

Desenvolvimento de um Indexer para comando XY de dois motores passo-a-passo

Carlos Miguel Borges Pinto

Dissertação submetida para satisfação parcial dos requisitos exigidos para a
obtenção de grau de mestre no âmbito do Mestrado Integrado em
Engenharia Mecânica, opção de Automação

Orientador: Prof. Dr. António José Pessoa de Magalhães

31 de Agosto de 2011



Mestrado Integrado em Engenharia Mecânica

Opção de Automação

© Carlos Miguel Borges Pinto, 2011

Resumo

Este trabalho procura desenvolver uma nova solução para um problema que já foi tema de outros trabalhos de fim de curso no passado. Encontra-se directamente relacionado com um sistema existente na SAIC (Secção de Automação, Instrumentação e Controlo), designado por Unidade de Demonstração de Produção Flexível. Esta unidade corresponde a um produto didáctico da marca Festo®, e é uma das soluções qualificadas, orientadas para processos de automação, no domínio da formação industrial e educação profissional.

Na unidade em questão, pretende-se simular um processo de maquinagem, sendo para tal necessário controlar um elemento móvel, segundo três eixos (X, Y e Z), ao qual está associado uma ferramenta de corte que executará a operação de maquinagem desejada numa peça com a forma geométrica de um disco, sendo a mesma fixada numa superfície base, dentro de uma mesa de trabalho.

O eixo dos ZZ corresponde ao movimento ascendente e descendente da ferramenta de corte, e é controlado por um PLC (Siemens S7_300), que é igualmente responsável pelo controlo parcial da unidade.

Por sua vez os eixos XX e YY estão associados à movimentação e posicionamento no espaço do elemento móvel referido anteriormente, sendo o movimento em cada eixo assegurado por um motor passo-a-passo independente. É nesta tarefa de posicionar e controlar dois motores passo-a-passo segundo dois eixos, que se enquadra o trabalho desenvolvido.

São utilizados motores passo-a-passo visto que, estes possuem uma interessante forma de actuação, dispensando o controlo em anel fechado. Diversos sistemas de posicionamento em que a precisão exigida não é particularmente elevada usam motores passo-a-passo.

O objectivo do presente trabalho é precisamente construir um sistema de comando XY, para este tipo de actuadores, baseado num PLC Siemens S7 200, com uma interface série a partir da qual, o mesmo troca mensagens com um sistema de comando hierarquicamente superior.

O trabalho teve uma forte componente de desenvolvimento experimental, tendo sido antecedido por uma compreensão global dos requisitos em que o sistema em causa teve de ser inserido (unidade de demonstração de produção flexível existente). Seguiu-se a criação de várias plataformas de teste, onde o desenvolvimento foi ensaiado antes da especificação do protocolo de comunicação a empregar, e respectiva inserção no sistema final. No seguimento, elaborou-se a integração com o sistema de produção através do sistema de comando hierarquicamente superior (um PLC S7 300). Por fim, realizou-se uma avaliação do funcionamento global do sistema de produção.

Abstract

This project seeks to develop a new solution for a problem that has been the subject of other final-year projects in the past. It is directly related with a system existing at the Control, Instrumentation and Automation Section, named Flexible Production Demonstration Unit. This unit is a didactic product from Festo® brand and it is a qualified and oriented solution for automation processes in the area of industrial training and professional education.

This unit intends to simulate a machining process for which is necessary to control a moving element according to three axes (X, Y and Z), which it is associated to a cutting tool that will perform the desired finishing operation in a disk shaped part, being this cutting tool fixed on a base surface within a work table.

The ZZ axis corresponds to the upward and downward movement of the cutting tool, controlled by a PLC (Siemens S7_300), which is also responsible for the partial unit control.

The XX and YY axes are associated to the movement and positioning of the moving element mentioned previously, being the movement on each axis assured by an independent stepper motor. The developed project fits itself in this task of position and control of two stepper motors through two axes.

Since stepper motors are interesting actuators dispensing the closed loop control, we used two in this work. Several positioning systems where the required precision is not particularly high use stepper motors.

Thus, the objective of this project was to build an indexing system XY for these kinds of actuators based on a PLC Siemens S7 200, with a serial interface from which, it exchanges messages with a high level system.

The all work had a strong experimental component that was preceded for a global understanding of the requirements that the system in question had to be inserted (existence of the Flexible Production Demonstration Unit). After, there were created several test platforms, on which development was tested before the specification of the communication protocol to employ and their integration into the final system.

Following this, it was implemented the integration with the production system through the high level system (PLC S7 300) and finally, the evaluation of the global operation of the production system.

Agradecimentos

Embora se trate de um trabalho individual, gostaria de expressar o meu sincero agradecimento a todos aqueles que me ajudaram ao longo da elaboração deste.

Os meus sinceros agradecimentos,

Ao Professor António Pessoa de Magalhães, pelo seu empenho, dedicação e disponibilidade, assim como, pelas linhas orientadoras ensinamentos e ideias ao longo deste trabalho.

Ao engenheiro Jorge Matos Reis, gostaria de expressar a minha gratidão pela sua disponibilidade, pelo tempo despendido e também pelas suas ideias.

Ao Professor Francisco Freitas, pela orientação e acompanhamento ao longo de todo este processo.

Aos colegas da FEUP, nomeadamente aos colegas da opção de automação o apoio e a compreensão prestada.

À minha família, pelo apoio incondicional, paciência e compreensão sempre demonstrada.

Índice de Conteúdos

Resumo	iii
Abstract.....	v
Agradecimentos	vii
Abreviaturas e Símbolos.....	xix
Capítulo 1	1
Introdução	1
1.1 Contexto	1
1.2 Objectivos	3
1.3 Organização da tese	5
Capítulo 2	7
Descrição do problema	7
2.1 Sistema MPS FESTO.....	7
2.2 Motores passo-a-passo e seu comando	11
2.2.1 Apresentação de motores passo-a-passo	11
2.2.2 Tipos de Motores passo-a-passo e seu funcionamento	12
2.2.3 Áreas de aplicação.....	15
2.2.4 Os pontos fortes e os pontos fracos.....	16
2.2.5 Controladores	18
2.2.6 Indexers	19
2.3 Estado da arte.....	21
2.4 Proposta de solução	23
Capítulo 3	27
Arquitectura da solução	27
3.1 Componentes electromecânicos.....	27
3.1.1 Especificidades do indexer.....	31
3.2 Modelo de software	34
3.3 Modelo de comunicações MS e PP.....	39
3.3.1 Comunicação entre dispositivos	39
3.3.2 Princípios de comunicação no modo Ponto-a-Ponto	40
3.3.3 Princípios de comunicação no modo Mestre/Escravo	41

3.3.4	Modo Freeport para controlar a porta de comunicação serie.....	41
3.3.5	Utilização do cabo PC/PPI com outros dispositivos e protocolo Freeport	44
3.3.6	Descrição da comunicação a implementar	47
3.3.7	Descrição do protocolo a implementar	48
Capítulo 4		55
Desenvolvimento		55
4.1	Software desenvolvido em relação ao modelo	55
4.1.1	Compreender o ciclo de varrimento (scan cycle) da CPU	55
4.1.2	Programação em diagrama de escada (Ladder)	57
4.1.3	Variáveis e Instruções relevantes	59
4.1.4	Programação Desenvolvida.....	69
Capítulo 5		89
Testes e análise de resultados		89
5.1	Testes realizados.....	89
5.2	Resultados obtidos	95
5.3	Validação	106
Capítulo 6		107
Conclusões.....		107
6.1	Resumo da tese	107
6.2	Resultados obtidos	108
6.3	Propostas de trabalhos futuros	110
6.4	Balanço pessoal.....	111
Referências bibliográficas		113
ANEXO A:	Blocos de código e dados mais importantes.....	115

Índice de Figuras

Figura 1.1 - Unidade de demonstração de produção flexível.	2
Figura 1.2 - Diferentes células do sistema MPS Festo.	3
Figura 2.1 - Peças a trabalhar.	8
Figura 2.2 - Exemplo de uma operação de maquinagem.	9
Figura 2.3 - Unidade de demonstração de produção flexível (Sistema MPS).....	10
Figura 2.4 - Princípio de comando dos MPPs.	12
Figura 2.5 - Motor de relutância variável.	13
Figura 2.6 - Motor de íman permanente.	13
Figura 2.7 - Motor Híbrido.	14
Figura 2.8 - Motor Unipolar.	15
Figura 2.9 - Motor Bipolar.	15
Figura 2.10 - Produto da Siemens para posicionamento de MPP.	22
Figura 2.11 - Esquema global da solução projectada.	24
Figura 3.1 - Exemplo de botoneiras a implementar na solução.....	28
Figura 3.2 - Conexão das Drives ao indexer e motores.....	28
Figura 3.3 - Indexer (Siemens® S7-200).	29
Figura 3.4 - Célula de maquinagem.....	30
Figura 3.5 - Motor passo-a-passo.	30
Figura 3.6 - Sistemas de accionamento para MPP.	31
Figura 3.7 - Modelo geral do programa.....	35
Figura 3.8 - Esquema do modo de comunicação ponto-a-ponto.	36
Figura 3.9 - Esquema do modo de comunicação mestre/escravo.....	37
Figura 3.10 - Formato do buffer de transmissão de dados.	42
Figura 3.11 - Formato do buffer de recepção de dados.	43
Figura 3.12 - Comunicações com o PLC em modo PPI.	45
Figura 3.13 - Transmissão Série Assíncrona.	46
Figura 3.14 - Flowchart da comunicação entre controladores.....	48
Figura 4.1 - Ciclo de varrimento da CPU do s7-200	56

Figura 4.2 - Aplicações do software possíveis ao utilizador.	57
Figura 4.3- Programa em diagrama de contactos.	58
Figura 4.4 - Exemplo de rampa de aceleração desenvolvida para um MPP.	61
Figura 4.5 – Fórmula utilizada para determinar o tempo de ciclo delta de um segmento.	62
Figura 4.6 – Exemplo de trajectória rectilínea.	63
Figura 4.7 – Inicialização do programa.	70
Figura 4.8 – Ordem de geração do trem de impulsos para o motor do eixo x.....	71
Figura 4.9 – Procura do comando introduzido pelo utilizador.	72
Figura 4.10 – Sentido de rotação do motor X.....	72
Figura 4.11 – Energização Motor X.	72
Figura 4.12 – Forçar a paragem da geração de impulsos do motor X.....	73
Figura 4.13 – Deslocamento para a origem.	74
Figura 4.14 – Descrever uma trajectória linear.	75
Figura 4.15 – Rampa de aceleração do motor X.	78
Figura 4.16 – Coordenadas absolutas motor X.	79
Figura 4.17 – Actualização de parâmetros do motor X.....	80
Figura 4.18 – Inicialização de comunicação.	81
Figura 4.19 – Byte de Status dos motores.	82
Figura 4.20 – Processar a mensagem ‘S’.....	82
Figura 4.21 – Processar a mensagem ‘P’.....	83
Figura 4.22 – Inicializar os Parâmetros de Comunicação via Porta 0.....	84
Figura 4.23 – Inicialização dos parâmetros a receber pela porta 0.....	84
Figura 4.24 – Mensagem de Estado.	85
Figura 4.25 – Mensagem recebida terminada: Atingido número máximo de caracteres permitidos.....	86
Figura 4.26 – Configuração para receber o próximo comando.	86
Figura 4.27 – Verificar se o carácter final foi recebido.....	87
Figura 4.28 – Indicação de erro.	88
Figura 5.1 - Banca de trabalho da primeira fase de trabalho.	90
Figura 5.2 – Desenvolvimento de Comunicações PC/Indexer.	92
Figura 5.3 - Interface entre PLC e as drives dos motores passo-a-passo.	94
Figura 5.4 - Banca de trabalho onde se interagiu com um motor.....	95

Figura 5.5 - Interface para comandar MPP.....	96
Figura 5.6 - Mensagem enviada para o indexer para executar uma funcionalidade na terceira fase.....	96
Figura 5.7 – Mensagem de feedback enviada pelo indexer na terceira fase de teste.	97
Figura 5.8 - Mensagem enviada para o indexer para executar uma funcionalidade na quarta fase de teste.....	97
Figura 5.9 - Mensagem de feedback enviada pelo indexer na quarta fase de teste.	98
Figura 5.10 - Banca de trabalho onde se interagiu com um motor.....	98

Índice de Tabelas

Tabela 3.1 - Grupo de comando.	49
Tabela 3.2 - Mensagens de erro.....	50
Tabela 3.3 - Comandos 'W' (escrever comando) do indexer.....	51
Tabela 3.4 - Byte associado ao comando SX, do grupo de comando R (Read).....	53
Tabela 3.5 - Comandos 'R' (Ler Status de comando) do indexer.	53
Tabela 3.6 - Comandos para seleccionar o modo de comunicação do Indexer.....	54
Tabela 3.7 - Grupo de comando para verificar estado da comunicação entre controladores.	54
Tabela 4.1 - Perfil de uma operação PTO de múltiplos segmentos.....	60
Tabela 4.2 - Variáveis especiais.	66
Tabela 4.3 - Byte da memória especial SMB30 (Controlo de Registos Freeport)	66
Tabela 4.4 - Controlo de registos do PTO/PWM.	67
Tabela 4.5 - Byte da memória especial SMB86 até 94 (Controlo de mensagens recebidas).....	68
Tabela 5.1 - Primeiro protocolo de comunicação desenvolvido para comando de um motor passo-a-passo	93
Tabela 5.2 - Primeira tabela elaborada de mensagens de erro.....	93
Tabela 5.3 - Procedimento executado para simulação da operação de maquinagem I.	100
Tabela 5.4 - Procedimento executado para simulação da operação de maquinagem II.	102
Tabela 5.5 - Procedimento executado para simulação da operação de maquinagem III.	104

Abreviaturas e Símbolos

ASCII (American Standard Code for Information Interchange) - Sistema de codificação usado para representar letras e caracteres. O código ASCII de sete bits pode representar 128 combinações diferentes. O código ASCII de oito bits (ASCII extenso) pode representar 256 combinações diferentes.

Backlash (*folga*) - A reacção de recuo provocada pela interacção de elementos desgastados ou devido a peças mal ajustadas num mecanismo. Efeito normalmente negativo provocado pela acção que ocorre entre diferentes partes de um mecanismo.

Baud rate (Cadência de transmissão) - Velocidade de comunicações em serie. Número de bits transmitidos por segundo. Por exemplo, RS-232 é normalmente usado com um baud rate de 9600, ou cerca 9600 bits por segundo. É preciso cerca 10 bits em serie para enviar um carácter ASCII, então um baud rate de 9600 transmite cerca de 960 caracteres por segundo.

Binário - Sistema baseado em dois números. É um sistema em que os 1s e os 0s são usados para representar números.

Bit - Dígito binário. O menor elemento de dados binários. Um bit tanto pode ser um 0 com um 1.

Byte - Oito bits ou dois nibbles. Um nibble são 4 bits.

Closed loop (Anel fechado) - Um sistema em anel fechado tem um feedback e pode automaticamente corrigir erros.

Comando - Quando se pretende referir apenas uma instrução entre todas as desenvolvidas, a essa instrução está associada uma determinada funcionalidade que foi programa. Relacionado com um dos campos do protocolo de comunicação do indexer.

Comunicações assíncronas - Método de comunicações que usam um serie de bits para enviar dados entre dispositivos. Existe o bit de inicio, bits de dados (7 ou 8), bit de paridade, e ainda o bit final (1, 1.5, ou 2). É transmitido um carácter de cada vez. RS-232 é o protocolo mais comum.

Dados - São uma representação dos factos, conceitos ou instruções de uma maneira normalizada que se adapte à comunicação, interpretação e processamento pelo ser humano ou através de máquinas automáticas. Os dados são representados por símbolos como por exemplo as letras do alfabeto: a, b, c, etc., mas não são em si a informação desejada.

Debugging - É um processo metódico de encontrar e reduzir o número de erros ou defeitos, num determinado programa de computador ou numa parte de hardware

electrónico, tornando-se assim o seu comportamento dentro daquilo que seria o esperado.

Encoder (*transdutor de posição*) - São transdutores de movimento capazes de converter movimentos lineares ou angulares em informações eléctricas, que podem ser transformadas em informações binárias e convertidas em informações que possam ser interpretadas como outra grandeza (distância, velocidade, força, etc.). Unidade de realimentação que informa acerca das posições actuais e faz a comparação com as posições desejadas, de maneira a que os movimentos possam ser planeados.

Flowchart - Usado para tornar o design de um programa mais simples e perceptível de compreender.

Freeport - É um protocolo promovido pela Siemens® e um dos protocolos de comunicação disponíveis no PLC S7-200 (indexer). Permite definir parâmetros de comunicação, e parâmetros de recepção e envio de mensagens pela porta série.

Full duplex - Esquema de comunicações onde é possível enviar e receber dados em simultâneo. Existem trocas de dados em ambos os sentidos simultaneamente.

Grupo de comando - Quando se pretende especificar uma ordem que resulte numa operação. Relaciona todos os campos do protocolo de comunicação do indexer de modo a especificar a operação que se pretende ver executada.

Half duplex - Esquema de comunicações onde é possível enviar e receber dados, mas apenas numa direcção de cada vez. Existem trocas de dados em ambos os sentidos mas não simultaneamente.

Hexadecimal - É um sistema de numeração que representa os números em base 16.

I/O (Input/Output): (*entrada/saída*) - Usado para se falar sobre o número de entradas e saídas necessárias para um sistema, ou o número de entradas e saídas que um controlador lógico programável consegue lidar.

Indexer - Tecnologia (hardware e software) desenvolvida que permite controlar os accionamentos, de modo a garantir o correcto posicionamento de um elemento móvel segundo dois ou mais eixos, sincronizadamente.

Interface - Ligação de um PLC a dispositivos externos.

Informação - é todo o conjunto de dados devidamente ordenados e organizados de forma a terem significado.

LSB (*least significant bit*) - Bit menos significativo.

Mestre - Numa rede, é o dispositivo que controla o tráfego das comunicações. O mestre de uma rede usualmente conduz cada um dos escravos de modo a verificar se algum

deles tem algo a transmitir. Numa configuração mestre/escravo, apenas o mestre activo pode colocar uma mensagem na rede de comunicação.

MPP (*motor passo-a-passo*) - Tipo de motor eléctrico, muito particular, usado quando algo tem que ser posicionado com precisão ou segundo um ângulo.

MPS (*modular production system*) - Soluções didácticas Festo. Sistemas orientados para a formação e qualificação de pessoas na área de automação industrial. Através de equipamentos modulares permite criar e expandir à medida, uma ou mais estações de trabalho.

MSB (*Most significant bit*) - Bit mais significativo.

Paridade - É o bit usado para ajudar na verificação da integridade dos dados, durante uma comunicação de dados.

PLC (*controlador lógico programável*) - É um computador digital robusto, um sistema de tempo real. Ao contrário dos computadores de uso geral um PLC é concebido para múltiplas entradas e arranjos de saída, resistir a faixas mais altas de temperatura, imunidade a ruídos eléctricos e resistência à vibração e impacto. Utilizado para automação de processos electromecânicos, são usados em muitas indústrias e máquinas.

SAIC (*secção de automação, instrumentação e controlo*) - Secção do departamento de Mecânica e Gestão Industrial da Faculdade de Engenharia do Porto.

Saída digital - Uma saída com dois estados: ligado (on), e desligado (off). Saídas referentes à lógica binária.

Software - É uma sequência de instruções a serem executadas, na manipulação ou modificação de um dado/informação ou acontecimento.

Escravo - Numa rede com configuração mestre/escravo, existem apenas um mestre para vários escravos. Os escravos são nós da rede que transmitem informações apenas para o mestre quando este invocar. O resto do tempo um escravo não transmite nada.

UDPF (*unidade de demonstração de produção flexível*) - Unidade didáctica MPS, disponível nos laboratórios da Feup.

Unidade de processamento central (CPU) - Porção do Microprocessador de um PLC. É a porção de um PLC que lida com a lógica.

Word - Extensão de dados em bits que um microprocessador consegue manipular.

Capítulo 1

Introdução

Neste capítulo é feita a contextualização do assunto abordado nesta dissertação. São referidos os problemas e os objectivos do projecto. Conclui-se o capítulo com a apresentação da organização do presente documento.

1.1 Contexto

Este trabalho está relacionado com o equipamento MPS (modular production system), que foi adquirido pela FEUP nos anos 90 à Festo, designado por Unidade de Demonstração de Produção Flexível (figura 1.1), o qual se encontra actualmente instalado no SAIC. Este é um sistema didáctico, que permite simular o funcionamento de uma linha de produção.

A questão fulcral e a mais delicada de todo o sistema é o posicionamento de um elemento terminal móvel.

Esta parte do sistema já foi alvo de duas soluções bem diferentes, tendo consistido a primeira solução instalada, num controlador remoto da Festo formado por vários módulos, onde se interagiu com o sistema através de um protocolo de comunicação. Essa solução entretanto deixou de funcionar, tendo sido substituída por uma outra dedicada que se baseava em Microprocessadores. Esta segunda solução, foi o resultado de um trabalho de fim de curso no passado, reconhecido e inclusivamente

premiado, mas que, alguns anos mais tarde, viria também a deixar de funcionar devido a uma avaria severa, cuja resolução se mostrou difícil não se verificando consequentemente a reparação do sistema.

Por esse motivo, com o presente trabalho pretende-se desenvolver uma nova solução, que terá semelhanças com a primeira, será económica e desejavelmente funcional e eficaz.



Figura 1.1 - Unidade de demonstração de produção flexível.

O sistema MPS Festo[®] é constituído por quatro diferentes células e o controlo geral dessas mesmas células é assegurado por vários controladores lógicos programáveis (Siemens[®]). Em cada célula são executadas diferentes operações com o intuito de se simular a actividade de uma linha de produção. Na figura 1.2 encontra-se uma ilustração dessas células.

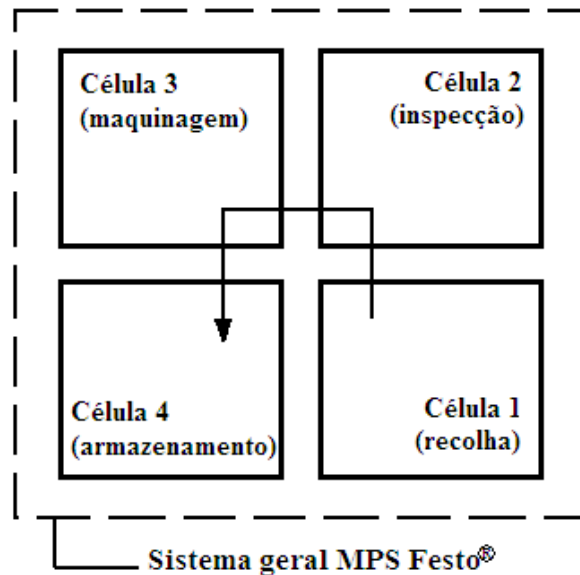


Figura 1.2 - Diferentes células do sistema MPS Festo®.

A célula 1 efectua a recolha de peças que posteriormente serão maquinadas, a célula 2 inspecciona o tipo de peça recolhida, já na célula 3 é onde se realiza o processo de maquinagem, e por fim a célula 4 fica encarregue do processo de armazenamento das peças maquinadas.

A tarefa do controlo geral da célula de maquinagem está encarregue a um PLC s7-300 (Siemens®). Esta célula conta também com um elemento móvel, onde se encontra instalada a ferramenta de trabalho (furadora), que se pode mover segundo XY dentro da área de trabalho. O posicionamento do elemento móvel é assegurado, grosso modo, por um conjunto de elementos, nomeadamente: fins de curso, indexer, motores passo-a-passo, as respectivas drives e fonte de alimentação.

Recorrer a motores passo-a-passo para efectuar posicionamento, é uma solução clássica e económica. Os motores passo-a-passo são ideais para aplicações onde a optimização de custos é mais importante do que a velocidade e a dinâmica.

1.2 Objectivos

Este trabalho tem por objectivo o desenvolvimento de um sistema de comando XY, para motores passo-a-passo, baseado num PLC s7 200 com uma interface série, a partir da qual, o mesmo trocará mensagens com um sistema hierarquicamente superior.

O sistema de comando XY deverá ser capaz de sincronizar o movimento de dois eixos, de modo a garantir que o elemento a posicionar efectue trajectórias lineares.

Pretende-se que o mesmo sistema possibilite ainda comandar diferentes acções de controlo sobre cada um dos motores passo-a-passo, tais como: definir o sentido de rotação, tempo entre dois impulsos e escala (ms ou μ S) a que cada passo é incrementado, tipo de passo e energizar ou não energizar a drive.

Requerem-se adicionalmente o desenvolvimento de diversas funcionalidades externas e internas, com o objectivo de assegurar o correcto funcionamento da unidade e facilitar o trabalho de posicionamento do sistema de comando XY.

As funcionalidades externas servirão em grande parte para fornecer indicações acerca de qual/quais a(s) acção/acções de controlo executada(s) sobre cada um dos motores, possibilitando a tomada de medidas de segurança. Este tipo de funcionalidades consistirão em sinalizações luminosas (disponíveis no PLC s7-200), onde se desenvolverá o sistema de comando XY e instalação de botoneiras na unidade.

As funcionalidades internas corresponderão a todas as acções de controlo disponíveis pelo indexer, a ser aplicadas nos motores passo-a-passo através de comandos. Estas funcionalidades internas serão executadas de um modo sequencial através de comandos que consistirão em mensagens definidas num protocolo de comunicação, estas irão consistir em linhas de código de programação a desenvolver ao qual corresponderá uma extensa parte de todo trabalho a desenvolver neste trabalho.

O sistema de comando que permite desempenhar as acções referidas anteriormente, designa-se por Indexer. Este controlará o posicionamento do elemento móvel segundo os eixos X e Y, apenas com movimentos lineares, dentro da área de trabalho.

A cada um dos eixos estará associado um motor passo-a-passo.

Para que seja possível efectuar uma determinada operação de maquinagem, é necessário que o indexer consiga comunicar e sincronizar-se com o controlador (s7-300) geral da célula de maquinagem, que actuará no eixo da ferramenta de corte, fixada no elemento móvel (eixo Z).

Do trabalho deverá resultar uma solução que possa ser usada em demonstrações, que permita ser adaptada e instalada em outro sistema idêntico, e ainda, que permita dar continuidade à pesquisa e desenvolvimento do assunto abordado nesta dissertação, inserida numa área particular do Controlo e Automação.

1.3 Organização da tese

O presente documento é composto por seis capítulos. Tendo em conta os objectivos enunciados no ponto anterior, pretende-se detalhar com rigor o trabalho desenvolvido e implementado, permitindo e facilitando uma futura utilização por terceiros.

Assim, no Capítulo 1 é apresentada a introdução ao projecto, onde se declara a motivação e o contexto do presente trabalho. São ainda indicados os objectivos propostos para o projecto.

No Capítulo 2 é realizada uma descrição mais detalhada do problema. São anunciados os aspectos e características gerais do sistema MPS, onde se pretende implementar a solução final obtida. É referida a função e disponibilidade dos principais elementos, que irão formar a solução pretendida. No final, é apresentada a proposta de solução a desenvolver.

No Capítulo 3 aborda-se a arquitectura seguida para a solução pretendida. É descrito o modelo traçado ao nível de hardware, software, bem como o modelo definido para as comunicações, que puderam ser efectuadas no modo mestre/escravo ou ponto-a-ponto.

No Capítulo 4 são apresentadas algumas especificidades relacionadas com desenvolvimento do projecto. Tais especificidades permitirão perceber como foi possível ao nível do software/programação, configurar e implementar alguns aspectos abordados no capítulo anterior (modelo da solução).

No Capítulo 5 são indicados os testes realizados e os resultados alcançados ao longo do projecto. É feita uma validação do sistema obtido de modo a se perceber, se o

sistema satisfaz os requisitos relativos à comunicação entre controladores e ao comando didacticamente interessante da mesa XY.

Posteriormente, no capítulo 6 apresentam-se as principais conclusões do projecto. Apresenta-se o resumo da tese em jeito de conclusão, comentando-se os resultados obtidos tendo em conta as perspectivas iniciais e apresentando algumas propostas para trabalhos futuros. Termina ainda, com um balanço pessoal.

Por fim, são mencionadas as referências da bibliografia consultada para realização da presente dissertação. Os blocos de código e os dados mais relevantes são apresentados em anexo.

Capítulo 2

Descrição do problema

Este capítulo aborda os aspectos gerais do sistema MPS da Festo, elaborando-se uma descrição alargada de vários aspectos acerca dos motores passo-a-passo utilizados neste trabalho, tais como: características, função, disponibilidade, relevância, enquadramento, entre outros.

É referida ainda, a função dos controladores, bem como dos indexers actualmente disponíveis no mercado.

No final deste capítulo apresenta-se também uma delineação da proposta de solução pensada e desenvolvida.

2.1 Sistema MPS FESTO

2.1.1 Aspectos gerais e simulação da maquinagem

A linha de produção de uma fábrica pode ser constituída por células de produção individuais. Cada célula tem uma função específica no processo (distribuição, inspecção, processamento, manipulação, montagem, armazenamento, entre outras).

O sistema MPS da Festo[®], é um sistema que permite seleccionar e combinar de uma maneira eficaz, um conjunto de estações individuais, de modo a formar um sistema de produção particular que atenda as exigências pré-estabelecidas. Esta tecnologia é

orientada para o treino, consultoria e formação de todo o espectro da automação, em fábricas, escolas e universidades.

A Festo Didactic[®] é uma divisão do grupo Festo[®] que se dedica, entre outras coisas, ao desenvolvimento deste género de soluções [7].

O sistema MPS existente na Feup (figura 2.1), no laboratório da secção de automação, instrumentação e controlo, é uma solução que permite simular uma linha de produção efectuando-se a recolha, inspecção, maquinagem e o armazenamento das peças ilustradas na figura 2.2.



Figura 2.2 - Peças a trabalhar.

É especificamente na célula de maquinagem deste sistema MPS, onde se efectua a operação de maquinagem na peça, que se enquadra este trabalho. Nesta célula de maquinagem, pretende-se realizar operações de maquinagem nas peças ilustradas acima, quando estas se encontrem na mesa de trabalho rotativa e na respectiva posição de trabalho. A operação de maquinagem consiste em executar um perfil de furação pré-definido através de uma ferramenta de corte, a qual se situa no elemento móvel.

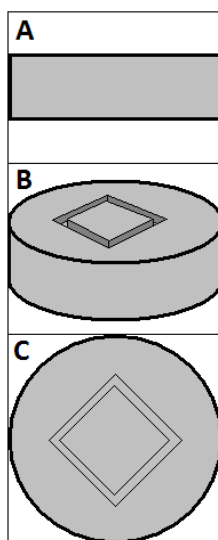
O elemento móvel pode-se deslocar segundo dois eixos perpendiculares X, Y. O perfil de maquinagem é consumado pela actuação e comando de dois motores passo-a-passo instalados na estrutura do sistema, em que cada um deles coloca o elemento móvel a trasladar segundo o respectivo eixo. Quanto ao eixo Z, correspondente ao movimento ascendente e descendente da ferramenta de corte, este é assegurado por actuadores pneumáticos.

É importante referir que os movimentos X, Y e o movimento Z, de posicionamento do elemento móvel, são assegurados por dois controladores diferentes.

A célula de maquinagem do sistema MPS é controlada por um PLC Siemens® s7-300 que controla também, o actuador pneumático que faz movimentar segundo o eixo Z o elemento móvel.

Quanto ao controlador que vai permitir executar sincronizadamente os movimentos segundo XY, irá desenvolver-se uma solução para aplicações mais específicas, que envolvam requisitos e funcionalidades diferente. Este tipo de controlador designa-se por indexer.

O Indexer apenas irá controlar o posicionamento do elemento móvel segundo os eixos XY, ou seja, moverá o elemento móvel dentro da área de trabalho. Sendo no entanto necessário o indexer conseguir comunicar e sincronizar-se com o controlador da unidade (s7-300) que deslocará no sentido ascendente ou descendente, segundo o eixo Z, a ferramenta de corte fixada no elemento móvel, de forma a realizar o tipo de maquinagem superficial desejada para a peça.



O indexer irá comandar dois motores passo-a-passo, que permitiram ao elemento móvel desempenhe trajectórias lineares (como por exemplo: figura 2.3), as quais sem o Indexer não seriam possíveis de obter. Este é composto por tecnologia Siemens, cujo hardware é constituído por um Micro PLC s7-200, tendo sido utilizado o software V4.0 STEP 7 MicroWIN SP5 para o desenvolvimento da solução.

Legenda:

A - vista de frente

B - vista 3D

C - vista de cima (superfície a ser trabalhada)

Figura 2.3 - Exemplo de uma operação de maquinagem.

