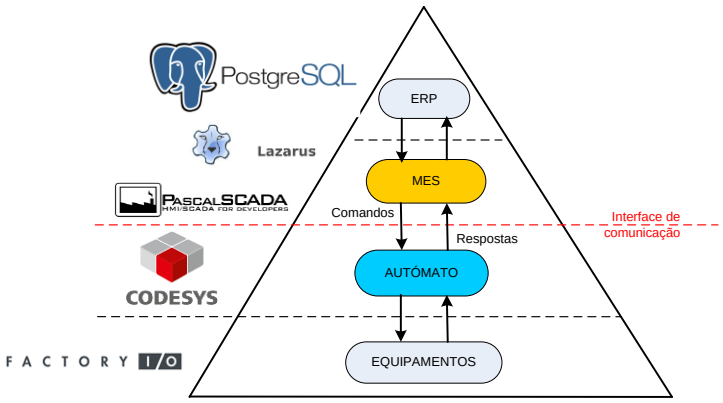
Informática Industrial 2020/21

Interface de comunicação entre o mini-MES e o Autómato

Sistema de gestão industrial integrado: da gestão comercial à gestão da fábrica

INTRODUÇÃO

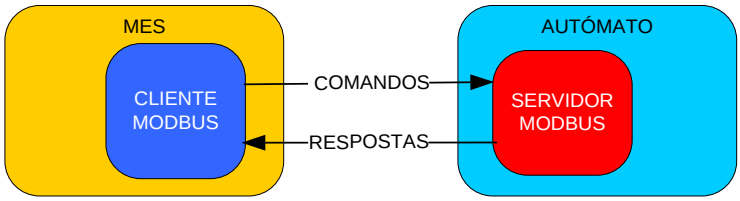
Este documento descreve a interface de comunicação entre o mini-MES e o Autómato.



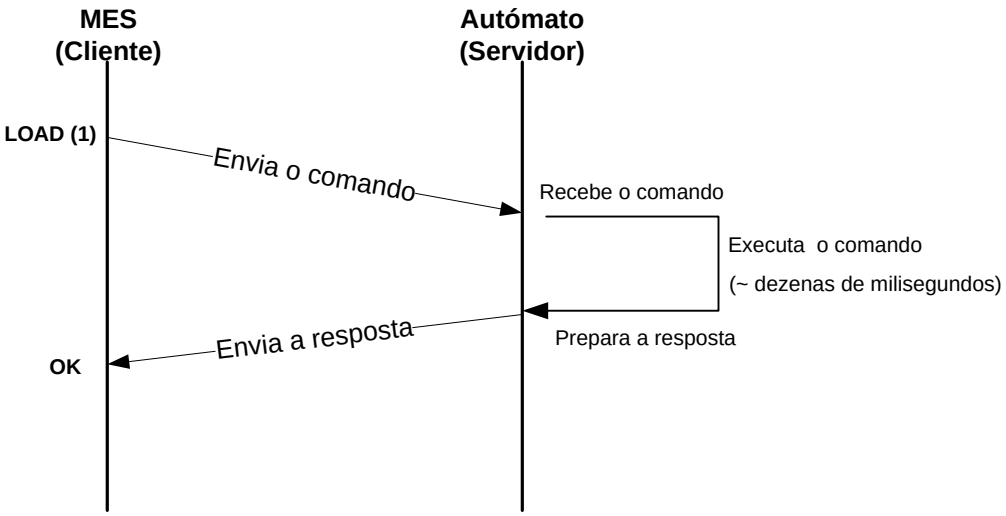
MODELO DE INTERAÇÃO

A interação entre o mini-MES e o Autómato realiza-se através de um modelo Cliente-Servidor da seguinte forma:

- **mini-MES**. Funciona como Cliente, enviando pedidos para o MES e recebendo as respostas do autómato.
- **Autómato**. Funciona como Servidor, executando os pedidos do MES e processando as respetivas respostas.



Os pedidos estão organizados sob a forma de **comandos**. Cada comando tem uma estrutura específica e representa uma ação que deve ser realizada pelo autómato (ex. retirar uma peça do armazém). Quando é enviado um comando para o servidor este gera uma **resposta** que indica a forma como o comando foi executado.



Os comandos são enviados através do protocolo **Modbus**, que será estudado oportunamente.

