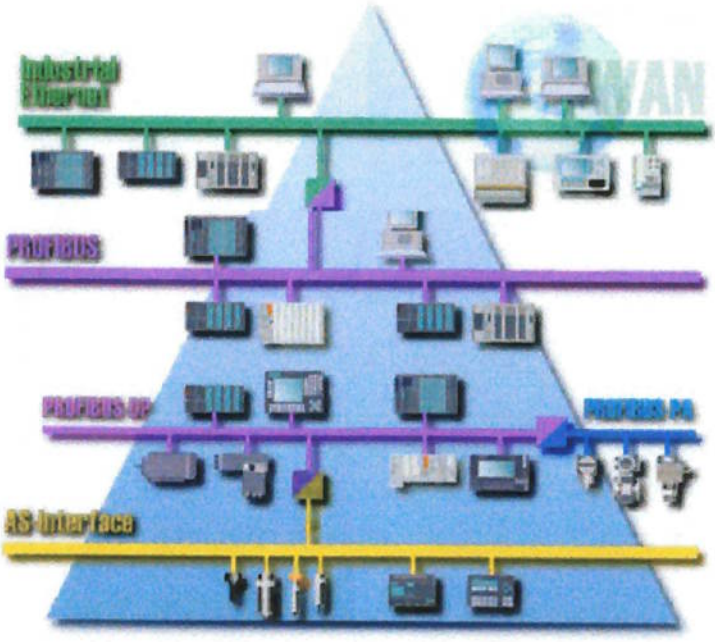
1. O robô recebe sinal que há peça no alimentador e a pega;
2. Fica em uma posição intermediária aguardando sinal de liberação da fresadora;
3. O robô coloca a peça na fresadora;
4. Vai para uma posição de espera, durante a usinagem;
5. O robô recebe o sinal de fim de usinagem e retira a peça da fresadora;
6. Fica em uma posição intermediária aguardando sinal de liberação do torno;
7. O robô coloca a peça no torno;
8. Vai para uma posição de espera, durante o torneamento;
9. O robô recebe o sinal de fim de usinagem e retira a peça do torno;
10. Coloca a peça no rack de peças prontas, e fica pronto para reiniciar o ciclo.

## 9.1 A Função do PLC

O PLC é responsável pela sincronia dos sinais provenientes dos elementos de campo (máquinas ferramentas, sensores, ...), elementos de interface homem-máquina e o robô.

Conforme a complexidade dos elementos e quantidade, adotamos uma ou outra arquitetura de rede. Por utilizarmos o PLC da linha SIMATIC S7 da SIEMENS, podemos optar pela rede MPI (Multipoint Interface), PROFIBUS e Industrial Ethernet. Atualmente já existe uma interface para redes à nível de PC's trafegando dados em TCP/IP.

### 9.1.1 Seleção da Rede




As diferenças em performance entre MPI, PROFIBUS e Industrial Ethernet dependem das condições marginais como o número de nós ou o tamanho das mensagens de dados. A principal condição marginal é o tempo de

acesso, ou seja, o tempo que um nó deve esperar antes que possa enviar uma mensagem. Quanto menor o tempo de acesso à rede, melhor a performance.

Outro critério de seleção é o custo x performance. O custo de instalação e manutenção são levados em conta. Por esse critério, a rede MPI apresenta o custo inferior e a Industrial Ethernet o superior.

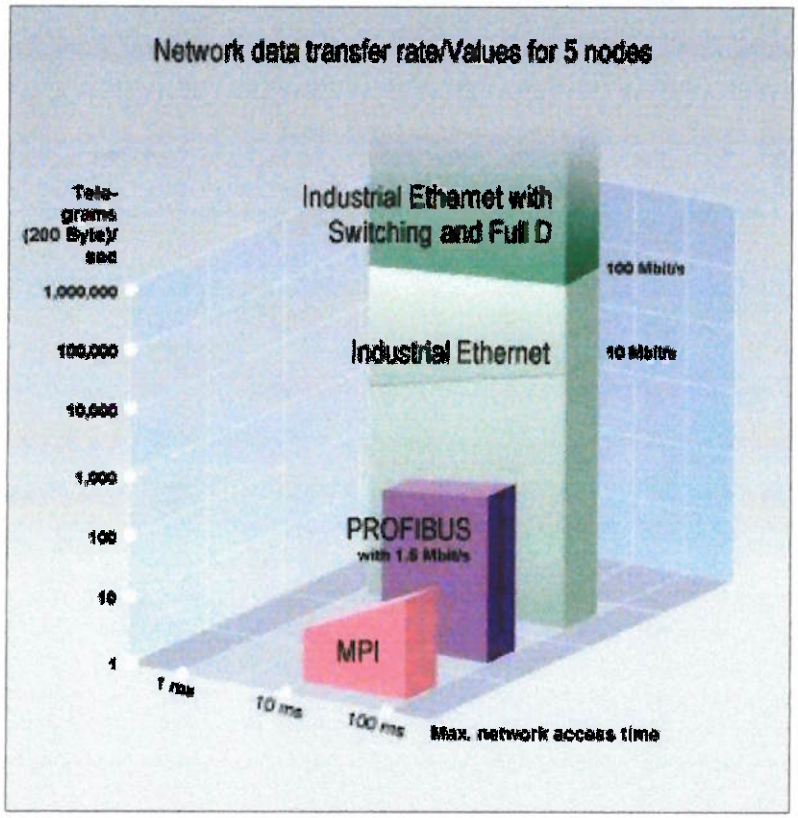



**Figura 11 - Comparação entre os tipos de Rede**

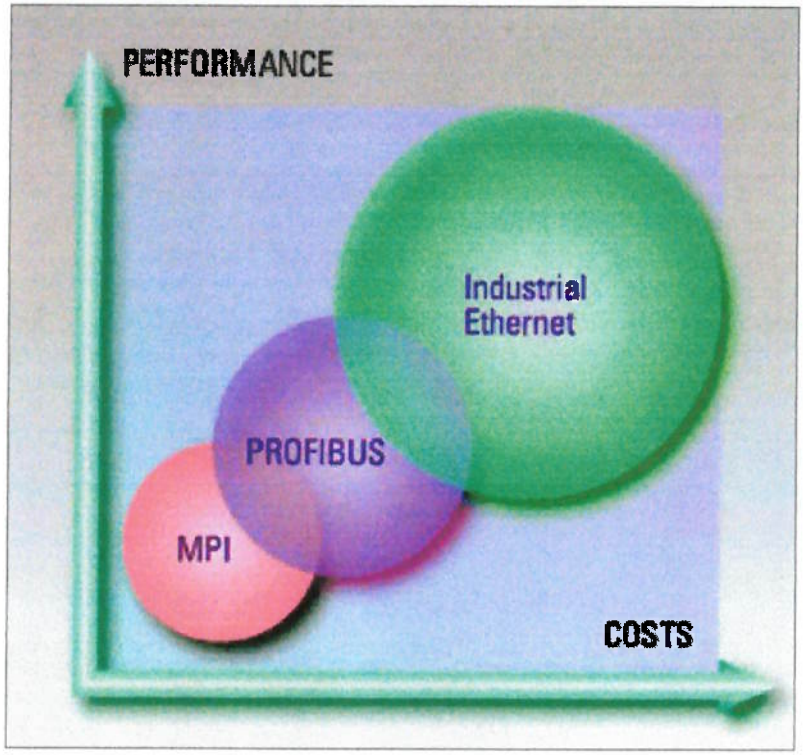



**Figura 12 - Performance x Custo de redes**

#### 9.1.1.1 Tabela para Seleção

| | MPI | PROFIBUS | Industrial Ethernet |
|---|---|---|---|
| Sistemas de Conexão | SIMATIC S7/M7/C7<br>SIMATIC PG/PC<br>SIMATIC HMI | SIMATIC S7/M7/C7<br>SIMATIC PG/PC<br>SIMATIC HMI<br>SMIATIC S5<br>SIMATIC 505 | SIMATIC S7/M7/C7<br>SIMATIC PG/PC<br>SIMATIC HMI<br>WORKSTATION<br>SIMATIC S5<br>SIMATIC 505 |
| Número de nós<br>- Típico<br>- Máximo | <br>2 a 10<br>32 | <br>2 a 16<br>126 | <br>2 a 100<br>acima de 1000 |
| Tamanho de dados típico por mensagem | 64 bytes | 120 bytes | 250 bytes |
| Tamanho da rede (LAN) | Até 100m | Até 9.6Km | Até 1.5Km |

| | | | |
|---|---|---|---|
| | | | (até 200Km para FO) |
| Topologia | Linha (bus) | Linha, árvore, anel simples e redundante, estrela | Linha, árvore, anel redundante e estrela |

A rede escolhida foi a MPI pois possuímos um número reduzido de sinais sendo gerenciados por apenas um PLC.

A implementação dessa rede envolve um cabo MPI que se conecta à CPU do PLC e, através de um conversor MPI-RS232, conectamo-lo à saída serial do computador. Dessa forma, podemos programar a lógica do PLC através de um software instalado no computador (STEP 7) e carregá-la no espaço de memória do PLC.

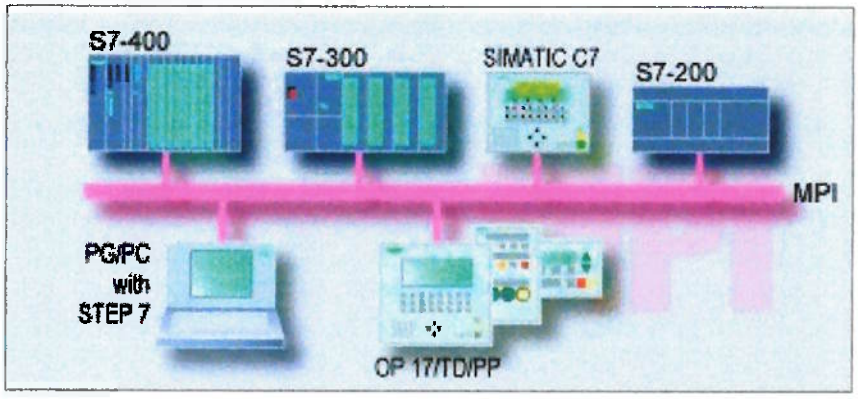



**Figura 13 - Rede MPI**

### 9.1.2 Sinais da Célula de Manufatura

Os sinais recebidos pelo PLC são do tipo booleano (1 ou 0):

| Informações para o PLC | Status dos Sinais |
|---|---|
| Alimentador de Peças | 1 – presença de peça<br>0 – ausência de peça |
| Fresadora Disponível | 1 – livre<br>0 – ocupada |
| Fresadora Usinou | 1 – usinou<br>0 – ainda usinando |
| Torno Disponível | 1 – livre<br>0 – ocupado |
| Torno Usinou | 1 – usinou<br>0 – ainda usinando |

Os sinais Alimentador de Peças, Fresadora Disponível e Torno Disponível são simulados através de botões (modo remoto do Supervisório) ou através de botões na tela do Supervisório (modo local).