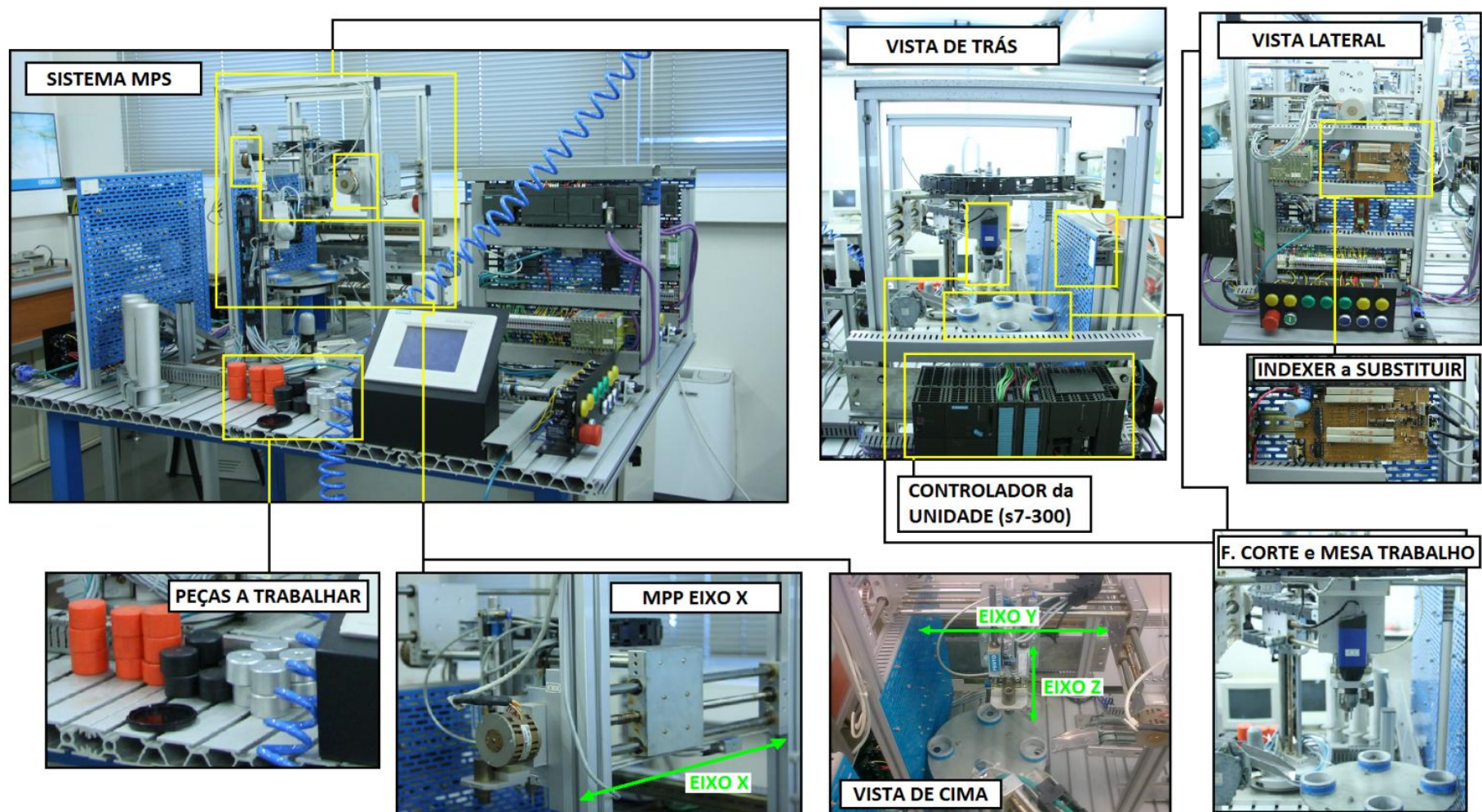


Figura 2.1 - Unidade de demonstração de produção flexível (Sistema MPS).

2.2 Motores passo-a-passo e seu comando

2.2.1 Apresentação de motores passo-a-passo

Os motores passo-a-passo são actuadores electromecânicos que convertem impulsos eléctricos em deslocamentos mecânicos. Um motor DC padrão, por exemplo, gira continuamente mas um motor passo-a-passo gira em incrementos fixos sempre que for dado um impulso eléctrico à drive. Um motor DC normal seria considerado um dispositivo analógico, enquanto um motor passo-a-passo seria considerado digital. [14]

A cada impulso de comando corresponde um deslocamento incremental bem determinado que toma o nome de passo. O tamanho do passo, ou o ângulo do passo (ângulo do passo é estabelecido em fracções de 360°), é determinado pela construção do motor ou pelo tipo de controlador a que vai estar ligado o motor. Por exemplo, uma resolução de 90° de um motor passo-a-passo corresponderia a quatro passos por rotação (revolução). Uma resolução de 15° corresponderia a 12 passos por rotação, ou ainda $1,8^\circ$ indicariam 200 passos por revolução. Motores Microstep são capazes de milhares de passos por revolução.

A rotação e o sentido a que um motor passo-a-passo funciona, estão directamente relacionados com os impulsos eléctricos que são recebidos, isto é, com a sequência a que os mesmos impulsos são aplicados. A velocidade de funcionamento do motor é dada pela frequência dos impulsos recebidos e o espaço percorrido está directamente relacionado com o número de impulsos aplicados.

Devido à sua exactidão de rotação, motores passo-a-passo são usados em sistemas de controlo em anel aberto (openloop) onde o posicionamento exigido é elevado. Existem também aplicações onde a margem de erro é mínima, situações que para além de ser necessário um posicionamento muitíssimo preciso, os parâmetros de funcionamento a que os motores serão sujeitos poderão originar erros, nessas situações particulares são usados encoders ou um dispositivo de realimentação da posição para confirmar a posição actual do eixo do motor.

Os motores passo-a-passo exigem uma drive, essa drive é composta por uma unidade de controlo electrónico e por um dispositivo de alimentação. O Indexer é o controlador que fornece os impulsos dos passos e direcção à(s) drive(s) que controlam a

quantidade de corrente que chega a cada fase. A figura 2.4 indica o princípio de comando dos motores passo-a-passo.

O motor passo-a-passo é essencialmente um motor sem escovas (brushless) que pode fornecer um binário elevado a velocidade nula, sem deslizar da posição em que se encontra. A direcção do motor pode ser invertida, invertendo a direcção dos impulsos a partir do controlador.

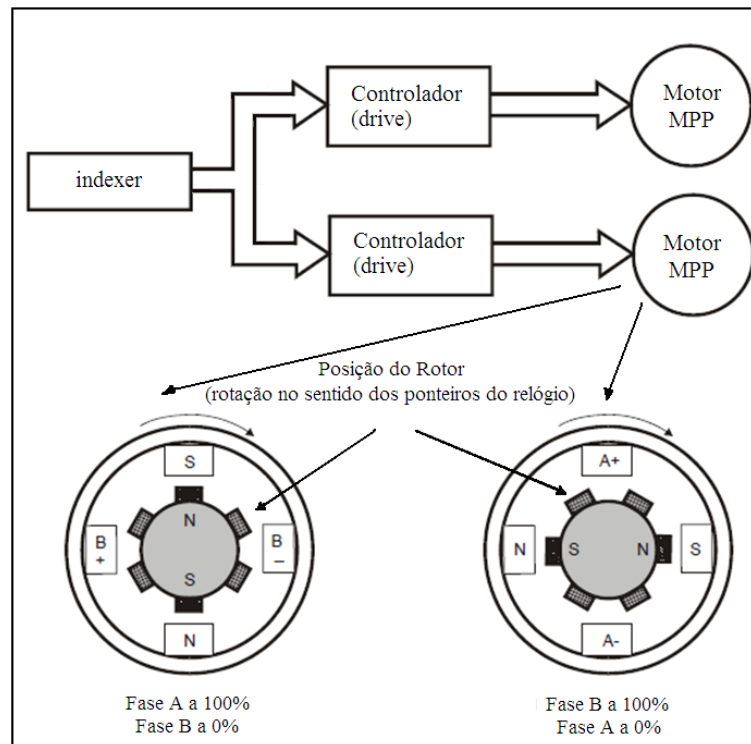


Figura 2.4 - Princípio de comando dos MPPs. [14]

2.2.2 Tipos de Motores passo-a-passo e seu funcionamento

Os tipos mais comuns de motores passo-a-passo são, provavelmente, o de íman permanente e o de relutância variável. No diagrama da figura 2.4 trata-se de um motor do tipo íman permanente. [3, 22]

Quanto à estrutura de um motor de relutância variável (figura 2.5), esta consiste num rotor de ferro, com múltiplos dentes e um estator com enrolamentos. Os enrolamentos do estator são energizados com corrente DC onde os pólos ficam

magnetizados. A rotação ocorre quando os dentes do estator são atraídos para os pólos do estator energizado.

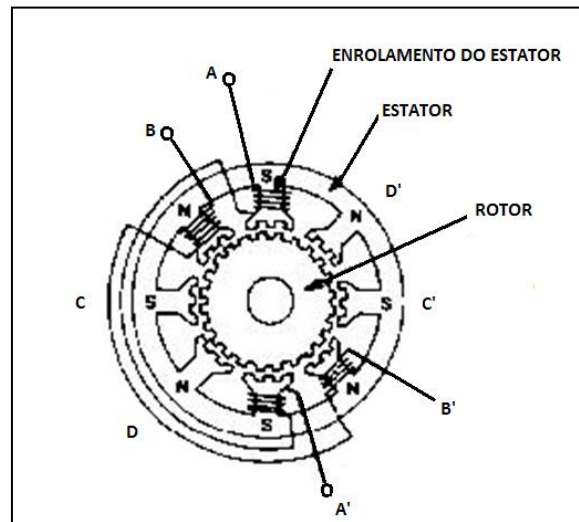


Figura 2.5 - Motor de relutância variável. [3]

Os motores de íman permanente (figura 2.6) têm baixo custo e baixa resolução, com passos típicos de 7.5° a 15° (48 - 24 passos/revolução). O rotor é construído por ímanes permanentes e não possui dentes. Os pólos magnetizados do rotor provêm uma maior intensidade de fluxo magnético, e por esse motivo, o motor de íman permanente exibe uma melhor característica de binário quando comparado ao motor de relutância variável.

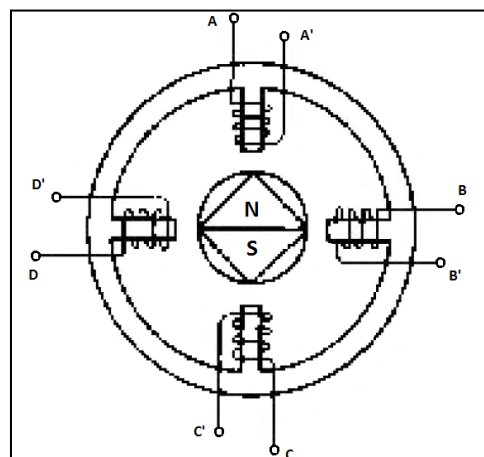


Figura 2.6 - Motor de íman permanente. [3]

O motor passo-a-passo híbrido (figura 2.7) é mais caro do que o de íman permanente, mas permite obter melhores desempenhos no que diz respeito à resolução

de passo, binário e velocidade. Os ângulos de passo típico de motores híbridos estão entre 3.6° a 0.9° (100-400 passos por volta). Este motor combina as melhores características dos motores de íman permanente e motores de relutância variável. O rotor é multi-dentado como no motor de relutância variável, contendo um íman permanente em redor do eixo. Os dentes do rotor fornecem um trilho melhor, o que ajuda a guiar o fluxo magnético.

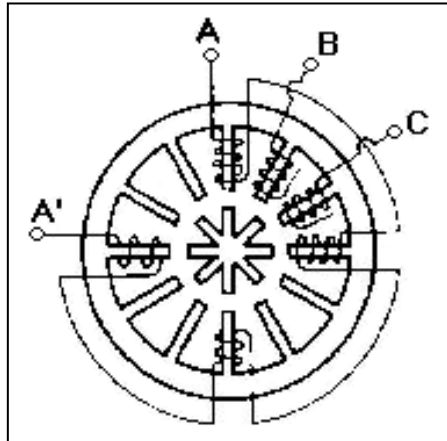


Figura 2.7 - Motor Híbrido. [3]

Para além dos tipos de motores supra mencionados, os quais cobrem a maior parte das aplicações de motores passo-a-passo, existem duas outras estruturas que merecem uma referência: o motor passo-a-passo linear e o motor passo-a-passo electrohidráulico.

Quanto à forma de funcionamento dos motores passo-a-passo, esta pode ser de dois modos: motores unipolares (figura 2.8) que têm dois enrolamentos por fase, um para cada sentido da corrente; e os motores bipolares (figura 2.9) que têm um único enrolamento por fase.

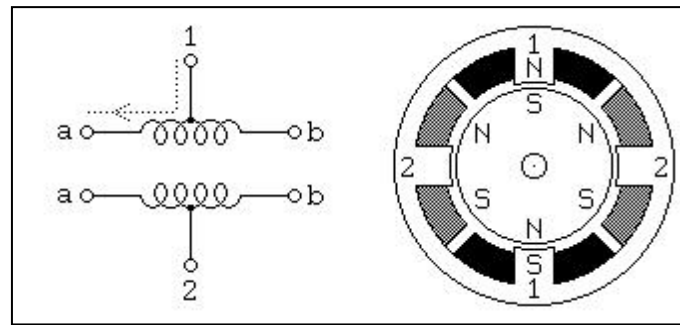


Figura 2.8 - Motor Unipolar. [3]

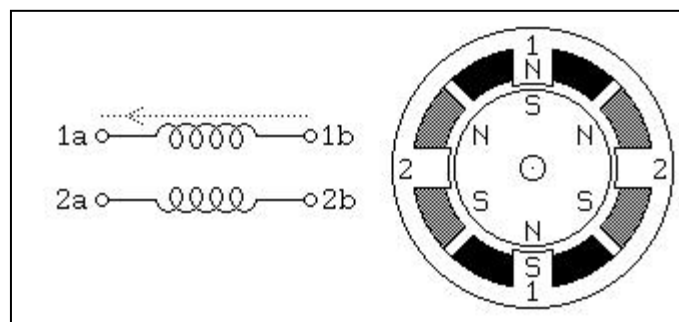


Figura 2.9 - Motor Bipolar. [3]

2.2.3 Áreas de aplicação

Existem vários aspectos a ter em considerações no que diz respeito a utilização de um ou mais motores passo-a-passo numa determinada área de aplicação. Tendo em conta as principais limitações e as virtudes deste tipo de actuadores pode-se dizer que um motor passo-a-passo pode ser uma boa escolha sempre que for necessário efectuar movimentos precisos numa determinada aplicação. [3, 14, 22, 23]

Motores passo-a-passo podem ser usados em aplicações onde seja necessário controlar vários factores tais como: ângulo de rotação, velocidade, posição e sincronismo. O ponto forte de um motor passo-a-passo não é o seu binário, tão pouco a sua capacidade de desenvolver altas velocidades ao contrário da maioria dos outros motores eléctricos, mas sim, a possibilidade de controlar os seus movimentos de forma precisa. Devido a esse factor, este é amplamente usado em impressoras, traçadores de gráficos, equipamento médico, scanners, robôs, câmaras de vídeo, equipamento

fotográfico, máquinas de controlo numérico, brinquedos, automação industrial, entre outros dispositivos electrónicos que requerem precisão posicional.

2.2.4 Os pontos fortes e os pontos fracos

Os motores passo-a-passo em relação aos outros tipos de motores disponíveis são mais vantajosos nos seguintes pontos:

- *Seguem uma lógica digital:*

Diz-se que o motor passo-a-passo segue uma lógica digital, pois o seu accionamento é feito através de impulsos eléctricos que activam sequencialmente as bobinas, fazendo o rotor alinhar-se com as mesmas, provocando o seu deslocamento.

- *Precisão de posicionamento:*

O posicionamento de um motor passo-a-passo é preciso, uma vez que, o rotor se movimenta sempre segundo ângulos bem determinados (passos), cujo erro de posicionamento é pequeno e não acumulativo (em geral 5%).

- *Precisão do binário aplicado:*

As variações de binário aplicado de um motor passo-a-passo são pequenas, tendo em consideração o seu funcionamento.

- *Excelente resposta de aceleração e desaceleração:*

O movimento que um motor passo-a-passo produz, é o resultado da activação sequencial das suas bobinas. A resposta para tais solicitações de aceleração e desaceleração é rápida, pois o rotor alinha-se rapidamente com a(s) bobina(s) que se encontra(m) energizada(s).

Em relação com outros tipos de motores podemos destacar os seguintes aspectos como desvantagens no uso de motores passo-a-passo:

- *Baixo desempenho em altas velocidades:*

O aumento de rotações num motor passo-a-passo (aceleração) é gerado pela variação no tempo entre o accionamento de uma bobina e a seguinte. Entretanto é necessário um rápido bloqueamento de um solenóide energizado para o próximo solenóide, de forma a permitir que a velocidade seja mantida, o que, muitas vezes é complexo e pouco eficiente.

- *Requer um certo grau de complexidade para ser operado:*

Pelo facto de usar uma lógica digital, não basta apenas ligar um motor passo-a-passo a uma fonte de energia para colocar o mesmo a rodar sem parar. Dada a sua complexidade, são necessários, outros elementos para controlar um motor passo-a-passo.

O “custo computacional” e a complexidade do dispositivo de controlo aumentam à medida que o número de passos também aumenta, visto que, mais passos requerem um maior número de terminais (fios) a serem activados e controlados.

- *Ocorrência de ressonância devido a um controlo inadequado:*

Tal como todos os objectos que existem, um MPP também têm uma frequência de ressonância. Caso as revoluções do mesmo se dêem nesta frequência, este pode começar a oscilar, aquecer e perder passos. Este problema pode ser contornado mudando-se o modo de operação do motor, utilizando-se meio-passo (half-step) ou o passo-completo (full-step) com as bobinas energizadas duas a duas. [3]

2.2.5 Controladores

Sistemas computadorizados supervisionam automaticamente e acrescentam ao processo muitos aspectos positivos, nomeadamente, ao nível da flexibilidade, segurança e cadência de produção, em comparação com um sistema de produção semelhante não computadorizado.

Em processos industriais, o uso de PLCs (Controladores Lógicos Programáveis) é típico. PLCs são controladores que trabalham efectivamente no lugar de um ou mais operadores que controlam manualmente uma estação. Na unidade de demonstração de produção flexível associada a este trabalho, a utilização de PLCs é vital, sendo todo o sistema controlado por vários PLCs, mas também o próprio indexer que será desenvolvido sobre um PLC.

O Controlador é o dispositivo electrónico que fornece um sinal em tensão ou corrente para a drive. Este, por sua vez, envia um certo nível de energia ao respectivo motor. Um controlador processa os sinais que são introduzidos através das entradas disponíveis (porta de comunicações ou I/O auxiliar), e executa um programa que permite de uma maneira autónoma executar as acções pretendidas pelo utilizador.

A configuração dos controladores para o comando de motores passo-a-passo é efectuada normalmente através de um módulo de comunicação segundo um protocolo de comunicação, onde o movimento pretendido pode ser programado (programação de alto nível). Neste género de controladores o modo de operação pode divergir um pouco, sendo inevitavelmente constituído pelo menos por três etapas. É indispensável haver um modo de configuração onde se definam especificamente quais as operações que se pretendem executar, um modo de comando dos motores onde se execute tudo aquilo que previamente ficou configurado, e ainda um modo de diagnóstico que permita verificar erros de programação e cablagem. [10, 14, 20]

São ainda providenciadas entradas digitais principalmente por motivos de segurança e de flexibilidade de comando, que permitem interagir rapidamente com o controlador, como por exemplo: activar ou desactivar o trem de impulsos, energizar ou não o(s) motor(es), definir o sentido de rotação. E no caso de, o controlador estar associado a uma aplicação delicada onde não existe grande margem para erro, onde é imprescindível a obtenção de movimentos com posições iniciais e finais definidas com

precisão, são normalmente disponibilizadas entradas diferenciais para activar ou não encoders.

A drive de um motor passo-a-passo em muitas bibliografias é referida como o controlador do motor. Essa denominação deve-se à função que as drives exercem sobre os motores passo-a-passo. Cada modelo de drive tem especificações técnicas particulares, que devem-se enquadrar com o tipo de motor passo-a-passo a controlar, bem como a tarefa a executar. Existem drives mais adequadas a providenciar controlo de precisão (micropassos), binário, ou velocidade.

As drives serão o dispositivo intermédio entre indexer e respectivo motor passo-a-passo. Cada drive tem disponíveis quatro entradas de sinais para cada motor, essas entradas permitirão; energizar ou não o respectivo motor, definir o tipo de passo do motor como full-step (passo completo) ou half-step (meio passo), definir o sentido de rotação como CW (sentido dos ponteiros do relógio) ou CCW (sentido contrário dos ponteiros do relógio), e ainda uma entrada que gerará a corrente para cada uma das fases do motor de acordo com o trem de impulsos proveniente do indexer (PLC:s7-200). As características mais importantes para controlar um motor passo-a-passo são a tensão de alimentação e a corrente eléctrica que as suas bobinas suportam.

2.2.6 Indexers

O indexer é um controlador que, tal como os controladores abordados anteriormente, fornece sinais de referência para a drive. A maioria das aplicações de posicionamento e controlo de motores passo-a-passo, requerem que o indexer governe outras funções de controlo como aceleração, desaceleração, número de passos por segundo e distância a percorrer, para dois ou mais eixos de movimento de um modo sincronizado. O indexer pode também realizar interface com outros controladores (dispositivos) e controlar muitos outros sinais externos. [2, 14, 20]

Geralmente existe um controlador/computador hierarquicamente superior ao indexer. A comunicação entre os dois dispositivos é efectuada por norma através de uma porta serie RS-232 e em alguns casos, através de uma porta RS-485. Em ambos os

casos, o indexer é capaz de receber comandos de alto nível (comandos ASCII) a partir do dito controlador/computador, e gerar os impulsos necessários para a drive.

Um indexer normalmente contém I/O auxiliar para monitorizar as entradas das fontes externas, tais como o interruptor Go (andar), Stop (parar), Home (origem) ou Limit (limite). Pode também iniciar funções de outra máquina através dos pinos de saída I/O.

Uma vez efectuado o *download* para a memória não-volátil e inicializado o programa de movimento, é possível controlar o movimento a partir de diferentes interfaces de operação, tais como um teclado, uma consola ou a partir de interruptores através de entradas auxiliares de I/O.

Um sistema de controlo de um motor passo-a-passo é inevitavelmente constituído por uma fonte de alimentação e por uma drive.

Para aplicações que requerem um controlo multi-eixo o indexer pode ter mais de um motor passo-a-passo para controlar. Nesse caso, um indexer tem disponível um sistema de controlo multi-eixo síncrono. Com esse sistema é possível ter até quatro drives de motores passo-a-passo conectadas (em alguns casos mais), onde cada drive fica separada e ligada ao respectivo motor passo-a-passo. Este sistema possibilita movimentos coordenados para aplicações que requerem um elevado grau de sincronização, como a interpolação circular ou linear.

2.3 Estado da arte

Para o estado da arte deste trabalho foi efectuada uma pesquisa acerca de indexers, de modo a se para perceber qualitativamente e quantitativamente quais o tipo de soluções que actualmente estão disponíveis no mercado. Procurou-se encontrar indexeres que resolvessem o problema do posicionamento/controlo de motores passo-a-passo bem como o problema das comunicações entre controladores, pois sem isso não é possível sincronizar os movimentos dos diferentes eixos.

Durante a pesquisa foi possível constatar que os resultados que se obtêm ao pesquisar pela palavra ou por algo relacionado com um indexer, não são muito reveladores. Esse facto por si só, indica que, não existem abundantemente soluções no mercado, que permitam responder positivamente e de uma maneira generalizada, aos mais diversos requisitos de cada um dos sistemas onde é necessário através da actuação e comando de motores passo-a-passo, proceder-se ao posicionamento de um elemento móvel.

Verificou-se que, as grandes marcas a nível mundial da área da automação, como a Allen-Bradley[®], Siemens[®] ou Schneider[®], entre outras, dispõem de soluções para o problema em questão. A figura 2.10 ilustra uma dessas soluções. [11, 12, 13]

A grande maioria dessas soluções consiste em módulos específicos compatíveis com o respectivo PLC ou família de PLCs (processadores) da mesma gama e marca. Esses módulos são anexados e ligados ao PLC, de modo a que o operador (programador) disponha de um vasto leque de funções para o comando de motores passo-a-passo, segundo um ou mais eixos de qualquer aplicação onde este equipamento seja uma solução válida.

Esses módulos de controlo de movimento vêm já com aplicações programadas dedicadas mas também permitem a programação de outras aplicações. São produtos que permitem vários modos comunicação com outros dispositivos relacionados, bem como vários modos de interação com o controlador. A interação com o controlador é normalmente efectuada através de uma HMI.

No que toca ao hardware e software deste tipo de solução, existem algumas pequenas vantagens em as utilizar pois permitem simplificar um pouco a tarefa de

operação. Essa situação deve-se, no que diz respeito ao hardware, à já inclusão das drives e fontes de alimentação adequadas ao controlo de motores passo-a-passo no próprio módulo, dando assim garantias de compatibilidade e afastando o utilizador da necessidade de efectuar as ligações entre os vários dispositivos. Quanto ao software, para além da vantagem de ser possível realizar nova programação, têm disponíveis algumas instruções desenvolvidas propositadamente a pensar no comando de motores passo-a-passo, evitando assim a programação de raiz para algumas funções pretendidas.

Tendo em conta que se propõem obter no final deste trabalho uma solução aplicável a um sistema genérico, onde se pretende obter o posicionamento de motores passo-a-passo segundo dois eixos perpendiculares no mesmo plano, através de um modo de comando didacticamente interessante segundo um protocolo de comunicação eficaz e funcional, que permita efectuar a comunicação entre dispositivos num de dois modos possíveis, aumentando assim a sua flexibilidade. E tendo ainda em conta os tipos de soluções que existem actualmente no mercado, o seu custo e as suas características e limitações, em comparação com aquilo que é possível desenvolver recorrendo aos recursos tecnológicos indispensáveis para uma solução do género, o tempo e dedicação suficiente. Pode-se dizer que o desenvolvimento de uma solução para este trabalho que responda aos requisitos já referidos é sem qualquer dúvida uma proposta muitíssimo interessante e perfeitamente viável.



Figura 2.10 – Produto da Siemens para posicionamento de MPP. [19]

2.4 Proposta de solução

O objectivo deste trabalho é elaborar um indexer que permita controlar o posicionamento do elemento móvel do sistema MPS Festo® disponível na Feup, segundo os eixos X e Y, dentro de uma área de trabalho limitada.

É necessário que este consiga comunicar e sincronizar-se com o controlador (s7-300) do sistema, o mesmo que tem como função deslocar a ferramenta de corte segundo o eixo Z, de forma perpendicular aos eixos X e Y. A ferramenta de corte encontra-se fixa ao elemento móvel a posicionar.

A solução, em grande parte, consiste em desenvolver um programa relativo ao indexer que assegure o correcto funcionamento de determinadas funcionalidades, bem como, habilitar a comunicação via porta serie entre dois controladores.

Uma vez que se pretende testar a solução deste projecto no sistema MPS Festo, a comunicação a ser realizada será entre o controlador (s7-300) da unidade e o indexer (s7-200), de forma a obter-se sincronização dos movimentos segundo os eixos X e Y com o movimento segundo o eixo Z. Por outras palavras, obter-se a sincronização da tarefa de posicionamento XY com a tarefa do movimento ascendente ou descendente da ferramenta de corte.

Uma outra fracção da proposta de solução está relacionada com a especificação e implementação dos elementos de hardware que irão constituir a solução. Na figura 2.11 está ilustrado o esquema do projecto onde estão representados esses elementos.

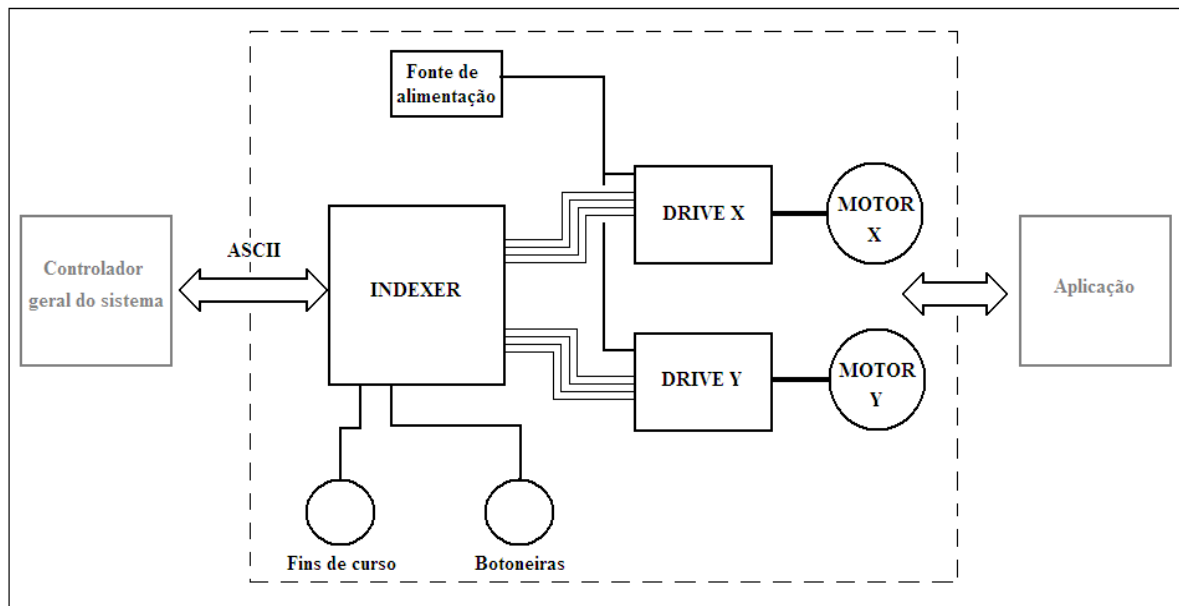


Figura 2.11 - Esquema global da solução projectada.

Definição de Protocolo de comunicação entre Controladores

Pretende-se desenvolver diversas funcionalidades internas e externas que serão úteis no comando de dois motores passo-a-passo. As funcionalidades externas baseiam-se em dispositivos sinalizadores e botoneiras. Quanto as funcionalidades internas, consistem num programa implementado através do software Siemens STEP7 MicroWIN, que permitirá controlar/comandar diversas acções para cada um dos motores passo-a-passo.

As funcionalidades programadas são executadas de um modo sequencial, recorrendo a comandos que consistem em mensagens definidas num protocolo de comunicação, como se apresentará de seguida.

A comunicação entre controladores via porta serie RS-232 através do protocolo de comunicação implementado, facultará controlar remotamente o indexer.

De modo a executar-se o tipo de acção pretendida para cada um dos motores passo-a-passo deverá ser elaborado um protocolo de comunicação onde deverão estar presentes, de uma forma clara e objectiva, todos os comandos disponíveis no indexer bem como toda a informação relativa ao modo de como utilizar cada um desses

comandos facultados pelo indexer (s7-200). Esses comandos serão transmitidos em código (ASCII) e deverão respeitar uma estrutura tipo, para que haja certa uma ordem e lógica de comando.

Para controlar os movimentos dos motores e os processos internos do indexer a ser integrado na unidade de demonstração de produção flexível, deverão ser enviados comandos (mensagens) via porta serie RS-232, do controlador S7-300 para o indexer (s7-200).

Deseja-se ter dois modos de comunicação: modo *mestre/escravo* e o modo *ponto-a-ponto*. Por defeito o indexer deverá encontrar-se no modo *mestre/escravo*, para evitar colisões entre mensagens enviadas e recebidas de controladores, visto que em modo *mestre/escravo* o “escravo” (indexer) apenas responde quando o “mestre” (controlador que interage com o indexer) lhe pedir.

Pretende-se também que seja possível comutar do modo de comunicação de *mestre/escravo* para *ponto-a-ponto* e vice-versa, através dos comandos apropriados. Depois de executado qualquer comando, existirá uma mensagem de feedback por parte do indexer, caso este esteja a comunicar segundo o modo *ponto-a-ponto*. Esta mensagem permitirá perceber se os comandos foram recebidos com sucesso por parte do mesmo. Os comandos não terão de ser constituídos por uma mensagem formada sempre pelo mesmo número de caracteres. Pode inclusivamente haver a necessidade de o mesmo comando conter um número superior ou inferior de caracteres, como é o caso do comando que permitirá definir o número de passos consoante o movimento desejado.

Capítulo 3

Arquitectura da solução

Este capítulo aborda a estrutura e a constituição física da solução desenvolvida. Descreve a função e exhibe a organização dos elementos essenciais que compõem o indexer a partir de um ponto de vista elementar e fundamental.

3.1 Componentes electromecânicos

O hardware relativo à solução deve assegurar o correcto posicionamento dos motores passo-a-passo, bem como assegurar uma correcta conexão e troca de energia entre os elementos constituintes da solução.

A solução inclui os seguintes componentes: sistema de accionamento, máquina (célula de maquinagem do sistema MPS), motores, drives, controlador e fontes de alimentação. Estes são também os componentes principais envolvidos em qualquer tipo de sistema de posicionamento, a juntar a estes a solução projectada ainda compreende fins de curso, botoneiras, e por fim o controlador (s7-300) da célula de maquinagem. O esquema de hardware da solução está representado na figura 2.11.

