

## DeviceNet Development Considerations

By

**Viktor Schiffer**, [vschiffer@ra.rockwell.com](mailto:vschiffer@ra.rockwell.com)

**Ray Romito**, [rromito@ra.rockwell.com](mailto:rromito@ra.rockwell.com)

---

### Disclaimer:

The following paper is not claiming to solve any of the problems you might encounter during the development of your DeviceNet product. Any decision you take in favor of or against a particular design concept is done at your own risk.

---

### Development considerations:

*Please consider the following issues before starting a development:*

1. What functionality does the product require today and in future applications?
  - Slave functionality
  - Master functionality
  - Peer-to-peer messaging
  - Combination of the above
2. What are your physical layer requirements? Is IP 65/67 required?
3. What type of hardware should be chosen for this product?
4. What kind of software should be used for this product?
5. Should you develop hardware and/or software internally or have it designed by an outside company?
6. What are your configuration requirements? Do you intend to have your product tested for conformance and interoperability?
7. What design and verification tools should be used?
8. What is an absolute must before your products can be placed on the market?

*Let me point out the major details associated with the above issues one by one:*

### Issue #1, Product functionality:

Most DeviceNet products on the market are and will be products with slave functionality only. Therefore, during most of the discussion within this paper, a slave design is being considered. If you do consider designing a product with master functionality we suggest you contact one or several of those companies offering services for DeviceNet. A list of such companies can be found in the annex to this developers guideline or in the ODVA product catalogue.

The first thing to consider during a slave development is the I/O communication method. During the initial phase of DeviceNet, only bit-strobe and polled I/O communications were defined and available in master products. However, with more master products becoming available that have

Change of State (COS) or Cyclic I/O data exchange methods incorporated (e.g. all Allen-Bradley scanners), these I/O data exchange methods should be considered, even though today's customers may not request these yet. In fact, these connections are preferred because of their superior use of the bandwidth.

Bit-strobe I/O is mainly used for sensor devices or other slaves that have little (1 bit) or no output data requirements. Apart from such devices, Bit-strobe may be used for output data synchronisation across several devices. However, since this use of the strobe frame is not standardised in the DeviceNet spec, such applications may be limited to your devices only.

Polling is the "bread and butter" type of I/O exchange. It should be considered for all implementations since this method is supported by all master devices.

Change of state I/O data exchange is a very powerful method to increase throughput of the network and reduce network loading at the same time. These methods allow to make full use of the inherent multi-master capability of the CAN protocol and should therefore be considered for all new developments.

While bit-strobe is limited to a maximum of 1 bit of consumed data and 64 bits of produced data, polling, change of state and cyclic I/O data exchange methods do not have such limitation. Consider using I/O data fragmentation if your I/O requirements are higher than the 64 bits (8 bytes) of the native CAN frame. The user of your device will not have to deal with any fragmentation details since fragmentation and re-assembly are automatically performed by the DeviceNet protocol.

The second consideration should go to the explicit messaging functionality of the device. The DeviceNet protocol requires all devices to support explicit messaging. As a minimum, the Identity, DeviceNet and Communication objects must be accessible through explicit messaging to the extent specified as mandatory requirements in the DeviceNet spec. However, if your configuration requirements go beyond setting a few switches, you should consider configuration of the device through explicit messaging. The use of proprietary configuration ports is strongly discouraged. See #6 for further configuration considerations.

Fragmentation, although not a must, should at least be considered for explicit message responses to allow to make full use of the product name field with up to 32 characters. If you intend to support features like configuration download or firmware update through the DeviceNet port, you must use explicit message fragmentation for both transmitted and received messages.

## **Issue #2, Physical layer requirements:**

The DeviceNet spec allows four types of connectors: mini, micro, open style and screw terminals. If ever possible, use mini, micro or open style connectors to allow plug and play installation with readily available cabling components. All non-IP65/67 devices should use the opens style connectors available from Phoenix and other manufacturers. The use of screw terminals should be restricted to those cases that cannot use any of the other methods. Please be aware that the screw terminal connectors also have to support disconnection of the network without severing the trunk. If you do not use mini, micro or open style connectors, it is recommended to contact one of the conformance test labs during an early stage of the development for their opinion on the conformity of the connection method considered. Please make sure that only gold plated connectors are being used.

If you should consider converting an SDS product to DeviceNet or implement the SDS protocol in your device at a later stage, be aware of the fact that the pinning of the micro and open style connectors differ from those defined in the DeviceNet spec. Also, the electronic circuitry of

existing SDS and CAN open devices may not meet the mis-wiring protection requirements of DeviceNet so that further modification is required. To the best of our knowledge, the mis-wiring protection circuitry required for DeviceNet does not preclude their use in an SDS or CAN open environment.

Only the 125, 250 and 500 kBaud data rates are supported in DeviceNet. The 1 MBaud supported by some other CAN based networks is not supported due to the severe network length limitations associated with this baud rate. DeviceNet does not require that all 3 rates be supported, though a product that does not support all is certainly at a market disadvantage.