Os comandos disponíveis são os seguintes:

Comando	**Descrição**
INITIALIZE	Inicializa uma posição do armazém com um tipo de peça
LOAD	Carrega uma peça para uma posição do armazém
UNLOAD	Retira uma peça de uma posição do armazém
GET_FREE	Obtém o número de posições livres no armazém
GET_USED	Obtém o número de posições ocupadas no armazém
GET_PART_INFO	Obtém informações sobre uma peça armazenada no armazém
SET_N_PARTS_DEFECTIVE	Define N peças com defeito
GET_PART_QUALITY	Obtém informação sobre a qualidade de uma peça
DO_PRODUCTION	Realiza uma operação de produção para obter um produto final
DO_EXPEDITION	Transporta um produto final para a expedição
DO_INBOUND	Recebe matéria-prima vinda do exterior da fábrica
DO_SCRATCH	Remove uma peça sem qualidade do armazém
GET_FACTORY_STATUS	Obtém o estado dos equipamentos da fábrica

DESCRIÇÃO FUNCIONAL

Neste documento é discutida a interação entre o mini-MES e o Autómato de uma perspetiva funcional, ou seja, apresentando apenas a funcionalidade dos comandos e ignorando os aspetos de implementação.

ESTRUTURA DOS COMANDOS

Um comando pode ser perspetivado como uma **função** que é invocada pelo cliente junto do servidor para ser executada. Um comando pode ter vários **parâmetros**. Após o comando ser executado pelo servidor, este retorna uma **resposta**. Assim, o cliente envia um comando para o servidor, que o executa, e retorna um resultado ao cliente. De forma geral, os comandos têm a seguinte sintaxe:

{resposta} = **Comando** (P1, …., Pn)

em que:

- **Comando** é o nome do comando que se pretende executar no servidor
- **P1, …, Pn** são parâmetros desse comando (o número é variável e depende de cada comando)
- *resposta* é o valor retornado por esse comando. A *resposta* é um valor numérico (número inteiro) ou um vetor de valores numéricos.
- Uma *resposta* com um valor numérico **negativo** indica que ocorreu um erro durante a execução do comando.

ESPECIFICAÇÃO DOS COMANDOS

Nesta secção são apresentados os vários comandos e as respetivas respostas.

Neste documento, a palavra **peça** é usada indistintamente para indicar **matérias-primas** ou **produtos finais**.

INITIALIZE

O comando **INITIALIZE** permite inicializar uma posição **X** do armazém com uma peça do tipo **P.** Este comando deve ser utilizado para definir o stock inicial do armazém (matérias-primas e produtos finais).

O formato do comando é o seguinte:

resposta = INITIALIZE (**X**, **P**)

em que:

- **X** é a posição do armazém. Podem ser inicializadas apenas as posições correspondentes à 1ª coluna do armazém, isto é **X** = **1**, **10**, **19**, **28**, **37** e **46**[1].
- **P** é o tipo de peça, na gama {1,2,4,5,7,8}[2]

A *resposta* do comando pode tomar os seguintes valores:

Valor	Significado
1	Comando válido. A peça P vai ser criada na posição X do armazém.
-101	Erro, a posição X está ocupada
-103	Erro, a posição X está fora da gama do armazém {1..54}
-107	Erro, o código da peça (P) está fora da gama permitida {1,2,4,5,7,8}
-108	Erro, a posição X está fora da gama permitida {1,10,19,28,37,46}
-109	Erro, o armazém está ocupado a executar outro comando

[1] Ver seção com a identificação das posições do armazém

[2] Ver seção com a identificação das peças

LOAD

O comando **LOAD** permite carregar uma peça presente no tapete de entrada do armazém e coloca-la na posição **X** do armazém.

O formato do comando é o seguinte:

resposta = LOAD (**X**)

em que:

- **X** é a posição do armazém na gama **1..54**

A *resposta* do comando pode tomar os seguintes valores:

Valor	Significado
1	Comando válido. A peça vai ser inserida na posição X do armazém
-101	Erro, a posição X está ocupada
-103	Erro, a posição X está fora da gama do armazém {1..54}
-104	Erro, não existe peça no tapete de entrada do armazém
-109	Erro, o armazém está ocupado a executar outro comando

UNLOAD

O comando **UNLOAD** permite retirar uma peça da posição **X** do armazém e coloca-la no respetivo tapete de saída.

