

Profibus: Por dentro dos *Identifier Formats*

Autor: César Cassiolato - Gerente de Produtos
Smar Equipamentos Industriais Ltda.

Introdução

O arquivo gsd é como se fosse um datasheet eletrônico do equipamento que trás detalhes de revisão de hardware e software, bus timing do equipamento e informações sobre a troca de dados cíclicos. As informações de troca de dados cíclicos para cada módulo permitido do equipamento são demarcadas pelas palavras-chaves “*Module*” e “*EndModule*”. Entenda como troca de dado cíclico a informação requisitada ou enviada pelo mestre classe 1(PLC, por exemplo), de alta prioridade, e que é parte fundamental no controle e tomada de decisão.

Cada módulo possui um conjunto de *Identifier Bytes* ou *Identifier Formats*. Este artigo nos mostrará como cada identificador é interpretado pelo mestre classe 1.

Entendendo os *Identifier Formats*

Os *Identifier Formats* são usados na configuração entre o mestre classe 1 e seus escravos. Após a energização (conhecida como power up) os equipamentos escravos estão prontos para a troca de dados cíclicos com o mestre classe 1, mas para isto, a parametrização no mestre para aquele escravo deve estar correta. Estas informações são obtidas através dos arquivos gsds, que deve ser um para cada equipamento. Através dos comandos abaixo, o mestre executa todo processo de inicialização com os equipamentos:

- *Get_Cfg*: carrega a configuração dos escravos e verifica a configuração da rede;
- *Set_Prm*: escreve em parâmetros dos escravos e executa serviços de parametrização da rede;
- *Set_Cfg*: configura os escravos segundo entradas e saídas;
- *Get_Cfg*: um segundo comando, onde o mestre verificará a configuração dos escravos.

Todos estes serviços são baseados nas informações obtidas dos arquivos gsds.

Existem 3 tipos de *Identifier Formats*, onde a principal diferença entre eles é a quantidade de bits e bytes que eles podem representar:

- *Simple*: podem representar 8 e 16 bits de dados
- *Especial*: podem representar 8 e 16 bits de dados e ainda tipos em formatos especiais
- *Especial para Profibus DP-V1*: podem representar 8 e 16 bits de dados e tipos padrões definidos de acordo com o DP-V1.

Exemplo de *Identifier Formats*:

```
;Modules for Analog Input
Module = "Analog Input (short) " 0x94
EndModule
Module = "Analog Input (long) " 0x42, 0x84, 0x08, 0x05
EndModule
```

Identifier Formats simples

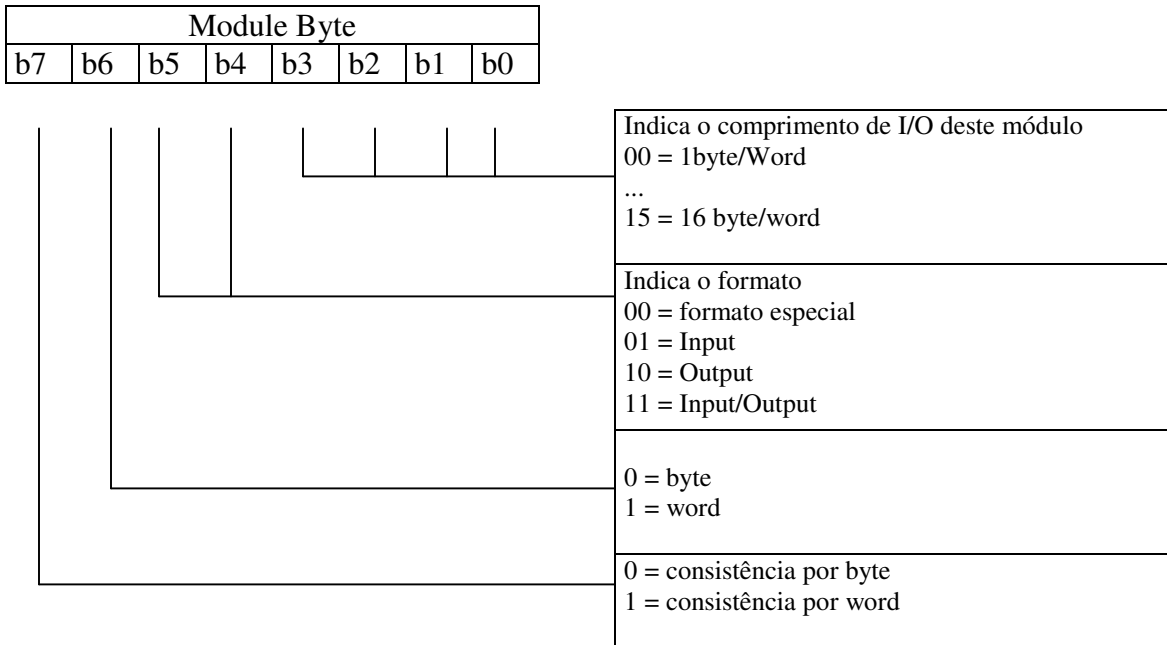


Figura 1 - Identifier Formats simples

Quando o formato for especial, veja *Identifier Formats especiais*.