DeviceNet requires transceivers that exceed the ISO 11898 requirements, mainly due to the extension to 64 physical devices on the DeviceNet link. Therefore, only those transceivers that meet the DeviceNet transceiver specifications may be used. Devices that meet these specifications are:

- Philips 82C250
- Philips 82C251
- Unitrode UC5350

The DeviceNet physical layer may be built with or without isolation between the transceiver section and the CAN chip section of the device. Whether you use an isolated physical layer or not typically depends on the type of product. Devices that have no electrical connection to the outside world and that are completely powered from the bus, e.g. sensors, typically do not need isolation, while devices with outside connections will almost invariably need to have physical layer isolation. The speed of the opto isolators used is critical, since it will add to the overall transceiver delay. The specification requires a maximum of 40ns delay through the opto isolators. Remember that a fast opto doesn't necessarily mean a low propagation delay. The typical parts used on DeviceNet nodes today are 500Mhz parts.

All DeviceNet nodes are required to power the CAN transceiver section from the bus. This makes sure that there are no unpowered transceivers attached to the network that might disturb the data traffic. The reason that traffic would be disturbed is because the transceiver impedance goes to about 1/2 its powered value. This will severely load the signals on the bus.

### **Issue #3, CAN/microprocessor hardware:**

Since the use of various CAN and microprocessor chips is a very broad subject, only some general do's and don'ts can be given within the scope of this paper:

Do not use a SLIO. There is no way you can incorporate the minimum requirements of the DeviceNet protocol in today's SLIO devices.

- All CAN chips that use the 11 bit identifier can be used. The use of long (29 bit) identifier is neither required nor tolerated in a DeviceNet network.
- Basic CAN chips can be used with good success for slave devices when restricting this device to a group2 only slave. The group 2 only aspect of the protocol has been optimised for basic CAN.
- While the use of a microprocessor with integrated CAN chip will minimise component count, this may only be recommended in those case where the micro/CAN chip combination exactly fits the device requirements. Choosing a CAN chip independently from a microprocessor will allow more flexibility in the design. The availability of an in-circuit emulator for any of the micros with integrated CAN chips should also be considered before taking such a decision.

- Every DeviceNet node must accommodate a 32 bit serial number that is unique for each and every device from this manufacturer. Therefore alterable non-volatile may be required for your device. If you intend to have settable parameters implemented in your device, non-volatile storage is required anyway.
- Pay particular attention to the state of the CAN H and CAN L lines during times when the CAN controller is reset and during power up/down. CAN chips will drift or be driven to different levels during these times which can result in the bus being driven dominant. The use of passive pull ups or pull downs, control register set-up and an inverter in the path from TXD to the transceiver can be used to guarantee that these conditions result in a harmless recessive state on the CAN lines.
- Do not allow the unused inputs to the CAN controller (RX0 or RX1) to float. Tie them to Vcc/2 on the transceiver or use a voltage divider. Leaving this input to float will almost certainly cause error frames. Also, some CAN controllers that provide a register to turn off this unused input appear to still be subject to this phenomenon, even when the pin is disabled. The safe bet is to tie it to the proper level.

#### **Issue #4, Software to be used:**

While nobody is required to buy DeviceNet software from anybody, there are various DeviceNet software packages on the market that can successfully be incorporated into DeviceNet products. The decision to use a particular software package should be made based on the features implemented and the support provided by the vendor. Pricing can be an issue but generally speaking you only get what you pay for.

*Some of the basic issues to consider are:*

- Is the software under consideration usable for my hardware?
- Is there any assembly code that needs to be rewritten?
- To what extent will hardware drivers have to be rewritten?
- Is the speed performance suitable for my application?
- Are all communication features (I/O and Explicit Messaging) supported that are required for my application?
- Is fragmentation supported (if required for my application)?
- What compiler is to be used? Is this a compiler I am familiar with?
- Is EDS support available?
- What support can I expect from the vendor of this software package?

#### **Issue #5, Buy vs. make:**

This is the common make vs. buy decision that every vendor of electronic equipment has to take. This decision has to be taken under consideration of the following issues:

- Is there enough in-house knowledge of the base technology, i.e. CAN and microprocessors?
- Is this a one-time implementation or do I expect further modifications and enhancements of this product?
- Simple slaves can easily be implemented in-house (some companies have managed to reach basic device functionality within a couple of weeks), but more complicated devices, particularly devices with master functionality are recommended to be designed on the basis of commercially available packages.

#### **Issue #6, Configuration requirements:**

Configuration of DeviceNet devices can be supported by a powerful tool called EDS (Electronic Data Sheet). It is strongly recommended to generate an EDS even if no or very few parameters are to be accessible for modification.

The configuration of a device can be split up into two areas.