O formato do comando é o seguinte:

resposta = UNLOAD (**X**)

em que:

- **X** é a posição do armazém na gama **1..54**

A *resposta* do comando pode tomar os seguintes valores:

Valor	Significado
1	Comando válido. Peça vai ser retirada na posição X do armazém
-102	Erro, a posição X está vazia (i.e. sem peça)
-103	Erro, a posição X está fora da gama do armazém {1..54}
-105	Erro, o tapete de saída do armazém está ocupado com uma peça
-109	Erro, o armazém está ocupado a executar outro comando

GET_FREE

O comando **GET_FREE** permite obter o número de posições livres no armazém.

O formato do comando é o seguinte:

resposta = GET_FREE ()

A *resposta* do comando é o número de posições livres no armazém, na gama **0…54**.

GET_STORED

O comando **GET_STORED** permite obter o quantas peças do tipo **P** estão no armazém.

O formato do comando é o seguinte:

resposta = STORED (**P**)

em que:

- **P** é o código da peça {1..9}. Se for invocado com **P**=0 retorna a soma de todas as peças que existem no armazém.

A *resposta* do comando pode tomar os seguintes valores:

Valor	**Significado**
N	Número de peças do tipo P no armazém ou o número total de peças no armazém (se P=0)
-107	Erro, o código da peça (P) está fora da gama permitida {1..9}

GET_PART_INFO

O comando **GET_PART_INFO** permite obter qual o tipo de peça que está armazenada na posição **X** do armazém.

O formato do comando é o seguinte:

resposta = GET_PART_INFO (**X**)

em que:

- **X** é a posição do armazém na gama **1..54**

A *resposta* do comando pode tomar os seguintes valores:

Valor	**Significado**
P	Código da peça **P**={1…9} armazenada na posição X do armazém
-102	Erro, a posição X está vazia (i.e. sem peça)
-103	Erro, a posição X está fora da gama do armazém {1..54}

GET_PART_QUALITY

O comando **GET_PART_QUALITY** permite obter informação sobre a qualidade da peça que está armazenada na posição **X** do armazém. Esta informação está guardada na etiqueta RFID associada à peça.

O formato do comando é o seguinte:

resposta = GET_PART_QUALITY (**X**)

em que:

- **X** é a posição do armazém na gama **1..54**

A *resposta* do comando pode tomar os seguintes valores:

Valor	**Significado**
1	A peça armazenada na posição X tem qualidade
2	A peça armazenada na posição X tem um defeito
-102	Erro, a posição X está vazia (i.e. sem peça)
-103	Erro, a posição X está fora da gama do armazém {1..54}

SET_N_PARTS_DEFECTIVE

O comando **SET_N_PARTS_DEFECTIVE** permite atribuir um defeito a **N** produtos finais do tipo **P** armazenados no armazém. Esta informação fica guardada na etiqueta RFID associada ao produto. Por omissão todas os produtos finais são considerados como tendo boa qualidade. O processo de atribuição é decorre da seguinte forma: o sistema atribui esta condição aos N primeiros produtos do tipo P que encontrar. O processo é atómico: ou atribui a todos ou não atribui a nenhum.

O formato do comando é o seguinte:

resposta = SET_N_PARTS_DEFECTIVE (**P, N**)

em que:

- **P** é o código do produto final P={4,5,7,8}
- **N** número de produtos finais a que se pretende atribuir um defeito.

A *resposta* do comando pode tomar os seguintes valores:

Valor	Significado
1	Comando válido. N peças do tipo P passam a ser consideradas com defeito
-107	Erro, o código do produto final (P) está fora da gama permitida {4,5,7,8}
-115	Erro, não existem N produtos finais do tipo P no armazém

DO_INBOUND

O comando **DO_INBOUND** permite receber uma matéria-prima do tipo **P** proveniente do exterior da fábrica. A matéria-prima depois de recebida é encaminhada automaticamente para o tapete de entrada do armazém.

O formato do comando é o seguinte:

resposta = DO_INBOUND (**P**)

em que:

- **P** é o código da matéria-prima P={1,2}

A *resposta* do comando pode tomar os seguintes valores:

Valor	Significado
1	Comando executado corretamente. A matéria-prima foi recebida na fábrica.
-107	Erro, o código da matéria-prima (P) está fora da gama permitida {1,2}
-112	Erro, ocupado a receber matérias-primas

DO_PRODUCTION

O comando **DO_PRODUCTION** permite produzir um produto final na célula **M**. A matéria-prima necessária para esta produção tem que estar previamente colocada no tapete de saída do armazém. O produto final, após ser processado na máquina, retorna automaticamente ao tapete de entrada do armazém.