Os sinais Fresadora Usinou e Torno Usinou são gerados no PLC, de forma a simular o tempo de usinagem. Na realidade, o status de término de usinagem seria gerado pelo CNC, indicando o término do programa de usinagem; na simulação, esse tempo de usinagem é atribuído no início do

processo através de um quadro de entrada de variáveis presente no software SCADA. O PLC recebe a informação do robô que a peça foi colocada na fresadora (ou no torno). Nesse momento um temporizador é disparado e, após o período definido no supervisório, ativa o bit correspondente.

O PLC ainda recebe três bits provenientes do robô que indicam o ponto de execução do programa do robô (seqüência de movimentos) e envia para ele cinco bits de comando que indicarão a tarefa que o robô deverá realizar.

Podemos observar a seguir um diagrama esquemático indicando os sinais digitais de entrada e saída do PLC.

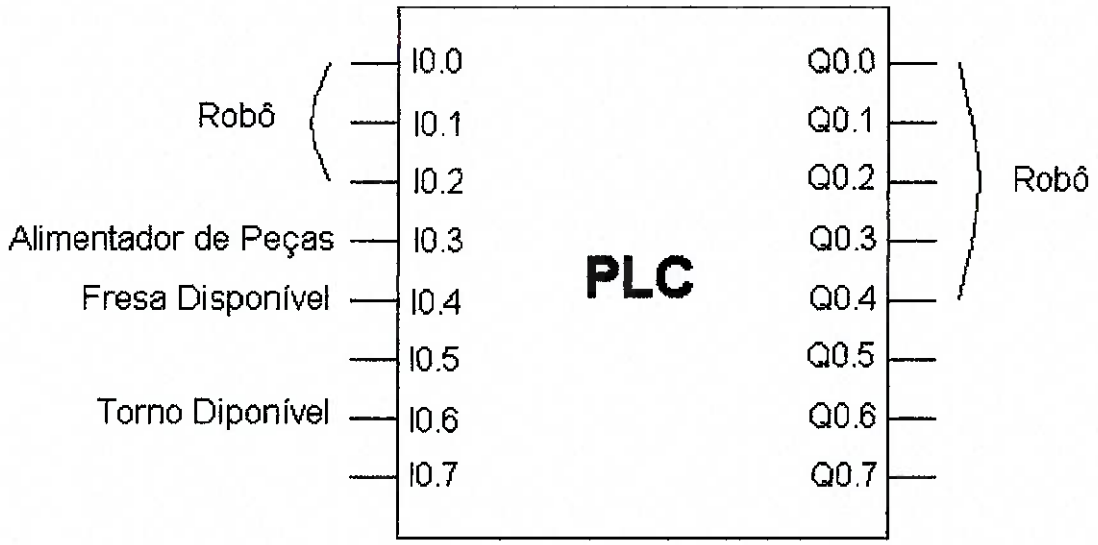



**Figura 14 - Esquema de Ligação do PLC**

No programa do robô foram reservados dois registradores: R4 é utilizado para armazenar os sinais de comando enviados pelo PLC e o registrador R5 é utilizado para enviar ao PLC o status do robô. A tabela a seguir mostra de

maneira simples a relação entre os diversos sinais e a tarefa correspondente que deverá ser executada pelo robô

| I0.2 | I0.1 | I0.0 | I0.3 | I0.4 | Interno | I0.6 | Interno | R4 | Tarefa indicada ao Robô |
|---|---|---|---|---|---|---|---|---|---|
| Robô (R5) | | | Alimentador | Fresa Disp. | Fresa Usinou | Torno Disp. | Torno Usinou | Saída - PLC | |
| 0 | 0 | 1 | 1 | x | x | x | x | Q0.0=1 | Pegue peça |
| 0 | 1 | 0 | x | 1 | x | x | x | Q0.1=1 | Coloque na fresa |
| 0 | 1 | 1 | x | x | 1 | x | x | Q0.2=1 | Retire da fresa |
| 1 | 0 | 0 | x | x | x | 1 | x | Q0.3=1 | Coloque no torno |
| 1 | 0 | 1 | x | x | x | x | 1 | Q0.4=1 | Retire do torno |

0 = sinal não ativo

1 = sinal ativo

x = irrelevante (0 ou 1)

## 9.2 Programa do Robô

### 9.2.1 Listagem da Lógica do Programa

O programa a ser carregado no robô deverá executar a seguinte lógica:

```
        Inicialização;          { Rotina de inicialização do robô }
        Faz  R5=1;              { R5=1 indica ao  PLC "robô pronto para operar" }
        Faz R4=0;               { Carrega R4=0. R4 será recebido do PLC }
LBL:    Envia R5 ao PLC;
        Recebe R4 do PLC;       { PLC indica ao robô que tarefa executar }
        Se R4=1 Jump to A;
        Se R4=2 Jump to B ;
        Se R4=4 Jump to C;
        Se R4=8 Jump to D;
        Se R4=16 Jump to E;
A:      Pega peça;
        Faz R5=2;               { Robô indica ao PLC "peguei a peça" }
        Jump to LBL;
B:      Coloca peça na fresadora;
        Faz R5=3;               { Robô indica ao PLC "coloquei na fresadora" }
        Jump to LBL;
```

C: Retira peça da fresadora;

Faz R5=4; { Robô indica ao PLC "retirei da fresadora" }

Jump to LBL;

D: Coloca peça no torno;

Faz R5=5; { Robô indica ao PLC "coloquei no torno" }

Jump to LBL;

E: Retira peça do torno;

Faz R5=1; { Robô indica ao PLC "retirei do torno" }

Jump to LBL;

## 9.3 Programa do PLC (STEP 7)

O programa simplificado do PLC é mostrado a seguir:

A I0.0

NA I0.1

NA I0.2

A I0.3

= Q0.0 {Se robô enviou 001 (R5=1) e há peça (I0.3=1) Q0.0 é ativada}

```
NA I0.0
A I0.1
NA I0.2
A I0.4
= Q0.1      {Se robô enviou 010 (R5=2) e fresa livre (I0.4=1) Q0.1 é ativada}

A I0.0
A I0.1
NA I0.2
A F0.5      { F0.5 é gerado internamente através de um temporizador }
= Q0.2      {Se robô enviou 011 (R5=3) e fresa usinou (F0.5=1) Q0.2 é ativada}

NA I0.0
NA I0.1
A I0.2
A I0.6
= Q0.3      {Se robô enviou 100 (R5=4) e torno livre (I0.6=1) Q0.3 é ativada}

A I0.0
NA I0.1
```

```
A I0.2
A F0.7          { F0.7 é gerado internamente através de um temporizador }
= Q0.4          {Se robô enviou 101 (R5=5) e torno usinou (F0.7=1) Q0.4 é
ativada}
```

Vale lembrar que o programa final do PLC apresenta mais algumas funções necessárias à interligação PLC – Supervisório. O trecho destacado anteriormente busca elucidar apenas o tratamento efetuado nos sinais dos dispositivos. O programa real pode ser observado no anexo.

O robô da ASEA dispõe de 7 entradas e 6 saídas digitais, além de 2 saídas digitais específicas para controle da garra. Na atual configuração estão sendo usadas 5 entradas e 3 saídas digitais do robô. Assim é possível expandir o sistema em futuros trabalhos. Quanto ao PLC, estão sendo usadas 5 entradas e 5 saídas digitais.

## 9.4 Programa SCADA – Supervisório

Vale lembrar que a função do WinCC é de monitoração do processo de manufatura. Numa situação real, utilizamos o software Supervisório numa sala de controle operacional que fica distante do chão de fábrica. O Supervisório permite, além da monitoração, o controle e gerenciamento de informações relevantes ao processo. Por exemplo, podemos armazenar num banco de

dados uma série de "receitas", chamadas "scripts" sobre os parâmetros envolvidos numa determinada operação. Podemos controlar a demanda de fabricação de uma determinada peça, ou a quantidade de uma certa substância a ser misturada num produto, etc.

Na nossa simulação o Supervisório irá monitorar os sinais relacionados às máquinas ferramentas e ao robô de forma a cumprir a sincronia das operações. Em outras palavras, o Supervisório será o responsável em interpretar os sinais recolhidos pelo PLC e mostrá-lo ao usuário que estará operando o microcomputador.

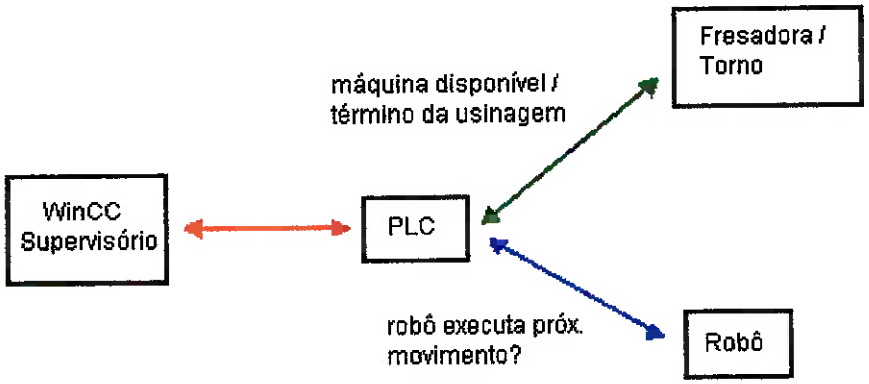
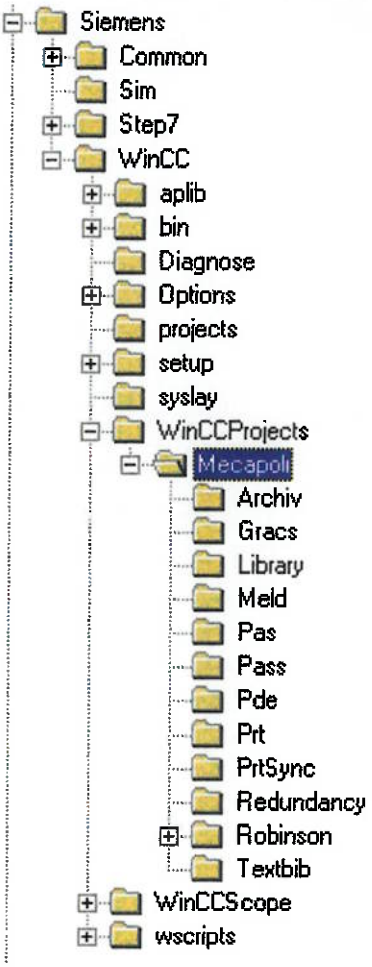



**Figura 15 - Esquema de Funcionamento do PLC**

### 9.4.1 Projeto MecaPoli

Quando iniciamos o WinCC pela primeira vez devemos criar um novo projeto que irá gerenciar todas as funções descritas anteriormente no item "PLC – CONTROLADOR LÓGICO PROGRAMÁVEL).