Botoneiras: As botoneiras (figura 3.1) serviram para executar funções indispensáveis em qualquer unidade de produção. É importante em caso de alguma coisa correr fora do previsto, existirem botoneiras para parar de emergência a operação que se está a executar no momento ou que possibilitem fazer reset ao indexer de maneira a evitar-se danos maiores. Puderam existir também botoneiras para activar ou

desactivar parâmetros tais como, energizar motores, alterar o tipo de passo, ou mesmo o sentido de rotação.

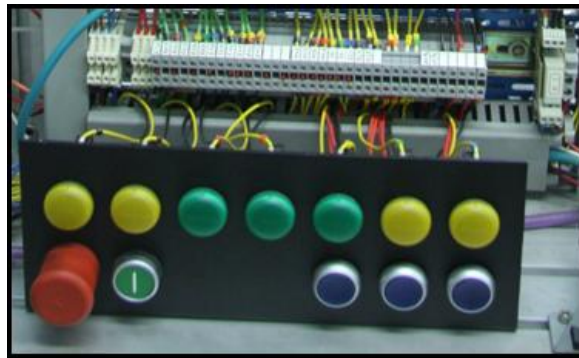


Figura 3.13 – Exemplo de botoneiras a implementar na solução.

Drive: A drive pode ser considerada o coração de todo o sistema. Este elemento controla a velocidade, tipo de passo, direção e a energização ou não do motor. Fazem parte da solução duas drives, em cada uma das drives estará associado um motor. A drive irá aplicar ao motor, níveis de corrente em conformidade com os sinais provenientes do indexer e desta forma controlar as acções do motor. A figura 3.2 ilustra a disposição e as ligações entre o indexer a drive e os motores. Entre as drives e o PLC (s7-200) foi ainda implementada uma placa de interface adaptadora (ver figura 5.3.).

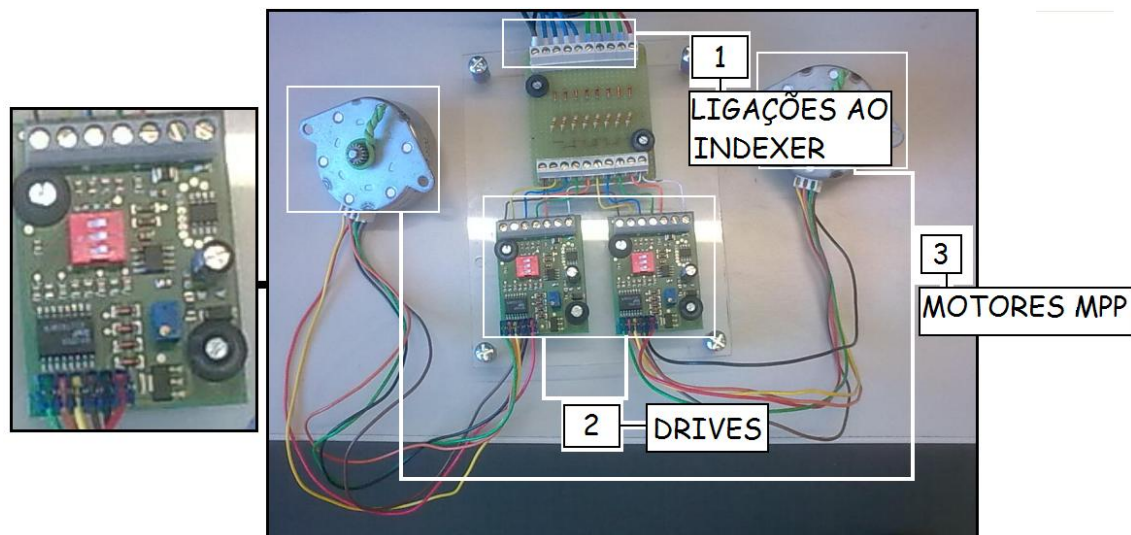


Figura 3.2 - Conexão das Drives ao indexer e motores.

Fins de curso: Na estrutura do sistema MPS Festo existem dois fins de curso instalados, cada fim de curso está associado a um motor MPP, e estes vão permitir ao

indexer identificar a posição zero (posição de origem). Os fins de curso irão limitar os movimentos dos motores de acordo com os limites da área de trabalho.

Fonte de alimentação: A drive para funcionar eficazmente deve ter uma fonte de energia. Se a drive for eléctrica, deve ter uma fonte de alimentação simples ou trifásica disponível. A drive irá receber esta energia e irá modifica-la para uma energia á saída que seja utilizável e indicada para o tipo de motor a utilizar.

Indexer: É o elemento que processa os sinais, faz “cálculos” de acordo com o estado lógico das variáveis programadas e/ou com base nos sinais das entradas I/O, e gera nas saídas utilizadas um sinal de referência. Este sinal de referência de saída indica à drive o tipo de movimento que deve gerar. O indexer (figura 3.3) será desenvolvido sobre um PLC (s7-200) da Siemens® e deverá ser capaz de permitir ao utilizador comandar dois motores passo-a-passo, através de comandos enviados por mensagens trocadas com o controlador hierarquicamente superior, via porta serie. Todo isto para possibilitar posicionar uma elemento móvel dentro de um plano XY limitado, desenvolvendo-se trajectórias lineares.



Figura 3.3 - Indexer (Siemens® S7-200). [19]

Máquina: A aplicação ou máquina é de certa forma a essência de qualquer sistema de comando, afinal é aqui que necessariamente se executará o trabalho. Neste trabalho a solução será desenvolvida, tendo em conta a aplicação relativa à célula de maquinaria do sistema MPS (figura 3.4), tendo sempre em consideração a preocupação de se procurar obter no final, uma solução que possa também ser aplicada a outra qualquer aplicação semelhante, que necessite de posicionar um elemento móvel através do comando de dois motores passo-a-passo. [14, 20]



Figura 3.4 - Célula de maquinagem.

Motor: Um motor passo-a-passo (figura 3.5) é um actuador electromagnético que converte impulsos eléctricos em deslocamentos mecânicos. A cada impulso de comando corresponde um deslocamento incremental bem determinado que toma o nome de passo. Essa característica confere ao motor passo-a-passo uma grande facilidade de interligação com sistemas digitais o que conduz à sua utilização em numerosos domínios. Existem vários tipos de motores passo-a-passo que usam várias formas de energia. A dimensão do motor normalmente determina a quantidade de movimento de rotação que pode gerar a partir de entrada de energia, sendo também verdade que existem algumas excepções a este princípio. [23]

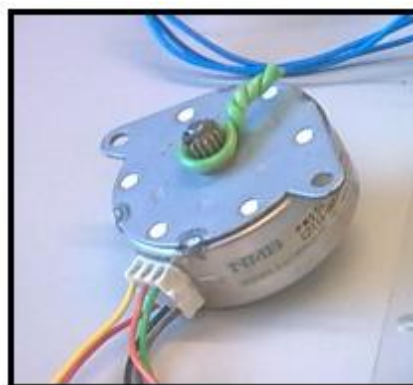


Figura 3.5 - Motor passo-a-passo.

Sistema de accionamento: É o dispositivo que faz a ligação do motor ao elemento móvel. A unidade dispõe de dois sistemas de accionamento (figura 3.6), um para cada eixo, para fazer deslocar o elemento móvel segundo o eixo X e Y. A sua tarefa básica é fazer uma conexão sólida entre o motor e a aplicação. Sistemas de accionamento podem aceitar um diâmetro de eixo do motor e converter a saída para um outro tamanho de eixo. Podem ser considerados dispositivos adaptativos devido à sua capacidade para fornecer energia sem problemas para a aplicação. De certa forma, este dispositivo pode também amortecer os choques transmitidos pelo motor para a aplicação.

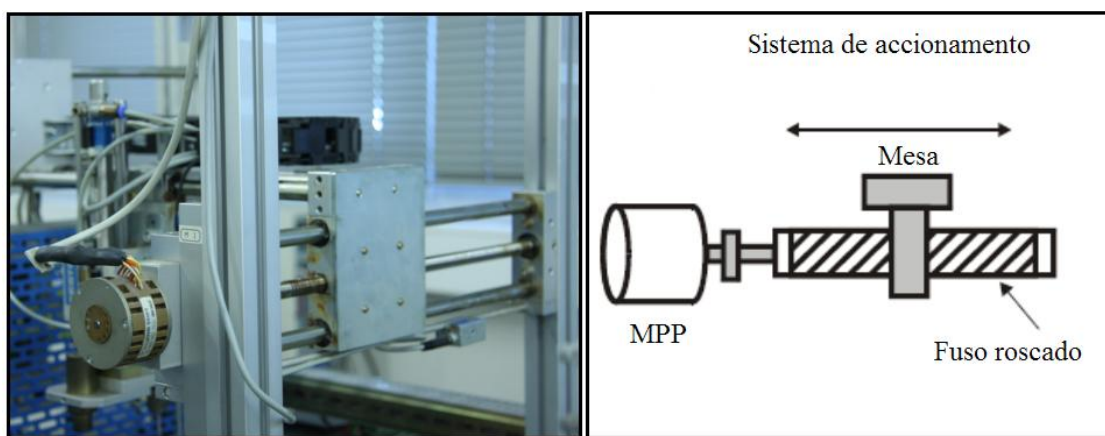


Figura 3.6 - Sistemas de accionamento para MPP. [14]

3.1.1 Especificidades do indexer

As especificidades que se pretende implementar no indexer são as seguintes:

- Todos os caracteres dirigidos ao indexer devem ser em código ASCII.
- Os parâmetros de comunicação definidos para o indexer são: baud rate de 9600, 7 bits de dados por carácter, e paridade ímpar.
- Os parâmetros de recepção de mensagem definidos para o indexer são: activar recepção de mensagens, usar o valor de 50ms para detectar a condição de linha inactiva, temporizador inter-caracteres de 4000ms, utilizar carácter de fim de mensagem, número máximo de caracteres a receber igual a 14.

- As mensagens a trocar entre os intervenientes devem ser as mais sucintas possíveis.
- Duas das entradas do PLC s7-200 devem estar reservadas para cada um dos fins de curso associado ao respectivo eixo de translação. Os fins de curso limitarão os movimentos dos motores de acordo com a área de trabalho. Quando cada um dos motores atingir a posição de origem (0,0), os fins de curso estarão actuados, e nesse instante automaticamente os motores irão parar.
- Deverão ser disponibilizadas entradas do PLC para se instalar botoneiras que sirvam para fazer *reset* ao programa, *paragem* de emergência dos motores, e *activação/desactivação* de funcionalidades relevantes.
- Os valores de limite máximo para as coordenadas X, Y até as quais os motores puderam-se deslocar, estão definidas internamente (programado). Sempre que se atingir esse valor é enviada uma mensagem de erro própria.
- Se enviarmos a mesma ordem repetidamente, de deslocar o(s) moto(res) para uma determinada posição, estes após se encontrarem na posição desejada não devem reagir mais e esse comando.
- Todas as ordens de posicionamento dirigidas ao indexer que envolvam o deslocamento dos dois motores devem produzir trajectórias lineares.
- Deve existir um comando que permita confirmar se a comunicação entre o indexer e o controlador hierarquicamente superior encontra-se activa ou não.
- Deve existir dois modos de comunicação opcionais (mestre/escravo e ponto-a-ponto) para permitir uma maior flexibilidade de comunicação e uma maior compatibilidade entre o indexer e um outro controlador, de acordo com a porta de comunicação utilizada (RS-232 ou RS-485).
- Deve existir uma diversidade de funcionalidades suficiente que permita ao indexer dispor de um comando útil e eficaz no posicionamento dos motores passo-a-passo.

- Deve existir uma diversidade de erros programados de maneira a informar acerca da origem do problema.
- O indexer deve contar com dois comandos independentes para efectuar o posicionamento. Um comando particular para executar um movimento de acordo com todos os parâmetros previamente definidos (comando G). E um outro para posicionar os motores na posição de origem de acordo com parâmetros definitivos (comando O).
- No modo de comunicação mestre/escravo, ao executar-se um grupo de comando W (Write) não deve de ocorrer posteriormente qualquer mensagem de feedback.
- No modo de comunicação ponto-a-ponto, ao executar-se um grupo de comando W (Write) deve ser enviada logo de seguida, em caso de sucesso (grupo de comando identificado) uma mensagem de feedback, constituída apenas pelo carácter do comando previamente inserido mais o carácter final.
- Em ambos os modos de comunicação é possível através do grupo de comando R (read) ter acesso a informações ou dados relevantes acerca do estado de funcionamento dos motores passo-a-passo.

3.2 Modelo de software

A programação foi dividida em três grandes grupos de modo a permitir estruturar o programa e separar as funções distintas que cada um desses grupos vai desempenhar. Na figura 3.7 está ilustrado o esquema delineado para a implementação do programa.

A delineação da estrutura do programa, facilita a compreensão de quem pretende assimilar como foi obtida a programação final, como o programa está organizado, e facilita também a manipulação e optimização da programação desenvolvida a quem necessitar de alterar/acrescentar algo.

Conseguiu-se assim que cada grupo de programação desempenhe as suas funções sem se preocupar com aquilo que se passou ou se vai passar nos outros grupos.

O programa distribuí-se em três grupos ligados entre eles por dois buffers, onde vão ficar alojados os valores a atribuir a determinadas variáveis, e as configurações desejadas.

Um desses grupos é o **hospedeiro de mensagens**, esta parte do programa tem como função principal estabelecer a comunicação entre controladores segundo o modo Freeport e definir os parâmetros de comunicação, de modo a habilitar o indexer a receber e enviar mensagens. Outro grupo é o **gestor de parâmetro** que está associado á parte dos cálculos necessários à execução das funcionalidades internas do indexer. Por fim, o terceiro e último grupo é o **controlador dos motores**, este tem como principal função colocar os motores a funcionar de acordo com a configuração prévia dos parâmetros de funcionamento dos motores.

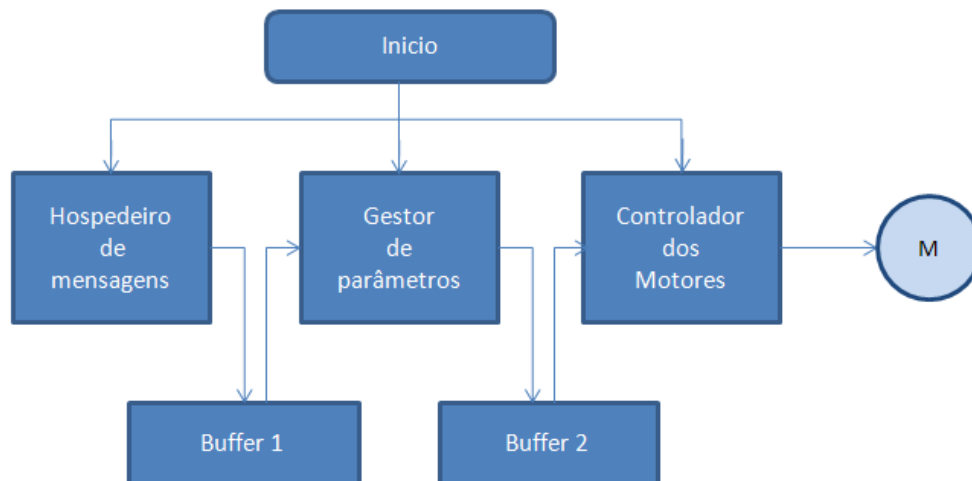


Figura 3.7 - Modelo geral do programa.

O hospedeiro de mensagens e o gestor de parâmetros são as duas primeiras etapas do programa a actuar. Ao inicializar o programa o gestor de parâmetros configura os valores pré-definidos de movimento, e o hospedeiro de mensagens habilita de alguma maneira transmitir ao indexer aquilo que se deseja ver executado.

O hospedeiro de mensagens é uma parte distinta do programa, que tem como função principal receber e enviar mensagens. As mensagens a ser recebidas pelo indexer são introduzidas por um utilizador, via porta serie RS-232 através de um controlador que possibilite enviar caracteres (formarão mensagens com os comandos desejados) e também admita receber caracteres, que nos fornecerá dados e informação acerca do funcionamento dos motores.

Na fase de operação com todo sistema MPS, o indexer irá comunicar com o controlador da unidade (s7-300). Já numa fase de testes, esse controlador poderá ser por exemplo; um PC equipado com um software particular que permita enviar e receber caracteres.

A comunicação entre controladores pode ser feita de dois modos, comunicação ponto-a-ponto (figura 3.8) ou comunicação mestre/escravo (figura 3.9). Este ultimo modo de comunicação é o modo que por defeito está ativo e é ideal para comunicações em que o indexer seja composto por um conjunto de módulos (o indexer será constituído por apenas um módulo, mas em trabalhos futuros pode vir a ser formado por um conjunto de módulos). No modo de comunicação ponto-a-ponto, sempre que o

indexer receber uma mensagem identificada como sendo uma ordem de comando (funcionalidade programada) é enviada logo de seguida uma outra mensagem de feedback, uma vez que, neste modo de comunicação a probabilidade de ocorrer colisões de mensagens é reduzida.

Se a comunicação entre controladores for efectuada segundo o modo mestre/escravo, não é enviada qualquer mensagem de feedback como resposta ao grupo de comando enviado anteriormente. Neste modo de comunicação o indexer apenas responde quando lhe for perguntado, endereçado um grupo de comando Read (Ler) e quem responde é apenas o módulo a que for endereçado esse grupo de comando.

Para se verificar se um determinado módulo está activo, existe um comando específico que pode ser endereçado ao indexer. Esse comando apenas pode ser utilizado no modo de comunicação ponto-a-ponto, na situação de o módulo estar activo é enviada uma mensagem específica. No modo de comunicação mestre/escravo é possível verificar se um módulo está activo através das indicações luminosas disponíveis no PLC caso se envie um comando que permita activar ou desactivar uma parâmetro de funcionamento, como por exemplo activar a energização de um motor passo-a-passo. Independentemente do modo de comunicação definido no programa, quando um grupo de comando é enviado a um módulo inexistente, não é enviada qualquer mensagem de volta como sinal de ausência do módulo.

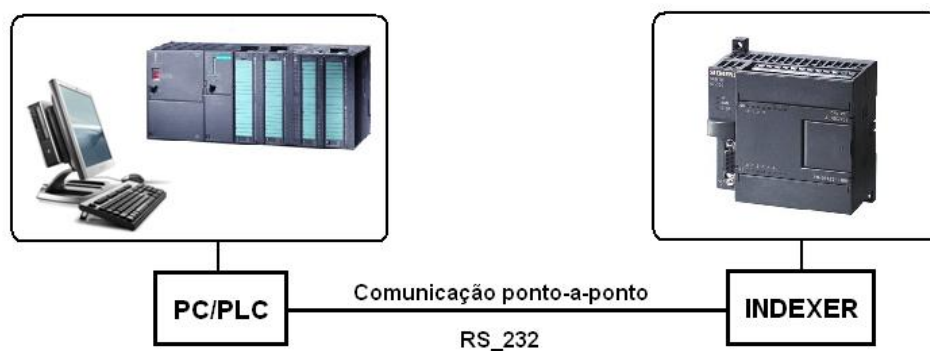


Figura 3.8 - Esquema do modo de comunicação ponto-a-ponto.

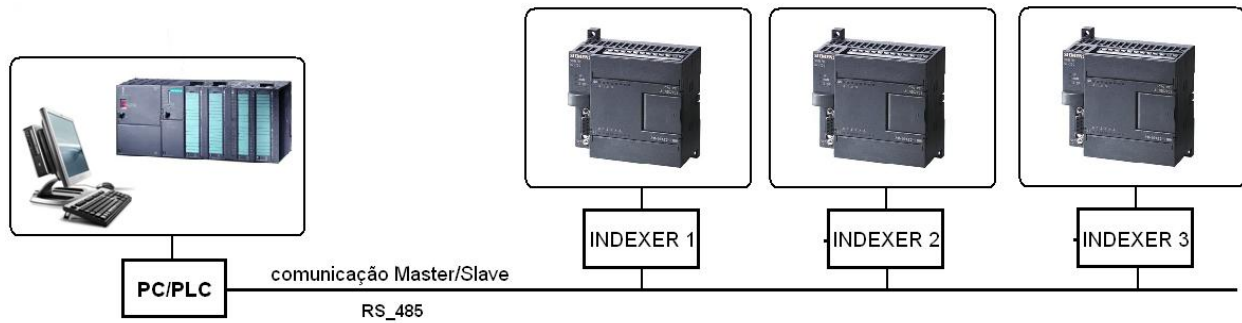


Figura 3.9 - Esquema do modo de comunicação mestre/escravo.

O **gestor de parâmetros** é o grupo do programa dedicado a todas as funcionalidades internas programadas; é a parte do programa que vai activar as funcionalidades desejadas, e é também quem efectua os cálculos necessários para verificar se determinada funcionalidade pode ser realizada ou não.

Nesta parte do programa, após o indexer receber uma mensagem onde é identificado um dos comandos definidos no protocolo de comunicação (tabela 3.3), o valor lógico da variável associada a esse comando é alterado (set/reset), e a partir desse instante a variável á qual foi alterado o valor lógico está relacionada a uma determinada funcionalidade que passa a estar activada, e pronta a ser executada.

São realizados cálculos de uma forma autónoma para a execução das rampas de aceleração e desaceleração, e para a execução de trajectórias lineares. Para que estas duas funcionalidades ocorram é necessário apenas definir anteriormente um valor para a posição e velocidade pretendida. Para estas serem executadas, no caso das rampas é preciso activar a funcionalidade através do respectivo grupo de comando. Já no caso das trajectórias lineares, esta funcionalidade está permanentemente activada.

Quando é enviado um grupo de comando para deslocar o(s) motor(es) passo-a-passo até uma determinada posição, seja o movimento em coordenadas absolutas ou relativas, é verificado se o valor da posição até a qual se deseja deslocar o(s) motor(es) é permitida ou se ultrapassa o valor limite da área de trabalho. Sempre que forem introduzidos valores de forma incorrecta para a posição ou velocidade pretendida, será enviada uma mensagem de erro a informar acerca do problema.

Associado ao gestor de parâmetros estão ainda dois buffers de dados, que estabelecem a ligação deste ao hospedeiro de mensagens, que vai processar todas as

mensagens do indexer, e ao controlador dos motores que vai colocar os dois motores passo-a-passo a trabalhar segundo os parâmetros que ficaram definidos anteriormente no gestor de parâmetros.

O **Controlador dos motores** é a última etapa do programa a ser percorrida. O PLC quando é inicializado, é feito no primeiro scan do PLC uma configuração dos parâmetros, onde se colocam os dois motores desenergizados, com passo completo (full-step), e com rotação no sentido que permita a ferramenta de corte afastar-se da posição de origem, são também definidos valores por defeito para o número de passos e tempo de ciclo.

Esta parte do programa ignora as comunicações entre controladores, o tipo de mensagens que chegaram ao indexer, se são mensagens que irão originar erros ou se são mensagens válidas, também não se “preocupa” com qual ou quais foram as funcionalidades internas afectadas, muito menos se preocupa se tais funcionalidades podem ser activadas para serem posteriormente executadas pelo controlador dos motores, todo isto já foi tratado atrás pelos outros dois grupos do programa, o controlador dos motores não se preocupa com nenhum destes aspetos, apenas interpreta os valores que chegaram ao buffer 2 provenientes do gestor de parâmetros e coloca os motores a operarem de acordo com a configuração que aí se encontra.

3.3 Modelo de comunicações MS e PP

3.3.1 Comunicação entre dispositivos

Porque os PLC não funcionam isoladamente, eles não devem ser unicamente encarados como máquinas com capacidade para processar informação; também as suas possibilidades de comunicação com os sistemas que o rodeiam, especialmente outros dispositivos, é um aspecto importante. [20]

As melhorias no hardware e software dos computadores actuais fazem com que as comunicações entre dispositivos de controlo sejam muito mais simples. Muitos dos softwares de comunicação que estão disponíveis, tornam fácil a comunicação com PLCs e outros dispositivos.

A maioria das máquinas oferece capacidade de comunicação série, usando o modo de comunicação assíncrona e tendo uma porta série RS-232 disponível. Podemos ser levados a pensar que os dispositivos com porta série RS-232 podem comunicar facilmente com outro qualquer dispositivo que disponha de uma porta série RS-232, o que definitivamente não é verdade, cada dispositivo pode estar definido para parâmetros de comunicação diferentes. [10]

Existem diversas normas relativas à representação de um “1” e de um “0” no contexto da comunicação série. A norma mais divulgada é a “RS-232-C”, normalmente referida apenas por RS-232. Praticamente todos os sistemas operativos a suportam. Não menos relevante é o facto da generalidade dos sistemas industriais programáveis utilizarem precisamente esta interface para a descarga de programas desenvolvidos normalmente em computadores pessoais. É o caso dos controladores lógicos programáveis, controladores de temperatura, controladores de motores, etc. É também através desta interface que os programas em execução são analisados no computador em que foram desenvolvidos com recurso a ‘software’ próprio (‘debuggers’).

A comunicação estabelecida entre o indexer e um outro controlador hierarquicamente superior é promovida pelo protocolo Freeport. É um modo de operação através da qual é possível controlar a porta de comunicação da CPU S7-200. Este protocolo é facultado pela Siemens® e permite definir os parâmetros de comunicação e de recepção de mensagem desejados. Usando o modo Freeport, pode-se

implementar protocolos de comunicação de interface para diversos tipos de dispositivos. Um programa ao ser desenvolvido pode controlar as operações da porta de comunicação através de interrupções de recepção e transmissão, e das instruções de transmissão (XMT) e de recepção (RCV). O protocolo de comunicação é controlado totalmente pelo programa enquanto estiver no modo Freeport. O modo Freeport é activado por meio da variável SMB30.

Dado a que o PLC (s7-200) sobre o qual será desenvolvido o indexer, comunicará a partir da porta RS-485 e através de um conversor poderá comunicar também a partir da porta RS-232. Nesta situação faz sentido existir dois modos de comunicação onde cada um se adapte melhor ao tipo de porta a utilizar.

O programa desenvolvido para o indexer permite seleccionar qual dos dois modos de comunicação referidos anteriormente se deseja activar.

Operar com o indexer no modo de comunicação *mestre/escravo* traz simplicidade ao processo de executar os comandos, inexistência de colisões no tráfego da rede, permite formar uma rede de comunicação e nesse caso é mais fácil a detecção de falhas de ligação. Já a comunicação no modo ponto-a-ponto é ideal para uma comunicação entre dois dispositivos, introduz a capacidade de comunicar situações ao controlador hierarquicamente superior por iniciativa própria. Em todo o caso existe um grupo comando (tabela 3.6) que permite obter informações relevantes ao indexer em qualquer um dos dois modos de comunicação.

3.3.2 Princípios de comunicação no modo Ponto-a-Ponto

Um canal de comunicação que liga dois (e só dois) computadores denomina-se uma ligação *ponto-a-ponto*. Uma comunicação estabelecida sobre um canal deste tipo denomina-se comunicação *ponto-a-ponto*. [21]

Há duas grandes formas de comunicação *ponto-a-ponto*: a comunicação paralela e a comunicação série, esta última é a utilizada no programa do indexer. A diferença entre elas tem a ver com o canal de comunicação, ou seja, o recurso físico que conduz a informação desde o emissor ao receptor.

No que diz respeito a solução a desenvolver neste trabalho, dados os princípios de funcionamento de cada um dos modos de comunicação, ao contrário da comunicação *mestre/escravo*, na comunicação *ponto-a-ponto* existe sempre uma mensagem de feedback. Sempre que o indexer recebe uma mensagem, envia de volta uma outra mensagem para indicar: que ocorreu um ou mais erros, ou então na mensagem recebida foi identificado um grupo de comando e nesse caso é enviada uma mensagem de volta com o carácter do respectivo comando.

3.3.3 Princípios de comunicação no modo Mestre/Escravo

No modo *mestre/escravo* desenvolvido sobre o protocolo Freeport, o indexer desempenha o papel de *escravo*, o outro dispositivo que irá comunicar com o indexer será o *mestre*. Tal como em outros protocolos *mestre/escravo*, aqui também apenas o *mestre* pode estabelecer uma comunicação. O protocolo de comunicação apenas permite ter um *mestre* mas é permitido ter mais do que um *escravo*. Aos *escravos* não é permitido comunicar entre si, apenas é permitido responder às solicitações do *mestre*. Neste projecto o indexer apenas irá ser constituído por um módulo (um *escravo*) mas a programação foi desenvolvida a pensar que em trabalhos futuros poderá ser útil o indexer ser constituído por outros módulos (para acrescentar outro tipo de soluções).

3.3.4 Modo Freeport para controlar a porta de comunicação serie

Foi seleccionado o modo Freeport para controlar a porta de comunicação serie do indexer (Micro-PLC s7-200) através do programa desenvolvido. Quando o modo Freeport é seleccionado, o programa passa a controlar a actividade da porta de comunicação através do uso de interrupções e instruções de recepção e transmissão. O protocolo de comunicação é inteiramente controlado pelo programa em diagrama de escada, enquanto estiver no modo Freeport. SMB30 (para porta 0) é a memória especial usada para seleccionar os parâmetros da comunicação, tais como, a taxa de transmissão e paridade. [16]

O modo Freeport é desactivado e a comunicação normal é restabelecida (por exemplo, para ter acesso ao dispositivo de programação), quando o indexer s7-200 se encontra no modo de operação STOP (programa não está a ser executado).

Através deste modo de comunicação pretende-se enviar mensagens, que transmitiram uma ordem de comando a um outro controlador, e também receber mensagem desse mesmo controlador de maneira a se obter informações e dados relevantes. Para tal é necessário, conceber um programa para suportar o protocolo que será usado pelo controlador com o qual o indexer s7-200 comunicará, enquanto estiver no modo Freeport.

A comunicação no modo Freeport só é possível quando o PLC que implementa o indexer estiver no modo de operação RUN (programa está a ser executado). O modo Freeport é activado definindo um valor de 01 no campo de selecção de protocolo, do byte SMB30 (Porta 0). Enquanto estiver no modo Freeport, a comunicação com o dispositivo de programação não é possível.

SMB30 (tabela 4.3) descrevem o byte de controlo Freeport. É gerado um bit de stop para todas as configurações.

Transmissão de dados

A instrução de transmissão permite enviar um buffer de um ou mais caracteres, até ao máximo de 255. A figura 3.10 mostra o formato do buffer de transmissão. [16]

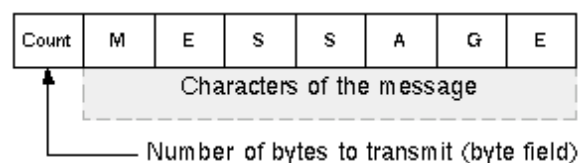


Figura 3.10 - Formato do buffer de transmissão de dados. [1]

O indexer gera uma interrupção após o último carácter do buffer ser enviado, devido á rotina de interrupção (evento 9 de interrupção para a porta 0) estar anexa ao evento- transmissão completa. No programa são também efectuadas transmissões sem o uso de interrupções, tal é conseguido monitorizando a variavel SM4.5 ou SM4.6 para sinalizar quando a transmissão é concluída.

Recepção de dados

A instrução de Receber, à semelhança com a instrução de transmitir, permite receber um buffer de um ou mais caracteres, até ao máximo de 255. A figura 3.11 mostra o formato do buffer de recepção. [16]

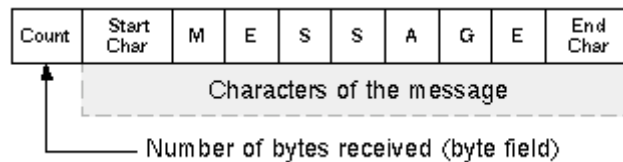


Figura 3.11 - Formato do buffer de recepção de dados. [1]

Foi anexa uma rotina de interrupção ao evento-recepção de mensagem completa, para que o indexer gere uma interrupção (evento de interrupção 23 para a porta 0) após o último carater do buffer ser recebido.

O indexer pode receber mensagens também sem o uso de interrupções, através da monitorização da variável SMB86 (porta 0). Este byte será diferente de zero quando a instrução de recepção de mensagens estiver inactiva ou tenha sido terminada. Será zero quando a recepção de uma mensagem estiver em progressão.

A instrução de recepção permite seleccionar o início da mensagem e as condições de fim de mensagem. SMB86 até SMB94 foi utilizada para controlar e ler, o estado da instrução de recepção de mensagem para a porta 0.

Condições de início e fim para a Instrução de recepção

A instrução de recepção usa os bits de controlo de recepção de mensagem (SMB87), para definir as condições pretendidas de início e fim de mensagem. A instrução de recepção suporta varias condições de iniciar uma mensagem, bem como também suporta várias maneiras de terminar uma mensagem. A mensagem pode iniciar e terminar de acordo com uma das opções disponíveis, ou uma combinação de opções. No indexer temos apenas a condição de fim de mensagem, por esse motivo vamos referir apenas essa condição.

O carácter final é o carácter CR: *carriage return* (013: em decimal) do código ASCII, e é usado para indicar o final de uma mensagem. Assim que o carácter final for recebido, são agregados a este, os caracteres anteriores e todos serão colocados no buffer de mensagem, nesse instante a recepção da mensagem é terminada.

A detecção do carácter final é usada com o protocolo ASCII onde cada mensagem termina com um carácter específico.

