iCC 1996

3rd international CAN Conference

in Paris (France)

Sponsored by

Motorola Semiconductor
National Semiconductor
Philips Semiconductors

Organized by

CAN in Automation (CiA)
international users and manufacturers group
Am Weichselgarten 26
D-91058 Erlangen
Phone +49-9131-69086-0
Fax +49-9131-69086-79
Email:headquarters@can-cia.de
URL: http://www.can-cia.de

DeviceNet Configuration Using an Electronic Data Sheet

David Brod
Rockwell Automation - Allen-Bradley
Mequon, WI

Scott Braun
Rockwell Automation - Allen-Bradley
Mequon, WI

Abstract - During the development of the DeviceNet Specification, it became apparent that it would be necessary to provide a mechanism for common user friendly configuration of all DeviceNet products. It would have to be robust enough to cover a wide range of products, from the very simple, to the very complex. It would be required to be used by arbitrary configuration tools and be independent of the implementation of these tools. The configuration mechanism would also have to be easy to implement in order to facilitate its adoption by product developers. It would also have to be cost effective to implement so as not to add excessive cost to inexpensive devices. This paper describes how the Electronic Data Sheet achieves these goals.

I. INTRODUCTION

DeviceNet[â] is a communications network that connects industrial devices; such as, actuators, sensors, push buttons, motor drives, and programmable logic controllers. DeviceNet allows for device configuration across the network and for embedding configuration or diagnostic data in devices. Any device containing configuration data accessible through the DeviceNet communications interface requires a configuration tool to access and possibly modify the data. A device configured only through the use of external switches, jumpers, thumbwheels, or other propriety interfaces may still require a means for a tool to access and determine the state of the hardware configuration switches.

The DeviceNet specification allows data residing within a device to be represented as arbitrary data with arbitrary object paths (Class/Instance/Attribute). No limitations are placed on the attribute path, nor the meaning of the configuration data of a device. In order to access and interpret the data stored in the device, either a configuration tool or the user will require knowledge about the data available in the device. This paper describes four methods of accessing and interpreting internal device data:

- Custom configuration software
- Device data sheet
- Parameter objects
- Electronic Data Sheet

Each method is reviewed in the paper with special emphasis on the Electronic Data Sheet.

A. Custom Configuration Software

A possible method of configuring a device requires the user to purchase and utilize product specific custom software to configure the device. This method burdens the device developers to develop both the actual device and the configuration software. Customers of these types of devices benefit from a configuration tool highly responsive to the demands and needs of that vendor's specific type of device.

An alternative to product specific custom configuration software, would be for vendor specific or Special Interest Group (SIG) specific software. Vendor specific software would provide access only to those devices sold and distributed by a specific vendor. This would allow for a specific vendor to provide unique value to their devices that may utilize particular vendor specific objects, services, or attributes possible on DeviceNet. The software would need to be somewhat generic in order to address possible differences between unique product types.

SIG specific configuration software allows multiple vendors of similar products the ability to provide a common configuration or commissioning software tool by specific product category. All vendors supporting the documented SIG structures could then utilize a single tool that was highly optimized to the needs of the users of those products.

The disadvantage of any custom configuration software tool is the inability to address a complete solution to all possible devices on the network. The application of custom tools has a negative side effect of limiting access or capabilities within the open system. Therefore, the specific customization features of the tool works against the benefits of the actual product when used in conjunction with other unsupported products.

DeviceNet supports the ability to develop and implement custom configuration software, but also realizes additional benefits from the ability to provide standard methods to promote open structures for configuration of any and all devices.

B. Device Data Sheet

[â] DeviceNet is a trademark of the Open DeviceNet Vendor Association, Inc.

A simple and generic method of configuring a device requires the user to refer to a device's data sheet provided by the manufacturer. This data sheet describes the data available in the device and how to access it. In this situation, the user has to tell the configuration tool how to access the data by specifying the data size, the desired service, and the class, instance, and attribute index by numeric or symbolic reference. The service specifies what action the tool is to perform, such as getting or setting a value. The class, instance, and attribute specify the location of a particular datum stored in the device. However, the configuration tool would not be able to format nor provide any additional information about the configuration or diagnostic data other than the user's interpretation of the returned value. Therefore, data sheets alone would not provide an acceptable user-friendly method for configuring devices.