O Projeto criado chama-se MecaPoli e todas as informações relativas ao projeto ficam armazenadas no diretório C:\Siemens\WinCC\WinCCProjects\MecaPoli.

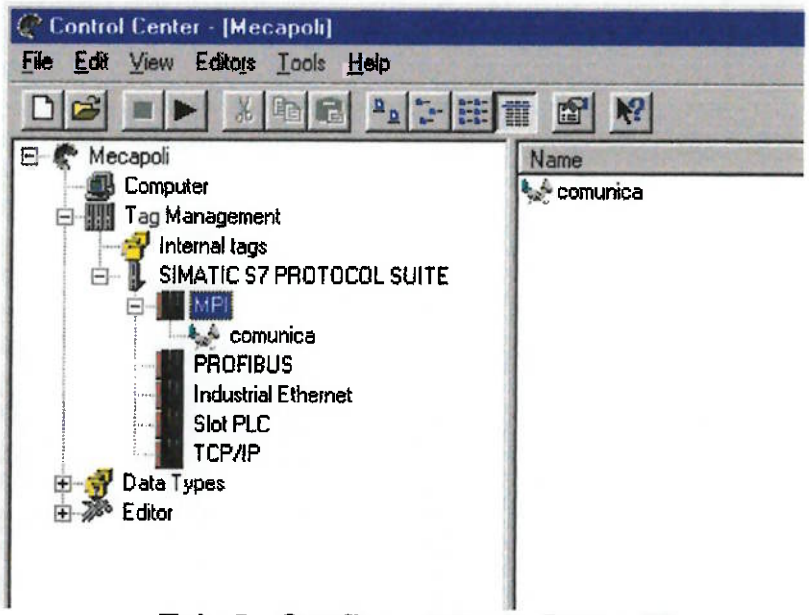
O WinCC possui licenças referentes a criação e utilização de TAG's. A versão DEMO permite a utilização de todos os recursos porém limita a quantidade de TAG's.

TAG's são variáveis que representam uma determinada área de memória do PLC. Por exemplo, se no PLC chamarmos uma variável de entrada, de tamanho 1bit, de I0.0, podemos associar uma TAG, também chamado "mnemônico", de nome "BOTÃO 1" à esse endereço e que possuirá o tamanho de 1 bit.

#### 9.4.1.1 Gerenciamento de TAG's

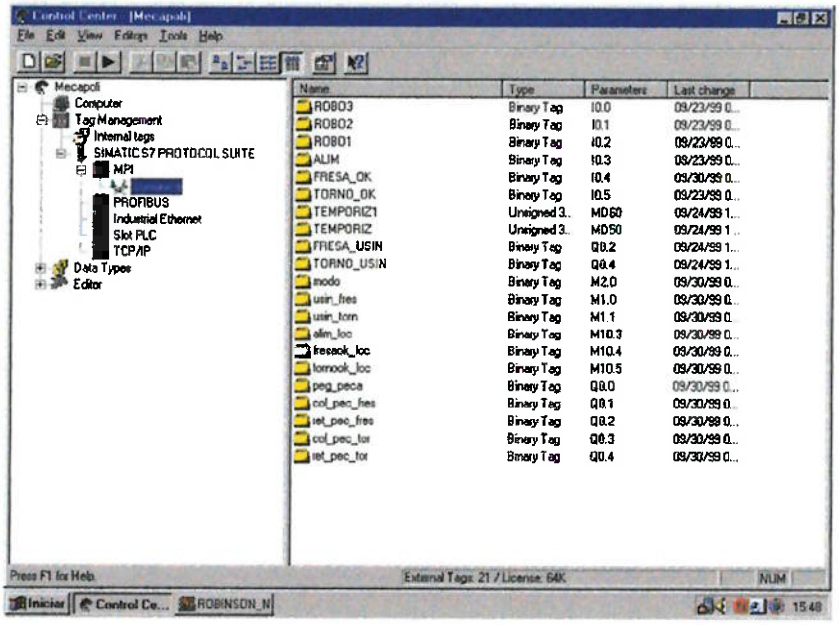
No projeto, devemos configurar a comunicação entre o supervisório (instalado num PC) e o PLC.

Para isso instalamos um drive para o reconhecimento do PLC da linha SIMATIC S7 e então definimos o protocolo de comunicação (MPI).




**Tela 5 - Configuração da Rede MPI**

Configurado a rede a ser utilizado, basta agora definirmos as variáveis que queremos monitorar. Na criação de um novo TAG atribuímos um nome "mnemônico", estabelecemos o endereço referente à variável no PLC e podemos também definir o valor inicial.



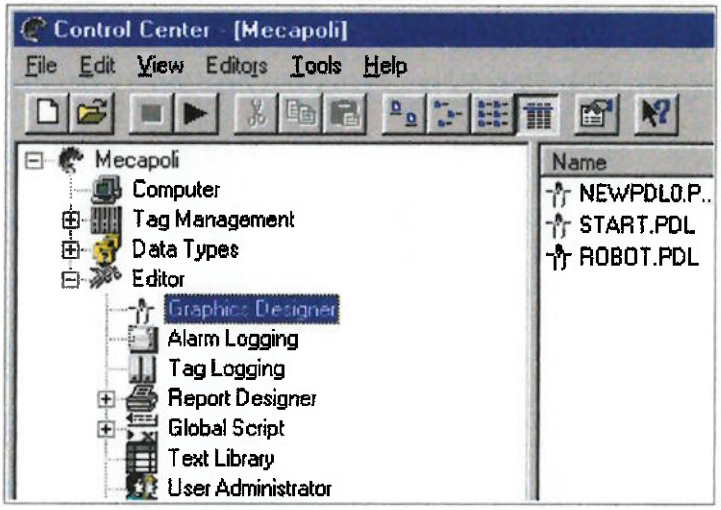

**Tela 6 - Lista de TAG's**

| TAG | Tipo | Endereço | Função |
|---|---|---|---|
| ROBO3 | Binário | I0.0 | Enviar ao PLC o status do Robô (término de algum movimento) |
| ROBO2 | Binário | I0.1 | |
| ROBO1 | Binário | I0.2 | |
| ALIM | Binário | I0.3 | Status de há/não há peças |
| FRESA_OK | Binário | I0.4 | Status de fresadora disponível (sem peça) |
| TORNO_OK | Binário | I0.5 | Status de torno disponível (sem peça) |
| usin_fres | Binário | M1.0 | Status de que a fresadora usinou |
| usin_torn | Binário | M1.1 | Status de que o torno usinou |
| alim_loc | Binário | M10.3 | Status de há/não há peças (modo local) |
| fresaok_loc | Binário | M10.4 | Status de fresadora disponível (modo local) |
| tornook_loc | Binário | M10.5 | Status de torno disponível (modo local0 |
| modo | Binário | M2.0 | Define o modo de operação (local/remoto) |
| TEMPORIZ | 32-bit (unsigned) | MD50 | Armazena o tempo de usinagem da fresadora |
| TEMPORIZ1 | 32-bit (unsigned) | MD60 | Armazena o tempo de usinagem do torno |
| peg_peca | Binário | Q0.0 | PLC avisa o Robô se pode pegar peça |
| col_pec_fres | Binário | Q0.1 | PLC avisa o Robô se pode colocar peça na fresadora |
| FRESA_USIN | Binário | Q0.2 | PLC avisa o Robô se a fresadora já usinou |
| col_pec_tor | Binário | Q0.3 | PLC avisa o Robô se pode colocar peça no torno |
| TORNO_USIN | Binário | Q0.4 | PLC avisa o Robô se o torno já usinou |

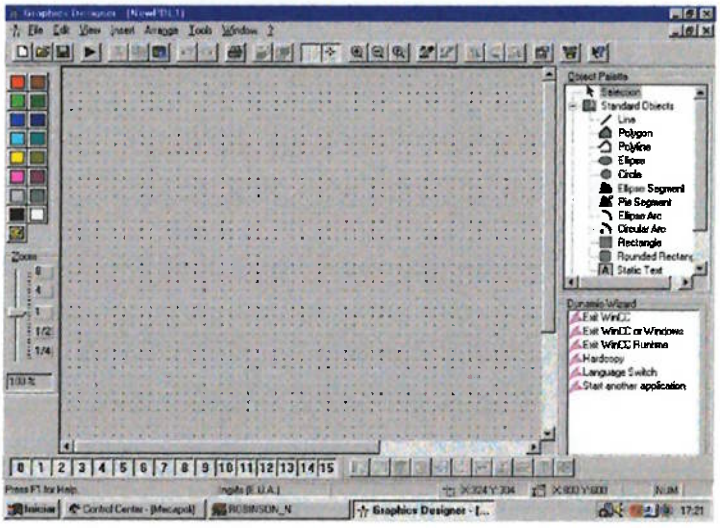
### 9.4.1.2 Editor de Telas (Monitoração)

Tendo sido definidas as variáveis de monitoração, podemos criar telas que as utilizarão.

O editor de telas do WinCC assemelha-se à tela de uma linguagem visual. Inclusive podemos criar objetos e programar ações e intertravamentos em linguagem C.




**Tela 7 - Editor de Telas do WinCC**

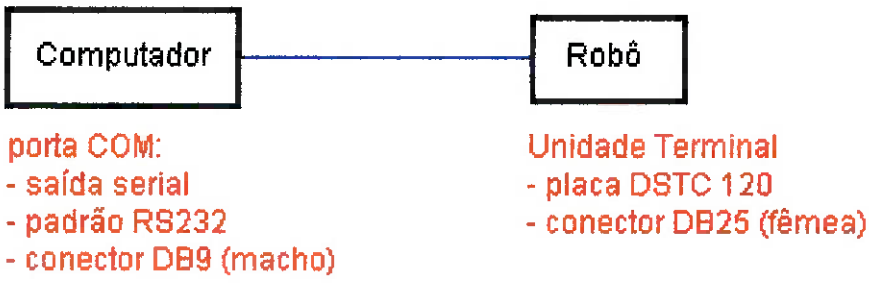


## 9.5 Comunicação Serial

A Integração do Trabalho de Formatura "Transferência de Dados entre o Robô e o PC" – Jussara Koga e Marcelo Massaki Kawaguchi – 1998 ao Trabalho de Formatura em questão se daria no escopo da execução do programa em linguagem C desenvolvido através do Supervisório.