Exemplo:

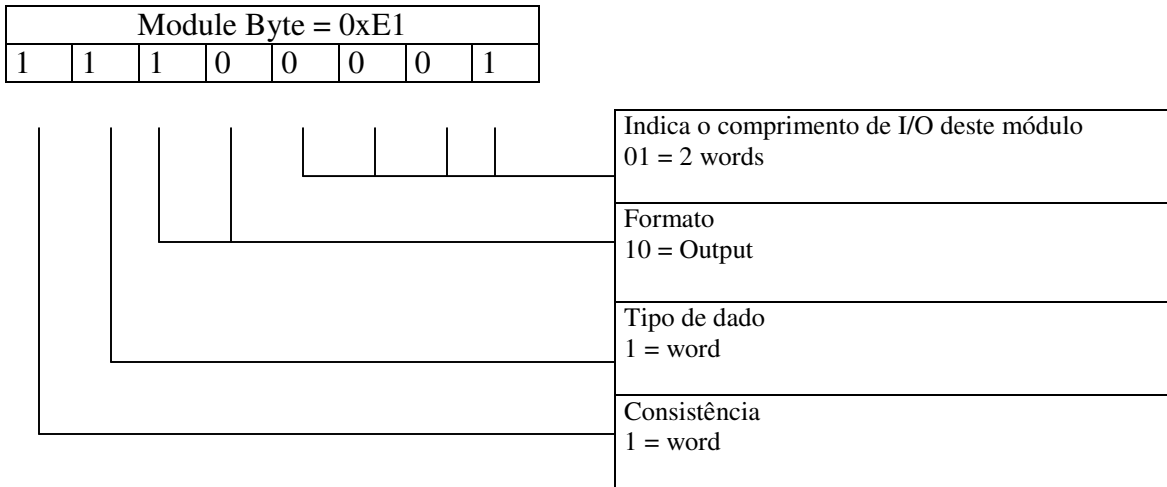


Figura 1a – Exemplo de Identifier Formats simples

Identifier Formats especial

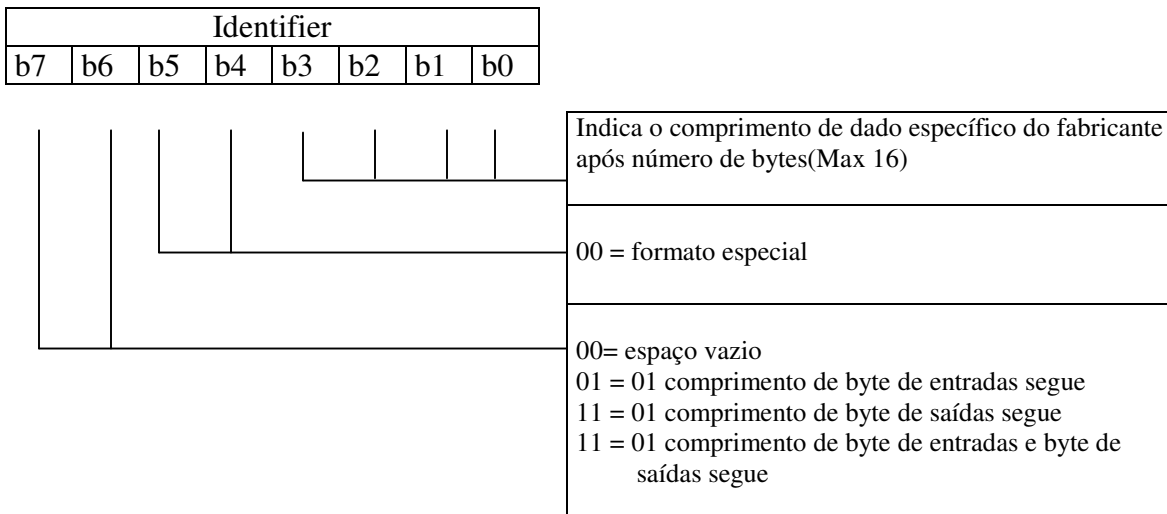


Figura 2 - Identifier Formats especial

Comprimento de bytes para Identifier Formats especial

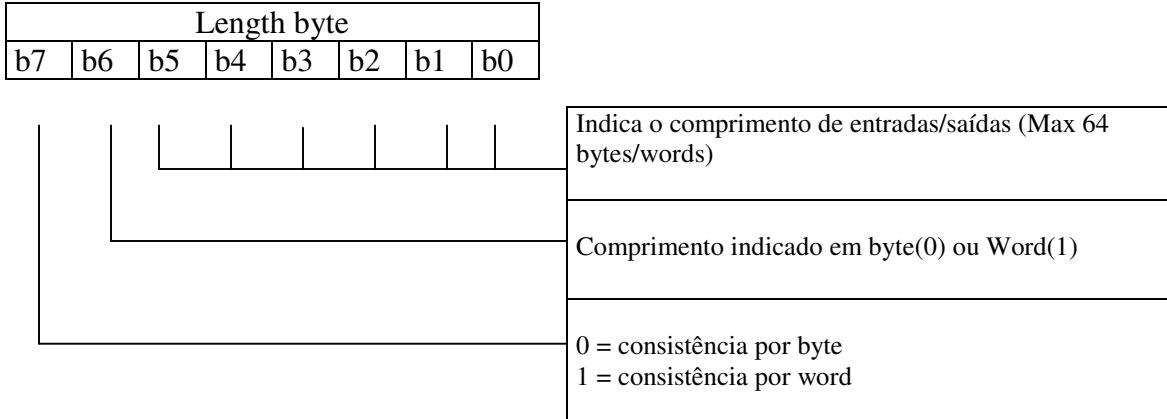


Figura 3 - Comprimento de bytes para Identifier Formats especial

Note que através do formato especial pode-se descrever dados em 8 e 16 bits e que permitem módulos de até 64bytes/words. Vejamos o exemplo abaixo, onde para o campo de identificador temos 1 byte de comprimento de saída, 1 byte de comprimento de entrada em formato especial e um byte de dados de usuário. De acordo com o Length Byte 1, temos 64 words de saída com consistência em Word e de acordo com o Length Byte 2, temos 64 words de entrada com consistência em Word. Depois no Length Byte 2, temos um byte de dados de usuário que é específico e não pode ser descrito como padrão:

Identifier							
1	1	0	0	0	0	0	1
Length byte 1							
0	1	1	1	1	1	1	1
Length byte 2							
0	1	1	1	1	1	1	1
Length byte 3							
X	X	X	X	X	X	X	X

Figura 3a – Exemplo de Identifier Formats especial

Identifier Formats de acordo com Profibus DP-V1

Este formato pode representar 8 e 16 bits de dados e tipos padrões definidos de acordo com o DP-V1. A tabela a seguir lista alguns destes tipos.

Tipo de dados – Profibus DP-V1	Número de bytes	Código
Boolean	1	1
Integer 8	1	2
Integer 16	2	3
Integer 32	3	4
Unsigned 8	1	5
Unsigned 16	2	6
Unsigned 32	4	7
Float Point	4	8
Visible String	1,2,3...	9
Octet String	1,2,3...	10