C. Parameter Objects

A device manufacturer can embed all of the required information concerning the configuration or diagnostic data within the device through the use of parameter objects. Parameter objects contain an open and optional method to define a common data structure accessible to any DeviceNet product developer. Parameter objects provide a common public interface to a device's accessible data where each parameter object corresponds to a single configuration or diagnostic datum. The value attribute of the parameter object may reference itself or directly reference an attribute of another object as specified by the link path attribute of the parameter object.

A full parameter object contains all of the information necessary for a configuration tool to access, display, and modify any product's datum:

- link path to the referenced datum value
- descriptor, data size, and data type
- data limits and default values
- formatting information
- descriptive text

A device that provides access to its data by implementing full parameter objects enables a configuration tool to provide online user-friendly device configuration. However, full parameter objects require additional memory and resources in the device. Therefore, the developer of a simple, low cost device may choose not to implement parameter objects due to memory and/or cost constraints.

A partially defined parameter object, called a parameter object stub, excludes some of the information included in a full parameter object; such as, descriptive text, datum limits, and scaling information. However, the parameter object stub can still establish a link to the configuration data in the device. The limitation of this method is that tools cannot display the datum values in user acceptable units without the aid of additional product specific software components to provide the necessary missing attributes.

D. Electronic Data Sheet

The DeviceNet standard defines a specially formatted ASCII text file, or standardized software component, called an Electronic Data Sheet (EDS) to provide an open method of defining the accessible data and the input/output (I/O) characteristics of a device. The format of the data in the EDS is independent of the configuration tool or file system.

An EDS provides information necessary for a configuration tool to access diagnostic data or modify the configurable data within a device. A device manufacturer can incorporate specific information in an EDS to enable configuration tools to provide the following features:

- visually format the configuration and diagnostic data
- automate limit checking of user entered values
- provide the user with a selection of choices for each configurable datum
- help automate the device configuration process

Non-standard information may also be included in the EDS through the use of comment lines. However, this information may not be interpreted by the configuration tools.

In addition, configuring devices via an EDS adds no cost to the devices themselves, except for the possible cost of transferring the EDS files to the user. Depending on the method or methods chosen, this transfer cost can be minimized.

The remainder of this paper describes device configuration using an EDS and a brief overview of the EDS format. The Appendix contains an example of an EDS. The reader is referred to the DeviceNet specification for a complete description of the EDS file format.

II. EDS File Specification of a Device's Data

A configuration tool, designed to utilize Electronic Data Sheets, reads and interprets an EDS file to determine what configuration or diagnostic data is stored in the device, where to access the data, and how to display or modify the data in the device. Basically, the type of information stored in an EDS file determines the level of functionality that a configuration tool can provide to the user. An EDS represents the device's configuration or diagnostic data by utilizing an ASCII representation of either the full parameter object or the parameter object stub.

The use of parameter object stubs in an EDS allows a configuration tool to access and to modify the configuration data in the device. However, the user will still require a data sheet to determine the data's context, content, and format. An EDS parameter object stub provides the following information:

- link path (pointer) to the corresponding datum
- descriptor, data size, and data type

The link path points to a specific datum in the device. This referenced data corresponds to the value attribute of the particular parameter object or to an attribute of another object implemented in the device.

The use of full parameter objects in an EDS provides additional information about the device's configuration data:

- data limits and default values
- formatting information
- descriptive parameter names and units
- optional help text

This information allows a configuration tool to access and to display the product's data in the format specified by the EDS. This format includes the ability to display descriptive text strings. If the data is configurable, then the tool can prompt the user with a list of choices or data entry fields to modify the data. In addition, the tool can validate the user's data choices.

A device manufacturer may decide to implement parameter object stubs in the device to provide a known public interface to the device's configuration data for simple network tools, while still providing an EDS file to supply the remaining parameter information for more complex tools. This hybrid approach minimizes the amount of memory required by the actual device to store all the database information internally while still providing some degree of information on-line.