Estando funcionando o programa que possibilitaria a comunicação direta PC-Robô, o objetivo do Supervisório seria a execução dos comandos de transferência de dados na tela de monitoramento de todo o processo.

A comunicação entre o PC e o Robô é feita através de um cabo seguindo a pinagem sugerida pelos formandos de 1998



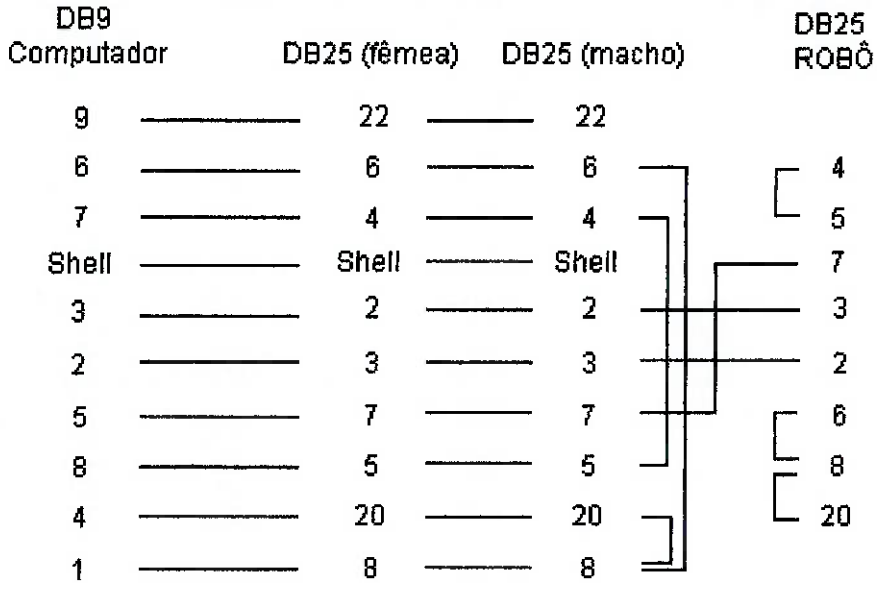



**Figura 16 - Esquema da Pinagem da Comunicação Serial**

### 9.5.1 Programa

O programa estabelece a comunicação entre o PC (Superior Computer) e o Robô (IRB) e possui um menu de opções de transferência de dados.

Primeiramente detecta-se qual porta de comunicação está sendo utilizada (COM1 ou COM2), envia-se um caracter de controle ENQ e espera-se um ACK, a seguir envia-se um DLE e um STX avisando que se inicializará a transferência de dados.

As opções contidas no menu são as seguintes:

```
Voce tem as seguintes opcoes:

1 - Enviar um programa ou um bloco de programa ao robo.

2 - Iniciar um programa no robo

3 - Parar um programa que esta rodando.

4 - Enviar um valor a um registrador do robo.

5 - Receber um programa ou um bloco de programa do robo.

6 - Sair

Digite a sua opcao:
```

**Tela 8 - Programa de Comunicação Serial (PC-Robô)**

### 9.5.2 Problemas com a Comunicação

Após um estudo detalhado sobre a comunicação serial entre o PC e o Robô, protocolos de comunicação, Trabalho de Formatura de 1998 e o programa feito em linguagem C, encontramos alguns problemas agravantes que não possibilitaram a integração deste com o atual Trabalho de Formatura.

Um dos problemas encontrados foi a pinagem do cabo de comunicação que não estava de acordo com o explicitado no documento. Havia dois cabos, sendo que um convertia DB9 (saída do PC) em DB25 e outro que efetivamente comunicava com a porta serial do Robô (ambos com conectores DB25). Para solucionarmos esse problema refizemos o primeiro cabo de acordo com a

norma para conversão (DB9-DB25 em RS232) e o segundo de acordo com a pinagem explicitada no documento.

Outro problema encontrado foi que, apesar de termos refeito os cabos, o programa não funcionava de acordo com o esperado. Chegamos a observar a troca dos caracteres de controle (ENQ, ACK, DLE, STX) mas as opções do menu não mostravam quaisquer efeitos. Por fim decidimos estudar mais cuidadosamente cada linha do programa e das bibliotecas para identificar possíveis erros lógicos na programação mas não conseguimos solucionar o problema. É importante ressaltar que em momento algum alteramos o programa fonte.

# 10. CONSIDERAÇÕES FINAIS

A grande complexidade do problema proposto envolvendo diversos conceitos diferentes nos levou a adotarmos uma estratégia de buscarmos o entendimento de cada parte isoladamente para então integrá-los.

Primeiramente observamos quais eram essas partes e então estudamos a literatura disponível a respeito.

Efetuamos alguns testes ao longo do projeto e cabe salientar que nosso objetivo era criar uma simulação que estivesse mais próxima da realidade. Para isso inicialmente fizemos um programa simplificado integrando o PLC e o Robô cujos resultados foram satisfatórios. A seguir estudamos a relação entre o PLC e o Supervisório.

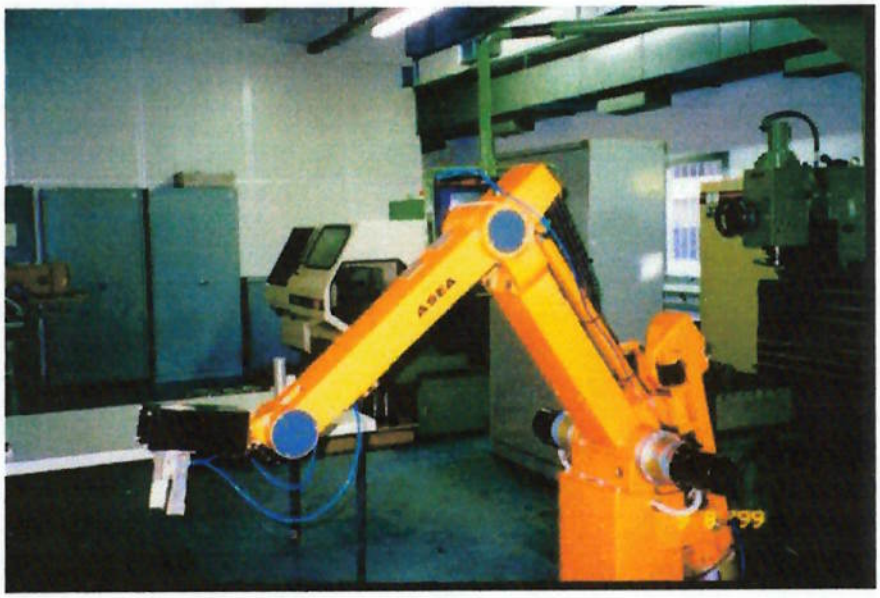
Isoladamente estudamos o hardware e o software referentes ao trilho conseguindo colocá-lo em operação. Quanto à comunicação, após estudo detalhado do programa, descobrimos a inviabilidade de agregamo-lo ao projeto.

Por fim, reunimos todas essas informações e elaboramos o programa final do Robô, o programa final do PLC e o programa final do Software Supervisório conseguindo uma simulação próxima ao pretendido.

É importante destacar a impossibilidade de executarmos esses testes finais devido aos problemas incorridos com o robô quando fomos colocá-lo em funcionamento.