Esses caracteres não são mais do que ‘*bits*’ recolhidos a um ritmo dado pelo ‘*baud rate*’ acordado com o emissor. Na configuração dos parâmetros do programa, ficou definido um ‘*baud rate*’ de 9600 e a transmissão de 7 ‘*bits*’ de dados, ‘*bits*’ esses enviados por ordem crescente. Imediatamente após os ‘*bits*’ de dados, é enviado um ‘*bit*’ de *paridade*. Foi definida uma paridade par, este ‘*bit*’, redundante do ponto de vista do conteúdo da mensagem a trocar, permite que o receptor detecte eventuais erros de comunicação.

Dado a que as mensagens são enviadas usando a comunicação assíncrona, cada mensagem é fragmentada em caracteres individuais e é transmitido um bit de cada vez. O programa do indexer utiliza o sistema ASCII para enviar e receber mensagens. Em ASCII cada número, letra, e alguns caracteres especiais, têm um valor binário-codificado equivalente. Existe o código ASCII básico 7-bits e um código ASCII expandido de 8 bits. O código ASCII utilizado foi o ASCII 7-bits que representa 128 diferentes, letras, números e caracteres especiais.

3.3.5 Utilização do cabo PC/PPI com outros dispositivos e protocolo Freeport

O cabo PC/PPI e as funções de comunicação do Freeport foram usadas para se estabelecer a comunicação entre o indexer (S7-200) e os outros dispositivos compatíveis com a norma RS-232 standard. [16]

Um cabo isolado PC/PPI com uma porta RS-232, tem cinco comutadores para definição da cadência de transmissão entre outros itens de configuração. Ver figura 3.12.

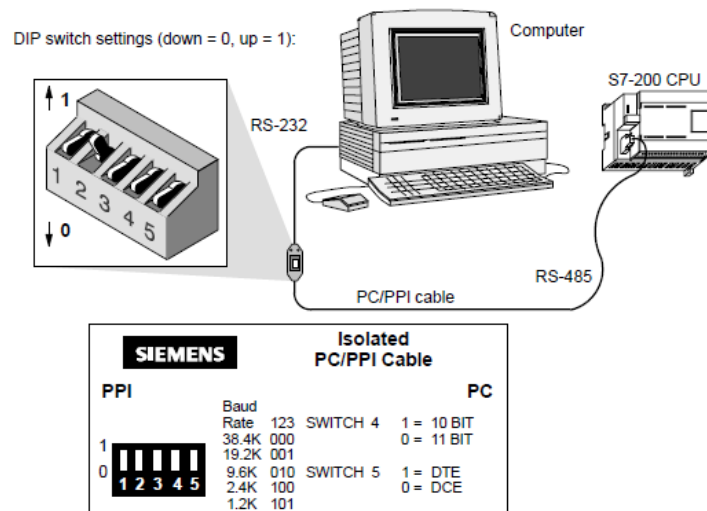


Figura 3.12 - Comunicações com o PLC em modo PPI. [16]

O cabo muda imediatamente de o modo receber para o modo transmitir, assim que é detectado um carácter na linha de transmissão RS232. O cabo volta para o modo receber quando a linha de transmissão está no estado inactivo por um período de tempo definido como o tempo de retorno do cabo. Este tempo depende da selecção da cadência de transmissão, e permite ao cabo PC/PPI temporizar o suficiente para mudar do modo *transmitir* para o modo *receber*, de maneira a que os todos os dados possam ser transferidos a partir da porta RS-485 para a porta RS-232.

Todos os aspetos da comunicação entre o emissor e o receptor são resolvidos por acordos. Por exemplo, se o utilizador de um computador pretender comunicar com o utilizador de outro computador por meio de uma comunicação série, eles poderão “combinar” entre si que os ‘bits’ serão transmitidos por ordem crescente (isto é, no âmbito de cada ‘byte’, o ‘bit’ 0 é o primeiro a ser enviado e o 7 o último). [10, 21]

Os restantes aspetos da comunicação têm um contexto *temporal*, passam também por acordos entre o emissor e receptor. Um importante acordo é a *cadência de transmissão*, ou, equivalentemente, o *tempo de duração* da transmissão de cada ‘bit’.

Pode ocorrer algum desentendimento entre emissor e receptor acerca do tempo de transmissão de cada ‘bit’. Dai a necessidade de haver um acordo entre intervenientes e de cada um deles *possuir um relógio*!

A ressincronização é conseguida por troca de mensagens, tem-se que, no caso limite, não é possível transferir informação entre os intervenientes, porque estes usam

inteiramente o canal de comunicação para proceder a ressincronizações. Daí que um parâmetro de interesse no âmbito da comunicação série seja a taxa de ocupação do canal com informações afectas a processos de ressincronização. Como é evidente, quanto maior for essa taxa menor é a eficiência do sistema de comunicação.

O modo *assíncrono* é o modo de comunicação série utilizado entre o indexer e outro qualquer dispositivo cuja comunicação seja compatível. A figura 3.13 permite compreender a função dos ‘bits’ delimitadores e a essência da comunicação série assíncrona.

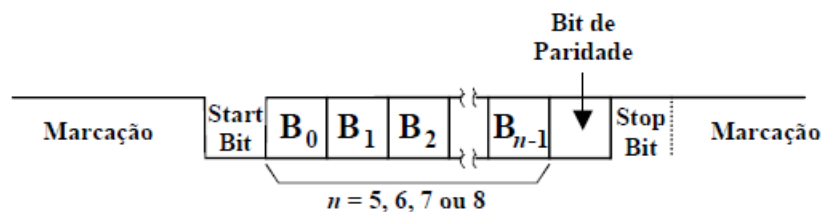


Figura 3.13 - Transmissão Série Assíncrona. [1]

No âmbito da comunicação assíncrona conforme foi salientado, a comunicação série envolve uma série de acordos entre o emissor e o receptor. Um deles prende-se com o número de ‘bits’ transmitidos após cada ressincronização e ao significado de cada um. Outro acordo, é o *tempo de duração* de cada ‘bit’, ou seja, o tempo em que o canal de comunicação é mantido em “0” ou “1”, assinalando o valor lógico do ‘bit’ a transmitir. O inverso do tempo de duração de um ‘bit’ dá a frequência com que os ‘bits’ são enviados para o canal de comunicação. Esta frequência designa-se por ‘*baud rate*’. No contexto da comunicação série assíncrona, a ressincronização de relógios ocorre sempre que se inicia a transmissão de cada uma das palavras em que uma mensagem foi decomposta, e apenas nesses instantes. A ressincronização é normalmente coordenada por 2 ‘bits’ especiais que são acrescentados a cada palavra a transmitir, os quais se designam por ‘*start bit*’ e ‘*stop bit*’. Dadas as suas posições estes dois ‘bits’ delimitam cada palavra enviada: o ‘*start bit*’ inicia-a, e o ‘*stop bit*’ finaliza-a.

3.3.6 Descrição da comunicação a implementar

O programa comunica com uma interface, usando as instruções (XMT) e (RCV) do software STEP 7-Micro/WIN para transmitir e receber dados respectivamente.

Foram implementados vários estados de comunicação na programação do indexer, para poder realizar as comunicações. O programa é composto por cinco estados comunicação:

- INACTIVO (0)
- RECEBE (1)
- TRANSMITE (2)
- ESPERA (3)
- ERRO (4)

Consulte a figura 3.14 onde se encontra representado graficamente os estados de comunicação e como estes evoluem.

O modo como as comunicações se processam é essencialmente o seguinte:

- Inicialização do programa.
- Definição do tipo de comunicação.
- Definição dos parâmetros de comunicação.
- A função de receber uma mensagem é autorizada.
- O programa faz a transição para o estado INACTIVO, esperando por um comando ser recebido.
- Um grupo de comando é recebido.
- Identificação do grupo de comando, verificação se existe ou não.
- Verificação da mensagem recebida para perceber se existe algum erro.
- O comando é executado, a funcionalidade é activada.
- O programa faz a transição para o estado de transmissão (TRANSMITE).
- Caso exista alguma mensagem a transmitir, essa é transmitida.
- A função de receber uma mensagem é novamente autorizada.

Nota: O estado de comunicação ESPERA (3) "tempo de espera de dois caracteres" só é necessário se estiver a ser usando um cabo PC/PPI. O tempo de espera de dois caracteres é o tempo necessário para se dar um ciclo completo de comunicação. Se estiver a ser usando um cabo de rede RS-485, pode-se omitir o estado de comunicação ESPERA (3).

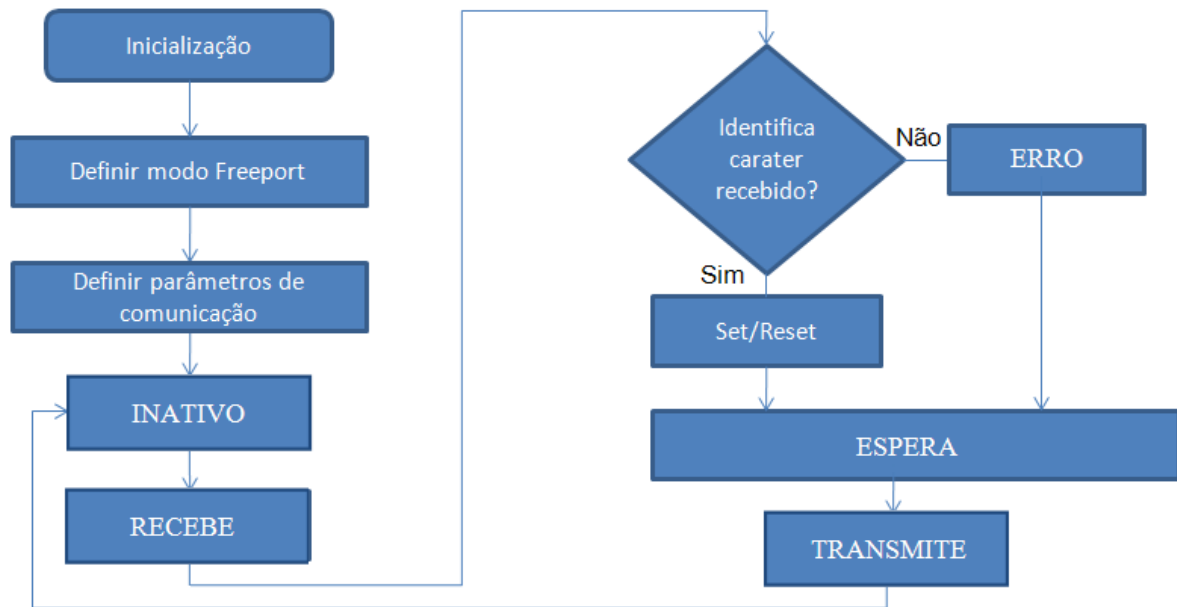


Figura 3.14 - Flowchart da comunicação entre controladores.

3.3.7 Descrição do protocolo a implementar

O protocolo idealizado para este trabalho será o resultado da reflexão sobre vários protocolos estudados. Não é prudente nem existe qualquer tipo de necessidade de se engendrar algo completamente oposto a todo aquilo que já foi pensado e repensado por muita gente com bastante experiencia nesta área, e que com toda a certeza sentiram na pele e tiveram a necessidade de desenvolver um tipo de protocolo que responde-se o melhor possível as exigências.

As principais referências consultadas para a análise do protocolo a elaborar para este trabalho foram os manuais técnicos das referências bibliográficas número [5], [11], [12], e [13], sendo o manual da referência bibliográfica [5] aquele que mais influência teve, por ser bastante intuitivo e se aproximar mais das ideias pensadas para o protocolo deste trabalho.

Todos os comandos deste protocolo serão transmitidos em código (ASCII) e devem respeitar uma estrutura tipo, para que haja certa uma ordem e lógica de comando. Dai pretende-se que estes sejam introduzidos da seguinte forma:

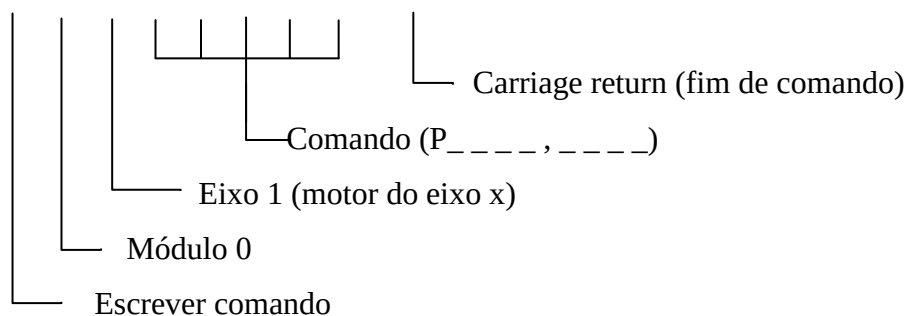
Tabela 3.1 - Grupo de comando.

Tipo de operação:	W – Para escrever uma operação R – Para ler o estado do movimento
Endereço do módulo:	0 até 9
Qual o eixo:	1 – Para o eixo x 2 – Para o eixo y 3 – Para ambos os eixos
Inserir comando:	(inserir apenas um comando)
Fim de comando:	CR

Enviar um grupo de comando (tabela 3.1), consiste em definir o tipo de operação, módulo solicitado, o(s) eixo(s) a que nos pretendemos dirigir, definir qual o comando que se pretende executar (tabela 3.3: funcionalidades programadas), e no final o grupo de comando deve terminar com o carácter final (*carriage return* – ‘CR’). Esse grupo de comando permitirá especificar e executar com mais detalhe, uma determinada operação desejada, como por exemplo:

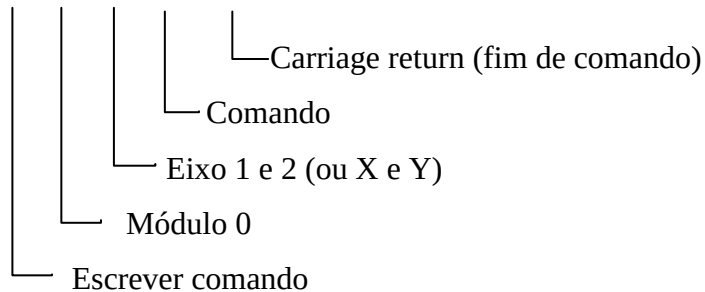
Exemplo 1: Comando “W01P0500<CR>”, onde se pretende (escrever comando ‘W’, endereçado ao módulo ‘0’, ordem apenas para o eixo ‘1’, onde se pretende executar o comando ‘P’ (tabela 3.3), ‘0500’ consiste em mandar o motor andar 500 passos).

W 0 1 P 0 5 0 0 <CR>



Exemplo 2: O comando “W03L<CR>”, onde se pretende (escrever comando ‘W’, endereçado ao módulo ‘0’, ordem dirigida aos dois eixos ‘3’, onde se pretende executar o comando ‘L’ energização de motor(es); (tabela 3.3).

W 0 3 L <CR>



O grupo de comando acaba com o símbolo de *carriage return* (ASCII 13). Os comandos serão executados sequencialmente, sendo que, em caso de se alterar algum parâmetro de funcionamento dos motores quando estes ainda se encontram em funcionamento, esse parâmetro será alterado apenas quando o motor parar.

Em caso de alguma mensagem recebida pelo indexer não estar de acordo com o protocolo, surgirá uma outra mensagem com o respectivo erro. Quando ocorrer mais do que um erro, o erro indicado será o primeiro. As mensagens de erro que se pretende implementar são as indicadas na tabela 3.2:

Tabela 3.2 - Mensagens de erro.

Mensagens de Erro	
ERRO	DESIGNAÇÃO
E0	Mensagem recebida terminada: Comando introduzido pelo utilizador
E1	Mensagem recebida terminada: Falta carácter final
E2	Mensagem recebida terminada: Tempo expirado
E3	Mensagem recebida terminada: Atingido número máximo de caracteres permitidos
E4	Mensagem recebida terminada: Erro de paridade
E5	Mensagem recebida terminada: Grupo de Comando Inválido
E6	Número de caracteres introduzido errado
E7	Funcionalidade Inexistente
E8	Atingido número máximo de coordenadas X
E9	Atingido número máximo de coordenadas Y

E10	Atingido número máximo de Coordenadas XY
E11	Valor de tempo de ciclo incorrecto

Os seguintes comandos serão os usados para o controlo do Indexer (tabela 3.3). Quando este (s7-200) for colocado em modo RUN, será feito num primeiro scan o *reset* a todas as variáveis de memória utilizadas, sendo definidos valores por defeito para o tempo de ciclo (frequência dos impulsos) e o número de passos a ser executado em cada motor.

Tabela 3.3 - Comandos ‘W’ (escrever comando) do indexer.

Valores por defeito: 15 ms (tempo de ciclo). 200 (número de passos). Estado inicial: Alimentação dos motores não activada. Tipo de passo <i>half-Step</i>. Rotação dos motores para a esquerda (CCW). -Exemplo de uma acção de comando enviada com sucesso para o PLC, em modo de comunicação ponto-a-ponto: grupo de comando “W03L+<CR>”, então PLC energiza motores e “responde”: “L+<CR>”. -Exemplo de uma acção de comando enviada sem sucesso para o PLC. grupo de comando “W03L” sem <i>carriage return</i> <CR>, depois de enviado este grupo de comando, o PLC espera alguns segundos pré-definidos, a ver se é enviada uma nova mensagem com <CR>, se tal não acontecer então PLC “responde”: “E2” - (Tempo expirado).	
Comandos de controlo W (write)	
Comando	Designação – exemplo de aplicação
...v____ ↵	(v)elocidade: escala do cycle time em (ms) e o respectivo valor pretendido com 4 caracteres (0000 até 9999). Exemplo: v0110 = tempo de ciclo 110 ms. nota: valores máximos impostos pelo PLC são: (1<ms<65536 ; 49<µs<65536).
...V____ ↵	(V)elocidade: escala do cycle time em (µS) e o respectivo valor pretendido com 4 caracteres (0000 até 9999). Exemplo: v0050 = tempo de ciclo 50 µS. nota: valores máximos impostos pelo PLC são: (1<ms<65536 ; 49<µs<65536).
...P____,____ ↵	(P)osição: Valor pretendido para as coordenadas x e y do elemento móvel, com 4 caracteres (0000 até 9999) para cada coordenada.

Exemplo: P0250,1020 = posição x e y (250 e 1020) correspondente.	
...D ↵	(D)ireita: Colocar motor a rodar para a direita (sentido horário).
...E ↵	(E)squerda: Colocar motor a rodar para a esquerda (sentido anti-horário).
...L ↵	(L)igar: Permite ligar a alimentação dos motores (energizar).
...A ↵	(A)pagar: Permite desligar a alimentação dos motores (não energizar).
...F ↵	(F)ull-step: Motores executam passos completos.
...H ↵	(H)all-step: Motores executam ½ passos.
...O ↵	(O)rigem: Desloca os dois motores até a posição de origem (0,0).
...G ↵	(G)o: Desloca os dois motores segundo os parâmetros até então definidos.
...R ↵	(R)ampa: Motores são accionados com rampas de aceleração.
...r ↵	(r)ampa: Motores são accionados sem rampas de aceleração.
...Z ↵	(Z)ero: Coordenadas absolutas, movimentação dos motores relativamente há posição de origem (0,0).
...I ↵	(I)ncremental: Coordenadas relativas, movimentação dos motores em relação há posição anterior. nota: posicionamento dos motores limitado pelo espaço da mesa de trabalho, no caso de ser dada uma ordem para o motores andar mais para além do limite possível deve ser enviada uma mensagem de imediato do Indexer para o controlador a informar da situação.
...S ↵	(S)top: Permite parar o(s) motor(es) imediatamente.

Depois de enviar um grupo de comando onde esteja contido um dos comandos da tabela 5, será enviada uma mensagem de volta, contendo dados acerca do estado do movimento dos dois motores.

Ao comando SX estará associado um byte, onde cada bit tem o significado descrito na tabela 3.4.

Tabela 3.4 - Byte associado ao comando SX, do grupo de comando R (Read).

Descrição da função READ “SX”							
MSB				LSB			
7				0			
A	B	C	D	E	F	G	H

Byte de status da mensagem a enviar (valor entre 00 e FF)

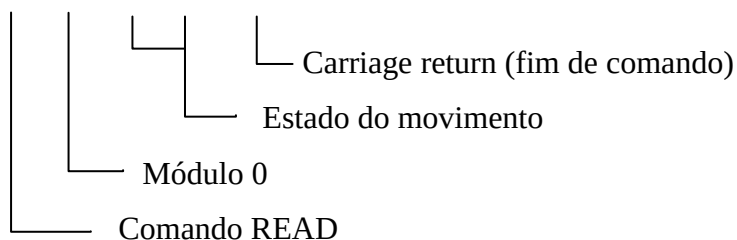
A: 1 = Motor em funcionamento ----- motor X (executar passos)
B: 1 = Sentido de rotação ----- motor X
C: 1 = Motor energizado ----- motor X
D: 1 = Tipo de passo ----- motor X
E: 1 = Motor em funcionamento ----- motor Y (executar passos)
F: 1 = Sentido de rotação ----- motor Y
G: 1 = Motor energizado ----- motor Y
H: 1 = Tipo de passo ----- motor Y

Tabela 3.5 - Comandos 'R' (Ler Status de comando) do indexer.

Comando	Designação – exemplo de aplicação
...SX ↵	Comando Read (SX): (STATUS dos motores)
...ST ↵	Comando Read (ST): STATUS velocidade dos passos uS OU ms)
...St ↵	Comando Read (St): STATUS Número de Passos

Exemplo 1: O comando “R0SX<CR>”, onde se pretende (ler o estado do movimento ‘R’, endereçado ao módulo ‘0’, onde se pretende executar o comando ‘ST’ (tabela 3.5), lê a velocidade actual dos passos).

R 0 S T <CR>



O comando que se deverá introduzir para seleccionar o modo de comunicação desejado (tabela 3.6), irá consistir numa mensagem de apenas três caracteres, os dois primeiros associados ao respectivo modo de comunicação, o terceiro diz respeito ao carácter de fim de comando <CR>.

Tabela 3.6 - Comandos para seleccionar o modo de comunicação do Indexer.

Comando	Designação – exemplo de aplicação
MS ↵	(M)aster/(S)lave: modo de comunicação. Exemplo: “MS+<CR>”.
PP ↵	(P)onto-a-(P)onto: modo de comunicação. Exemplo: “PP+<CR>”.

O grupo de comando da tabela 3.7 envia uma mensagem particular ao Indexer, para averiguar se nesse instante o módulo a qual é endereçada a mensagem, se encontra a comunicar com o controlador do sistema.

Tabela 3.7 - Grupo de comando para verificar estado da comunicação entre controladores.

Comando	Designação – exemplo de aplicação
W0? ↵	(?) Verificação do estado da comunicação. Exemplo: “W0?+<CR>”.

Nota: O símbolo ↵ representa a sequência do símbolo <CR>, com que todos os comandos são finalizados.

Capítulo 4

Desenvolvimento

Este capítulo examina aspetos fundamentais da programação em diagrama de escada (*ladder logic*). Também analisa particularidades da programação desenvolvida através do software da Siemens® STEP 7 MicroWIN SP5, bem como os recursos fundamentais desse mesmo software.

4.1 Software desenvolvido em relação ao modelo

4.1.1 Compreender o ciclo de varrimento (scan cycle) da CPU

A CPU do s7-200 é projectada para executar uma série de tarefas, incluindo o nosso programa, repetitivamente. Esta execução cíclica de tarefas é chamada de ciclo de varrimento. Durante o ciclo de varrimento ilustrado na figura 4.1, a CPU executa a maioria ou mesmo todas as seguintes tarefas: [16]

- Leitura das entradas
- Execução do programa
- Processamento de qualquer pedido de comunicação
- Execução da CPU de auto-teste de diagnóstico
- Escrever para as saídas

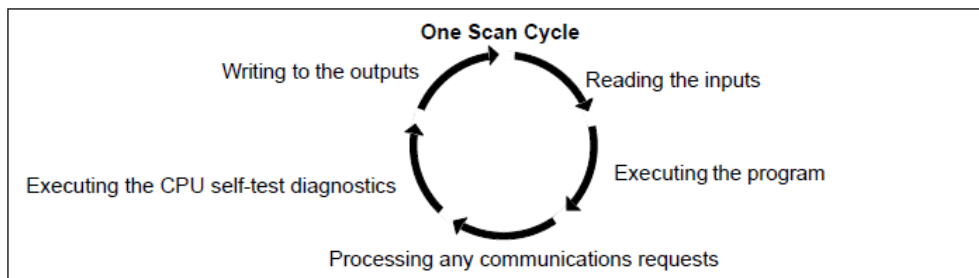


Figura 4.1 - Ciclo de varrimento da CPU do s7-200 [16]

Essas tarefas que são executadas durante o ciclo de varrimento dependem do modo de operação da CPU. A CPU S7-200 tem dois modos de operação, STOP e modo RUN. No que diz respeito ao ciclo de scan, no modo RUN o programa é executado, e no modo STOP não.

Programas para uma CPU S7-200 são construídos a partir de três elementos básicos: o principal programa, subrotinas (opcional), e rotinas de interrupção (opcional). O programa realizado no S7-200 está estruturado nos seguintes elementos organizacionais:

Programa principal (Main program): O programa principal é o corpo central do programa (conhecido como OB1), é onde se colocam as instruções que controlam a aplicação. As instruções no programa principal são executadas sequencialmente, uma vez por cada ciclo de scan da CPU.

Subrotinas: A subrotina é um conjunto opcional de instruções localizadas num bloco separado. Essas instruções são executadas somente quando a subrotina é chamada a partir do programa principal, rotina de interrupção ou mesmo de outra subrotina. Subrotinas são úteis nos casos em que se deseja executar uma função repetidamente.

Interrupções: As rotinas de interrupção são utilizadas para lidar com eventos pré-definidos, esses eventos são designados por eventos de interrupção. As rotinas de interrupção não são chamadas pelo programa principal, mas sim pelo sistema operacional do PLC. Pode-se dizer que uma rotina de interrupção é um conjunto opcional de instruções localizadas num bloco separado, que é executado somente quando um evento (interrupção) ocorre.

As várias componentes do software STEP 7 MicroWIN SP5 possíveis ao utilizador estão ilustradas na figura 4.2.

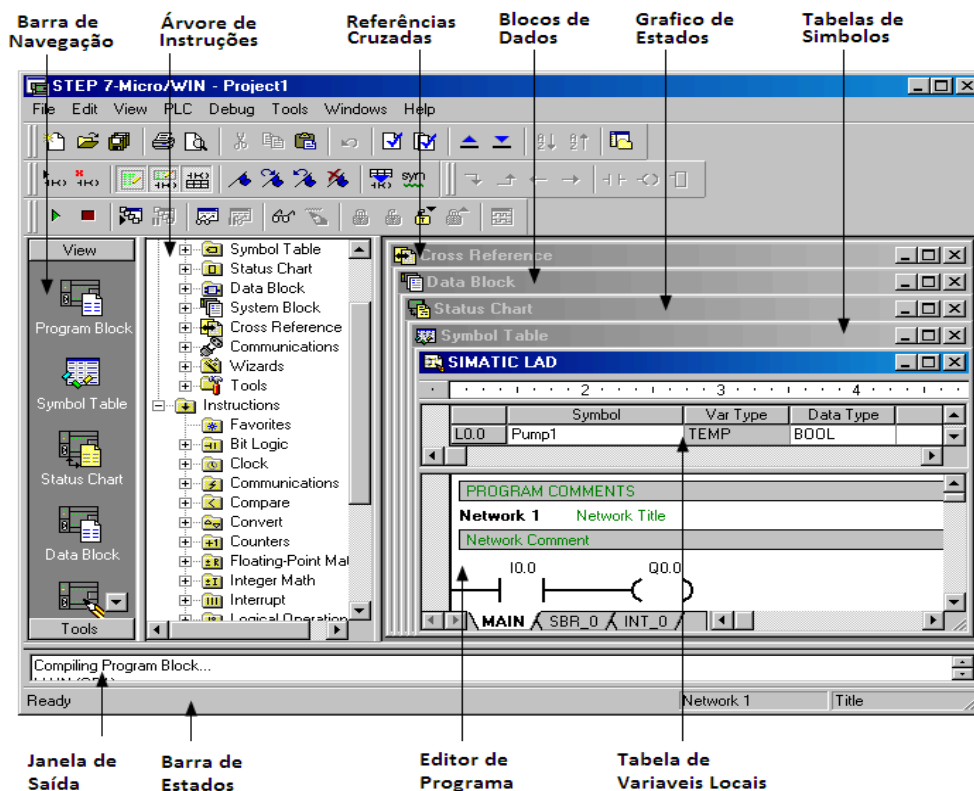


Figura 4.2 - Aplicações do software possíveis ao utilizador. [16]

4.1.2 Programação em diagrama de escada (Ladder)

O software sob o qual foi desenvolvida a solução, foi o STEP 7- MicroWIN versão 4.0. Este software permite a programação sob três diferentes editores de programação, programação em: listas de instruções 'STL', diagrama de blocos de funções 'FBD' e em diagrama de escada 'LAD'.

Diagrama de escada foi o editor de programação utilizado, e os pontos principais que levaram a seleccionar este editor foram:

- Lógica de programação mais indicada para os programadores com pouca experiência.
- A representação gráfica é muitas vezes a mais fácil de compreender, e é popular em todo o mundo.
- É possível usar o editor de STL para exibir um programa criado com o editor LAD.

O editor de diagrama de escada permite construir programas que se assemelham a um diagrama de um circuito elétrico. Programação em diagrama de escada é o método de eleição para muitos programadores de PLCs e pessoal de manutenção, é uma linguagem excelente para programadores principiantes. Os programas em diagramas de escada permitem a CPU emular o fluxo de corrente eléctrica a partir de uma fonte de energia, através de uma série de condições de entradas lógicas, que por sua vez habilitam condições de saídas lógicas. A lógica de programação é separada em redes de programação. O programa é executado consoante a programação ditar, uma rede de programação de cada vez, da esquerda para a direita e de cima para baixo. A CPU uma vez chegado ao fim do programa, ele começa todo de novo (novo ciclo), a partir do topo do programa. A figura 4.3 mostra um exemplo de um programa em diagrama de escada. [10, 16]

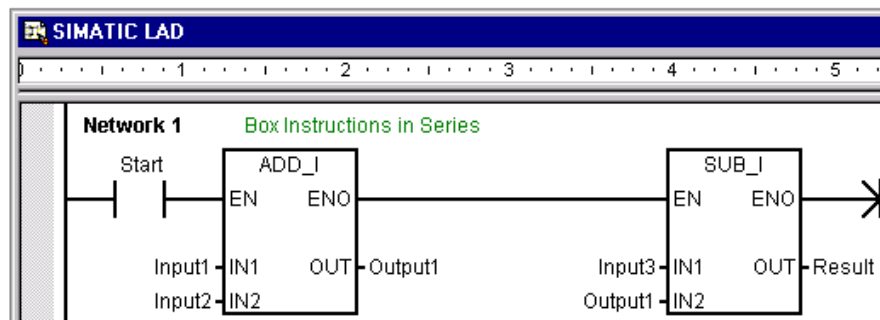


Figura 4.3 - Programa em diagrama de contactos. [16]

As várias instruções são representadas por símbolos gráficos e incluem três formas básicas.



Contactos - representa as condições de entrada lógica que podem ser impostas, por exemplo: interruptores, comutadores, variáveis internas e assim por diante.



Bobinas - geralmente representam os resultados das saídas lógicas, por exemplo: lâmpadas, motores, relés, variáveis interna de saída e assim por diante.



Blocos - representam instruções adicionais, tais como temporizadores, contadores, ou instruções de matemática.

4.1.3 Variáveis e Instruções relevantes

PTO

A instrução de saída de alta frequência (PTO) foi uma das instruções utilizadas mais importantes, Através desta instrução é possível gerar os impulsos necessários para controlar aos motores passo-a-passo. A CPU s7-200 suporta saídas de altas frequências. Os geradores PTO/PWM usam as saídas Q0.0 e Q0.1. Quando uma função PTO ou PWM está activa para uma das saídas Q0.0 ou Q0.1, o gerador PTO/PWM tem o controlo dessas saídas, e o uso normal dessas saídas é inibido (ver Tabela 4.4 - Controlo de registos do PTO/PWM). [16]

A função de trem de impulsos fornece uma onda quadrada (50% ciclo de trabalho) de saída, para um determinado número de impulsos e um tempo de ciclo especificado. O número de impulsos pode ser especificados de 1 a 4294967295 impulsos. O tempo de ciclo pode ser especificado em incrementos de microssegundos ou de milissegundos, os valores possíveis para o tempo de ciclo vão desde de 50 (μ S) até 65.535 (μ S) ou desde 2 (ms) até 65.535 (ms). Especificar qualquer número ímpar de microssegundos ou milissegundos (como 75 ms) pode causar alguma distorção no duty cycle.

A função de trem de impulso de saída (PTO) pode ser programada para produzir um trem de impulsos, ou pode ser programada para produzir um perfil de impulsos que consiste em gerar múltiplos trens de impulsos. No modo de perfil de impulsos, a função PTO pode ser programada para controlar um motor passo-a-passo através de sequências simples de rampas de aceleração e desaceleração, ou então sequências mais complexas. Um Perfil de impulsos é constituído por múltiplos segmentos (ver tabela 4.1). O número de segmentos pode ultrapassar os 255. A variável associada a função de geração de um trem de impulsos é a variável PLS.