The first area is the setting of DeviceNet communication related parameters. These are the baud rate and the MAC ID. Many devices set these parameters by mechanical switches while others may allow access via the DeviceNet link. This process is called "device commissioning" in the Allen-Bradley DeviceNet Manager configuration software. Still other products provide for auto baud rate detection in their products. This simplifies the set-up for customers somewhat, but it's clearly evident that some device must be capable of establishing the rate on a network. In other words, auto baud requires at least one node to assume a baud rate, or be pre-configured for baud rate. For this reason it is suggested that the master device not implement auto baud detection to provide a known source for baud rate.

The second and larger area of device configuration is the setting of application related parameters. If an EDS exists for the device to be configured, all parameters (settable and read-only) can be displayed together with text and help instructions. Modifications to the parameters can be done in plain English and actual formatted units, no messing with bits and bytes is necessary. The parameter display within the "Enhanced Configuration" of a device will also allow on-line monitoring of the actual parameters. This allows monitoring of process values provided these are mirrored to parameters accessible through the EDS parameter list. Please note, however, that these parameters will be read through explicit messaging, therefore fast transients may be missed while static values present no problem.

EDS files are easy to create, and if the product supports instances of the parameter class, one can even be created on line by the DeviceNet Manager software. The DeviceNet specification, Volume 2, Chapter 4 contains the details and examples of what goes into this file.

## **Issue #7, Conformance testing:**

Conformance testing is possible through official ODVA test sites today. It includes protocol compliance as well as physical layer compliance. Both tests are pass/fail. Interoperability testing is evolving, but no official test procedure has been agreed upon so far. Results of the interoperability test are provided to the vendor, but are not pass/fail.

ODVA or DeviceNet does not force you to perform any conformance testing, although it is highly recommended and is even required by some customers. Passing conformance testing will enhance your customers' trust in your product and it will help you design a better product. The conformance test is typically done in two steps:

In step #1 the product is described with all object details by the designer using the description tool provided in the conformance test software (available from ODVA). This description is required for the conformance test software to test all implemented features of the protocol. The developer can perform a pre-compliance test during the development of the product since the conformance test software can run in a development mode that allows testing of certain aspects of the protocol only. It is strongly suggested that use of the compliance test software begin in the very early stages of code debug. The Conformance SIG visualises this as a development tool as well as a protocol verification tool.

Once the product passes the tests in the developer's lab, it can proceed to step #2 by registering the product for conformance testing with the ODVA. The developer then has to contact one of the ODVA approved test labs to arrange for a test session. No representative of the developer needs

to be present during the actual test although this is recommended to quickly sort out minor problems.

If the product passes, then the result may be published by the ODVA, if the product fails, only the test lab and ODVA will know.

### **Issue #8, Design tools:**

This summary does not intend to provide a complete list of products required for the design and verification of your product, it is rather meant to point out which type of tools are needed.

It is generally assumed that you are sufficiently equipped to successfully perform a microprocessor development. Therefore, this guide will only talk about DeviceNet (CAN) related tools.

As a minimum, you need to have some kind of CAN monitor. This is typically a CAN card sitting in a PC together with the appropriate software tool. DeviceNet compatible PC cards are available from companies like Softing, STZP, Huron Networks, S-S Technologies and others. Consult the latest edition of the ODVA product catalogue or the online version on the WWW to find a complete list of vendors offering such cards. There will be some variation in price and performance. Before taking a decision for a specific vendor, make sure his product is compatible with your PC and all types of software you intend to run on this card.

DeviceNet or CAN monitor software packages vary greatly in price and performance. A typical low end tool is the Allen-Bradley Slave Development Tools while the Vector Informatik CANALYZER clearly is a high end tool with a price tag of roughly ten times of what you pay for the Allen-Bradley product. Generally speaking, you only get what you pay for. Therefore, a low end tool may be a good starting point and usable throughout the whole development of a simple product, but complex products will certainly require more sophisticated tools. Again, the ODVA catalogue (paper or on-line) is a good source of information. If you accept working on the CAN level only, then there are many more companies supporting monitoring in a layer 2 CAN level. CiA (CAN in Automation) can help you locate such products.

Once your product becomes operational, you may consider running it in a more typical industrial control environment. Your first priority should be to run it in the environment that will be typical for most of your products. This should include running a configuration tool to see how your product reacts to explicit messaging requests to change configurable attributes in your device. This will also allow you to check the EDS.

### **Issue #9, Minimal formal requirements:**

DeviceNet is a very open network with very few restrictions. However, a few rules have to be considered, typically before starting a development, that have to be complied with before the product may be placed on the market.

The only registration required is the registration for a vendor ID. This vendor ID is required to distinguish your products from other vendors' product on an existing network. You get your vendor ID number from the ODVA after purchasing the DeviceNet specification, signing the included terms of usage agreement and returning it to the ODVA. Once you have received your vendor ID number this has to be used for all products sold by your company. Different divisions may ask for different vendor IDs, but they will have to purchase the spec and sign the terms of usage agreement separately.

*Please be advised that no conformance testing may take place unless a vendor ID number has been assigned to your company by the ODVA*