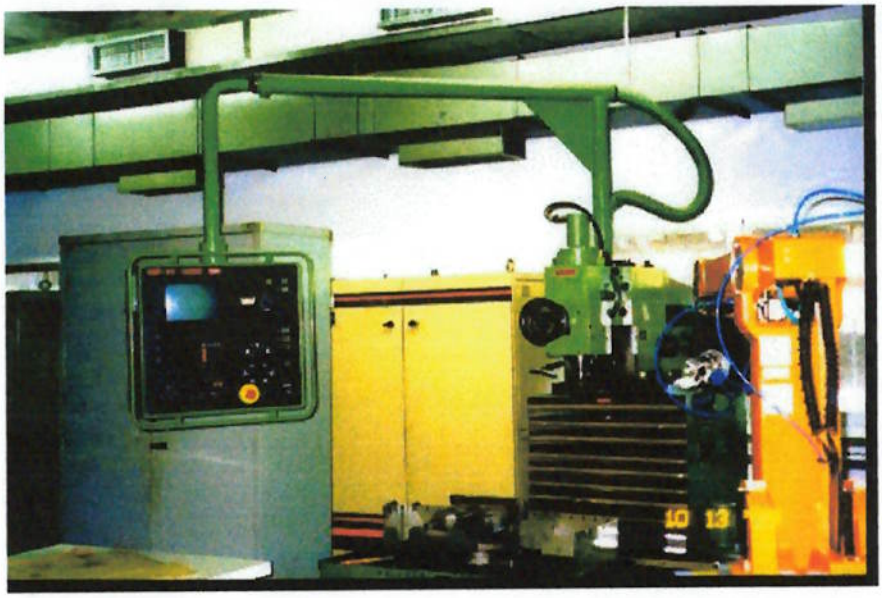
# 11. ANEXOS

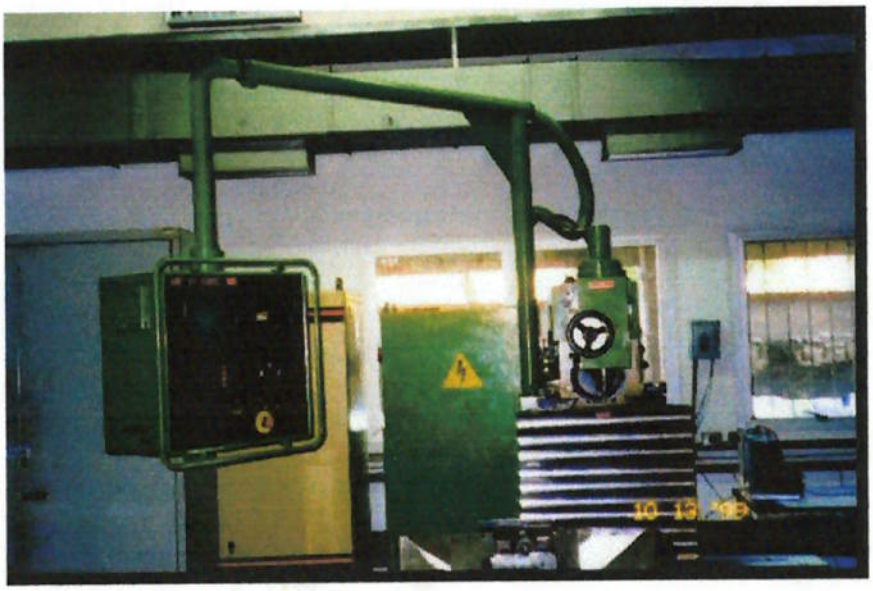
## 11.1 Fotos do Robô em Operação


## 11.2 Fotos dos Componentes da Célula de Manufatura


ASEA






## 11.3 Listagem do Programa de Comunicação Serial

### 11.3.1 Programa Principal (Protocol.c)

```
/*Comunicacao direta com a porta serial atraves da programação da*/
/*UART*/

//Valores de configuracao da porta serial

#define TAXA 9600L      /*bps*/
#define PARIDADE 2  /*0=nehuma, 1=impar, 2=par*/
#define STOP 1
#define BITS_DADOS 8

#define TRUE 1

#include <moldura.h>
#include <wait.h>
```

```
#include <stdio.h>
#include <inicia.h>
#include <stdlib.h>
#include <receber.h>
#include <comunica.h>
#include <envia.h>

int COM0;
FILE *f;

void main()
{
       int i,ok,status,func,funs,param[5],resp[10];
       unsigned char ch, stat,bcs,bcs1;
       int opcao,opcao1,envia,fim,prog;
       char *nome,op;

  while (TRUE)
     {
     do
       {
       clrscr();
       //Moldura();
       gotoxy (15,13);
       cprintf ("Qual a porta de comunicacao? (1) ou (2)");
       gotoxy (15,16);
       while (!kbhit());
       if (kbhit())
        {
        opcao=getch()-48;
        cprintf("%d",opcao);
```

```
        }
        }
      while ((opcao<1)||(opcao>2));
      if (opcao==1)
        {
        COM0=0x3f8;
        break;
        }
      if (opcao==2)
        {
        COM0=0x2f8;
        break;
        }
  }
//Configura a porta serial de acordo com a configuracao do robo
//e com a porta serial utilizada pelo computador

  inicializa (COM0,TAXA,PARIDADE,STOP,BITS_DADOS);

// Montagem da tela inicial

  while (TRUE)
   {
   outportb(COM0+5,0);
   status=0;
   do
      {
      opcao=0;
      clrscr();
      //Moldura();
      gotoxy (15,7);
```

```
        cprintf ("Voce tem as seguintes opcoes:");
        gotoxy (15,9);
        cprintf ("1 - Enviar um programa ou um bloco de programa ao robo.");
        gotoxy (15,11);
        cprintf ("2 - Iniciar um programa no robo");
        gotoxy (15,13);
        cprintf ("3 - Parar um programa que esta rodando.");
        gotoxy (15,15);
        cprintf ("4 - Enviar um valor a um registrador do robo.");
        gotoxy (15,17);
        cprintf ("5 - Receber um programa ou um bloco de programa do robo.");
        gotoxy (15,19);
        cprintf ("6 - Sair");
        gotoxy (15,21);
        cprintf ("Digite a sua opcao: ");

// Ocorre interrupcao quando alguma tecla e pressionada ou
// quando ha um dado novo na porta serial
        while (!kbhit())
          {
          status=0;
          status=inportb(COM0+5);
          if (status&1)
            {
            receber_da_serial(COM0);
            break;
            }
          }
        if (kbhit())
          {
          opcao=getch()-48;          // Verifica qual a tecla pressionada
```

```
        cprintf ("%d",opcao);
        }
      }
   while ((opcao==5)||(opcao>8)||(opcao<1));

// Executa as funcoes de acordo com a tecla pressionada
   switch (opcao)
     {
     case 1:    // Transferencia de programa ou bloco de programa
      //Moldura();
      envia_prog(COM0);
      break;
     case 2:
      ok=0;
      while (!ok)
       {
       do
       {
       //Moldura();
       gotoxy(15,11);
       cprintf ("Opcao 2 - Iniciar programa no robo");
       gotoxy(15,13);
       cprintf ("Tecle a opcao:");
       gotoxy(15,14);
       cprintf("A) Reiniciar o programa");
       gotoxy(15,15);
       cprintf("B) Iniciar um outro programa");
       gotoxy(15,17);
       cprintf("OPCAO==> ");
       scanf("%c",&op);
       }
```

```
while ((op!='a')&&(op!='b')&&(op!='A')&&(op!='B'));
gotoxy (15,17);
cprintf("OPCAO ESCOLHIDA: %c",op);
if ((op=='a')||(op=='A'))
 {
 param[0]=1;
 param[3]=0;
 }
else
 {
 param[0]=0;
 do
  {
  gotoxy(10,18);
  cprintf ("Entre com o numero do programa");
  scanf("%d",&param[3]);
  }
 while ((param[3]<0)||(param[3]>9999));
 }
param[1]=param[3]>>8;
param[2]=param[3]&0x00ff;
comunica(COM0);
outportb(COM0,0);     // Envia o numero de bytes
bcs=0;
espera_envio(COM0);
outportb(COM0,10);     // Envia o numero de bytes
bcs=bcs^10;
espera_envio(COM0);
outportb(COM0,120);     // Envia o endereco de destino
bcs=bcs^120;
espera_envio(COM0);
```

```
outportb(COM0,0);     // Envia o endereco fonte
bcs=bcs^0;
espera_envio(COM0);
outportb(COM0,2);     // Envia o codigo da funcao
bcs=bcs^2;
espera_envio(COM0);
outportb(COM0,1);
bcs=bcs^1;
espera_envio(COM0);
outportb(COM0,0);     // Envia o sufixo da funcao
bcs=bcs^0;
espera_envio(COM0);
outportb(COM0,param[0]);     // Envia o sufixo da funcao
bcs=bcs^param[0];
espera_envio(COM0);
outportb(COM0,param[1]);     // Envia o numero do programa
bcs=bcs^param[1];
espera_envio(COM0);
outportb(COM0,param[2]);     // Envia o numero do programa
bcs=bcs^param[2];
espera_envio(COM0);
outportb(COM0,0X10);       //Envia DLE
espera_envio(COM0);
outportb(COM0,0X03);       //Envia ETX
bcs=bcs^0x03;
espera_envio(COM0);
outportb(COM0,bcs);        //Envia BCS
espera(COM0);
bcs1=inportb(COM0);
if (bcs1==0x06)
 {
```

```
outportb(COM0,0X10);      //Envia DLE
espera_envio(COM0);
outportb(COM0,0X04);      //Envia EOT
espera_envio(COM0);
ok=1;
}
}

//Verifica se os dados foram enviados com sucesso
espera(COM0);
param[0]=inportb(COM0);
if (param[0]==5)
{
outportb(COM0,6);
espera(COM0);
param[1]=inportb(COM0);
if (param[1]==0x10)
  {
  espera(COM0);
  param[2]=inportb(COM0);
  if (param[2]==2)
    {
    for (param[3]=0;param[3]<10;param[3]++)
      {
      espera(COM0);
      resp[param[3]]=inportb(COM0);
      }
    if (resp[1]==8)
      {
      gotoxy(15,20);
      cprintf("COMANDO EXECUTADO COM SUCESSO!!");
```

```
        }
      else
        {
        gotoxy (15,19);
        cprintf ("OCORREU UM ERRO!!");
        param[1]=resp[8]<<8;
        param[1]=param[1]+resp[9];
        gotoxy(15,20);
        cprintf("Vefique o erro numero: %d",param[1]);
        }
      }
    }
  }
  gotoxy(15,22);
  cprintf ("Pressione alguma tecla para continuar");
  while (!kbhit());
  op=getch();
  break;

case 3:
 ok=0;
 while (!ok)
  {
  //Moldura();
  gotoxy (15,10);
  cprintf ("Opcao 3 - Para um programa que esta rodando no robo:");
  param[1]=0;
  param[2]=0;
  comunica(COM0);
  outportb(COM0,0);     // Envia o numero de bits
  bcs=0;
```

```
espera_envio(COM0);
outportb(COM0,8);     // Envia o numero de bits
bcs=bcs^8;
espera_envio(COM0);
outportb(COM0,120);     // Envia o endereco de destino
bcs=bcs^120;
espera_envio(COM0);
outportb(COM0,0);     // Envia o endereco fonte
bcs=bcs^0;
espera_envio(COM0);
outportb(COM0,3);     // Envia o codigo da funcao
bcs=bcs^3;
espera_envio(COM0);
outportb(COM0,1);
bcs=bcs^1;
espera_envio(COM0);
outportb(COM0,0);     // Envia o sufixo da funcao
bcs=bcs^0;
espera_envio(COM0);
outportb(COM0,0);     // Envia o sufixo da funcao
bcs=bcs^param[0];
espera_envio(COM0);
outportb(COM0,0X10);
espera_envio(COM0);
outportb(COM0,0X03);
bcs=bcs^0x03;
espera_envio(COM0);
outportb(COM0,bcs);
espera(COM0);
bcs1=inportb(COM0);
if (bcs1==0x06)
```

```
 {
 outportb(COM0,0X10);  //Envia DLE
 espera_envio(COM0);
 outportb(COM0,0X04);  //Envia EOT
 espera_envio(COM0);
 ok=1;
 }
}

//Verifica se os dados foram enviados com sucesso
espera(COM0);
param[0]=inportb(COM0);
if (param[0]==5)
 {
 outportb(COM0,6);
 espera(COM0);
 param[1]=inportb(COM0);
 if (param[1]==0x10)
   {
   espera(COM0);
   param[2]=inportb(COM0);
   if (param[2]==2)
     {
     for (param[3]=0;param[3]<10;param[3]++)
       {
       espera(COM0);
       resp[param[3]]=inportb(COM0);
       }
     if (resp[1]==8)
       {
       gotoxy(15,20);
```

```
            cprintf("COMANDO EXECUTADO COM SUCESSO!!");
            }
          else
            {
            gotoxy (15,19);
            cprintf ("OCORREU UM ERRO!!");
            param[1]=resp[8]<<8;
            param[1]=param[1]+resp[9];
            gotoxy(15,20);
            cprintf("Vefique o erro numero: %d",param[1]);
            }
          }
        }
      }
    gotoxy(15,22);
    cprintf ("Pressione alguma tecla para continuar");
    while (!kbhit());
    op=getch();
    break;

  case 4:
   ok=0;
   while (!ok)
    {
    //Moldura();
    gotoxy (15,10);
    cprintf ("Entre com o numero do registrador (0 a 255):");
    do
     {
     gotoxy (15,11);
     scanf ("%d",&param[0]);
```

```
        }
        while ((param[0]<0)||(param[0]>255));
        gotoxy (15,13);
        cprintf ("Valor a ser enviado para o registrador %d(0 a
9999)",param[0]);
        do
        {
        gotoxy(15,14);
        scanf ("%d",&param[3]);
        }
        while ((param[3]<0)||(param[3]>9999));
        param[1]=param[3]>>8;
        param[2]=param[3]&0x00ff;
        comunica(COM0);
        outportb(COM0,0);     // Envia o numero de bits
        bcs=0;
        espera_envio(COM0);
        outportb(COM0,10);     // Envia o numero de bits
        bcs=bcs^10;
        espera_envio(COM0);
        outportb(COM0,120);     // Envia o endereco de destino
        bcs=bcs^120;
        espera_envio(COM0);
        outportb(COM0,0);     // Envia o endereco fonte
        bcs=bcs^0;
        espera_envio(COM0);
        outportb(COM0,14);     // Envia o codigo da funcao
        bcs=bcs^14;
        espera_envio(COM0);
        outportb(COM0,1);
        bcs=bcs^1;
```

```
espera_envio(COM0);
outportb(COM0,0);     // Envia o sufixo da funcao
bcs=bcs^0;
espera_envio(COM0);
outportb(COM0,param[0]);     // Envia o sufixo da funcao
bcs=bcs^param[0];
espera_envio(COM0);
outportb(COM0,param[1]);     // Envia o valor do registrador
bcs=bcs^param[1];
espera_envio(COM0);
outportb(COM0,param[2]);     // Envia o restante do valor desejado
bcs=bcs^param[2];
espera_envio(COM0);
outportb(COM0,0X10);     // Envia DLE
espera_envio(COM0);
outportb(COM0,0X03);     // Envia ETX
bcs=bcs^0x03;
espera_envio(COM0);
outportb(COM0,bcs);      // Envia BCS
espera(COM0);
bcs1=inportb(COM0);
if (bcs1==0x06)
 {
 outportb(COM0,0X10);    //Envia DLE
 espera_envio(COM0);
 outportb(COM0,0X04);
 espera_envio(COM0);     //Envia EOT
 ok=1;
 }
}
```

```
//Verifica se os dados foram enviados com sucesso.
espera(COM0);
param[0]=inportb(COM0);
if (param[0]==5)
 {
 outportb(COM0,6);
 espera(COM0);
 param[1]=inportb(COM0);
 if (param[1]==0x10)
   {
   espera(COM0);
   param[2]=inportb(COM0);
   if (param[2]==2)
     {
     for (param[3]=0;param[3]<10;param[3]++)
       {
       espera(COM0);
       resp[param[3]]=inportb(COM0);
       }
     if (resp[1]==8)
       {
       gotoxy(15,20);
       cprintf("COMANDO EXECUTADO COM SUCESSO!!");
       }
     else
       {
       gotoxy (15,19);
       cprintf ("OCORREU UM ERRO!!");
       param[1]=resp[8]<<8;
       param[1]=param[1]+resp[9];
       gotoxy(15,20);
```

```
            cprintf("Vefique o erro numero: %d",param[1]);
            }
          }
        }
      }
    gotoxy(15,22);
    cprintf ("Pressione alguma tecla para continuar");
    while (!kbhit());
    op=getch();
    break;

  case 5:
    //comunica(COM0);
    ok=0;
    //Moldura();
    gotoxy(10,10);
    cprintf("Entre com o dado");
    gotoxy(10,11);
    scanf("%d",&bcs);
    while (!kbhit())
   outportb(COM0,bcs);

    break;
  case 6:
    textbackground(0);
    textcolor(15);
    system("cls");
    exit(1);
  default:
    break;
  }
```

```
   }
}
```