Quando o trem de impulsos estiver concluído, a saída de um novo trem de impulsos começa imediatamente. Isto permite a continuidade entre subseqüentes trens de impulso de saída.

Tabela 4.1 - Perfil de uma operação PTO de múltiplos segmentos. [7]

Deslocamento de byte	Número de segmento	Descrição das entradas na tabela
0		Número de segmentos (1 até 255)
1	#1	Tempo de ciclo inicial (2 até 65535)
3		Tempo de ciclo delta por impulso (-32768 até 32767 unidade de base de tempo)
5		Número de impulsos (1 até 4294967295)
9	#2	Tempo de ciclo inicial (2 até 65535)
11		Tempo de ciclo delta por impulso (-32768 até 32767 unidade de base de tempo)
13		Número de impulsos (1 até 4294967295)
:	:	:
:	:	:

Desenvolver uma rampa de aceleração é uma das várias funcionalidades programadas disponíveis. Na figura 4.4 está ilustrado um exemplo de perfil de rampa de aceleração. Programada através da instrução PTO para um perfil de impulsos, para tal são determinados os valores necessários para gerar uma onda de saída que acelera um motor passo-a-passo, coloca o motor a funcionar uma velocidade constante, e, em seguida, desacelera o motor.

O objectivo das rampas é essencialmente permitir obter maiores velocidades de deslocamento dos motores sem que estes percam passos, nos períodos em que os motores não irão sofrer qualquer binário adicional (não está a ser executada uma operação de maquinagem).

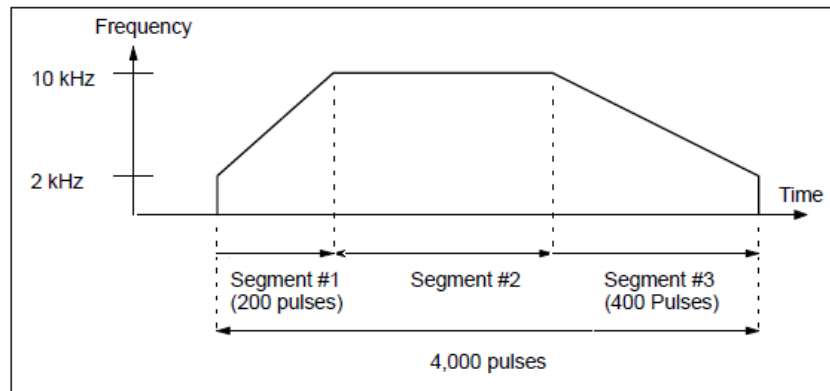


Figura 4.44 - Exemplo de rampa de aceleração desenvolvida para um MPP. [16]

O valor do número de impulsos total é o correspondente ao valor de passos pretendidos para a trajectória desejada, por outras palavras é o valor total dos impulsos dos três segmentos.

O valor dos passos para o primeiro segmento (aceleração), segundo segmento (velocidade constante) e terceiro segmento (desaceleração) corresponde a 5%, 85% e 10% do valor total de passos para a trajectória desejada, respectivamente. Foram testados nas rampas outros valores para o número de passos em cada segmento, mas os valores que ficaram definidos foram os que melhor se ajustaram aos testes de movimento realizados (número de passos e gama de velocidade aplicadas).

O tempo de ciclo para o primeiro segmento (aceleração) corresponde a um valor de 5 vezes o valor do tempo de ciclo desejado para a trajectória. O mesmo acontece para o tempo de ciclo do terceiro segmento (desaceleração) mas inversamente. Quanto ao segundo segmento (velocidade constante) o valor do tempo ciclo é o mesmo do tempo de ciclo pretendido para a trajectória.

A variação do tempo entre dois impulsos é designada por tempo de ciclo delta. No que diz respeito ao valor do tempo de ciclo delta para cada segmento, esse foi calculado através da fórmula representada na figura 4.5.

Todos os valores e cálculos necessários para reproduzir uma rampa de aceleração são realizados automaticamente, através das linhas de código criadas para o efeito, após se activar a respectiva funcionalidade.

$\text{delta cycle time for a given segment} = \text{ECT} - \text{ICT} / Q$ where, ECT = ending cycle time for this segment ICT = initial cycle time for this segment Q = quantity of pulses in this segment

Figura 5 – Fórmula utilizada para determinar o tempo de ciclo delta de um segmento. [16]

Outras instruções relevantes

Foram também utilizadas inúmeras instruções de comparação, matemáticas, conversão, interrupção, comunicação etc.

As instruções de comparação permitem essencialmente comparar dois valores, a comparação pode ser entre valores de duas variáveis diferentes, ou então comparar o valor de uma variável com um valor fixo pré-definido, foram utilizadas muitas vezes para comparar os valores dos caracteres contidos nas mensagens recebidas, com o valor estabelecido nas instruções de programação de acordo com os comandos do protocolo de comunicação.

Quanto as instruções matemáticas foram utilizadas aquelas que possibilitam efectuar operações matemáticas básicas: soma “ADD”, subtracção “SUB”, divisão “DIV”, multiplicação “MUL” e raízes quadradas “SQRT”. Estas operações matemáticas, entre outras tarefas foram utilizadas para se conseguir colocar os dois motores a desempenhar trajetórias lineares, e rampas de aceleração.

Já como referido varias vezes anteriormente, o indexer permite desenvolver trajetórias lineares no plano XY. Tal façanha é conseguida essencialmente através da utilização de instruções matemáticas num raciocínio que tem como base o teorema de Pitágoras. O cálculo e a lógica por de trás da operação que permite ao indexer desenvolver trajetórias lineares é o seguinte:

Após se introduzir no indexer o valor correspondente a velocidade a que se pretende executar a trajetória, o valor desejado para a coordenada X (número de passos para o motor X), e ainda o valor desejado para a coordenada Y (número de passos para o motor Y). Esses valores (tal como muitos outros) ficaram armazenados numa determinada área da memória do PLC, e serão utilizados para se efectuar os cálculos

que nos permitiram, não só calcular o valor do número de passos da trajectória, como também o valor da velocidade a que cada motor terá de se deslocar de maneira a que os movimentos dos dois motores ocorram sincronizadamente, uma vez que, no caso do valor das coordenadas X e Y serem diferentes, os motores para começarem e acabarem o seu movimento em simultâneo, é necessário calcular a velocidade adequada para cada eixo, tendo em conta o valor do número de passo que se introduziu. Caso esta situação não se verificasse, a trajectória que os motores iriam desenvolver seria bastante irregular.

Na figura 4.6 encontra-se um exemplo de uma trajectória desenvolvida pelo indexer. Onde se pretende posicionar o elemento móvel nas coordenadas (500,1000) com um tempo de passo de 15 (ms). Para tal é enviada uma ordem ao motor X para este “andar” 500 passos, e ao motor Y enviada uma ordem para “andar” 1000 passos.

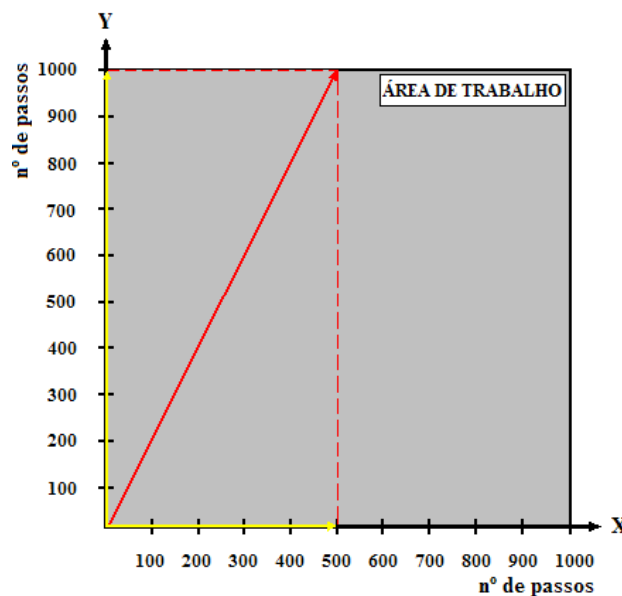


Figura 4.6 – Exemplo de trajectória rectilínea.

Neste exemplo estão representados os cálculos que o indexer realiza a fim de obter o número de passos da trajectória, e o valor de tempo de passo (tempo entre dois impulsos) para cada um dos eixos. Estes são os seguintes:

Legenda:

- a* - Número de passos pretendido para o motor x.
- b* - Número de passos pretendido para o motor y.
- c* - Número de passos da trajectória.
- d* - Tempo de passo pretendido para trajectória.
- e* - Tempo de duração da trajectória.

Através do teorema de Pitágoras e sabendo o número de passos correspondentes as coordenadas X e Y, é calculado o número equivalente de passos da trajectória.

- *O número equivalentes de passos da trajectória:*

$$\frac{\sqrt{a^2 + b^2}}{c} \quad (\text{passos})$$

Sabendo o número de passos equivalentes da trajectória e o tempo de passo para a mesma, podemos obter o tempo de duração da trajectória.

- *Tempo de duração da trajectória:*

O tempo de passo para cada um dos eixos é calculado dividindo o tempo de duração da trajectória pelo número de passos para cada um dos eixos.

- *Tempo de passo para o motor do eixo Y e motor X, respectivamente:*

$$\frac{e}{c} \quad \frac{e}{c}$$

As instruções de conversão permitem converter um valor de uma variável de um determinado tipo, para um valor de uma variável de outro tipo. Como por exemplo a instrução “B_I” esta instrução converte o valor de um byte (8 bits) para um valor de um inteiro (Word_16 bits) e coloca o resultado na variável de saída especificada. São usadas para manipular os valores das variáveis, de maneira a permitir compatibilizar e relacionar adequadamente diferentes instruções e/ou tarefas.

Todas as instruções de interrupção e comunicação utilizadas, foram convenientemente combinadas para se estruturar o programa da melhor maneira possível e também para se obter as comunicações pretendidas entre controladores.

Uma outra instrução utilizada importante, é a instrução *saltar*. A instrução *saltar* é normalmente usada para saltar de uma secção ‘origem’ (rede de programação) do programa para uma outra secção ‘destino’ pretendida, evitando-se assim, passar por determinadas linhas de programação indesejáveis. Isso é conseguido, atribuindo a mesma referência de identificação as duas instruções que permitem tal operação. A instrução *saltar* deve ser aplicada na mesma janela de programação (janela principal, subrotina ou interrupção), isto é, não pode saltar por exemplo de uma subrotina para uma outra subrotina.

Variáveis especiais

Existem muitas variáveis especiais para funções particulares/dedicadas disponíveis ao programador, onde cada uma delas desempenha uma função específica. As variáveis especiais tal como muitas instruções desenvolvidas pelos fabricantes, simplificam muito a programação pois permitem executar funções que se outro modo levariam muitas horas de programação, e tornariam o programa muito extenso e confuso. Variáveis especiais permitem também recolher dados importantes ao utilizador, nomeadamente no âmbito das comunicações do PLC. As variáveis especiais são bytes de memória, como por exemplo Special Memory Byte 0 ou SMB0 (SM0.0 - SM0.7), que fornecem oito bits que são actualizados pela CPU s7-200 no fim de cada ciclo de varrimento. Um programa pode ler o estado desses bits e tomar decisões baseadas no seu valor. As variáveis especiais (SM) utilizadas no programa do indexer estão listadas nas tabelas 4.2, 4.3, 4.4 e 4.5.

Tabela 4.2 – Variáveis especiais. [16]

Indexer s7-200	SM (special memory)	
Designação do símbolo	Endereço	Função
Sempre_a_1	SM0.0	Variável sempre a "1".
Primeiro_Scan	SM0.1	Variável a "1" só no primeiro varrimento.
Transm_Ok	SM4.5	Esta variável vai a "1" apenas quando a instrução transmitir (XMT) não está a enviar nada.
P0_Start_Rcv	SM86.5	Variável vai a "1" quando uma mensagem é recebida pela porta 0.

SMB30 controla a comunicação Freeport para porta 0. Pode-se ler e escrever em SMB30. Conforme descrito na Tabela 4.3, esses bytes configuram a porta de comunicação para o modo Freeport, e proporcionam a seleção de outro qualquer protocolo tabelado para além do protocolo Freeport.

Tabela 4.3 - Byte da memória especial SMB30 (Controlo de Registos Freeport) [16]

Porta 0	Descrição								
Formato de SMB30	MSB 7 LSB 0 Byte de controlo do modo Freeport <table><tr><td>p</td><td>p</td><td>D</td><td>b</td><td>b</td><td>b</td><td>m</td><td>m</td></tr></table>	p	p	D	b	b	b	m	m
p	p	D	b	b	b	m	m		
SM30.6 e SM30.7	pp: <i>seleccionar paridade</i> 00 = sem paridade 01 = paridade par 10 = sem paridade 11 = paridade impar								
SM30.5	d: <i>bits de dados por cada carácter</i> 0 = 8 bits por caracteres 1 = 7 bits por caracteres								
SM30.2 até SM30.4	bbb: <i>cadencia de transmissão (baud rate) no modo Freeport</i> 000 = 38,400 baud 001 = 19,200 baud 010 = 9,600 baud 011 = 4,800 baud 100 = 2,400 baud 101 = 1,200 baud 110 = 600 baud								

	111 = 300 baud
SM30.0 até SM30.1	mm: <i>Seleção de protocolo</i> 00 = protocolo de interface ponto-a-ponto (modo PPI/escravo) 01 = protocolo Freeport 10 = modo PPI/mestre 11 = reservado

Tabela 4.4 - Controlo de registos do PTO/PWM. [16]

Q0.0	Q0.1	Bits de Controlo
SM67.0	SM77.0	PTO/PWM actualização do valor do tempo de ciclo: 0 = não actualiza; 1 = actualiza tempo de ciclo
SM67.1	SM77.1	PWM actualiza valor da amplitude de impulso: 0 = não actualiza; 1 = actualiza amplitude de impulso
SM67.2	SM77.2	PTO actualiza valor do número de impulsos: 0 = não actualiza; 1 = actualiza número de impulsos
SM67.3	SM77.3	PTO/PWM selecção da base de tempo: 0 = 1 ms; 1 = 1ms
SM67.4	SM77.4	Método de actualização: 0 = actualização assíncrona; 1 = actualização síncrona
SM67.5	SM77.5	Modo de operação do PTO: 0 = operação de segmento singular; 1 = operação de múltiplos segmentos
SM67.6	SM77.6	Seleccção do modo PTO/PWM 0 = selecção de modo PTO; 1 = selecção do modo PWM
SM67.7	SM77.7	Activação PTO/PWM 0 = desactivação PTO/PWM; 1 = activação PTO/PWM
Q0.0	Q0.1	Outros registos PTO/PWM
SMW68	SMW78	Valor do tempo de ciclo para PTO/PWM (intervalo: 2 até 65535)
SMD72	SMD82	Valor do número de impulsos para PTO (intervalo: 1 até 4294967295)
SMB166	SMB176	Número de segmentos em progresso (usado apenas em operação PTO múltiplos segmentos)
SMW168	SMW178	Localização inicial da tabela de perfil (usado apenas em operação PTO múltiplos segmentos)

4.1.4 Programação Desenvolvida

No texto que se segue serão descritas as partes principais da programação desenvolvida.

Janela principal (MAIN)

Network 2 - Inicialização (só executada no 1º Ciclo: figura 4.7)

- Reset das Saídas Q0.0 a Q0.5, a serem utilizadas para:
 - Q0.0 geração do sinal de alta frequência (motor X);
 - Q0.1 geração do sinal de alta frequência (motor Y);
 - Q0.2 definir sentido de rotação (motor X);
 - Q0.3 definir sentido de rotação (motor Y);
 - Q0.4 definir o tipo de passo (motor X);
 - Q0.5 definir o tipo de passo (motor Y);
- Set das Saídas Q0.6 e Q0.7 para que os motores inicialmente encontrem-se não-energizados.
- Reset de variáveis internas (M2.0 a M7.0 excepto M6.0), para inicializar o programa.
- Inicialização e valores por defeito para PTO 0:
 - Em VB0 -> SMB67 e SMB77 - Activação de PTO0 e PTO1 como "single segment", base de tempo de 1 ms e autorização para actualização dos valores dos impulsos e tempo de ciclo.
 - Em VW1 -> SMW68 e SMW78 - Tempo de Ciclo: 15 (ms).
- Move-se o valor +4000 para a variável SM168 e o valor de 4050 para a variável SM178 para definir que os valores para o perfil de cada uma das rampas de aceleração começam a partir destes valores.
- Criar tabela a partir da variável VW800 com 20 linhas de espaço.
- Fazer reset aos valores que se encontram na variável VW900.
- Definir o valor das variáveis a partir das quais vão estar associadas as rampas de aceleração do motor X (PTO 0) e motor Y (PTO1).

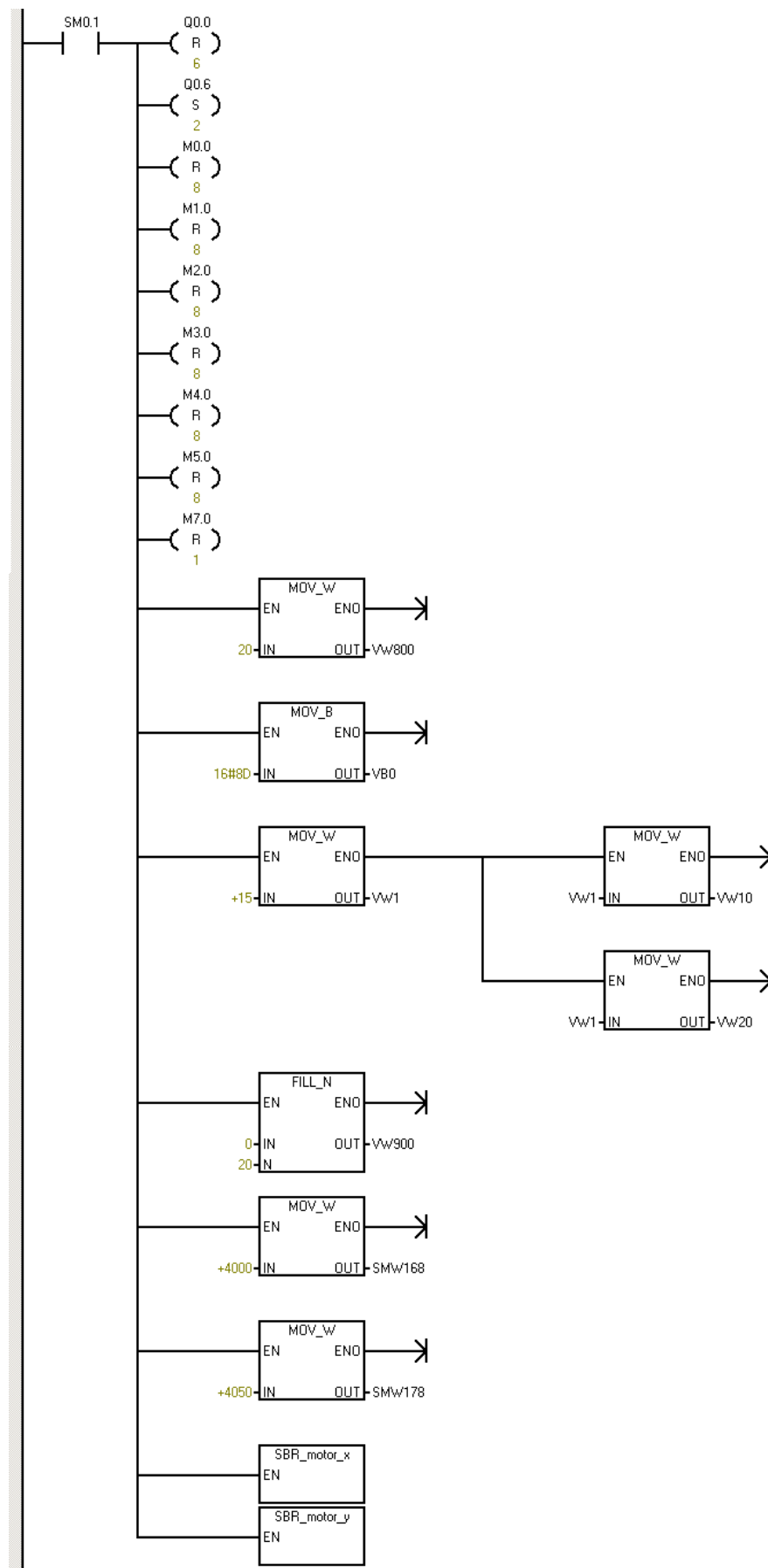


Figura 4.7 – Inicialização do programa.

Network 3 - Ordem de geração do trem de impulsos para o motor do eixo x (figura 4.8)

- Ordem de geração de impulsos para o motor X, a ordem só é válida com o motor parado (SM 66.7 =1).
- Os impulsos surgem na saída Q0.0,
- Os Impulsos são gerados pela transição de 0 para 1 das flags M0.0. e M0.1.
- Restrições:
 - M5.5 vai a 1 quando pretende-se que o motor X não seja actuado.
 - As ordens de movimento durante a geração de impulsos são ignoradas.
 - Número de passos máximo definido para 9998.
 - As instruções de comparação do valor da variável VD30 com o valor da variável VD560 estão associadas aos limites impostos para o número de passos da funcionalidade coordenadas absolutas (Network 21).

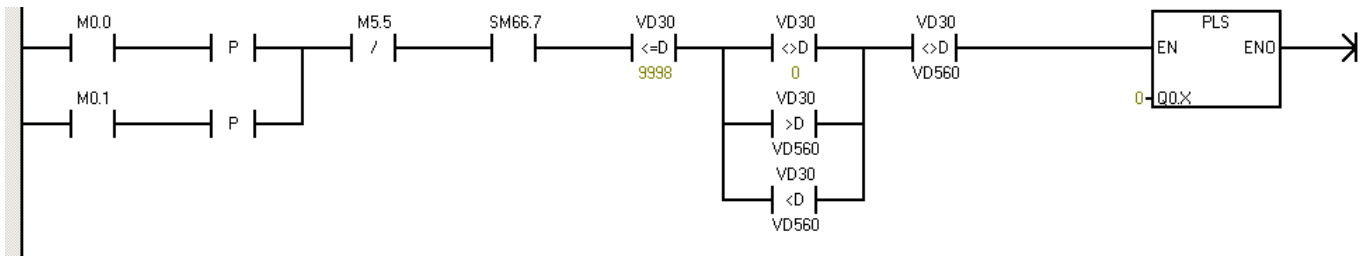


Figura 4.86 – Ordem de geração do trem de impulsos para o motor do eixo x.

Network 9 - Procura do comando introduzido pelo utilizador (figura 4.9)

- A procura é iniciada apenas se a ordem for endereçada ao módulo 0 => HEX (16#30) e se na mensagem recebida chegar o carácter final <CR> =>16#OD.
- A instrução MOVE_W neste caso é utilizada para fazer reset ao valor em INDX.
- O carácter a procurar na tabela é o valor contido na variavel VB2004 (receiver_buffer); é convertido para uma Word VW2040.
- É chamada a subrotina SBR_Fun onde se encontram todas as variáveis que vão afectar cada uma das funcionalidades programadas.

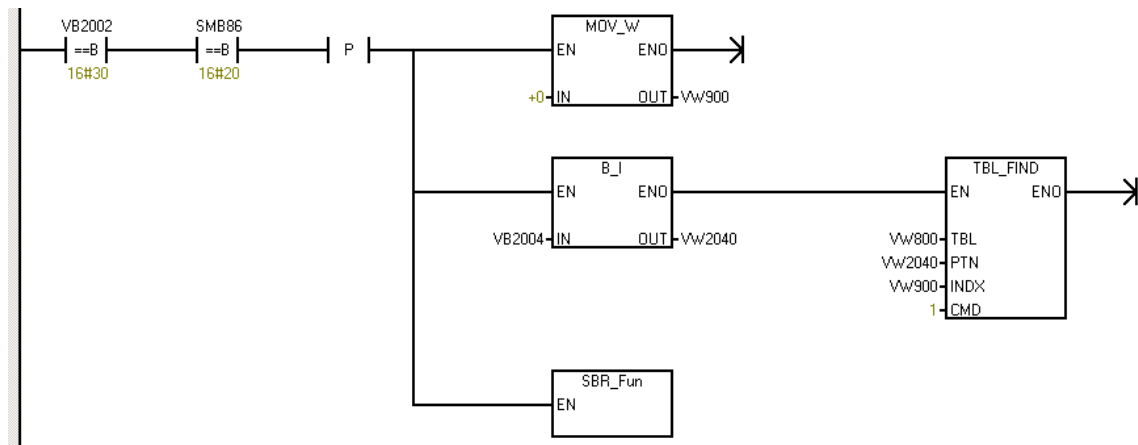


Figura 4.97 – Procura do comando introduzido pelo utilizador.

Network 10 - Sentido de rotação do motor X: (figura 4.10)

(Q0.2 = 1 => ccw; Q0.2 = 0 => cw)

- A definição só é valida (tem efeito) com o motor parado (SM 66.7 = 1).
- A definição é feita pelas flags M0.1. e M0.2.
- As flags efectam a saída Q0.2.

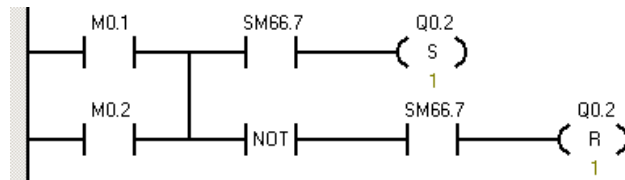


Figura 8 – Sentido de rotação do motor X.

Network 14 - Energização Motor X (figura 4.11)

(Q06 = 0) / Desenergização do andar de potência do drive Motor X (Q0.6 = 1)

- A definição só é actualizada com o motor parado (SM 66.7 = 1).
- A definição é feita pela flag M0.6.
- A flag afecta a saída Q0.6.

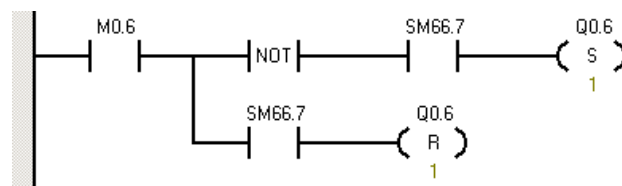


Figura 4.11 – Energização Motor X.

Network 17 – Forçar a paragem da geração de impulsos do motor X (figura 4.12)

- Para alterar o controlo PTO/PWM para a saída Q0.0, move-se o valor hexadecimal DB (binário 11011011) para byte de memória especial SMB67. Este permite a actualização do tempo de ciclo e a actualização do valor de largura de pulso, define a base de tempo de 1 ms, selecciona o modo PWM, e activa a saída Q0.0.
- Com o modo PWM seleccionado a saída Q0.0 pode ser interrompida. A largura de impulso é reduzida a 0, suprimindo o sinal da saída.

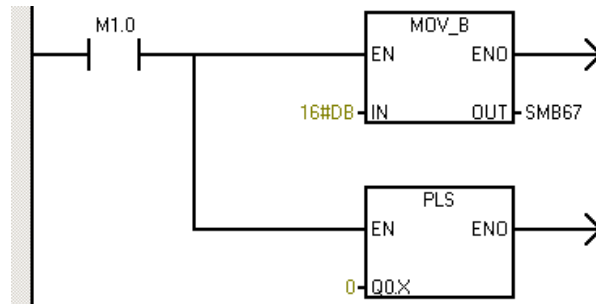


Figura 4.12 – Forçar a paragem da geração de impulsos do motor X.

Network 18 - Deslocamento para a origem (figura 4.13)

- Move-se o valor 3000 para as variáveis VD30 e VD40, de modo a definir um número de passos suficientes para colocar o elemento terminal na posição de origem (0,0).
- Move-se o valor 15 para as variáveis VW10 e VW20 de modo a definir um tempo de ciclo baixo (velocidade baixa).
- Move-se o valor hexadecimal 16#8D para a variável VB0 de modo definir o PTO0 e PTO1 como "single segment", base de tempo de 1 ms.
- Reset as variáveis M3.0 / M3.5 e M4.0 / M4.5 de modo a desactivar as rampas e o modo de coordenadas absolutas de cada motor respectivamente.

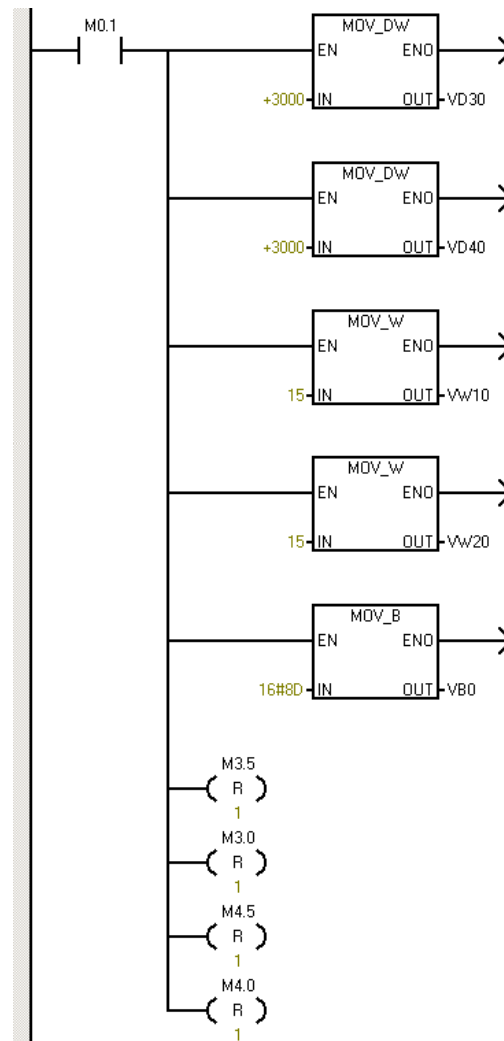


Figura 4.13 – Deslocamento para a origem.

Network 19 - Descrever uma trajetória linear (figura 4.14)

- O objectivo é pegar no valor do número de passos que se pretende executar segundo X, Y e fazer descrever com esses dois valores uma trajetória.
- Esta network está constantemente a ser executada em cada scan.
- O valor de passos pretendido para o eixo X, é fixado na variavel VD30.
- O valor de passos pretendido para o eixo Y, é fixado na variavel VD40.
- Através de instruções matemáticas simples, é calculado o teorema de Pitágoras com os valores do número de passos pretendidos para cada um dos eixos (VD30, VD40), obtém-se deste modo o valor do número de passos da trajetória, depois multiplicando esse valor pelo valor do tempo de ciclo definido para a trajetória (VW1), e de seguida dividindo o resultado pelo valor do número de passos pretendido para cada eixo, obtemos o valor arredondado do tempo de ciclo específico para cada eixo (VW10 e VW20).

- No final teremos os valores dos tempos de ciclo a serem endereçados aos respectivos motores de maneira a permitir desempenhar uma trajetória linear.

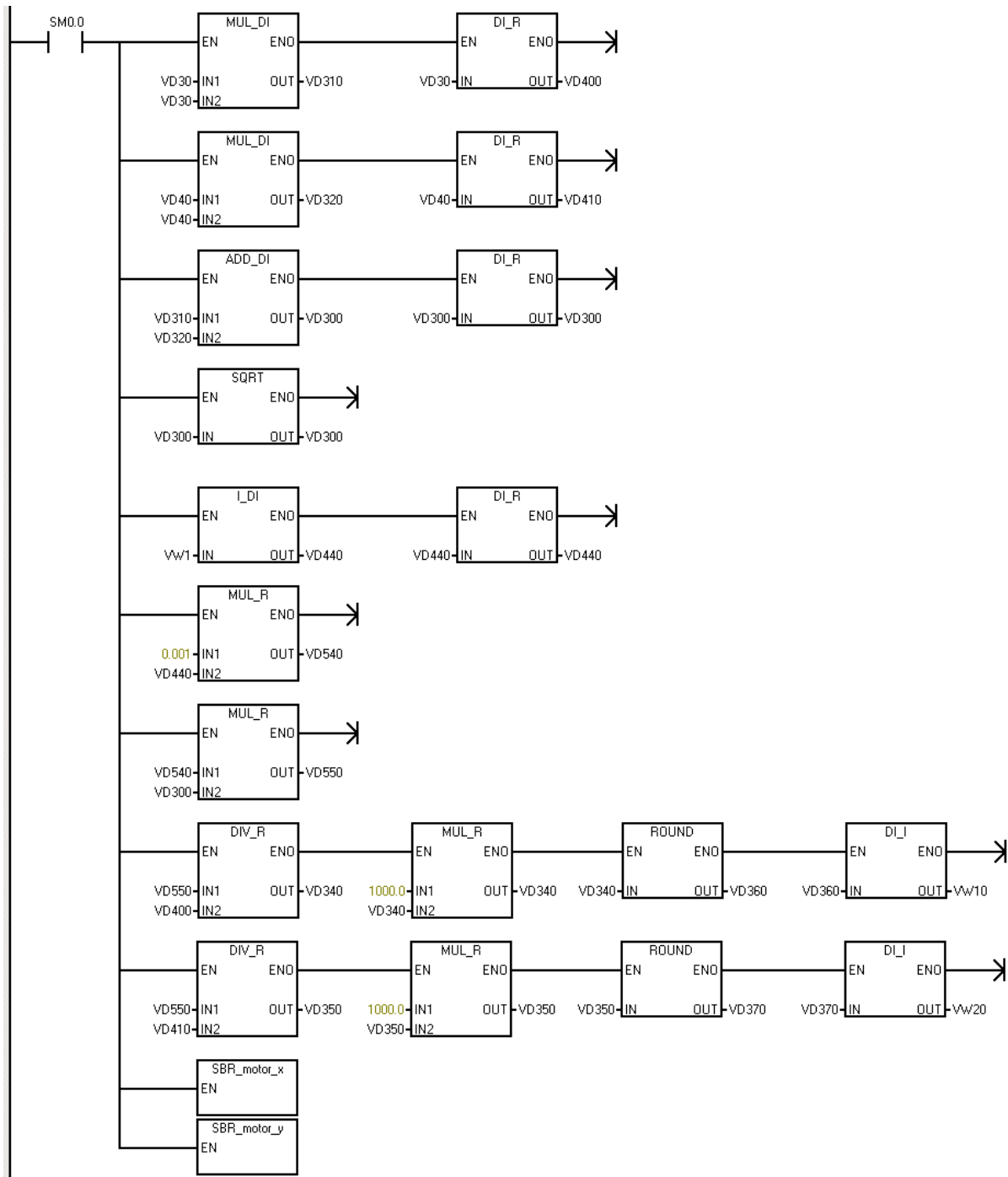


Figura 4.14 – Descrever uma trajetória linear.