III. Creating Electronic Data Sheets

Electronic Data Sheets can be created by a product development group via a simple text editor. These files can be shipped with the DeviceNet product or made available over a customer accessible bulletin board or an Internet server. It is also acceptable that the device manufacturer only provide a printed listing of the EDS and thereby require the user to create the EDS by text entry. However, this method places an unnecessary burden on the end user, can lead to customer typographical errors, and is strongly discouraged.

Devices that support full access to their database over DeviceNet by supporting Parameter objects, can have a major portion of their product's EDS file uploaded directly over the network. This method frees product developers from having to issue computer media containing the EDS file with their products or from requiring users to access electronic bulletin boards to download the EDS files. However, the user would be required or prompted to specify the device's I/O characteristics since these can not be uploaded from the device in a common, open method today.

Any of these methods, or a combination of any of them will provide any DeviceNet tool the knowledge and understanding to access and configure the network devices.

IV.EDS File Organization

The EDS file has several sections that describe both the file and the device for application by the configuration tool. The following figure, Fig. 1, shows the structure of an EDS file. The bold text (within brackets) in Fig. 1 are keywords defined by the EDS specification that indicate the start of a section.

[File]
File Description Section

[Device]
Device Description Section

[IO_Info]
I/O Characteristics Section

[ParamClass]
Parameter Class Section

[Params]
Parameter Section

[Groups]
Parameter Groups Section (optional)

[EnumPar]
Parameter Enumeration Section (optional)

Fig. 1 - EDS File Structure

A. File Description Section

The file description section contains information about the EDS file; such as:

- File Description Text (required)
- File Creation Date and Time (required)
- Last Modification Date and Time (optional)
- EDS Revision (optional)

A configuration tool can read this information, format it, and display it to the user. A user can modify the file description information in the EDS to track any changes the user may make to the file.

B. Device Description Section

The Device Description section contains information about the device. This information can be compared to the actual device on the network by the tool to insure that the EDS matches the actual device to be configured. This information could also be referenced for use by other device's who may be responsible for the application of this device or other devices in the system. This feature could provide for a safety mechanism to insure correct network behavior as new or different devices are added or replaced. The following lists some of the entries located in this section:

- Vendor Name
- Product Type
- Product Code
- Revision
- Catalog Number

C. I/O Characteristics

A DeviceNet device can produce and consume I/O data over a connection. In order for a configuration tool to be able to interpret the device's I/O capabilities, an EDS file must specify the I/O characteristics or behaviors of a DeviceNet device. The available choices for I/O behavior are a part of the EDS file specification. The EDS may contain the following information:

- Default I/O Information (required)
- I/O Poll Information (optional)
- I/O Strobe Information (optional)
- Device's Producing Connection (optional)
- Device's Consumer Connection (optional)

D. Parameter Class Section

The parameter class section contain the following information:

- Maximum Instances
- Parameter Class Descriptor

Maximum Instances specify the number of parameters defined in the EDS. The Parameter Class Descriptor specifies if the device supports parameter objects. If the device supports parameter objects, then the descriptor also specifies whether full parameter objects or parameter object stubs are implemented and if device data are stored in non-volatile storage within the device.

E. Parameters Section

The parameters section identifies all of the configuration or diagnostic data in a device. The data fields follow the format of the parameter object. Each entry in the parameter section corresponds to a specific instance of user data in the device. The actual data may correspond to an actual parameter object value in the device or to any attribute of another object. Each parameter entry contains the following required information:

- Parameter Value
- Path and Path Size
- Descriptor, Data Type, and Data Size

The path is a text string that describes the location of the data in the device while the path size specifies the number of characters in the path string. If the path size is zero and the path string is empty, then the parameter entry references a parameter object in the device. The descriptor, data type, and data size specify the components of the parameter value that give the parameter value scaleability. The descriptor also specifies whether the parameter object supports scaling and if the value attribute can be set by the user.

A parameter entry may contain additional information to allow a configuration tool to provide more useful visual interpretation to the user:

- Parameter Name, Units, and Help Strings
- Minimum, Maximum, and Default Values
- Scaling Information
- Scaling Links
- Decimal Precision

A configuration tool can use the descriptor, scaling information, and decimal precision to format and display the data. The minimum and maximum values allow a tool to validate a user's data choice. Scaling links optionally specify that a parameter value is scaled based on the data value of another parameter.