### 11.3.2 Comunica.h

```
#include <stdio.h>
#include <dos.h>


void comunica(int COM0)
{
unsigned char status,num,ok;

Moldura();
gotoxy(10,13);
cprintf ("Efetuando comunicacao");
ok=0;
outportb(COM0+5,0);
while (!ok)
  {
  outportb(COM0,0x05);  // Envia ENQ para o robo
  gotoxy (10,15);
  cprintf("ENQ enviado");
  espera(COM0);
  num=inportb(COM0);
  switch (num)
       {
       case 0x06:            // Se a resposta for ACKNOLODGE
        {
        outportb(COM0,0x10);   // Envia DLE
```

```
        gotoxy (10,16);
        cprintf("DLE enviado ");
        espera_envio(COM0);
        outportb(COM0,0X02);       // Envia STX par
        gotoxy (10,17);
        cprintf("STX enviado ");
        espera_envio(COM0);
        ok=1;
        break;
        }
       case 0x0e:                  // Se a resposta for WACK
        {
        ok=0;
        break;
        }
       case 0x15:                  // Se a resposta for NACK
        {
        gotoxy (10,15);
        cprintf("FALHA NA COMUNICACAO!");
        gotoxy (10,16);
        cprintf ("TENTE NOVAMENTE");
        ok=1;
        break;
        }
       }
     }
}
```

### 11.3.3 Envia.h

```
#include <stdio.h>
#include <conio.h>

void envia_prog(int COM0);

void envia_prog(int COM0)
{
int ok,i,bcs,bcs1,cont,param[7];
FILE *arqui;
unsigned char arquivo[20],buffer[121];

ok=0;
while (!ok)
    {
    Moldura();
    gotoxy(15,11);
    cprintf ("Opcao 1 - Transferir programa");
    gotoxy(15,13);
    cprintf ("Entre com o caminho e o nome do arquivo:");
    gotoxy(15,14);
    scanf ("%s",&arquivo);
    arqui=fopen (arquivo,"rt");
    cont=1;
    while ((!feof(arqui))&&(cont!=121))
      {
      buffer[cont]=fgetc(arqui);
      cont++;
      }
```

```
param[0]=0; // Estamos enviando um bloco
gotoxy(15,16);
cprintf ("Entre com o numero do programa:");
gotoxy(15,17);
scanf ("%s",&param[3]);

param[1]=param[3]>>8;
param[2]=param[3]&0x00ff;
comunica(COM0);
cprintf ("Parametros: %d   %d ",param[1],param[2]);
outportb(COM0,0);    // Envia o numero de bytes
cprintf ("\n\rPrimeiro NOB enviado");
bcs=0;
espera_envio(COM0);
outportb(COM0,cont);    // Envia o numero de bytes
cprintf("\n\rNOB enviado");
bcs=bcs^cont;
espera_envio(COM0);
outportb(COM0,120);    // Envia o endereco de destino
cprintf("\n\rEndereco de destino enviado");
bcs=bcs^120;
espera_envio(COM0);
outportb(COM0,0);    // Envia o endereco fonte
bcs=bcs^0;
espera_envio(COM0);
outportb(COM0,1);    // Envia o codigo da funcao
bcs=bcs^1;
espera_envio(COM0);
outportb(COM0,1);
bcs=bcs^1;
espera_envio(COM0);
```

```
outportb(COM0,0);     // Envia o sufixo da funcao
bcs=bcs^0;
espera_envio(COM0);
outportb(COM0,param[0]);     // Envia o sufixo da funcao
bcs=bcs^param[0];
espera_envio(COM0);
outportb(COM0,param[1]);     // Envia o numero do programa
bcs=bcs^param[1];
espera_envio(COM0);
outportb(COM0,param[2]);     // Envia o numero do programa
bcs=bcs^param[2];
espera_envio(COM0);
outportb(COM0,param[1]);     // Envia o numero do bloco
bcs=bcs^param[1];
espera_envio(COM0);
outportb(COM0,param[2]);     // Envia o numero do bloco
bcs=bcs^param[2];
espera_envio(COM0);
for (i=1;i<=cont;i++)
  {
  outportb(COM0,buffer[i]);     // Envia o numero do bloco
  bcs=bcs^buffer[i];
  espera_envio(COM0);
  }
outportb(COM0,0X10);
cprintf ("   Enviando DLE");
espera_envio(COM0);
outportb(COM0,0X03);
cprintf ("   Enviando ETX");
bcs=bcs^0x03;
espera_envio(COM0);
```

```
    outportb(COM0,bcs);
    espera(COM0);
    bcs1=inportb(COM0);
    cprintf ("\n\r   resposta ao bcs:%x",bcs1);
    if (bcs1==0x06)
      {
      outportb(COM0,0X10);
      espera_envio(COM0);
      outportb(COM0,0X04);
      espera_envio(COM0);
      ok=1;
      }
    }
scanf("%d",&bcs1);
}
```

### 11.3.4 Inicia.h

```
// Funcao que inicializa a porta de comunicacao escolhida com os parametros
// identicos aos do robo.

#include <stdio.h>
#include <dos.h>


void inicializa (unsigned int porta,long bps,int p,int stop,int bits_dados)

{
      union
   {  unsigned int nr;
```

```
    unsigned char hilo[2];
}
divisor;
unsigned char byte_de_param;
/*Calcula valor a ser enviado para registradores
  de taxa de transmissao */
divisor.nr=(unsigned int)(1843200L/(bps*16L));
/*Liga bit do registrador de controle das caract.
  que permite acesso aos registradores de taxa de
  transmissao*/
outportb(porta+3,0x80);
/*Byte menos significativo para porta */
outportb(porta,divisor.hilo[0]);
/*Byte mais significativo para porta */
outportb(porta+1,divisor.hilo[1]);
/*Formata byte de parâmetros       */
byte_de_param=bits_dados-5;
byte_de_param |=(stop-1)<<2;
if(p)
{   byte_de_param |=0x08;      //liga bit de paridade
          byte_de_param |=p<<3&0x10;//paridade par ou impar
}
/*Envia byte de parâmetros para registrador de controle
  de linha  */
outportb(porta+3,byte_de_param);
}
```

### 11.3.5 Moldura.h

```
#include <stdio.h>
#include <conio.h>

void Moldura();

void Moldura()
{
int i,j;
textbackground(2);
for (i=1;i<80;i++);
  for (j=1;j<23;j++);
    printf (" ");
textcolor(14);
clrscr();
gotoxy (1,1);
cprintf ("%c",201);
for (i=2;i<80;i++)
  cprintf ("%c",205);
cprintf ("%c",187);
gotoxy (1,24);
cprintf ("%c",200);
for (i=2;i<80;i++)
  cprintf ("%c",205);
cprintf ("%c",188);
for (i=2;i<24;i++)
  {
  gotoxy(1,i);
  cprintf ("%c",186);
```

```
  gotoxy(80,i);
  cprintf ("%c",186);
  }
gotoxy(1,3);
cprintf ("%c",204);
for (i=2;i<80;i++)
  cprintf ("%c",205);
cprintf ("%c",185);
textcolor (6);
gotoxy (23,2);
cprintf ("PROGRAMA DE COMUNICACAO ROBO<==>PC");
textcolor (15);
}
```