Network 20 - Rampa de aceleração do motor X (figura 4.15)

- Variável M3.0 vai a "1" quando for enviada a ordem de comando "R" e a "0" quando a ordem de comando for "r".
- Os valores usados para gerar a rampa de aceleração desejada:

Endereço Port 0	Valor	Descrição
VB4000	3	(numero total de segmentos)
VW4001	500	(tempo de ciclo inicial - segmento número 1)
VW4003	-2	(tempo de ciclo delta - segmento número 1)
VD4005	200	(número de impulsos - segmento número 1)
VW4009	100	(tempo de ciclo inicial - segmento número 2)
VW4011	0	(tempo de ciclo delta - segmento número 2)
VD4013	3400	(número de impulsos - segmento número 2)
VW4017	100	(tempo de ciclo inicial - segmento número 3)
VW4019	1	(tempo de ciclo delta - segmento número 3)
VD4021	400	(número de impulsos - segmento número 3)

Descrição linha a linha de todos os cálculos utilizados:

- 1ª Linha de comando: números total de segmentos com que a rampa vai ser constituída, neste caso 3 segmentos, valor colocado na variável VB4000.
- 2ª Linha de comando: definida a velocidade inicial como sendo 5 vezes superior á velocidade pretendida para realizar a trajetória, valor colocado na variavel VW4001.
- 3ª Linha de comando: procedesse ao cálculo do delta cycle time para este segmento (1), isto é, a quantidade de aceleração que vai ser incrementada em cada impulso até atingir a velocidade, frequência ou cycle time pretendido, todas estas designações são diferentes modos de ver a mesma coisa, valor colocado na variável VW4003
- A fórmula calculada encontra-se no manual do PLC cap. 9-54, ($|ECT - ICT| / Q$):
- ECT = cycle time final deste segmento.
- ICT = cycle time inicial deste segmento.
- Q = quantidade de impulsos deste segmento.
- VW1 = cycle time definido.
- VD30 = número de impulsos ou passos totais definidos para este eixo.
- 4ª Linha de comando: é definido que o número de passos para este segmento é de 1/20 do número de passo totais.
- (exemplo: para um número de passos totais VD30 = 4000, o número de passos deste segmento vão ser de $200 = 1 * 4000 / 20$), valor colocado na variável VW4005.
- 5ª Linha de comando: definida a velocidade deste segmento (2) como sendo o valor pretendido de cycle time (VW1), valor colocado na variável VW4009.

6ª Linha de comando: valor do delta cycle time deste segmento nulo, valor colocado na variável VW4011.

7ª Linha de comando: é definido que o número de passos para este segmento é de 17/20 do número de passo totais.

(exemplo: para um numero de passos totais VD30 = 4000, o número de passos deste segmento vão ser de 3400= $17 \cdot 4000 / 20$), valor colocado na variável VW4013.

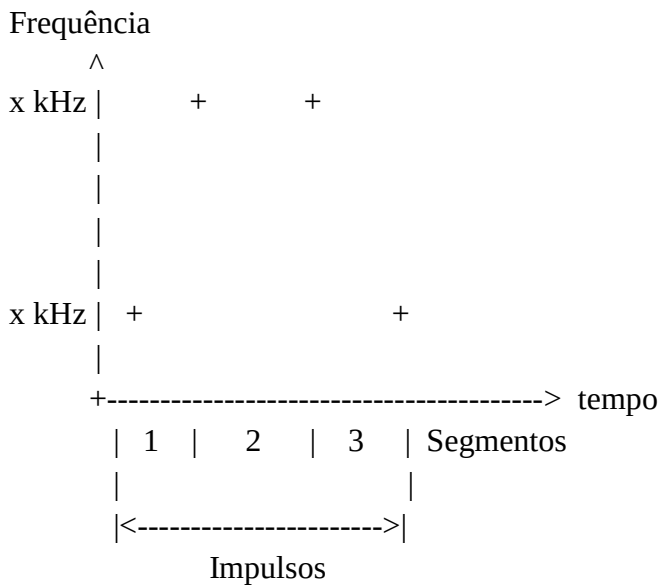
8ª Linha de comando: definida a velocidade inicial deste segmento (3) como sendo igual ao valor do cycle time anterior, valor colocado na variável VW4017.

9ª Linha de comando: procedesse ao cálculo do delta cycle time para este segmento (3), isto é, a quantidade de desaceleração que vai ser incrementada em cada impulso até atingir o cycle time pretendido, valor colocado na variável VW4019.

10ª Linha de comando: é definido que o número de passos para este segmento é de 2/20 do número de passo totais.

(exemplo: para um numero de passos totais VD30 = 4000, o número de passos deste segmento vão ser de 400= $2 \cdot 4000 / 20$), valor colocado na variável VW4021.

O seguinte diagrama representa o perfil da rampa:



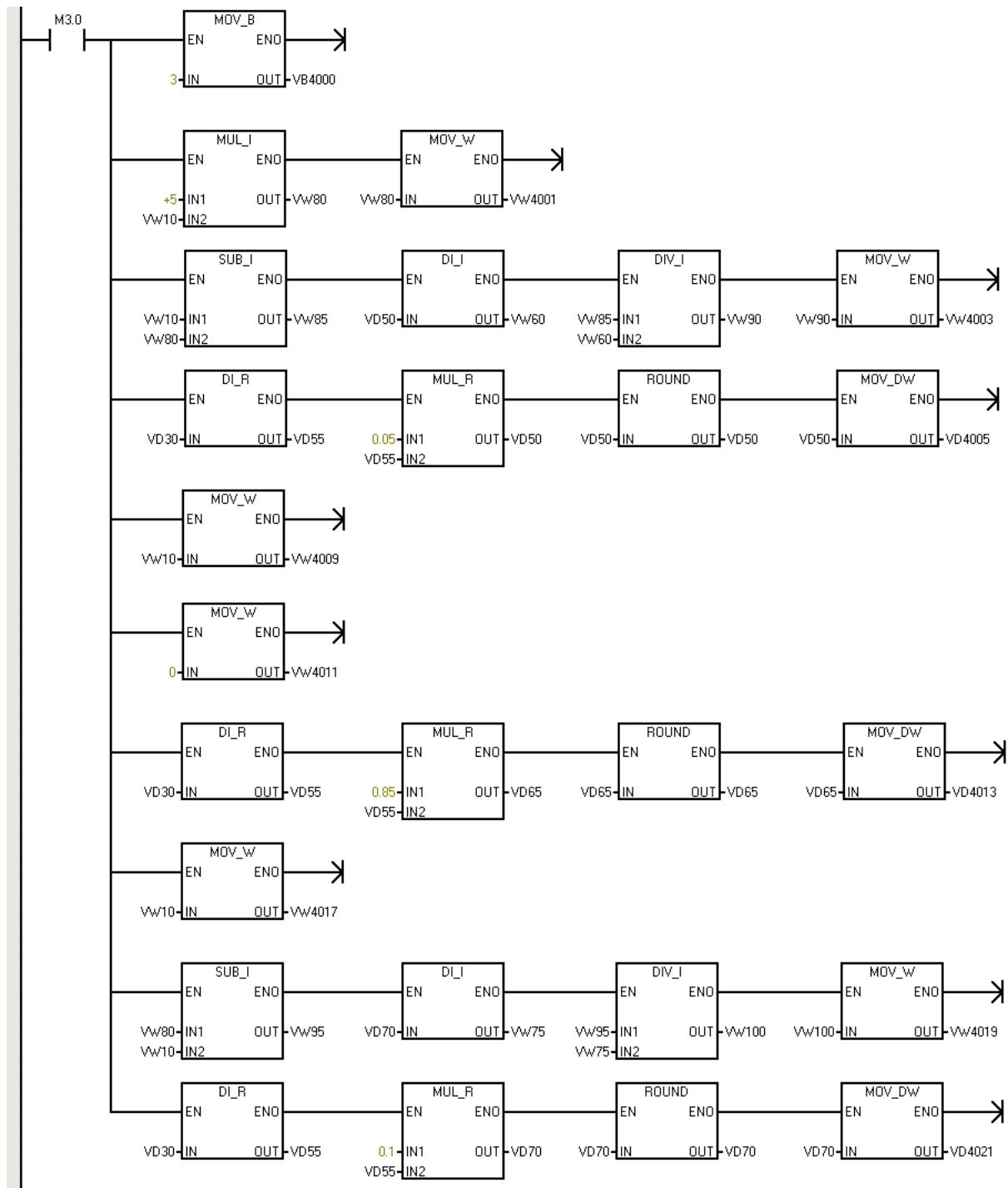
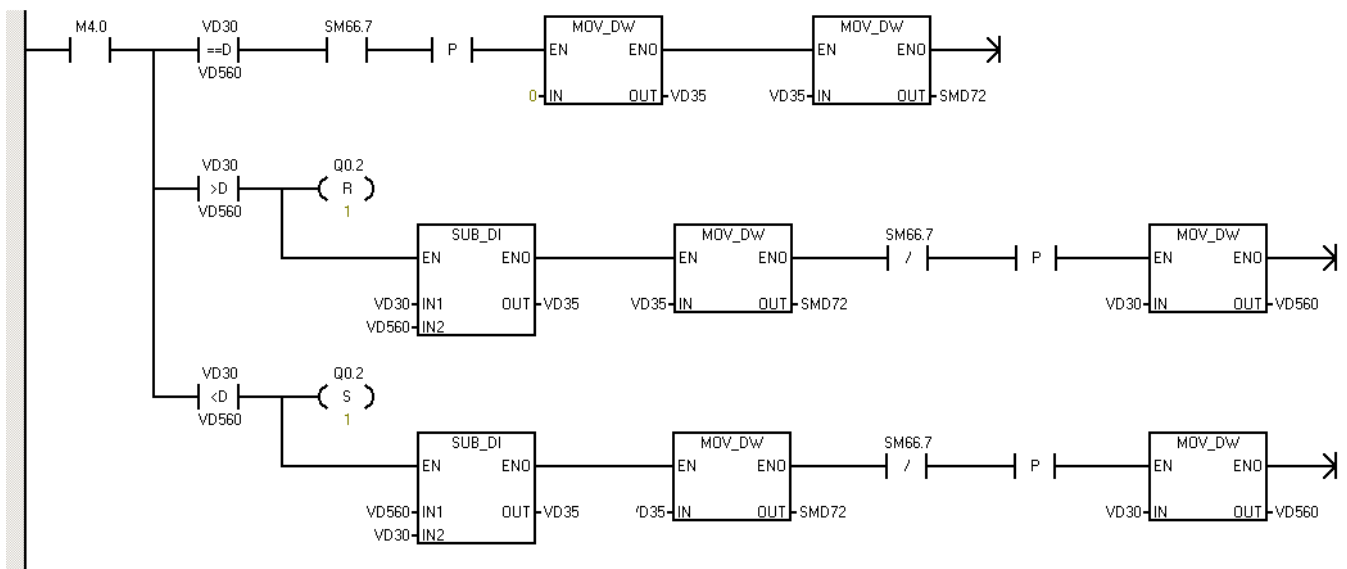


Figura 4.15 – Rampa de aceleração do motor X.

Network 22 - Coordenadas Absolutas motor X (figura 4.16)

- Variável M4.0 vai a "1" quando for enviada a ordem de comando "Z" e a "0" quando a ordem de comando for "I".
- São efectuadas três comparações entre, o número de passos pretendidos para o motor X (VD30) e o valor contido na variável VD560 (actualização do valor absoluto de passos).
- Quando o resultado da comparação das duas variáveis é:
 - Uma igualdade, o motor não se desloca.
 - Uma superioridade da variável VD30, coloca o motor a rodar para a esquerda até à coordenada equivalente ao número de passos definido.
 - Caso se verifique uma inferioridade da variável VD30, o motor passa a rodar para a direita, também até à coordenada equivalente ao número de passos definido.
- O valor contido na variável VD35 corresponde ao número de passos necessários, para ir da posição anterior para a nova posição.
- O valor da variável VD570 só é actualizado com o motor em andamento (SM 66.7 =1).

**Figura 4.16 – Coordenadas absolutas motor X.**

Network 24 – Actualização de parâmetros do motor X (figura 4.17)

- Esta rotina está sempre a ser chamada desde que o motor esteja parado.
- Permite a redefinição do período e número de passos, de acordo com os valores armazenados nas respectivas memórias M.

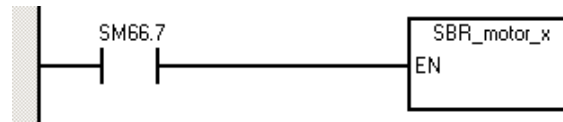


Figura 4.17 – Actualização de parâmetros do motor X.

Network 26 - Inicialização de comunicação (só executada no 1º Ciclo: figura 4.18)

- Chama a subrotina SBR_0 para inicializar os parâmetros de comunicação via porta 0.
- Chama a subrotina SBR_1 para inicializar os parâmetros das mensagens a receber via porta 0.
- Associa a interrupção evento 9 (Port 0: Transmit complete) há interrupção INT_0 para activar a INT_0, quando a transmissão é completa.
- Associa a interrupção evento 23 (Port 0: Receive Message complete) há interrupção INT_1 para activar a INT_1, quando a recepção é completa.
- Activa todos os eventos (interrupções).
- Move a constante 0 para a memória MB0 para "limpar" o byte de erro.
- Move a constante 0 para o byte da variavel de memória VB1, para inicializar o sub-estado de erro em 0.
- Move a constante 0 para o byte da variavel de memória VB0, para inicializar o estado de comunicação em 0.
- Inicialmente o modo de comunicação está definido como sendo mestre/escravo, daí colocar a instrução RCV pronta para receber mensagens para o buffer de recepção VB2000.

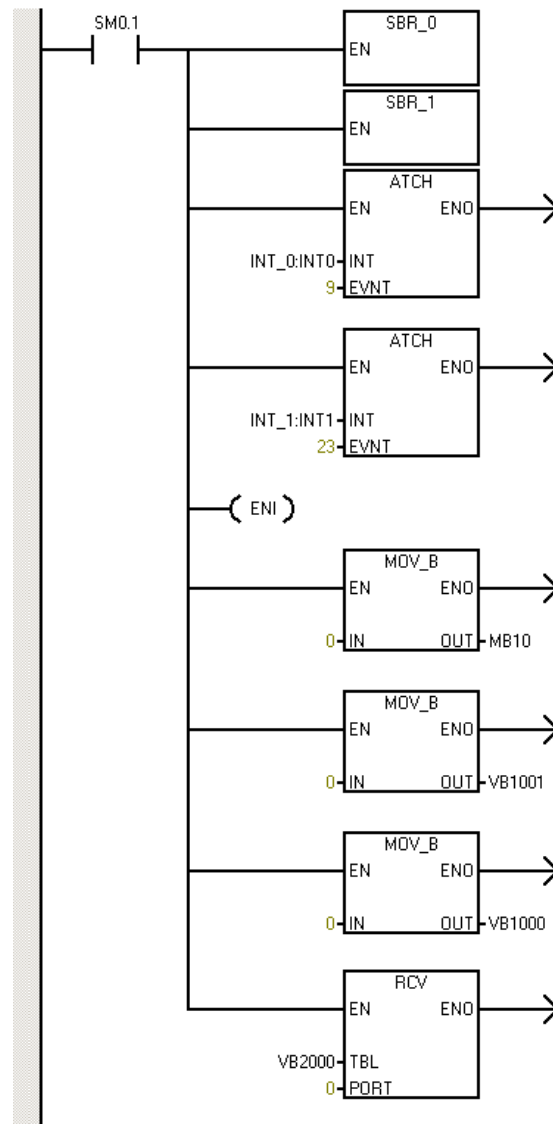


Figura 4.18 – Inicialização de comunicação.

Network 33 - Byte de Status dos motores (figura 4.19)

- Network sempre activa, percorrida em cada scan do PLC.
- No byte da variável de memória M6 fica armazenado o estado lógico de algumas variáveis de interesse.
- Indicação do Byte:
 - 0: motor x com tipo de passo em Full-Step
 - 1: motor x energizado;
 - 2: motor x com sentido de rotação para a direita;
 - 3: motor x em funcionamento;
 - 4: motor y com tipo de passo em Full-Step
 - 5: motor y energizado;
 - 6: motor y com sentido de rotação para a direita;
 - 7: motor y em funcionamento;

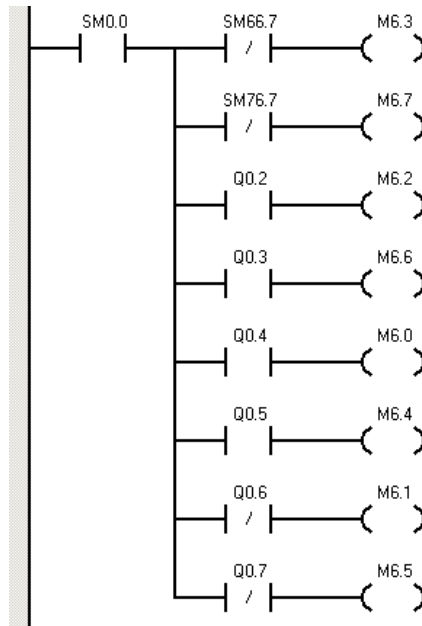


Figura 4.19 – Byte de Status dos motores.

Funcionalidades (subrotina)

Network 1 - Processar a mensagem 'S' (STOP: figura 4.20)

- A variável (word) VW900 é o buffer onde fica armazenado o valor associado a última ordem de comando recebida.
- As variáveis M5.5=1 e M5.0=1 são utilizadas para inibir o motor X e motor Y respectivamente.
- As variáveis M1.0=1 e M1.1=1 são utilizadas para parar instantaneamente os motores X e Y respectivamente.
- A instrução "JMP" permite saltar para o fim desta subrotina.

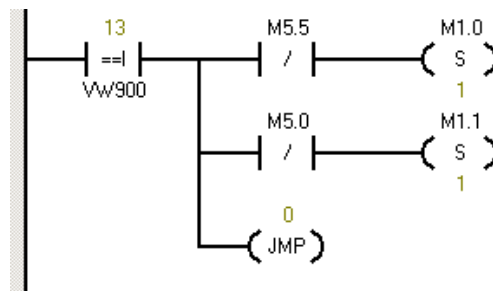
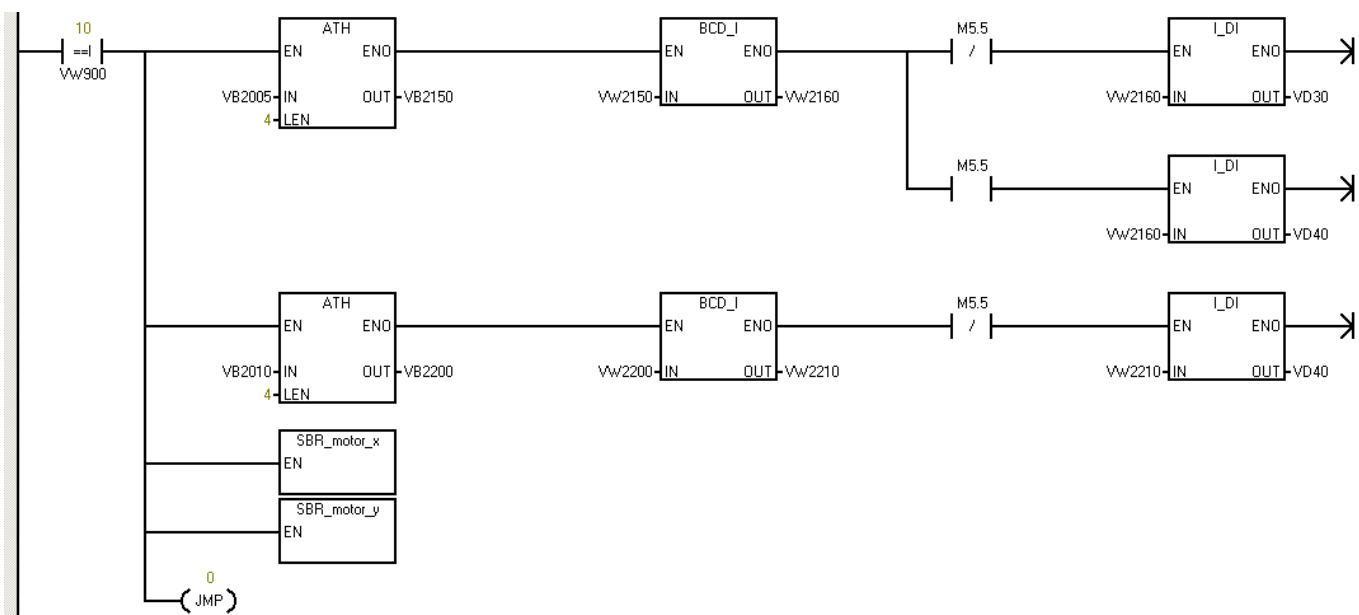


Figura 4.20 – Processar a mensagem 'S'.

Network 16 - Processar a mensagem "P____, ____" (número de passos: figura 4.21)

- A variável (word) VW900 é o buffer onde fica armazenado o valor associado a última ordem de comando recebida.
- As variáveis M5.5=1 e M5.0=1 são utilizadas para inibir o motor X e motor Y respectivamente.
- O valor pretendido para o número de passos a executar segundo uma determinada trajectória, fica armazenado no buffer de recepção a partir da variável VB2005 até há variável VB2008 para o motor x, e a partir da variável VB2010 até há variável VB2013 para o motor y.
- O valor pretendido para o número de passos do motor x fica alojado na variável de memória (double-word) VD30.
- O valor pretendido para o número de passos do motor y fica alojado na variável de memória (double-word) VD40.
- Chamada das subrotinas SBR_motor_x e SBR_motor_y, para actualização dos parâmetros dos motores.
- A instrução "JMP" permite saltar para o fim desta subrotina.

**Figura 4.21 – Processar a mensagem ‘P’.****SBR_0 (subrotina 0)****Network 1 - Inicializar os Parâmetros de Comunicação via Porta 0 (figura 4.22)**

- Mover o valor hexadecimal E9 (binário 1110 1001) para a memória especial (byte) SMB30 (Freeport control register for Port 0). Esta combinação de bits

configura a interface de comunicação para o modo Freeport, baud 9600, 7 bits de dados por carácter, e paridade ímpar.

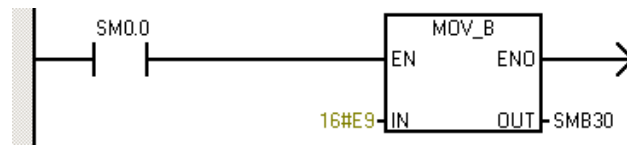


Figura 4.22 – Inicializar os Parâmetros de Comunicação via Porta 0.

SBR_1 (subrotina 1)

Network 1 - Inicialização dos parâmetros a receber pela porta 0 (figura 4.23)

- Move o valor hexadecimal B4 (binário 1011 0100) para o byte de memória especial SMB87 (the control byte for the Receive Message function of Port 0)
- Move a constante 50 para a memória especial (word) SMW90, para indicar um tempo de linha livre de 50 ms.
- Move a constante 4000 para a memória especial (word) SMW92 para indicar um temporizador inter-caracteres com um tempo livre de 4000 ms.
- Move a constante 14 para a memória especial (byte) SMB94 para definir que o número máximo de caracteres a receber =14.

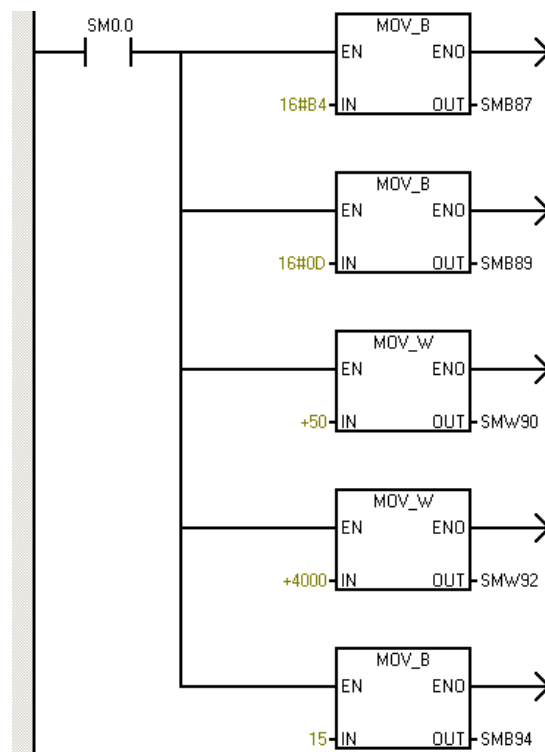


Figura 4.23 – Inicialização dos parâmetros a receber pela porta 0.

Receber mensagem (subrotina)

Network 8 - Mensagem de Estado "?" (figura 4.24)

- Utilizada para questionar se o respectivo módulo (indexer) está activo ou não.
- Na variável VB2000 fica armazenado o número de caracteres recebidos.
- Na variável VB2001 fica armazenado o primeiro carácter da mensagem recebida.
- Na variável VB2002 fica armazenado o segundo carácter da mensagem recebida e assim sucessivamente.
- São efectuadas várias comparações entre os caracteres esperados e os valores que chegam ao buffer de recepção, para se conseguir identificar a mensagem recebida.
- O valor hex 16#57 (ASCII = W) - comando Write
- O valor hex 16#30 (ASCII = 0) - modulo 0
- O valor hex 16#3F (ASCII = ?) - comando para motor X
- Move o valor 3 para a variável de memória (Byte) VB1000.
- A instrução "JMP" permite saltar para o fim desta subrotina.
- Move o valor 16#81 para a variavel de memória (Byte) MB10 para assinalar a ocorrência de um erro e o tipo de erro.
- Move o valor 4 para a variável de memória (Byte) VB1000.
- Move o valor 1 para a variável de memória (Byte) VB1001.

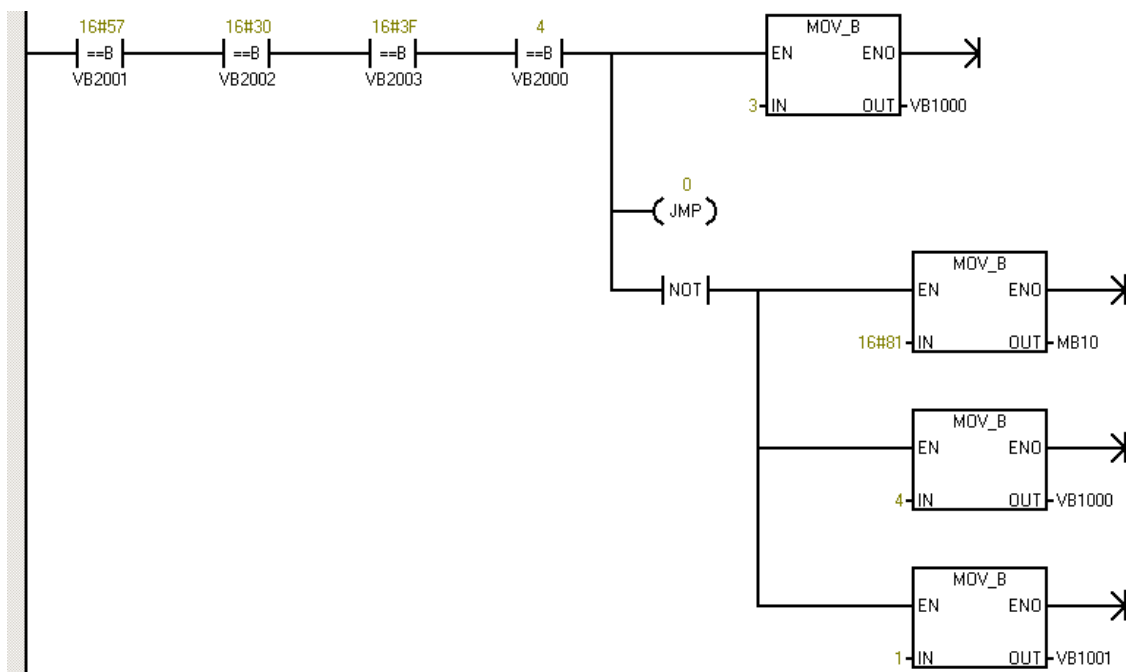


Figura 4.24 – Mensagem de Estado.

Erro (subrotina)

Network 5 - Mensagem recebida terminada: Atingido número máximo de caracteres permitidos (figura 4.25)

- Se o bit da memória especial SM86.1 está a '1' (ou seja, se a mensagem recebida foi terminada por ter sido atingido o número máximo de caracteres permitidos).
- Transmite via porta 0, o buffer de dados, que começa no byte de memória variável VB1130 ("E3").
- Mover a constante 2 para o byte da variável de memória VB1000 de modo a indicar que o estado de comunicação seguinte é TRANSMITE.
- Saltar para a Label LBL1, para o fim desta subrotina.

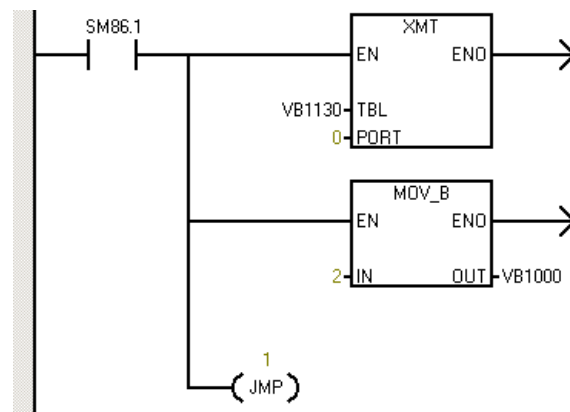


Figura 4.25 – Mensagem recebida terminada: Atingido número máximo de caracteres permitidos.

INT_0 (interrupção 0)

Network 1 - Configuração para receber o próximo comando (figura 4.26)

- É definido que o Bit de memória M0.7 (bit de erro grosseiro) é um contacto normalmente aberto e negado, para que a potência possa fluir, caso M0.7 não esteja a '1' (estado do sinal igual a 0).
- Se M0.7 não estiver a '1', armazena a mensagem que é recebida através da Porta 0 no buffer de dados, que começa no Byte de memória variável VB2000 (buffer de mensagens recebidas).

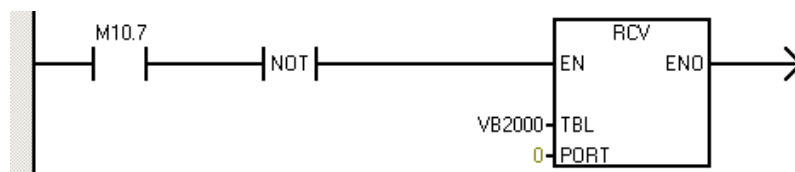
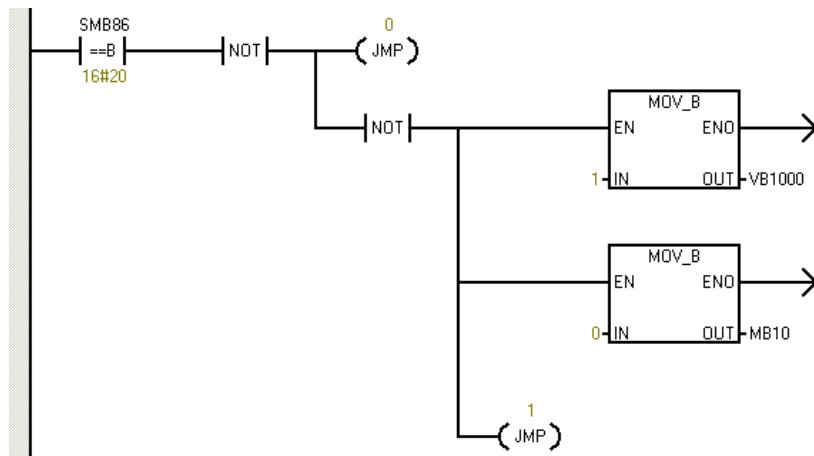


Figura 4.26 – Configuração para receber o próximo comando.