F. Parameter Groups Section

The parameter groups section identifies all of the parameter groupings in a device. Each parameter group can contain a list of the parameter object instances defined by the product. Parameter groups allow a configuration tool to present to the user several configuration data elements as a named or referenced group. This method provides a mechanism to segment similar parameter functions together to aid in the configuration or diagnosis of the product when applied.

Parameter groups can also be read or interpreted directly from the product through the available 'Parameter Group' object. Support of this object is beneficial, but optional.

A unique feature of the parameter group section is that end users can create their own groupings by editing the EDS. The addition or removal of particular groups can provide a customer with a method to customize a product interface or configuration tool to fit particular needs or situations. The user may also desire to change the names of the parameters or parameter groups to fit the application needs.

G. Parameter Enumeration Strings Section

The parameter enumeration strings section defines an enumerated list of textual strings for each parameter that supports enumeration strings. Each string represents a specific parameter value where subsequent entries increment by one. Enumeration allows a configuration tool to present a parameter value as a text string and provide the user a textual list of choices. This method provides the tool the ability to give the user a high level of parameter interpretation when accessing and utilizing an appropriate datum.

V. Conclusion

The EDS specification provides a common, consistent and compatible approach that enables network tools to interpret and access configuration and diagnostic data from devices connected to a DeviceNet communication network. An EDS file provides the mechanism that allows DeviceNet to provide an open and flexible means to standardize on a format to provide consistent interpretation of a device's accessible data.

Appendix

The following listing shows a sample Electronic Data Sheet.

```
$ Description:    The following file is the EDS
$ for a DeviceNet Photoelectric Sensor
$
[File]
  DescText = "DeviceNet Photoelectric Sensor EDS File";
  CreateDate = 4-22-95;
  CreateTime = 10:00:00;

[Device]
  VendCode = 99;
  ProdType = 6;
  ProdCode = 10;
  MajRev = 1;
  MinRev = 4;
  VendName = "XYZ Sensors, Inc.";
  ProdTypeStr = "Photoelectric Sensor";
  ProdName = "Transmitted Beam Receiver";
  Catalog = "";

[IO_Info]

  Default = 0x0002;              $ Strobe only
  PollInfo = 0, 0, 0;            $ Not supported
  StrobeInfo = 0x0002, 1, 1;     $ Use Input1 / Output1

  Input1 =
     1,                          $ 1 byte
     1,                          $ 1 bit used
     0x0002,                     $ Strobe only
     "Sensor Output",            $ Name
     6,                          $ Path size
     "20 04 24 01 30 03",        $ Path to value attribute
     "Output value.";            $ Help string

  Output1 =
     0,                          $ 0 byte
     0,                          $ 0 bit used
     0x0002,                     $ Strobe only
     "",                         $ Name
     0,                          $ Path size
     "",                         $ Path to value attribute
     "";                         $ Help string
```

```
[ParamClass]
  MaxInst = 2;
  Descriptor = 0x09;

[Params]
  Param1 =                       $ Operate Mode
     0,                          $ Data Placeholder
     6, "20 0f 24 01 30 01",     $ Path size and Path to Attr
     0x02,                       $ Descriptor
     2, 2,                       $ Data Type and Size
     "Operate Mode",             $ Name
     " ",                        $ Units (Not Used)
     ".",                        $ Help
     0,1,0,                      $ min, max, default
     1,1,1,0,                    $ scaling factors
     1,1,1,0,                    $ scaling links
     0;                          $ decimal places


  Param2 =                       $ Output
     0,                          $ Data Placeholder
     6, "20 0e 24 01 30 01",     $ Path to Output Attribute
     0x12,                       $ Descriptor
     4, 1,                       $ Data Type and Size
     "Output",                   $ Name
     " ",                        $ Units
     " ",                        $ Help
     0,1,0,                      $ min, max, default values
     1,1,1,0,                    $ Scaling
     1,1,1,0,                    $ Scaling links
     0;                          $ decimal places

[EnumPar]
  Param1 =              $ Output Mode Enumerated Strings
     "Light Operate",          $ For value = 0
     "Dark Operate";           $ For value = 1

  Param2 =                     $ Output Enumerated Strings
     "Off",                    $ For value = 0
     "On";                     $ For value = 1
```