### 11.3.6 Receber.h

```
#include <stdio.h>
#include <conio.h>

void receber_da_serial(int COM0)
{
   FILE *nom;
   unsigned int bcs,nob,carac1[128],carac;
   int i,STX,DLE,ok=0,continuacao=0;
   char nomearq[11];

STX=0x02;
clrscr();
Moldura();
gotoxy (15,10);
```

```
cprintf("Recebendo dados do robo...");
nom=fopen("irb","wt");
carac=inportb(COM0);
if (carac==0x05)    // Se o robo enviar ENQ
   {
   outportb(COM0,0X06);  // Envia ACK
   espera(COM0);
   carac=inportb(COM0);
   if (carac==0x10)  // Verifica se o robo enviou DLE
     {
     espera(COM0);
     carac=inportb(COM0);
     while (!ok)
       {
       if (carac==STX)  // Se o robo nao enviar STX termina a transmissao
        {
        for (i=0;i<128;i++)
          {
          espera(COM0);
          carac1[i]=inportb(COM0);
          printf("\n\r%d",carac1[i]);
          if (carac1[i]==0x10)
            DLE=1;    // Indica que recebeu DLE
          else
            {
            if ((carac1[i]==0x03)&&(DLE==1))
              { // Se ja recebeu DLE e esta recebendo ETX
              carac=i;
              i=128;
              }
            else
```

```
        DLE=0;
      }
    }
  fclose (nom);
  espera(COM0);
  caracl[129]=inportb(COM0);
  cprintf("BCS rebebido: %d",caracl[129]);
// Calcula o BCS.
  DLE=0;
  for (i=0;i<=carac;i++)
    {
    if ((caracl[i]!=0x10)||(DLE==1))
      {
      bcs=bcs^caracl[i];
      DLE=0;
      }
    else
      DLE=1;
    }
  gotoxy (15,15);
  cprintf ("BCS calculado: %d",bcs);
// Se o BCS calculado for igual ao recebido, copia os dados no arquivo
  if (bcs==caracl[129])
    {
    DLE=0;
    if (continuacao)
      nom=fopen("robo","ab");
    else
      nom=fopen("robo","wb");
    for (i=8;i<=carac;i++)
      {
```

```
      if ((carac1[i]!=0x10)||(DLE==1))  //DLE nao conta no calculo.
        {
        fprintf(nom,"%c",carac1[i]);
        DLE=0;
        }
      else
        DLE=1;
      }
    fclose(nom);
    if (STX==0X02)
      STX=0X82;
    else
      STX=0x02;
    continuacao=1;
    outportb(COM0,0X06);
    }
  else
    outportb(COM0,0x15);
  }
 else
  {
  if (carac==0x10)
    {
    espera(COM0);
    carac=inportb(COM0);
    if (carac==0x04)
      ok=1; //Se recebeu EOT.
    }
  else
    ok=1;
  }
```

```
        }
      }
   gotoxy (15,16);
   cprintf ("FIM DA TRANSMISSAO");
   gotoxy (15,17);
   cprintf ("Pressione alguma tecla para continuar");
   while (!kbhit());
   ok=getch();
   }
}
```

### 11.3.7 Wait.h

```
#include <stdio.h>
#include <conio.h>
#include <dos.h>

// Funcao que espera que a informacao enviada pela serial seja lida
// pelo outro elemento (computador ou robo).

void espera_envio(int COM0);
void espera(int COM0);

void espera_envio(int COM0)
{
unsigned char status;
do
  {
  status=0;
  status=inportb(COM0+5);
```

```
  status&=96;
  }
while(status!=96);
outportb(COM0+5,0);
}

// Funcao que espera o recebimento de alguma informacao pela serial
// para prosseguir

void espera(int COM0)
{
unsigned char stat,ok;

ok=0;
while (!ok)
  {
  stat=inport(COM0+5);
  if (stat&1)
    ok=1;
  }
outportb (COM0+5,0);
}
```

# 11.4 Diagramas Ladder (PLC)

## 11.4.1 FC3 – Seleção do Modo Remoto ou Local

FC3 : Title:

Comment:

Network 1: Title:

Comment:


Network 2 : Title:

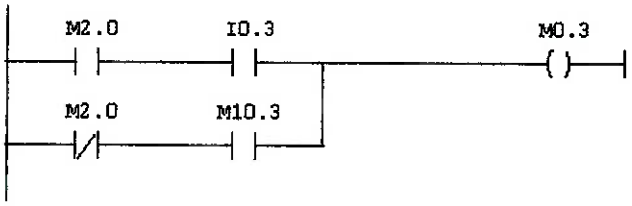
Comment:


Network 3 : Title:

Comment:

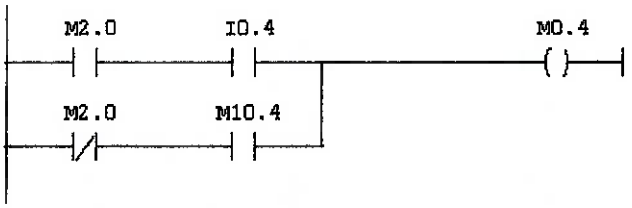

Network 4 : Title:

Comment:



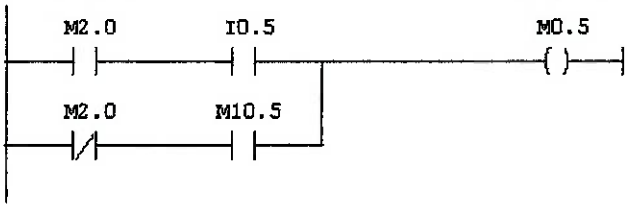
**Network 5 : Title:**

Comment:




**Network 6 : Title:**

Comment:



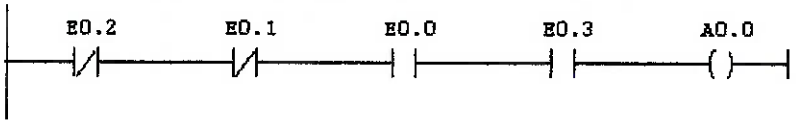
## 11.4.2 FC1 – Modo Remoto

FC1 : Modo Remoto

Botoes: Alimentador, Fresa Ok, Torno Ok sao simulados por botoes fisicos

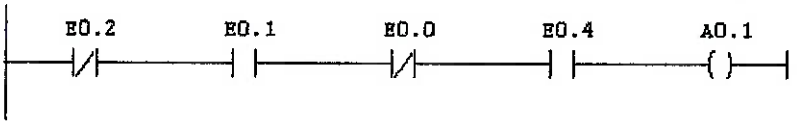
Network 1: Inicializou

Robo preparado, espera o sinal do alimentador de pecas



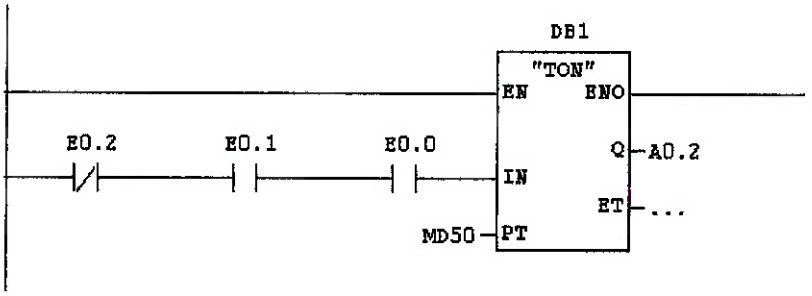

Network 2 : Pegou peca

Robo com a peca a ser fresada, espera o sinal da fresa



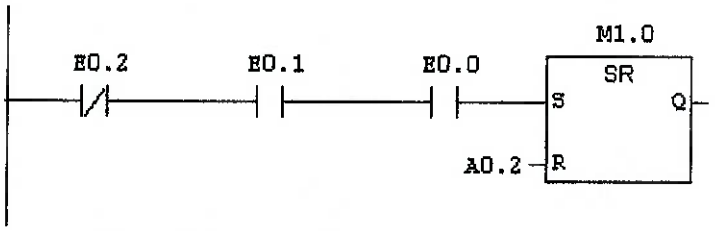

Network 3 : Colocou na fresa

NT3 e NT4) - Se robo mandar O11 (coloquei peca na fresa) devo estartar usinagem (fazer status = amarelo)e iniciar contagem de tempo.



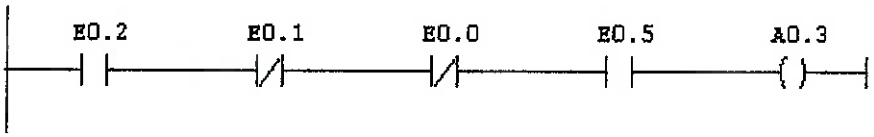
**Network 4 :** Title:

```
011 - seta m1.0 - inicia usinagem - acende amarelo
q0.2 - reseta m1.0  - fim da usinagem - acende branco
```



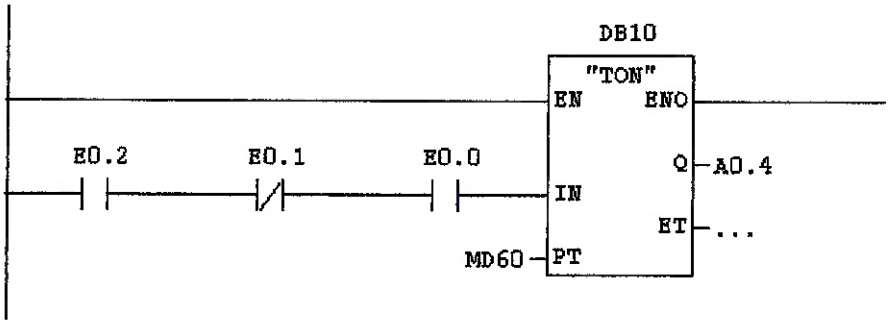

**Network 5 :** Retirou da fresa

```
Robo com a peca a ser torneada, espera o sinal do torno
```



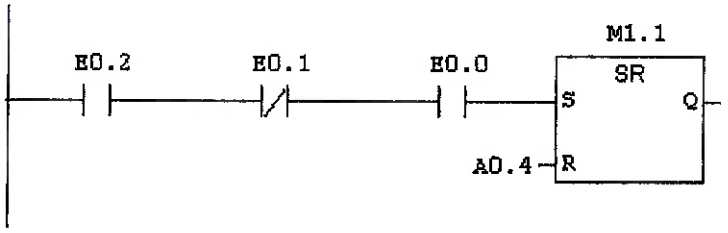

**Network 6 :** Colocou no torno

```
NT6 e NT7) - Se robo mandar 101 (coloquei peca no torno) devo estartar usinagem
(fazer status = amarelo)e iniciar contagem de tempo.
```