INT_1 (interrupção 1)**Network 1 - Verificar se o carácter final foi recebido (figura 4.27)**

- Compara o valor do byte de memória especial SMB86, com o valor hexadecimal 20 (binário 0010 0000).
- Se os valores forem iguais, isto significa que o carácter final foi recebido.
- Se o carácter final não for recebido, salta para a Label LBL0, para indicar que ocorreu um erro.
- Se o carácter final for recebido, move a constante 1 para o byte de memória variável VB0, de modo a indicar o estado de comunicação RECEBE, move a constante 0 para o byte de memória MB0 para limpar o byte de erro, e salta para a Label LBL1, para redefinir o temporizador de dois caracteres de atraso e activar um atraso por carácter para 5 ms.

**Figura 4.27 – Verificar se o carácter final foi recebido.****Network 3 – Indicação de erro (figura 4.28)**

- É utilizado o bit especial de memória SM0.0 para processar esta network em cada ciclo de scan.
- Move a constante 4 para o byte de memória variável VB0, para indicar o estado de comunicação ERRO.
- Move a constante 1 para o byte VB1, para indicar o sub-estado de erro de ESPERA.
- Move o valor hexadecimal 80 (binário 1000 0000) para o byte de memória MB0, para indicar que ocorreu um erro.

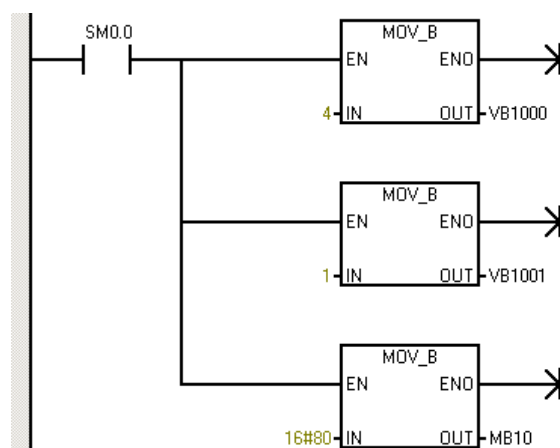


Figura 4.28 – Indicação de erro.

Capítulo 5

Testes e análise de resultados

Este capítulo descreve a evolução e os objectivos que foram sendo delineados em cada etapa ao longo do trabalho até ser atingido o resultado final. Também neste capítulo são abordados os resultados obtidos e nesse contexto será analisado o comportamento da solução final obtida durante os testes efectuados no laboratório de sistemas lógicos. Por fim, é efectuada a validação da solução alcançada, de maneira a se concluir o nível de satisfação desta.

5.1 Testes realizados

Tendo este trabalho uma elevada componente de desenvolvimento experimental, teve-se necessidade de efectuar de um modo faseado, a programação, os testes dos programas criados, e ainda a validação mediante os requisitos e os resultados obtidos.

O processo de desenvolvimento da solução passou por quatro fases. Na primeira fase o propósito era interagir com apenas um motor passo-a-passo através de um PLC (s7-200) e de uma consola (HMI Siemens®). Na segunda fase pretendia-se desenvolver um software que permitisse enviar e receber mensagens, sem que essas mensagens tivessem alguma função particular. Na terceira fase o grau de exigência aumentou, e aqui pretendia-se agora alterar e combinar o trabalho desenvolvido nas duas fases anteriores de maneira a se controlar um motor passo-a-passo, mas agora através de mensagens e não da consola. Na quarta e última fase os principais objectivos eram,

conseguir controlar dois motores passo-a-passo através de um protocolo de comunicação mais elaborado. Conseguir estabelecer uma comunicação (receber e enviar mensagens) entre o indexer e um outro controlador. E ainda conseguir desenvolver trajectórias lineares segundo o plano XY.

Primeira fase de teste

Na primeira fase após se realizar uma pesquisa bibliográfica e se ter compreendido minimamente as matérias que o tema desta dissertação abrange, desenvolveu-se um software e programou-se uma consola (Touch Panel TP 070) para se interagir com um motor passo-a-passo. Na figura 5.1 está representada a banca de trabalho onde se desenvolveu todo o trabalho desta primeira fase.

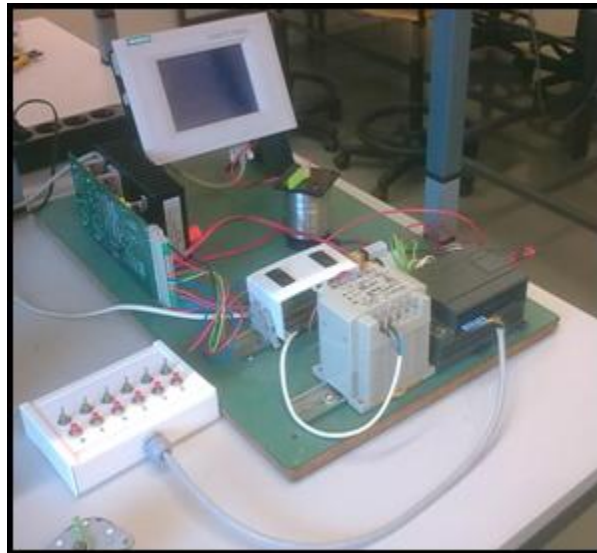


Figura 5.1 - Banca de trabalho da primeira fase de trabalho.

No início do trabalho era importante perceber como actuar um motor passo-a-passo e como colocar um motor deste tipo a executar as suas funções básicas. Por esse motivo, a interação com o motor foi realizada originalmente através uma consola (HMI Siemens®).

Ambicionava-se nesta fase desenvolver um software para o PLC e para a consola, que permitisse executar determinadas funcionalidade que seriam essenciais na solução final pretendida, as funcionalidades eram as seguintes:

- Energizar ou não o motor (ligar /desligar).
- Escolha do sentido de rotação do motor (sentido dos ponteiros do relógio ou sentido inverso).
- Colocar o motor a andar um número de passos entre 0 e 9999.
- Ordem para parar o motor de imediato.
- Escolha de tipo de passo do motor (passo completo ou meio passo).
- Ordenar o motor a deslocar-se a uma velocidade de valor 0 até 9999 (ms ou μ S).
- Escolher a base de tempo de ciclo (velocidade) em milissegundos (ms) ou microssegundos (μ S).
- Aparecer num campo da interface da consola uma mensagem de erro caso se introduzam valores fora dos limites pré-definidos.
- Programação de duas entradas do PLC para simular os fins de curso do sistema MPS Festo®.
- Apenas permitir que o motor se desloque para uma coordenada introduzida, se anteriormente este passou na posição de origem.
- Indicação da posição dos motores a informar se estes se encontram na posição de origem ou não.

Segunda fase de teste

Após na fase anterior se ter conseguido controlar as funcionalidades básicas do motor passo-a-passo através da consola, agora pretendia-se dar o primeiro passo para resolver o problema da comunicação entre controladores.

Nesta fase procedeu-se então ao desenvolvimento de um programa capaz de receber e enviar mensagens, através das instruções de comunicação e do protocolo Freeport disponíveis no software STEP 7-MicroWIN Siemens®.

A comunicação foi realizada entre o PLC e o software Windmill® (figura 5.2), foi deste modo também que se efectuaram praticamente todos os testes de comunicação entre um controlador (Computador) e o indexer. Windmill® é um software que possibilita a leitura e escrita de mensagens por intermédio da porta série do computador onde se encontra instalado.

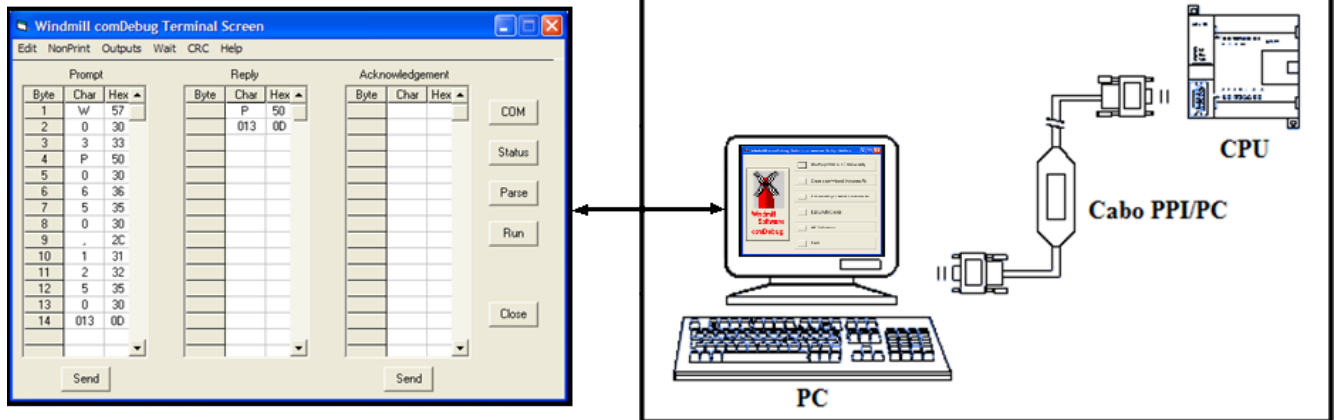


Figura 5.2 – Desenvolvimento de Comunicações PC/Indexer.

Terceira fase de teste

Para esta fase de testes tinha-se como objectivo agregar todo o trabalho desenvolvido até então e acrescentar melhoramentos. Para tal implementou-se um software que permitisse comandar um motor passo-a-passo mas agora por meio de mensagens. Criaram-se novas funcionalidades e desenvolveu-se um protocolo de comunicação simples onde cada funcionalidade era activada por uma mensagem formada por um carácter específico mais o carácter de fim de mensagem.

Assim sendo nesta fase, por intermédio do software Windmill[®] eram enviadas mensagens para o indexer que nos permitiam comandar um motor passo-a-passo, e também eram enviadas mensagens do indexer para o Windmill[®] que nos indicavam que a funcionalidade introduzida anteriormente tinha sido bem sucedida. Foram estudadas situações que poderiam surgir quando se estivesse a utilizar o software e por esse motivo criaram-se algumas mensagens de erro para informar o utilizador acerca do problema. As funcionalidades e as mensagens de erro programadas estão descritas nas tabelas 5.1 e 5.2 respectivamente:

Tabela 5.1 - Primeiro protocolo de comunicação desenvolvido para comando de um motor passo-a-passo

Funcionalidades do indexer	
Carácter	Designação
A	(A)pagar: Permite desligar a alimentação dos motores (não energizar).
D	(D)ireita: Colocar motor a rodar para a direita (sentido horário).
E	(E)squerda: Colocar motor a rodar para a esquerda (sentido anti-horário).
F	(F)ull-step: Motores executam passos completos.
G	(G)o: Desloca os dois motores segundo os parâmetros até então definidos.
H	(H)alf-step: Motores executam ½ passos.
L	(L)igar: Permite ligar a alimentação dos motores (energizar).
O	(O)rigem: Desloca os dois motores até a posição de origem (0,0).
P_ _ _ _	(P)osição: Valor pretendido para as coordenadas x e y do elemento móvel.
S	(S)top: Permite parar os motores imediatamente.
V_ _ _ _	(V)elocidade: Escala do tempo de ciclo em (µS) e o respectivo valor pretendido com 4 caracteres.
v_ _ _ _	(v)elocidade: Escala do tempo de ciclo em (ms) e o respectivo valor pretendido com 4 caracteres.

Tabela 5.2 - Primeira tabela elaborada de mensagens de erro

Mensagens de Erro	
Erro	Designação
E0	Mensagem recebida terminada: Comando introduzido pelo utilizador.
E1	Mensagem recebida terminada: Falta carácter final.
E2	Mensagem recebida terminada: Tempo expirado.
E3	Mensagem recebida terminada: Atingido número máximo de caracteres permitidos.
E4	Mensagem recebida terminada: Erro de paridade.
E5	Funcionalidade Inexistente.

Quarta fase de teste

Com a realização das três fases anteriores estávamos neste momento em condições de passar ao controlo de dois motores passo-a-passo, para finalmente se poder testar a capacidades de posicionamento do indexer.

No final desta fase aquilo que se ambicionava era conseguir alcançar uma solução eficaz, tanto na resolução do problema do posicionamento XY dos dois motores passo-a-passo como também do problema da comunicação entre controladores.

O primeiro passo a dar foi implementar entre as drives e o PLC (s7-200) uma placa de interface adaptadora, ilustrada na figura 5.3. Esta tem como funções, proteger e fazer a correspondência correcta entre os dois dispositivos, fazendo chegar aos motores passo-a-passo, sinais com os níveis de tensão ideais para o seu correcto funcionamento. Os sinais são provenientes do indexer (PLS s7-200), passam pelas drives, e são dirigidas aos motores.

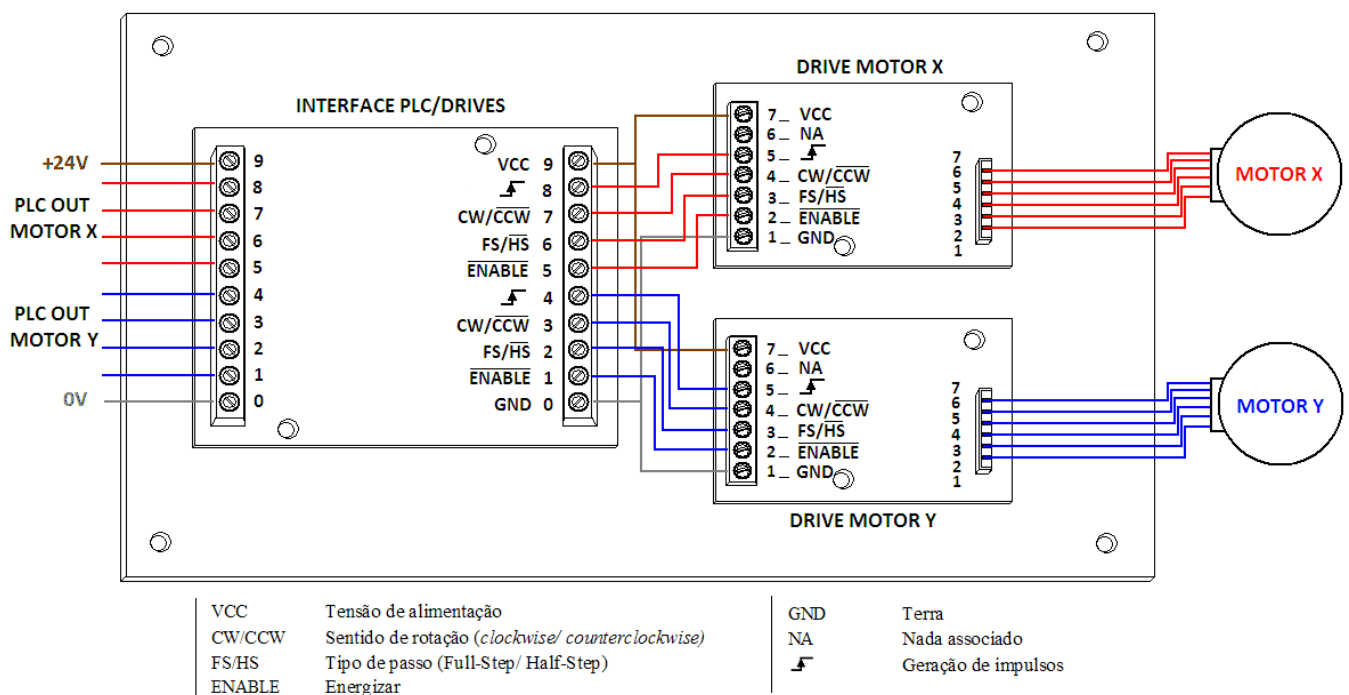


Figura 5.3 - Interface entre PLC e as drives dos motores passo-a-passo.

A seguir criaram-se ainda outras funcionalidades que se julgaram ser úteis e mais-valias na tarefa de posicionamento dos motores. Desenvolveu-se um protocolo de comunicação entre controladores mais completo e que possibilita especificar com mais detalhe a ordem de comando que se deseja ver executada. O resultado final nesta fase corresponde á proposta de solução que foi referida no capítulo 3 (3.3.7 Descrição do protocolo a implementar).

5.2 Resultados obtidos

Todas as funcionalidades que se pretendiam ver executadas nas diferentes fases (referidas no subtítulo anterior) do projecto foram elaboradas com sucesso. Para se evitar ao máximo o aparecimento de erros ou mal funcionamento nas soluções que foram criadas, a medida que o programa ia evoluindo, e iam sendo acrescentadas mais funcionalidades ao programa do indexer, cada uma delas foi sendo testada individualmente em pequenos programas criados apenas para esse efeito.

Visto que em muitas alturas até se conseguir corrigir um erro era necessário testar sistematicamente as mesmas funcionalidades, para facilitar e agilizar esses testes realizados nas diferentes fases, foram associadas algumas das entradas do PLC às funcionalidades mais utilizadas que foram sendo implementadas. Passou a ser possível activar ou não, essas funcionalidades através de interruptores.

A banca de trabalho onde se iniciou e se desenvolveu todo o trabalho para apenas um motor (primeira, segunda e terceira fase) é ilustrada na figura 5.4.

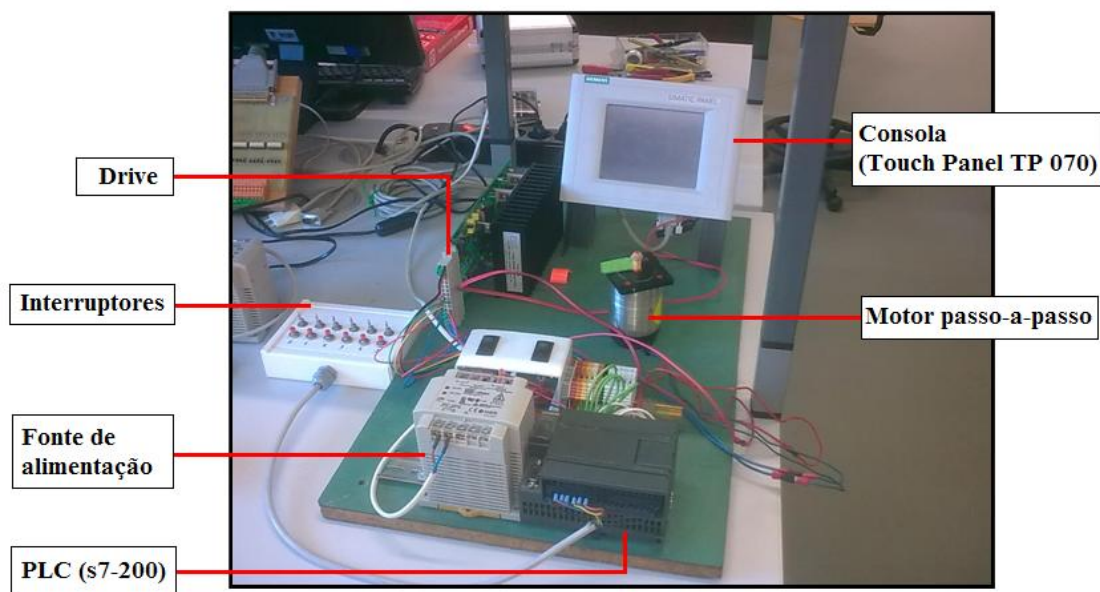


Figura 5.4 - Banca de trabalho onde se interagiu com um motor

Na figura 5.5 está representada a interface criada na primeira fase para comandar um motor através da consola.

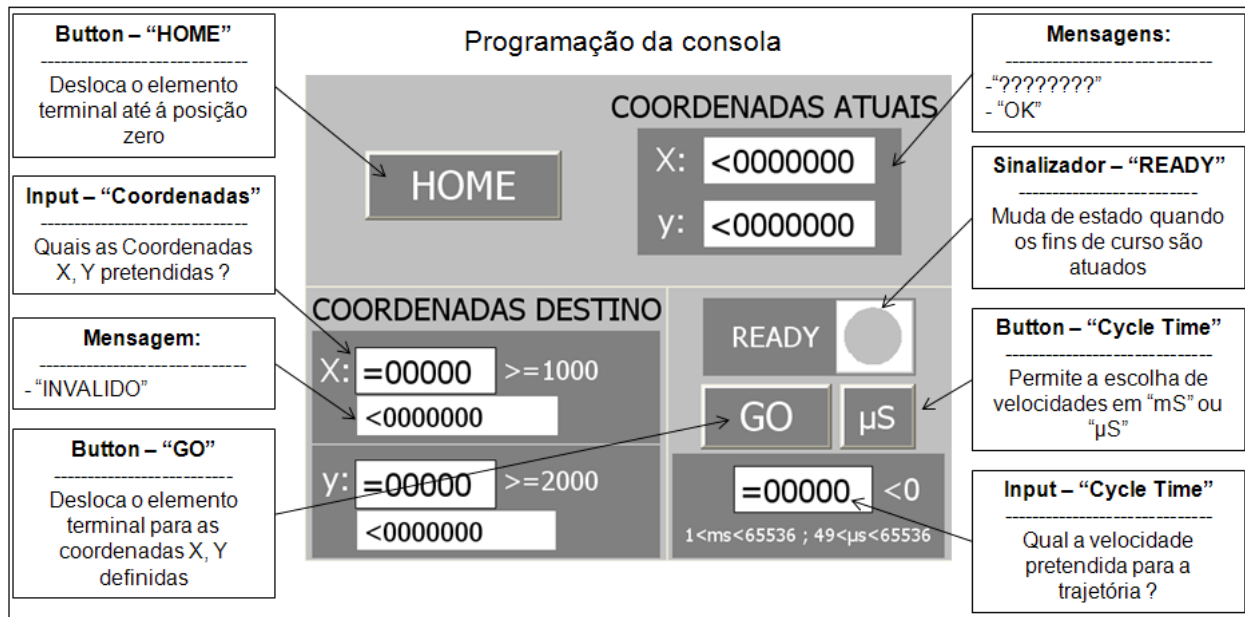


Figura 5.5 - Interface para comandar MPP.

Os testes de comunicação e funcionamento geral da solução realizados na segunda e terceira fase de testes do software foram como já referido no subtítulo anterior, efectuadas através do software Windmill[®]. Nas figuras 5.6, 5.7, 5.8 e 5.9 estão representados alguns exemplos desses testes.

Na figura 5.6 temos um exemplo de uma ordem de comando enviada por meio do Windmill[®] para o indexer segundo o protocolo desenvolvido na terceira fase de teste. Nesta fase o protocolo de comunicação produzido consistia em enviar sequencialmente mensagens, constituídas apenas por dois caracteres. Um dos caracteres diz respeito ao comando seleccionado, e outro carácter corresponde ao carácter de fim de mensagem (carriage return).

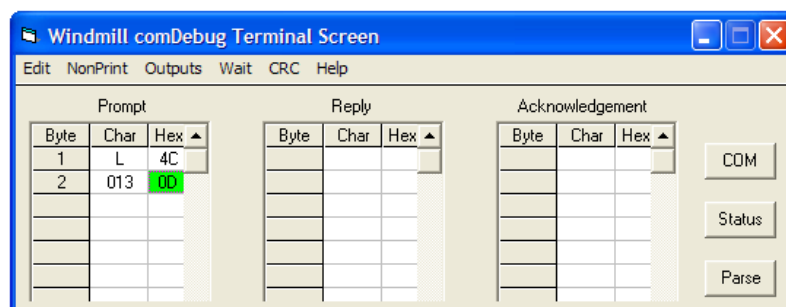


Figura 5.6 - Mensagem enviada para o indexer para executar uma funcionalidade na terceira fase

Na figura 5.7 temos um exemplo de uma mensagem de retorno, enviada pelo indexer para o Windmill®. Quando uma mensagem era enviada para o indexer, logo de seguida era sempre enviada uma outra mensagem de volta com o mesmo carácter da ordem de comando, para indicar que a ordem de comando foi recebida e executada com sucesso.

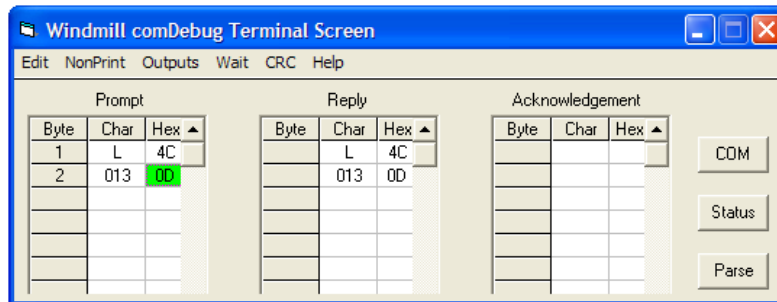


Figura 5.7 – Mensagem de feedback enviada pelo indexer na terceira fase de teste.

Na figura 5.8 temos um exemplo de um grupo de comando enviada por meio do Windmill® para o indexer segundo o protocolo desenvolvido na quarta fase de teste, nesta fase o protocolo de comunicação elaborado já permite especificar com mais detalhe o tipo de operação que se deseja executar. Neste caso o grupo de comando consistia em energizar os dois motores passo-a-passo.

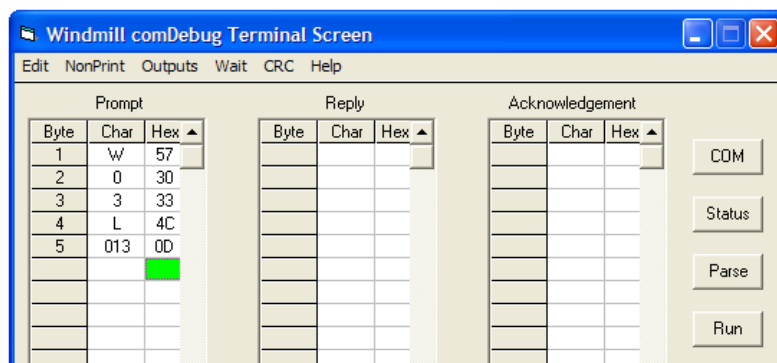


Figura 5.8 - Mensagem enviada para o indexer para executar uma funcionalidade na quarta fase de teste

A mensagem de feedback só é enviada pelo indexer se o modo de comunicação estiver definido para o modo ponto-a-ponto, caso contrário não é enviada qualquer mensagem por parte do indexer. A mensagem de retorno é constituída por apenas um

carácter, o mesmo carácter da funcionalidade que se deseja ver executado. Na figura 5.9 está representado um exemplo de uma mensagem de retorno.

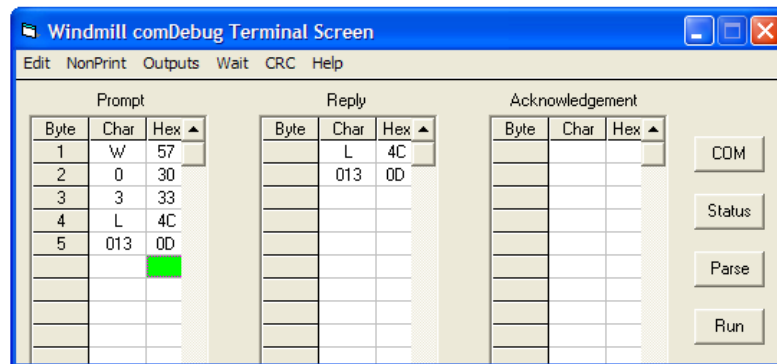


Figura 5.9 - Mensagem de feedback enviada pelo indexer na quarta fase de teste.

Nesta fase, até se alcançar uma solução final que responda a todos os requisitos, realizaram-se vários testes na banca de trabalho ilustrada na figura 5.10. Nesta banca foi possível testar todas as funcionalidades desenvolvidas com cada um dos motores e com os dois motores a funcionar em simultâneo. Introduziram-se propositadamente grupos de comando errados, no sentido de encontrar eventuais falhas que provocassem algum mal funcionamento.

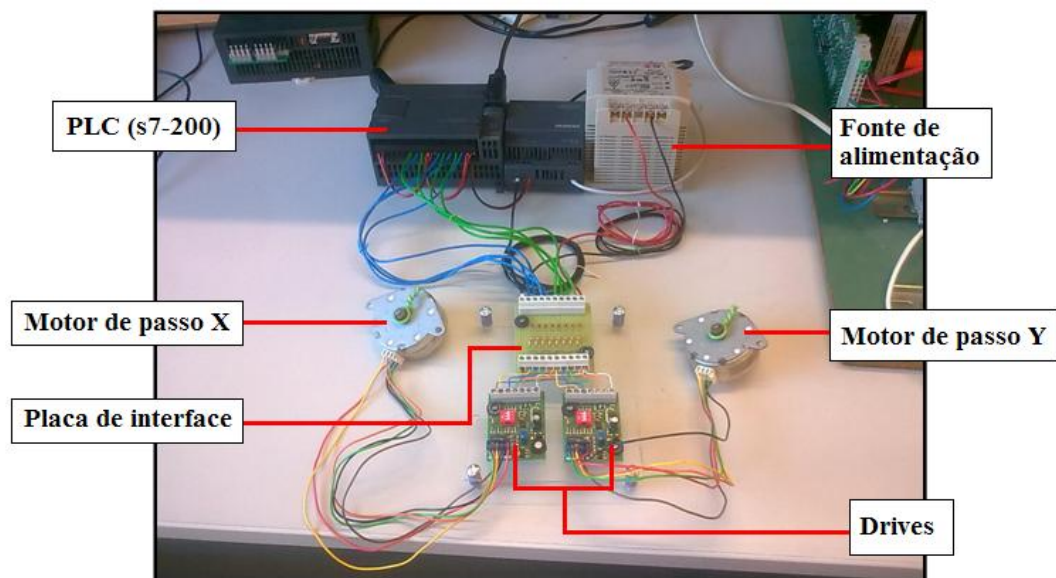


Figura 5.10 - Banca de trabalho onde se interagiu com um motor.

Após a obtenção da solução final, seria proveitoso adaptar essa solução ao sistema MPS Festo® para se efectuar um teste final. Esse teste consistiria em simular uma ou mais operações de maquinagem, de forma a avaliar o desempenho da solução no sistema. No entanto, dado o tempo limitado para a conclusão do projecto, não foi possível efectuar a simulação de uma operação de maquinagem no sistema MPS Festo®.

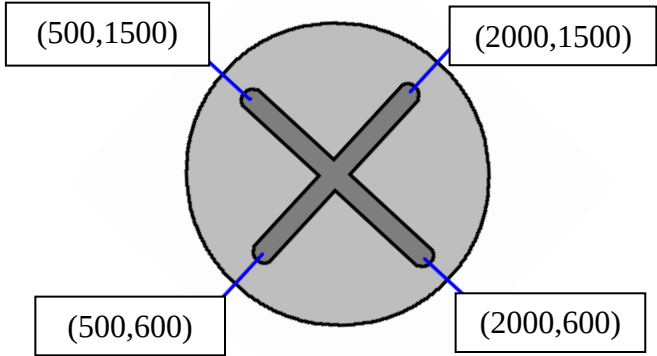
Em todo o caso, como teste final para a avaliação do desempenho da solução, foram realizadas simulações de três operações de maquinagem na banca de trabalho ilustrada na figura 5.10. Esses testes permitiram realizar na banca de trabalho, a mesma simulação que seria executada num sistema real, a excepção das etapas que não estão ao encargo do indexer.

Foi possível observar o correcto desempenho da solução final em todos os procedimentos de cada uma das diferentes operações de maquinagem. Esse teste final seguiu os procedimentos enumerados nas tabelas 5.3, 5.4, e 5.5.

Operações de maquinagem

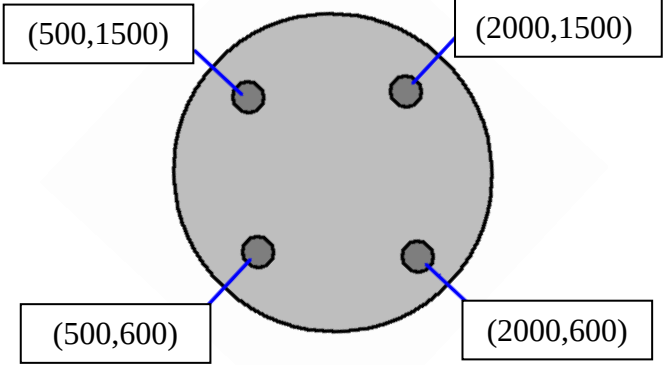
Testes finais/Procedimentos

Tabela 5.3 - Procedimento executado para simulação da operação de maquinagem I.