O formato do comando é o seguinte:

resposta = DO_PRODUCTION (**M**)

em que:

- **M** é a máquina de destino:
  - **1 :** Célula 1
  - **2 :** Célula 2

A *resposta* do comando pode tomar os seguintes valores:

Valor	Significado
1	Comando executado corretamente. O produto final vai ser produzido na célula **M**
-106	Erro, não existe peça no tapete de saída do armazém
-107	Erro, a peça que existe no tapete não é uma matéria prima
-110	Erro, o destino está fora da gama permitida {1,2}

Se se tentar enviar um produto final (i.e. já produzido) para qualquer uma das células (uma decisão errada), esse produto retorna automaticamente ao tapete de entrada do armazém sem passar pela célula (utiliza o tapete periférico que contorna o armazém). O comando retorna um erro nesta situação.

DO_EXPEDITION

O comando **DO_EXPEDITION** permite enviar um produto final para a expedição. O produto final tem que estar previamente colocado no tapete de saída do armazém.

O formato do comando é o seguinte:

resposta = DO_EXPEDITION (**O**)

em que:

- **O** é o destino da expedição:
  - **1:** Saída 1
  - **2:** Saída 2

A *resposta* do comando pode tomar os seguintes valores:

Valor	Significado
1	Comando executado corretamente. O produto final foi enviado para a expedição
-106	Erro, não existe uma peça no tapete de saída do armazém
-107	Erro, a peça que existe no tapete de saída não é um produto final
-110	Erro, destino fora da gama permitida {1,2}
-112	Erro, ocupado a expedir produtos finais

Nota: se se tentar expedir uma matéria-prima (uma decisão errada), essa matéria-prima retorna automaticamente ao tapete de entrada do armazém (utiliza o tapete periférico que contorna o armazém). O comando retorna um erro nesta situação.

DO_SCRATCH

O comando **DO_SCRATCH** remove automaticamente 1 produto final do tipo **P com defeito** que existe no armazém e envia-o automaticamente para a **Saída 3**. É removido o 1º produto deste tipo que for encontrado.

O formato do comando é o seguinte:

resposta = DO_SCRATCH (**P**)

em que:

- **P** é o tipo de produto final a remover {4,5,7,8}.

A *resposta* do comando pode tomar os seguintes valores:

Valor	Significado
X	Comando executado corretamente. Foi removido um produto final da posição **X=**{1..54} do armazém
-107	Erro, o código do produto final (P) está fora da gama permitida {4,5,7,8}
-115	Erro, não há nenhuma peça do tipo P com defeito para ser removida do armazém

Nota: se pretender remover N produtos terá que invocar este comando N vezes.

GET_FACTORY_STATUS

O comando **GET_FACTORY_STATUS** permite obter qual o estado dos vários equipamentos da fábrica.

A sintaxe do comando é a seguinte:

resposta = GET_FACTORTY_STATUS()

A *resposta* do comando é um vetor de inteiros com 8 posições.

Cada posição do vetor tem a seguinte informação:

Posição	Significado	Valor	
1	Estado do FactoryIO	1	Pronto para receber comandos
		0	Desligado (não pode ser utilizado)
2	Receção de matérias-primas (Inbound)	1	Pronto para receber comandos
		2	Ocupado a executar um comando
3	Estado do armazém	1	Pronto para receber comandos
		2	Ocupado a executar um comando
4	Peça no tapete de entrada do armazém	P	Com uma peça do tipo P={0..9} (0=sem peça)
5	Peça no tapete de saída do armazém	P	Com uma peça do tipo P={0..9} (0=sem peça)
6	Célula 1 (Robot, Bases)	P	Com uma peça do tipo P={0..9} (0=sem peça)
7	Célula 2 (Robot, Tampas)	P	Com uma peça do tipo P={0..9} (0=sem peça)
8	Pick & Place	P	O código da peça produzida é dado pela seguinte expressão: P=10*código_da_base + código_da_tampa (0=sem peça)

Comum a todos os comandos

Os seguintes elementos são comuns a todos os comandos:

- Respostas na gama >=0 indicam que o comando foi executado corretamente
- Respostas na gama < 0 indicam que existiu um erro:
  - -**100… -254** indicam que existiu um erro na execução do comando
  - -**255** indica que se trata de um comando inválido.