**Network 7 :** Title:

```
101 - seta m1.1 - inicia usinagem - acende amarelo
q0.4 - reseta m1.1 - fim da usinagem - acende branco
```



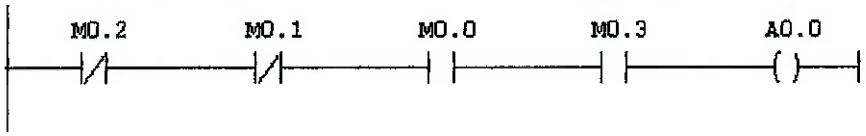
## 11.4.3 FC2 – Modo Local

FC2 : Modo Automatico

Botoes: Alimentador, Fresa Ok, Torno Ok sao simulados pelo WinCC

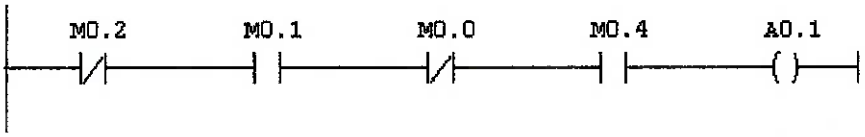
Network 1: Inicializou

Robo preparado, espera o sinal do alimentador de pecas



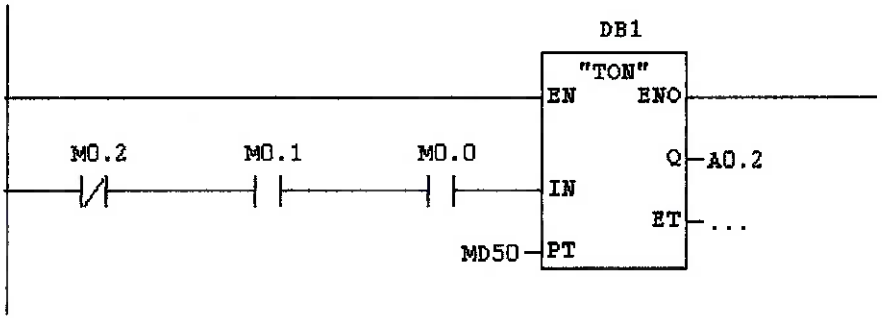

Network 2 : Pegou peca

Robo com a peca a ser fresada, espera o sinal da fresa



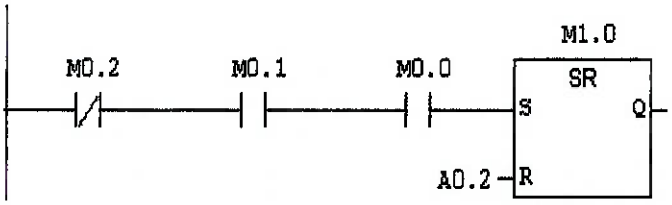

Network 3 : Colocou na fresa

NT3 e NT4) - Se robo mandar O11 (coloquei peca na fresa) devo estartar usinagem (fazer status = amarelo) e iniciar contagem de tempo.



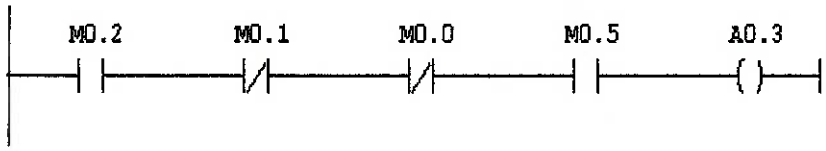
**Network 4** : Title:

011 - seta m1.0 - inicia usinagem - acende amarelo
q0.2 - reseta m1.0 - fim da usinagem - acende branco



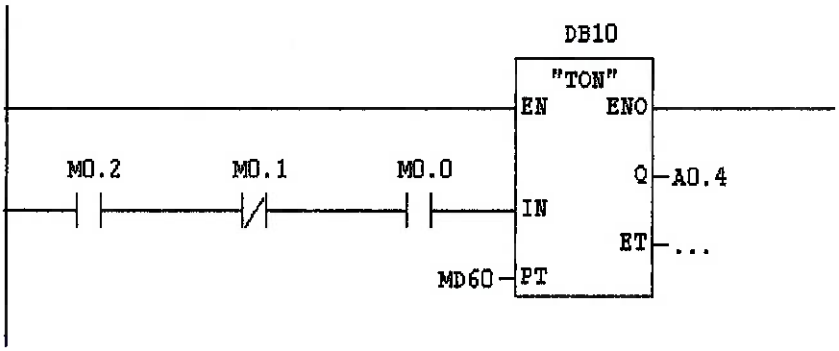

**Network 5** : Retirou da fresa

Robo com a peca a ser torneada, espera o sinal do torno



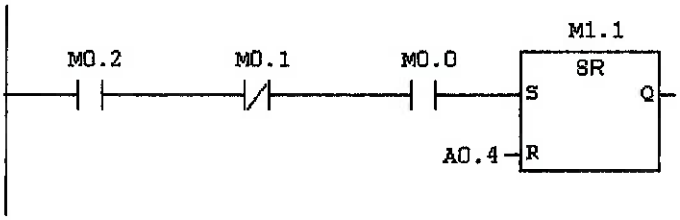

**Network 6** : Colocou no torno

NT6 e NT7) - Se robo mandar 101 (coloquei peca no torno) devo estartar usinagem (fazer status = amarelo)e iniciar contagem de tempo.

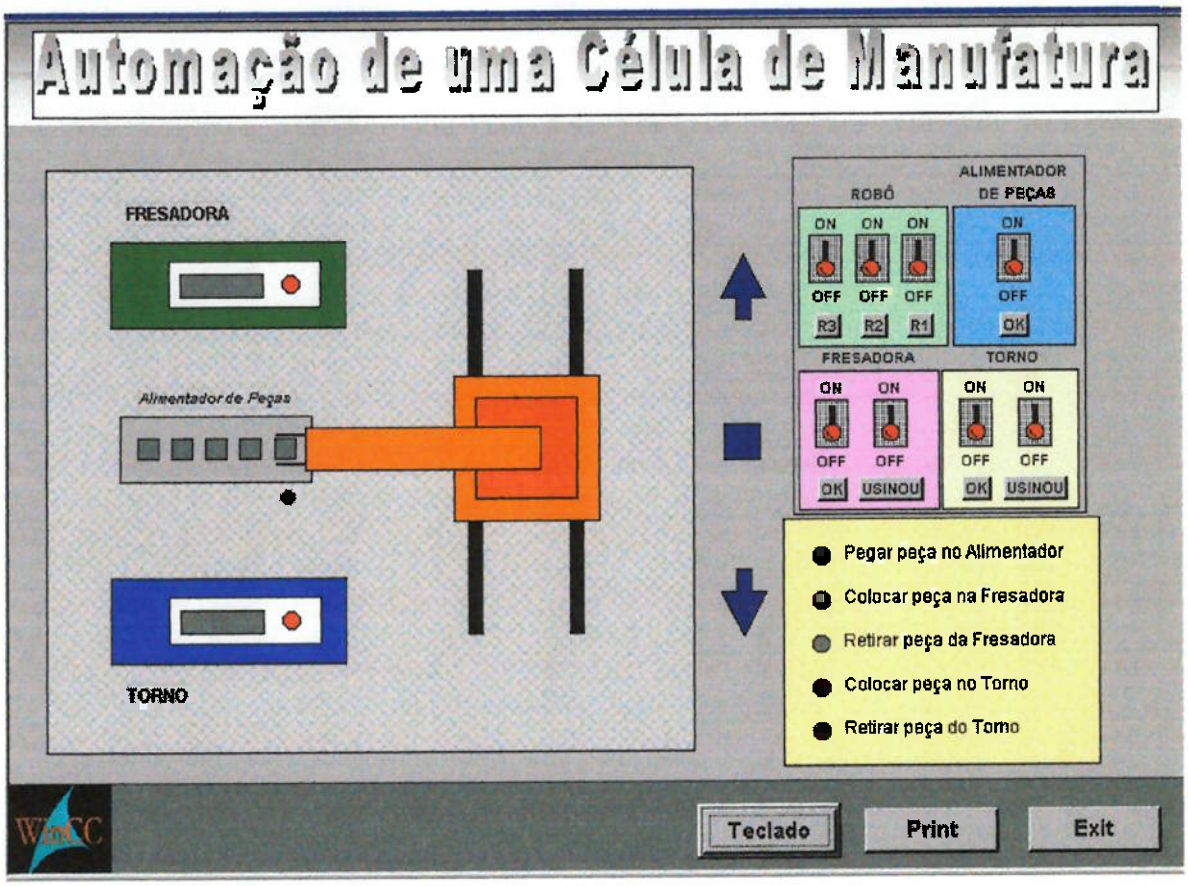



**Network 7** : Title:

101 - seta m1.1 - inicia usinagem - acende amarelo
q0.4 - reseta m1.1 - fim da usinagem - acende branco



## 11.5 Programa Supervisório




**Tela 9 - Tela da Simulação no WinCC**

# 12. REFERÊNCIAS BIBLIOGRÁFICAS

[1]. ASEA Industrial Robots – **Product Manual**

[2]. ASEA Industrial Robots – **Programming Manual**

[3]. SITE *Java Technology Home Page* em http://java.sun.com/

[4]. MANUAL de instalação Transaxe série 700 Servo-amplificadores 705 e 710. s.l., s.ed., 1992.

[5]. MANUAL placa Sensoray Model 421, s.l., s.ed., 1997.

[6]. GROOVER, M.P.; ZIMMERS, E.W., Jr. **CAD/CAM: computer-aided design and manufacturing** New Jersey, Prentice-Hall, 1984.

[7]. KUAE, L.K.N.; BONESIO, M.C.M.; VILLELA, M.C.O. **Manual para apresentação de dissertações e teses.** São Paulo, s.ed., 1991.

[8]. KOGA, J.; KAWAGUCHI, M.M. **Transferência de Dados entre o Robô e o PC.** São Paulo, s.ed., 1998.

[9]. SITE *Catálogo Online Siemens* em http://www.ad.siemens.de/

[10]. BRAGA, A C.; POLLI, B. T. **JNC – Java Numerical Control**. São Paulo, s.ed., 1998.

[11]. PREUSS, E.; CÉSAR J.. **Interligação Robô-PLC**. São Paulo, s.ed., 1998.