Maquinagem I			
			
	Operação	Grupo de comando	Designação
0	Modo de Comunicação	MS ou PP	Seleccionar modo de comunicação.
1	Energização	W 0 3 L <CR>	Os dois motores passam a estar activados (energizados).
2	Origem	W 0 3 O <CR>	Os dois motores deslocam-se da posição em que se encontram, para a posição de origem a uma velocidade pré-definida de 15 ms, ½ passo e sem rampa de aceleração.
3	Passos Completos	W 0 3 F <CR>	Motores passam a deslocar-se com passos completos.
4	Velocidade e Escala	W 0 3 V 3 0 0 0 <CR>	Define velocidade para a trajectória a realizar de 3000 na escala de tempo de μS.
5	Rampas Activas	W 0 3 R <CR>	Motores passam a deslocar-se com rampas de aceleração e desaceleração.
6	Coordenadas Absolutas	W 0 3 Z <CR>	Todas as coordenadas introduzidas a partir desta instrução serão relativas a posição de origem (0,0).
7	Posicionamento I	W 0 3 P 0 5 0 0 , 0 6 0 0 <CR>	Definição das coordenada 500 e 600 para o eixo X e Y respectivamente.
8	Executar	W 0 3 G <CR>	Colocar os motores a deslocarem-se de acordo com as definições previamente estabelecidas.

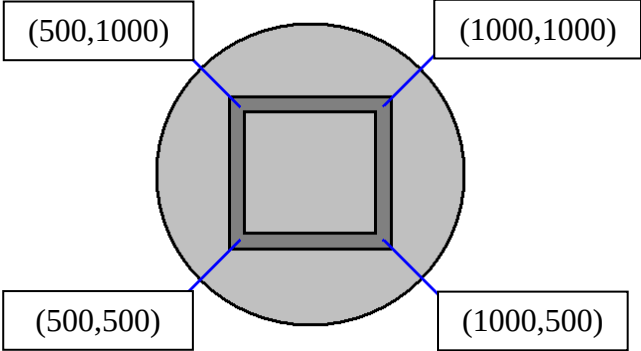
9	Ferramenta de Corte I	-	Colocar a ferramenta de corte em funcionamento e activar o movimento descendente da mesma.
10	Rampas Desactivas	W 0 3 r <CR>	Motores passam a deslocar-se sem rampas de aceleração e desaceleração.
11	Posicionamento II	W 0 3 P 2 0 0 0 , 1 5 0 0 <CR>	Definição das coordenadas 2000 e 1500 para o eixo X e Y respectivamente.
12	Executar	W 0 3 G <CR>	Colocar os motores a se deslocarem de acordo com as definições previamente estabelecidas.
13	Ferramenta de Corte II	-	Activar o movimento ascendente da ferramenta de corte e parar o funcionamento desta.
14	Rampas Activas	W 0 3 R <CR>	Motores passam a deslocar-se com rampas de aceleração e desaceleração.
15	Posicionamento III	W 0 3 P 0 5 0 0 , 1 5 0 0 <CR>	Definição das coordenadas 500 e 1500 para o eixo X e Y respectivamente.
16	Executar	W 0 3 G <CR>	Colocar os motores a se deslocarem de acordo com as definições previamente estabelecidas.
17	Ferramenta de Corte III	-	Colocar a ferramenta de corte em funcionamento e activar o movimento descendente da mesma.
18	Posicionamento IV	W 0 3 P 2 0 0 0 , 0 6 0 0 <CR>	Definição das coordenadas 2000 e 600 para o eixo X e Y respectivamente.
19	Rampas Desactivas	W 0 3 r <CR>	Motores passam a deslocar-se sem rampas de aceleração e desaceleração.
20	Executar	W 0 3 G <CR>	Colocar os motores a se deslocarem de acordo com as definições previamente estabelecidas.
21	Ferramenta de Corte IV	-	Activar o movimento ascendente da ferramenta de corte e parar o funcionamento desta.
22	Origem	W 0 3 O <CR>	Os dois motores deslocam-se da posição em que se encontram, para a posição de origem a uma velocidade pré-definida de 15 ms, ½ passo e sem rampa de aceleração.

Tabela 5.4 - Procedimento executado para simulação da operação de maquinagem II.

Maquinagem II			
			
	Operação	Grupo de comando	Designação
0	Modo de Comunicação	MS ou PP	Seleccionar modo de comunicação.
1	Energização	W 0 3 L <CR>	Os dois motores passam a estar activados (energizados).
2	Origem	W 0 3 O <CR>	Os dois motores deslocam-se da posição em que se encontram, para a posição de origem a uma velocidade pré-definida de 15 ms, ½ passo e sem rampa de aceleração.
3	Passos Completos	W 0 3 F <CR>	Motores passam a deslocar-se com passos completos.
4	Velocidade e Escala	W 0 3 V 3 0 0 0 <CR>	Definir uma velocidade para a trajectória a realizar de 3000 na escala de tempo de µS.
5	Rampas Activas	W 0 3 R <CR>	Motores passam a deslocar-se com rampas de aceleração e desaceleração.
6	Coordenadas Absolutas	W 0 3 Z <CR>	Todas as coordenadas introduzidas a partir desta instrução serão relativas a posição de origem (0,0).
7	Posicionamento I	W 0 3 P 0 5 0 0 , 0 6 0 0 <CR>	Definição das coordenada 500 e 600 para o eixo X e Y respectivamente.
8	Executar	W 0 3 G <CR>	Colocar os motores a deslocarem-se de acordo com as definições previamente estabelecidas.
9	Ferramenta de Corte I	-	Colocar a ferramenta de corte em funcionamento e activar o movimento descendente da mesma.
10	Ferramenta de Corte II	-	Activar o movimento ascendente da ferramenta de corte e parar o funcionamento desta.

11	Posicionamento II	W 0 3 P 2 0 0 0 , 1 5 0 0 <CR>	Definição das coordenadas 2000 e 1500 para o eixo X e Y respectivamente.
12	Executar	W 0 3 G <CR>	Colocar os motores a deslocarem-se de acordo com as definições previamente estabelecidas.
13	Ferramenta de Corte III	-	Colocar a ferramenta de corte em funcionamento e activar o movimento descendente da mesma.
14	Ferramenta de Corte IV	-	Activar o movimento ascendente da ferramenta de corte e parar o funcionamento desta.
15	Posicionamento III	W 0 3 P 0 5 0 0 , 1 5 0 0 <CR>	Definição das coordenadas 500 e 1500 para o eixo X e Y respectivamente.
16	Executar	W 0 3 G <CR>	Colocar os motores a deslocarem-se de acordo com as definições previamente estabelecidas.
17	Ferramenta de Corte V	-	Colocar a ferramenta de corte em funcionamento e activar o movimento descendente da mesma.
18	Ferramenta de Corte VI	-	Activar o movimento ascendente da ferramenta de corte e parar o funcionamento desta.
19	Posicionamento IV	W 0 3 P 2 0 0 0 , 0 6 0 0 <CR>	Definição das coordenadas 2000 e 600 para o eixo X e Y respectivamente.
20	Executar	W 0 3 G <CR>	Colocar os motores a deslocarem-se de acordo com as definições previamente estabelecidas.
21	Ferramenta de Corte VII	-	Colocar a ferramenta de corte em funcionamento e activar o movimento descendente da mesma.
22	Ferramenta de Corte VIII	-	Activar o movimento ascendente da ferramenta de corte e parar o funcionamento desta.
23	Origem	W 0 3 O <CR>	Os dois motores deslocam-se da posição em que se encontram, para a posição de origem a uma velocidade pré-definida de 15 ms, ½ passo e sem rampa de aceleração.

Tabela 5.5 - Procedimento executado para simulação da operação de maquinagem III.

Maquinagem III			
			
	Operação	Grupo de comando	Designação
0	Modo de Comunicação	MS ou PP	Seleccionar modo de comunicação.
1	Energização	W 0 3 L <CR>	Os dois motores passam a estar activados (energizados).
2	Origem	W 0 3 O <CR>	Os dois motores deslocam-se da posição em que se encontram, para a posição de origem a uma velocidade pré-definida de 15 ms, ½ passo e sem rampa de aceleração.
3	Passos Completos	W 0 3 F <CR>	Motores passam a deslocar-se com passos completos.
4	Velocidade e Escala	W 0 3 V 3 0 0 0 <CR>	Definir uma velocidade para a trajectória a realizar de 3000 na escala de tempo de µS.
5	Rampas Activas	W 0 3 R <CR>	Motores passam a deslocar-se com rampas de aceleração e desaceleração.
6	Coordenadas Absolutas	W 0 3 Z <CR>	Todas as coordenadas introduzidas a partir desta instrução serão relativas a posição de origem (0,0).
7	Posicionamento I	W 0 3 P 0 5 0 0 , 0 5 0 0 <CR>	Definição das coordenada 500 e 500 para o eixo X e Y respectivamente.
8	Executar	W 0 3 G <CR>	Colocar os motores a deslocarem-se de acordo com as definições previamente estabelecidas.
9	Ferramenta de Corte I	-	Colocar a ferramenta de corte em funcionamento e activar o movimento descendente da mesma.
10	Rampas Desactivas	W 0 3 r <CR>	Motores passam a deslocar-se sem rampas de aceleração e desaceleração.

11	Posicionamento II	W 0 3 P 1 0 0 0 , 0 5 0 0 <CR>	Definição das coordenada 1000 e 500 para o eixo X e Y respectivamente.
12	Executar	W 0 3 G <CR>	Colocar os motores a deslocarem-se de acordo com as definições previamente estabelecidas.
13	Posicionamento III	W 0 3 P 1 0 0 0 , 1 0 0 0 <CR>	Definição das coordenada 500 e 500 para o eixo X e Y respectivamente.
14	Executar	W 0 3 G <CR>	Colocar os motores a deslocarem-se de acordo com as definições previamente estabelecidas.
15	Posicionamento IV	W 0 3 P 0 5 0 0 , 1 0 0 0 <CR>	Definição das coordenada 500 e 1000 para o eixo X e Y respectivamente.
16	Executar	W 0 3 G <CR>	Colocar os motores a deslocarem-se de acordo com as definições previamente estabelecidas.
17	Ferramenta de Corte II	-	Activar o movimento ascendente da ferramenta de corte e parar o funcionamento desta.
18	Origem	W 0 3 O <CR>	Os dois motores deslocam-se da posição em que se encontram, para a posição de origem a uma velocidade pré-definida de 15 ms, ½ passo e sem rampa de aceleração.

5.3 Validação

Foi efectuada uma análise da solução final para o comando XY de dois motores passo-a-passo recorrendo a um PLC (s7-200). Essa análise permitiu compreender se a solução final satisfaz os requisitos que visam a implementação de uma metódica comunicação entre controladores e um comando XY didacticamente interessante.

Após realizarem-se todos os testes foi possível analisar que o desempenho do indexer, tanto ao nível das comunicações com um outro controlador, como também ao nível do tipo de comando XY implementado, corresponde às expectativas.

No que diz respeito às comunicações, o indexer efectua tal como pretendido, a troca de mensagens entre si e um outro controlador em função dos parâmetros de comunicação pré-definidos e dos parâmetros de recepção de mensagens. Tem disponíveis dois diferentes modos de comunicação (mestre/escravo ou ponto-a-ponto) para permitir uma maior flexibilidade na comunicação entre dispositivos.

Quanto ao tipo de comando XY implementado no indexer, este também responde positivamente à função principal que lhe é exigida. Procurou-se promover um tipo de comando didacticamente interessante, isto é, um tipo de comando que seja intuitivo e com um nível de abstracção relativamente elevado, que se afaste do código de máquina e seja mais próximo de uma linguagem de alto nível, que permita ainda especificar com algum detalhe mas também com simplicidade o tipo de operações que se desejam ver executadas. Procurou-se ainda ter disponíveis as funcionalidades necessárias e suficientes para um bom desempenho do indexer.

Concluiu-se que a solução final desenvolvida cumpre de uma forma geral os requisitos necessários para a simulação de uma operação de maquinagem num sistema genérico semelhante ao sistema MPS Festo[®].

Capítulo 6

Conclusões

6.1 Resumo da tese

Este trabalho foi desenvolvido com o intuito de criar um indexer baseado em hardware e software de um controlador lógico programável Siemens[®], para um sistema genérico que necessite através do comando de motores passo-a-passo efectuar o posicionamento X, Y de um elemento móvel segundo trajectórias lineares. Pretendeu-se ainda, que o indexer comunica-se com um controlador hierarquicamente superior através da porta série e que os comandos fossem inseridos por meio de mensagens de acordo com um protocolo implementado.

No entanto, dado que, o sistema que motivou o desenvolvimento desta solução foi um sistema didáctico MPS da Festo[®] existente na FEUP, designado por unidade de demonstração de produção flexível, a solução foi desenvolvida, tendo em consideração determinadas particularidades que essa unidade requer.

Na pesquisa de informação para este trabalho foi possível analisar o modo de operação dos motores passo-a-passo e compreendê-los mais um pouco, bem com também os controladores em geral e os indexers que se baseiam em PLCs. Foi possível constatar que existem actualmente poucas soluções no mercado que empregam a tecnologia dos controladores lógicos programáveis e que permitem efectuar posicionamento através do comando de motores passo-a-passo.

Apresentou-se a proposta da solução projectada, demonstrou-se a arquitectura da solução ao nível de hardware e software implementada, abordando-se a questão das comunicações entre controladores mais particular pois é uma das partes relevante deste trabalho.

Nesta dissertação foram também abordadas as matérias associadas ao software de programação utilizado que habilitou o desenvolvimento do indexer, matérias como a linguagem de programação utilizada entre outras particularidades do software de programação. As instruções mais relevantes para o comando de motores passo-a-passo e implementação do protocolo de comunicação.

São ainda apresentadas as várias fases de teste que foram realizadas ao longo da evolução do trabalho, onde se apresentam os objectivos delineados, os resultados obtidos em cada uma das fases de teste e por fim a validação da solução encontrada segundo os resultados obtidos.

Por fim, no último capítulo desta dissertação, são efectuadas as devidas conclusões do trabalho, tendo em consideração os resultados obtidos, junto de algumas propostas para trabalhos futuros e de um balanço pessoal relativamente a todas as experiências vivida neste trabalho.

6.2 Resultados obtidos

Ao longo deste trabalho foram concebidas várias aplicações, da evolução dessas aplicações deu origem um software final que possibilita o comando de motores passo-a-passo. Em cada uma dessas aplicações definiram-se diferentes requisitos onde o nível de exigência foi crescendo progressivamente. O resultado foi a obtenção de aplicações cada vez mais elaboradas até proporcionarem o desempenho final ambicionado.

Todos os requisitos ou objectivos traçados em cada fase de teste foram atingidos, como tal o indexer concebido até ao momento está configurado para funcionar da seguinte forma:

- O PLC que serviu de base ao indexer, uma vez ligado a um outro controlador hierarquicamente superior que permita comunicar de acordo com os parâmetros de comunicação definidos, quando colocado em modo de funcionamento (modo

‘RUN’), este não envia qualquer mensagem ao controlador pois inicializa sempre o seu funcionamento sob o modo de comunicação mestre/escravo.

- Sempre que se pretender comutar de modo de comunicação mestre/escravo para ponto-a-ponto ou vice-versa, basta enviar o respectivo comando (mensagem), referenciada na tabela 3.6.
- Em modo de comunicação ponto-a-ponto, após se enviar do indexer uma mensagem identificada como um comando ou grupo de comando, é sempre enviada logo de seguida uma mensagem a indicar que tal comando ou grupo de comando foi identificado e portanto imediatamente executado. Situação essa que, não se verifica em modo mestre/escravo.
- Independentemente do modo de comunicação definido para o indexer, este tem disponível em qualquer momento algumas funcionalidades (grupos de comando da tabela 3.5) que permitem fornecer informações ou dados acerca do estado de funcionamento do indexer.
- O indexer permite executar sequencialmente diversas funcionalidades (grupos de comando da tabela 3.3) criadas para possibilitar e facilitar a execução de uma operação de maquinagem.
- É desenvolvida uma trajectória linear sempre que se transmite a ordem que coloca os dois motores a se deslocarem segundo os parâmetros de controlo da saída de alta frequência, a movimentação dos motores é sincronizada de acordo com o número de passos pretendido para cada um dos eixos e a velocidade definida para a trajectória.
- Foram criadas algumas mensagem de erro (tabela 3.2) para alertar para diversas situações de erro prováveis.
- É possível enviar uma mensagem (tabela 3.7) a perguntar se o módulo do indexer ao qual nos estamos a referir se encontra activo ou não.

Por forma ao indexer concebido nesta dissertação puder ser uma opção válida para inúmeras aplicações que envolvam o comando de dois motores passo-a-passo, e tendo em consideração que o indexer se baseou em hardware e software de

controladores lógicos programáveis. Estes dois aspectos implicaram um elevado desenvolvimento da programação da solução e apesar de todo o esforço nesse sentido, o indexer elaborado neste trabalho dispõem ainda de uma ampla margem de progresso.

Nesta dissertação projectou-se e concebeu-se um indexer que permitiu executar todas as funcionalidades pretendidas, julgadas necessárias a facultar a realização de uma operação de maquinagem de uma máquina.

6.3 Propostas de trabalhos futuros

Para que em trabalhos futuros seja possível efectuar-se, de um modo agradável e eficaz, uma rápida contextualização acerca de tudo aquilo que foi realizado neste trabalho e assim permitir acrescentar melhorias ao indexer. Procurou-se obter um programa bem delineado.

O trabalho desenvolvido teve como objectivo a criação de um indexer para comando XY de dois motores passo-a-passo. Reflectindo sobre a performance do indexer nos vários testes realizados ao longo do trabalho, consegue-se perceber nesta fase que existem ainda algumas limitações, percebe-se também que são vários os aspectos que ainda tem muita margem de manobra, e são muitas as melhorias que se poderão ainda realizar em trabalhos futuros. Como tal, são várias as melhorias e desenvolvimentos que podem valorizar o trabalho efectuado até ao momento, das quais se destacam:

- O protocolo de comunicação é um dos aspectos que ainda dispõem de uma larga margem de progresso e aperfeiçoamento. Um aspecto é por exemplo, a inserção de novas funcionalidades que venham a substituir ou acrescentar algo de útil, ou que permitam corrigir alguma falha que se verifique e que afecte o correcto funcionamento do sistema.
- O PLC dispõe de várias entradas de sinais lógicos que neste momento não estão a ser utilizadas. Essas entradas poderiam ser utilizadas para facultar a interação entre o utilizador e indexer, principalmente por motivos de segurança. A essas entradas poderiam estar associadas botoneiras ou sensores. Implementar

botoneiras ao sistema para acções de emergência (como: “PARAR” ou “REINICIAR”) é algo a ter em conta.

- Ao nível da programação desenvolvida, o software criado tem também muitos aspectos a melhorar e com certeza outros aspectos a acrescentar a solução já existente, de maneira a tornar esta numa solução mais eficaz e dinâmica. Existem situações que ocorrem em que o indexer poderia responder de uma forma melhor e existem outras situações para a qual o indexer nem sequer está preparado para lidar.
- O indexer actualmente é apenas constituído por um módulo mas no futuro considera-se útil poder inserir outros módulos em anexo, para permitir ou indexer desempenhar outro tipo de acções ou complementar o que está feito até aqui.
- Seria útil e tornaria o actual indexer, num indexer mais versátil, se este fosse capaz de desempenhar também trajectórias curvilíneas para além das trajectórias lineares que este desempenha.

6.4 Balanço pessoal

Apesar de no mundo industrial ser mais fácil adquirir soluções prontas a instalar/utilizar – as ditas “chave na mão”, é importante ter-se a noção de que estas nem sempre existem ou não são as mais adequadas a uma determinada aplicação. Por esse motivo é importante ter noção da existência de alternativas, como é o caso do desenvolvimento de soluções como a apresentada neste trabalho. Este trabalho permitiu ter uma melhor noção acerca das condicionantes de desenvolvimento de uma solução deste tipo e respectivas possibilidades de aplicação.

Outro aspecto importante que este trabalho proporcionou foi a possibilidade de visualizar de uma forma mais clara as potencialidades das tecnologias envolvidas no projecto.

A sua concretização foi assim um verdadeiro desafio face às várias dificuldades sentidas e obstáculos a ultrapassar. Após a ultrapassagem de um determinado obstáculo, surgia um outro que exigia rapidamente uma nova solução.

A experiência e o conhecimento adquirido acerca das matérias que o tema envolve foram muito enriquecedores. Ganhou-se, com toda a certeza, conhecimentos que vão ser úteis no futuro e ampliou-se mais um pouco a nossa área de conhecimento acerca do vasto tema que é a Automação. O saber alcançado na área das comunicações e do funcionamento dos dispositivos da Siemens® (PLC, Consola) junto do respectivo software e instruções nele disponíveis, foi significativo. Acompanhar a evolução de todo o processo de implementação do protocolo de comunicação, foi em particular muito interessante.

Partir praticamente do zero, projectar uma solução para cumprir determinados requisitos e no final ver essa mesma solução desempenhar as funções para a qual foi projectada, foi para mim muito importante.

Referências bibliográficas

- [1] ASCII table and description <http://asciitable.com>
- [2] B&B Electronics Manufacturing Company - ‘*Technical Library*’ - Disponível na Internet em <http://www.bb-elec.com/techlib.asp>
- [3] Brites, G. F., Santos, V. (2008). Motor de Passo, Programa de Educação Tutorial. Niterói: Faculdade de engenharia, Universidade Federal Fluminense.
- [4] Chousal, M. (1991). *Controlo digital de motores passo-a-passo*. Dissertação apresentada para obtenção do grau de Mestre em Engenharia Electrotécnica e de Computadores. Porto: Faculdade de Engenharia da Universidade do Porto.
- [5] Eberhardt, Dr. R. Schulé. Festo Didactic KG (1991). *Remote Positioning Module Handbook*. Esslingen am Neckar. Germany.
- [6] Elizabeth A. Nichols, Joseph C. Nichols, Keith R. Musson. (1982). *Data Communications for Microcomputers: with Practical Applications and Experiments* / New York : McGraw-Hill Book, cop.
- [7] Festo Didactic <http://www.festo-didactic.com/int-en/>
- [8] Leon-Garcia, G., Widjaja, I. (2004). *Communication Networks, Fundamental Concepts and Key Architectures*. Second Edition. New York, NY: McGraw-Hill.
- [9] Mackay, S., Wright, E., Reynders, D., Park, J. (2004). *Practical Industrial Data Networks: Design, Installation and Troubleshooting*. First published, Elsevier/Newnes.
- [10] Magalhães, P. A. (2010). *Apontamentos da disciplina de Computação Industrial*. Porto: Faculdade de Engenharia da Universidade do Porto.
- [11] Manual Técnico da Allen-Bradley:
http://samplecode.rockwellautomation.com/idc/groups/literature/documents/um/1746-um121_-en-p.pdf
- [12] Manual Técnico da Schneider:
http://www.isquaredt.com/doc/scm220_manual.pdf
- [13] Manual Técnico da Siemens:
http://www.automation.siemens.com/doconweb/pdf/SINUMERIK_SIMODRIV_E_09_2009_E/FM353.pdf?p=1

- [14] Polka, Dave. (2003). *Motors and Drives – A practical technology guide*, USA: The Instrumentation, Systems, and Automation Society.
- [15] Siemens AG. Automation & Drives. SIMATIC Human Machine Interface, (2000). *SIMATIC HMI Touch Panel TP 070, Equipment Manual*. Nuremberg: Federal Republic of Germany.
- [16] Siemens AG. Automation and Drives. Industrial Automation Systems Division, (2000). *S7-200 Programmable Controller System Manual*. Nuremberg: Federal Republic of Germany.
- [17] Siemens AG. Automation and Drives. Industrial Automation Systems Division, (1999). *Micro system SIMATIC S7-200 One Hour Primer Manual*. Nuremberg: Federal Republic of Germany.
- [18] Siemens AG. Automation and Drives. Industrial Automation Systems Division, (2000). *Micro system SIMATIC S7-200 Two Hour Primer Manual*. Nuremberg: Federal Republic of Germany.
- [19] Siemens <http://www.siemens.com>
- [20] Stenerson, J. (2004). *Fundamentals of Programmable Logic Controllers, Sensors, and Communications*. Third edition. Upper Saddle River, New Jersey; Pearson Education/ Prentice Hall.
- [21] Tanenbaum, Andrew S. (1996). *Computer Networks*. Third Edition. Upper Saddle River, New Jersey: Prentice-Hall International, Inc.
- [22] Tenreiro Machado A. J. (1994). *Motores Passo a Passo: Aspectos Construtivos e Modos de Funcionamento*. Robótica e Automatização, Artigo de revista científica nacional nº 17, pp.17-20. Porto: Faculdade de Engenharia da Universidade do Porto.
- [23] Tenreiro Machado A. J. (1994). *Motores passo a passo: controlo e modos de funcionamento*. Porto: Publindústria.
- [24] William S. Davis. Redwood, CA. (1991). *Computing fundamentals concepts*: The Benjamin/Cummings Publishing Company.

Nota: todas as referências web indicadas estavam naturalmente disponíveis à data da submissão da presente dissertação (Agosto de 2011).

ANEXO A: Blocos de código e dados mais importantes

Toda a informação contida neste anexo auxilia a compreensão do programa desenvolvido segundo o software STEP 7 MicroWIN.

SIMATIC/ Mnemónicas Internacionais:

Description	SIMATIC	International
Ladder Diagram	KOP	LAD
Function Block Diagram	FUP	FBD
Statement List	AWL	STL
Input	E	I
Output	A	Q
Variable Memory	V	V
Internal Memory	M	M
Analog Input	AE	AI
Analog Output	AA	AQ
Timer	T	T
Counter	Z	C
High-Speed Counter	HC	HC
SCR	S	S
Special Memory	SM	SM
Accumulator	AC	AC
Local Variable Memory	L	L

Endereçamento de memória:

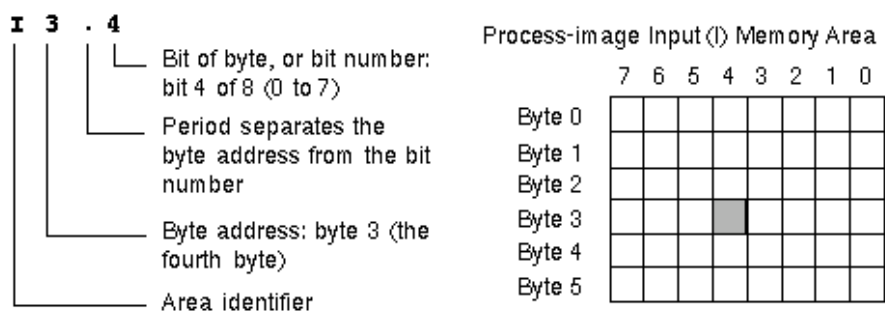
Os programas podem ser escritos em qualquer um dos três modos de endereçar as instruções de programação:

- Directo
- Simbólico
- Indirecto

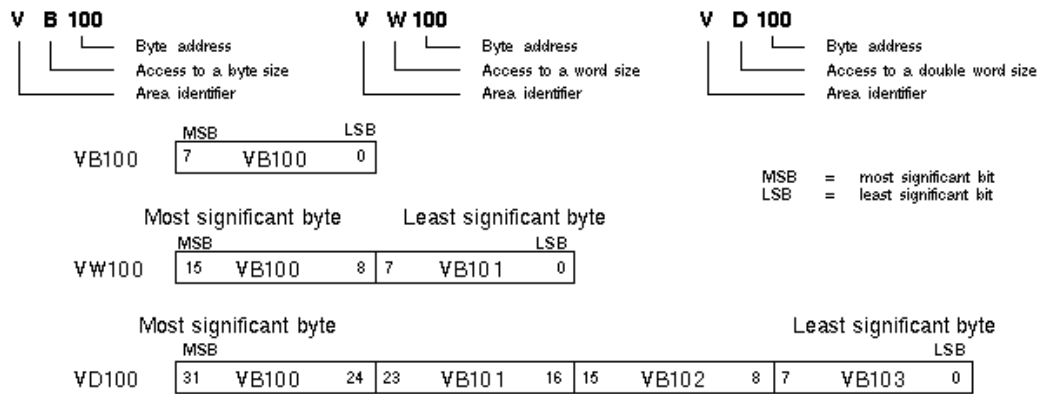
Endereçamento Directo:

O PLC s7-200 armazena informações em locais diferentes de memória que têm endereços únicos. É possível identificar explicitamente o endereço de memória que se deseja aceder. Isso permite ao programa ter um acesso directo às informações. Endereçamento directo especifica a área de memória, tamanho e localização, por exemplo VW790, refere-se a localização da Word 790, na área de memória V.

Para aceder a um bit numa área de memória, especifica-se o endereço, o que inclui o identificador de área de memória, o endereço de byte, e o número de bits precedido por um período. A figura abaixo mostra um exemplo de acesso a um bit (que é também chamado de endereçamento "byte.bit"). Neste exemplo, a área de memória e o byte de endereço (I = input, e 3 = byte 3) são seguidos por um ponto (".") para separar o bit de endereço (bit 4).



Pode-se aceder a dados na maioria das áreas de memória (V, I, Q, M, S, L, e SM), como *bytes*, *words* ou *double words* usando o formato de endereço byte. Para aceder a um byte, word ou double word de dados na memória, deve-se especificar o endereço de uma forma semelhante a especificar o endereço de um bit. Isto inclui um identificador área, designação da dimensão de dados, bem como o byte de endereço de início do byte, word, ou double word, como ilustrado na figura abaixo.



Dados em outras áreas de memória (como T, C, HC, e os acumuladores) são acedidos usando um formato de endereço que inclui um identificador de área e um número do dispositivo.

Endereçamento Simbólico:

Endereçamento simbólico usa uma combinação de caracteres alfanuméricos para identificar um endereço. Uma constante simbólica usa o nome do símbolo para identificar o número da constante ou o valor do carácter ASCII.

Para programas SIMATIC, é possível fazer atribuições de símbolos globais usando a tabela de símbolos. Para os programas IEC, faz-se atribuições de símbolos globais usando a tabela de variáveis globais.

Endereçamento Indirecto:

Endereçamento indirecto usa um ponteiro para aceder os dados na memória. Os ponteiros são locais de memória do tipo double words, que contêm o endereço de outro local de memória. Para criar um ponteiro, deve-se usar a instrução ‘Move Double Word’ para mover o endereço de um local de memória indirectamente endereçado ao local do ponteiro. Ponteiros também podem ser passados para uma subrotina como parâmetro.

Para aceder indirectamente aos dados de um endereço de memória, cria-se um ponteiro para esse local, digitando o símbolo (&) e a localização da memória a que nos queremos endereçar.

Inserir um asterisco (*) na frente de um operando para uma instrução específica que o operando é um ponteiro. Como mostrado na figura abaixo, definindo * em AC1 especifica que AC1 é um ponteiro para um valor de comprimento de uma Word a ser referenciado pela instrução Move Word (MOVW). Neste exemplo, os valores armazenados em VB200 e VB201 são movidos para o acumulador AC0.

Formato de identificação e Gama de Valores das constantes numéricas:

Ranges of Constant Number Values

Data Size:	Unsigned Integer Range		Signed Integer Range	
	Decimal:	Hexadecimal:	Decimal:	Hexadecimal:
B (Byte)	0 to 255	0 to FF	-128 to +127	80 to 7F
W (Word)	0 to 65535	0 to FFFF	-32768 to +32767	8000 to 7FFF
D (Double Word)	0 to 4294967295	0 to FFFF FFFF	-2147483648 to +2147483647	8000 0000 to 7FFF FFFF
Data Size:	Decimal Real (Positive Range)		Decimal Real (Negative Range)	
D (Double Word)	+1.175495E-38 to +3.402823E+38		-1.175495E-38 to -3.402823E+38	

Constant Format Identifier

Your program can use data constants in byte, word, or double word sizes in many instructions. Format identifiers control how a constant value is displayed (as binary, decimal, hexadecimal, or ASCII characters).

Program Constants are assumed to be decimal numbers unless you use a format identifier:

2#	Binary numbers
16#	Hexadecimal numbers
'abcd'	ASCII constant, begin and end the ASCII characters with single quote characters. Valid for most instructions that specify Constant in the parameter list. Stored as data bytes.
"string"	ASCII constant string, begin and end the ASCII string with double quote characters Valid for instruction parameters that specify Constant String in the parameter list. Stores as a length byte followed by data bytes.

Tabela com os Bytes de referência para o controlo PTO/PWM

Control Register (Hex Value)	Result of executing the PLS instruction							
	Enable	Select Mode	PTO Segment Operation	PWM Update Method	Time Base	Pulse Count	Pulse Width	Cycle Time
16#81	Yes	PTO	Single		1 μ s/cycle			Load
16#84	Yes	PTO	Single		1 μ s/cycle	Load		
16#85	Yes	PTO	Single		1 μ s/cycle	Load		Load
16#89	Yes	PTO	Single		1 ms/cycle			Load
16#8C	Yes	PTO	Single		1 ms/cycle	Load		
16#8D	Yes	PTO	Single		1 ms/cycle	Load		Load
16#A0	Yes	PTO	Multiple		1 μ s/cycle			
16#A8	Yes	PTO	Multiple		1 ms/cycle			
16#D1	Yes	PWM		Synchronous	1 μ s/cycle			Load
16#D2	Yes	PWM		Synchronous	1 μ s/cycle		Load	
16#D3	Yes	PWM		Synchronous	1 μ s/cycle		Load	Load
16#D9	Yes	PWM		Synchronous	1 ms/cycle			Load
16#DA	Yes	PWM		Synchronous	1 ms/cycle		Load	
16#DB	Yes	PWM		Synchronous	1 ms/cycle		Load	Load