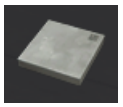
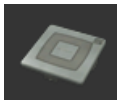
ASPETOS A CONSIDERAR NO ENVIO DOS COMANDOS

Devem ser tomados em conta os seguintes aspetos no envio dos comandos:

- Antes de ser enviado qualquer comando para o Autómato tem que existir uma conexão estabelecida entre o mini-MES e o Autómato.
- Se o FactoryIO não estiver pronto para receber comandos, os comandos enviados são ignorados.
- O Armazém só executa 1 comando de cada vez. Se estiver ocupado a executar um comando e receber um novo comando, esse comando é ignorado.
- O Inbound só executa só executa 1 comando de cada vez. Se estiver ocupado a executar um comando e receber um novo comando, esse comando é ignorado.

IDENTIFICAÇÃO DAS PEÇAS

Cada peça é identificada por um código numérico, na gama **1…9**, de acordo com a figura seguinte.

Código	Significado	Imagem
1	Matéria-prima AZUL	
2	Matéria-prima VERDE	
3	Matéria-prima METAL	
4	Base AZUL	
5	Base VERDE	
6	Base METAL	
7	Tampa AZUL	
8	Tampa VERDE	
9	Tampa METAL	

POSIÇÕES DO ARMAZÉM

As posições do armazém estão identificadas de acordo com a figura seguinte.

54						46
						37
						28
						19
						10
9				3	2	1

A numeração assume que se está na zona de Inbound 'de frente' para o armazém.

IMPLEMENTAÇÃO EM LAZARUS

Cada um dos comandos anteriores encontra-se implementado como uma função (FUNCTION) em Lazarus, disponível na Unit **comunit.pas** associado à Form **ComForm**. Deverá assim adicionar esta Unit e Form ao seu projeto para poder utilizar estas funções.

As funções têm o mesmo nome e os mesmos parâmetros do comando, tendo sido apenas adicionado o prefixo **M_**:

```
function M_Load(position: integer): integer;
function M_Unload(position: integer): integer;
function M_Get_Free(): integer;
function M_Get_Stored(piece: integer): integer;
function M_Get_Part_Info(position: integer): integer;
function M_Initialize(position, piece: integer): integer;
function M_Set_N_Parts_Defective(piece, number : integer): integer;
function M_Get_Part_Quality(position: integer): integer;
function M_Do_Scratch (piece: integer): integer;
function M_Do_Production(destination: integer): integer;
function M_Do_Inbound (piece: integer): integer;
function M_Do_Expedition(destination: integer): integer;
function M_Get_Factory_Status(): status_values;
```

Se estas funções forem invocadas com parâmetros errados retornam o valor **-1**. Neste caso o comando associado não é enviado para o autómato.

Antes de invocar estas funções, é necessário estabelecer uma conexão com o autómato. Para tal estão disponíveis as seguintes funções:

```
function M_Connect(): integer;
function M_Disconnect(): integer;
function M_Connection_Status(): integer;
```

A função **M_Connect()** permite estabelecer uma conexão com o autómato. Deve ser a 1ª função a ser chamada antes de enviar qualquer comando para o autómato. Se retornar um valor >0 a conexão foi estabelecida com sucesso. Se retornar um valor <0 é porque existiu um erro no estabelecimento da conexão (ex. o programa do autómato não está a ser executado). Em caso de erro, e como não há conexão, não podem ser enviados comandos.

A função **M_Connect()** permite encerrar a conexão com o autómato. Só deve ser chamada no fim da execução do MES e do ERP. Tal como no caso anterior, se retornar um valor >0 indica que a conexão foi encerrada com sucesso. Caso retorne um valor <0 é porque existiu um erro no encerramento da conexão (não há consequências nesta situação desde que o programa do MES seja encerrado).

A função **M_Connection_Status()** permite saber se a conexão está estabelecida ou não. Se retornar um valor >0 indica que a conexão está estabelecida. Caso retorne um valor <0 é porque a conexão foi interrompida. Se retornar um valor =0 é porque não foi realizada nenhuma conexão.

Adicionalmente estão disponíveis nesta Unit três estruturas de dados que auxiliam o desenvolvimento do código e que podem ser utilizadas no MES:

```
status_response_size = 8;


status_values = array[1..status_response_size] of integer;
M_cmd_msg: array [-115 ... -100] of string;
M_status_msg: array [1..status_response_size] of string;
```

- **status_values** é um tipo de dados (TYPE). Corresponde a um vetor onde é retornada a resposta da função **M_Get_Factory_Status**().
- O vetor **M_cmd_msg** contém os textos das mensagens de erro associadas às funções.
- O vetor **M_status_msg** contém os textos associados a cada uma das posições do vetor retornado pela função **M_Get_Factory_Status**().

FIM