Tabela 1 – Alguns tipos de dados manuseados de acordo com o Profibus DP-V1

Por dentro do *Identifier Formats* de acordo com Profibus DP-V1

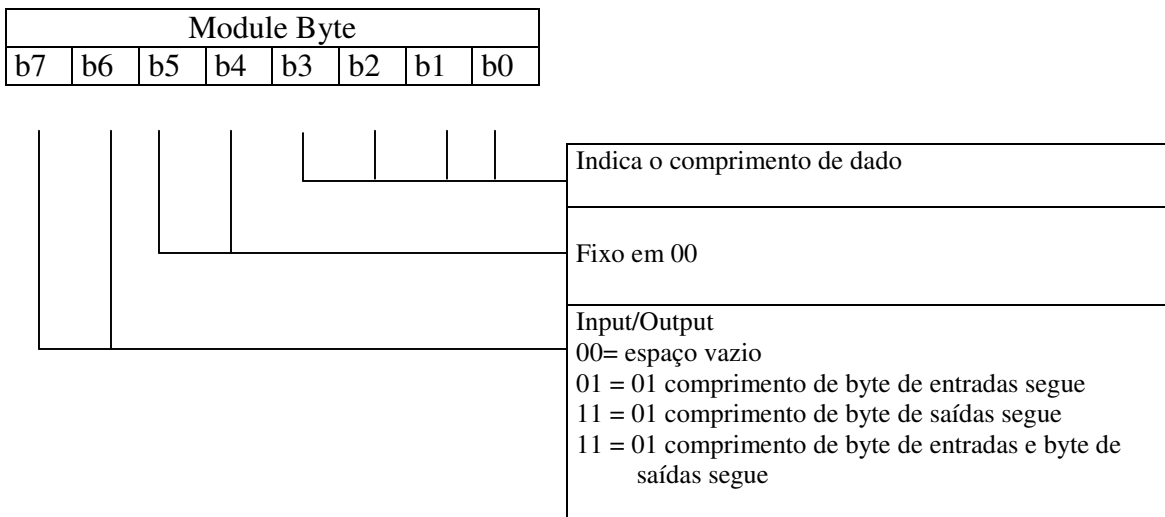


Figura 4 - Identifier Formats de acordo com Profibus DP-V1

Comprimento de bytes para *Identifier Formats* de acordo com Profibus DP-V1

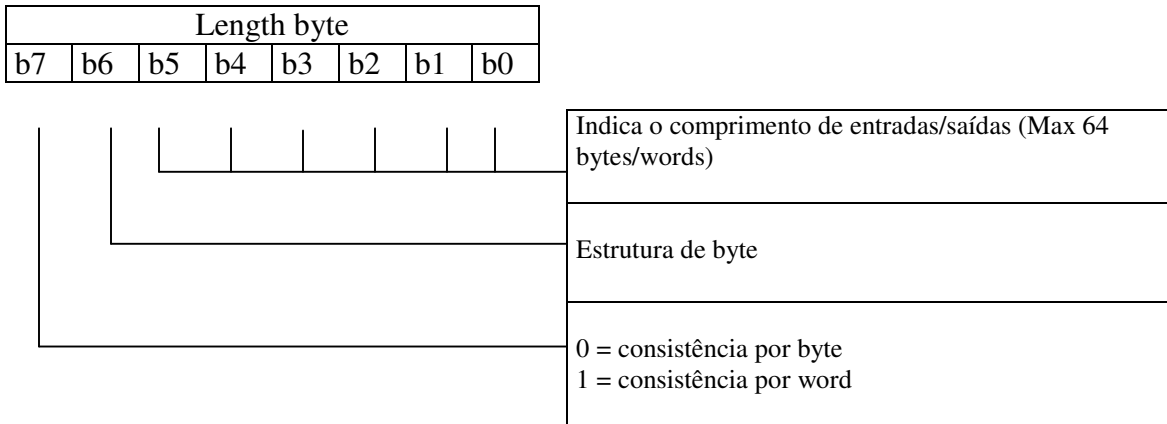


Figura 5 - Comprimento de bytes para Identifier Formats de acordo com Profibus DP-V1

No início do artigo, foi dado um exemplo para o Bloco Analog Input (AI). Note que podemos defini-lo em dois formatos: short e long. Vejamos o exemplo abaixo, considerando o formato long:

Identifier = 0x42							
Formato específico: 2 bytes de entrada							
0	1	0	0	0	0	1	0
Length byte 1 = 0x84							
Formato específico: 2 bytes de entrada							
1	0	0	0	0	1	0	0
Length byte 2 = 0x08							
Tipo de dado: Ponto Flutuante							
0	0	0	0	1	0	0	0
Length byte 3 = 0x05							
Tipo de dado: Unsigned 8							
0	0	0	0	0	1	0	1

Figura 6 – Exemplo de Identifier Formats de acordo com o Bloco Analog Input (AI) do Profibus DP-V1

Consideremos agora, um exemplo para o Bloco Analog Output (AO), onde temos a seguinte configuração possível entre várias:

```
Module = "eRCAS_IN + RCAS_OUT "
0xC4, 0x84, 0x84, 0x08, 0x05, 0x08, 0x05
EndModule
```

Identifier = 0xC4							
Formato específico: 4 bytes de entrada/ saída							
1	1	0	0	0	1	0	0
Length byte 1 = 0x84 5 bytes de saída com consistência por word							
1	0	0	0	0	1	0	0
Length byte 2 = 0x84 5 bytes de entrada com consistência por word							
1	0	0	0	0	1	0	0
Length byte 3 = 0x08 Tipo de dado: Ponto Flutuante							
0	0	0	0	1	0	0	0
Length byte 4 = 0x05 Tipo de dado: Unsigned 8							
0	0	0	0	0	1	0	1
Length byte 4 = 0x08 Tipo de dado: Ponto Flutuante							
0	0	0	0	1	0	0	0
Length byte 6 = 0x05 Tipo de dado: Unsigned 8							
0	0	0	0	0	1	0	1

Figura 7 – Exemplo de Identifier Formats de acordo com o Bloco Analog Output(AO) do Profibus DP-V1

Conclusão

Vimos através deste artigo a importância da interpretação dos *Identifier Formats* na tecnologia Profibus e suas particularidades.

Referências:

- Manuais Smar Profibus
- www.smar.com.br