Quick Restart of Disconnected DeviceNet Slave Devices

Viktor Schiffer, Rockwell Automation

In a standard DeviceNet system, reconnecting an existing slave device to a master can take between 2 and 10 seconds due to certain minimum wait times specified for the Master/Slave Connection Set. These wait times and the consequent safeguarding are acceptable for most DeviceNet systems where device replacement in a running system only occurs when a defective device needs to be replaced. However, such a long wait time is by far too much for highly dynamic systems with frequent device changes such as robots with exchangeable tools that contain active devices. To overcome this issue, the DeviceNet Quick Connect mechanism was invented. This mechanism bypasses the long wait times of standard devices and thus results in reconnect times of well under 1 second. This paper will discuss the standard wait times, explain the Quick Connect mechanisms and demonstrate the improvement in real applications.

Introduction

When DeviceNet was created more than 10 years ago, a logical communication access mechanism, called Duplicate MAC ID Algorithm, was included in the specification to prevent nodes going online when one or several duplicate node addresses are present on the network. The mechanism as such was never changed substantially, so it is still described in the specification today [1]. When this mechanism was conceived, the associated timing was made similar to what is required at system start anyway. At the same time, the associated long time-outs made sure that false detection of duplicates is minimized and that any further attempts to use an existing node number are safely detected and handled properly. Furthermore, the mechanisms that activate a slave device in an application were designed in a way that also needs substantial time, but still in the same order of magnitude as the overall system startup time.

However, newer applications, e.g. tool change on a robot require much faster start-up times. Therefore, a mechanism was defined that allows for start-up times that are appropriate to these applications. This mechanism was called “Quick Connect”.

Start-up times under normal conditions

As shown in Figure 1, the time required for a DeviceNet slave device to become operational in an application basically consists of three sections:

1. The time required for self test of the device until it can start communicating on the network.
2. The time required for the Duplicate Node ID Check Algorithm.
3. The time it takes for a scanner device to establish an operational connection to the slave device.

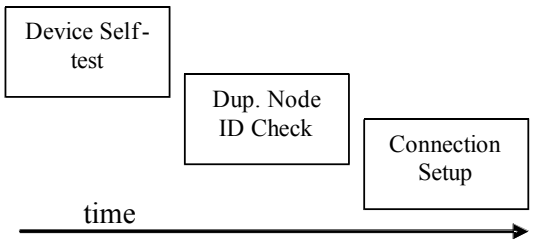

Figure 1: DeviceNet Start-up

The first value (device self-test) is independent of the network, so it will not be discussed in this context. The time required for the Duplicate Node ID Check Algorithm is defined in the DeviceNet specification [1]. Without Quick Connect, it

always takes 2 seconds to complete and there is nothing that can be varied per spec. The third part of the overall startup is variable. The worst case time it takes to establish an operational connection depends on how often the scanner tries to establish it. As long as no device responds, the scanner must keep repeating the following procedure:

1. Send a UCMM open message to establish an Explicit Messaging Connection to the slave device.
2. Wait at least 1 second.
3. Send a UCMM open message to establish an Explicit Messaging Connection to the slave device.
4. Wait at least 1 second.
5. Send an Allocate Message to allocate the Master/Slave Connection Set at least for Explicit Messaging.
6. Wait at least 1 second.
7. Send an Allocate Message to allocate the Master/Slave Connection Set at least for Explicit Messaging.
8. Wait at least 1 second.
9. Go to step #1

The 1-second delays are laid down in the spec; they cannot be shortened without violating the specification. The overall time for connection establishment may thus vary between a few milliseconds and about 3 seconds, depending on when the slave completes the Duplicate Node ID Check Algorithm with respect to the above cycle.

Start-up times under normal conditions

To overcome this deadlock, a modification of the start-up procedure was defined, called “Quick Connect”. There is nothing the DeviceNet specification can do about the device self-test, so the specification enhancement concentrates on the timing laid down in the spec. Both sides of the communication relationship (Master and slave) have to be improved to get the maximum possible delay reduction:

Slave modification:

The Duplicate Node ID Check Algorithm as described in the DeviceNet specification allows any node to listen only to Duplicate Node ID Check Responses during the two seconds of the procedure, i.e. while this check is running. This was done to avoid a device becoming operational before the procedure was completed. For Quick Connect, this wait time is not completely abolished, but a device, e.g. a slave, may already listen to other messages that try to open a connection. If a Duplicate Node ID Check Response shows up later, the device still goes to the offline state as it would do without Quick Connect.

Scanner (Master) modification:

With Quick Connect enabled, the scanner listens for the Duplicate Node ID Check request of the missing slave. As soon as the scanner sees this message, i.e. the first “sign of life” of the missing device, it interrupts the sequence described above (UCMM open, wait, UCMM open, wait, Allocate, wait, Allocate, wait etc.) and sends a UCMM open request and an Allocate request in short succession. The slave picks up whatever is appropriate; a UCMM-capable slave picks up the UCMM open request, the Group2 only slave (non-UCMM-capable) picks up the Allocate request. The scanner recognizes the slave type (with or without UCMM) and carries on with the communication setup according to the slave type detected.

Figure 2 shows a typical startup sequence of a scanner and a slave with Quick Connect enabled on both sides. Frames 1 through 8 show the typical sequence of UCMM open and Allocate requests issued by the scanner (node ID 0) as described above. As expected per spec, there is a wait time of about 1 second between each of these messages. Frame #9 is the first Duplicate Node ID Check request issued by this slave and the two frames that follow (#10 & #11) contain a UCMM open (#10) and an Allocate (#11) request. Since the slave shown in Figure 2 is UCMM-

capable, it responds to the UCMM open request with frame #12. What follows then is a typical conversation between scanner and slave to setup and configure whatever connections have been agreed upon between the two. Frame 32 is the first frame containing I/O data, so this marks the end of the communication setup procedure. As can be observed from the time stamps, it only took about 17 ms from the first “sign of life” of the slave to the beginning of the I/O exchange (poll request from the scanner).

Frame #	Time	Can Id	Grp	Msg ID	Mac ID	Size	Data	Ascii
1	0000:00'00"000000	0x42E	2	0x6	05	6	00 4B 03 01 01 00	.K....
2	0000:00'01"000502	0x42E	2	0x6	05	6	00 4B 03 01 01 00	.K....
3	0000:00'02"100114	0x780	3	0x6	00	4	05 4B 02 34	.K.4
4	0000:00'03"097528	0x780	3	0x6	00	4	05 4B 02 34	.K.4
5	0000:00'04"097778	0x42E	2	0x6	05	6	00 4B 03 01 01 00	.K....
6	0000:00'05"098303	0x42E	2	0x6	05	6	00 4B 03 01 01 00	.K....
7	0000:00'06"196130	0x780	3	0x6	00	4	05 4B 02 34	.K.4
8	0000:00'07"195748	0x780	3	0x6	00	4	05 4B 02 34	.K.4
9	0000:00'07"680981	0x42F	2	0x7	05	7	00 01 00 BE 18 16 1A	.....
10	0000:00'07"681847	0x780	3	0x6	00	4	05 4B 02 34	.K.4
11	0000:00'07"682075	0x42E	2	0x6	05	6	00 4B 03 01 01 00	.K....
12	0000:00'07"683073	0x745	3	0x5	05	6	00 CB 02 00 05 00	.....
13	0000:00'07"685230	0x700	3	0x4	00	8	05 4B 03 00 01 00 02 00	.K.....
14	0000:00'07"686261	0x605	3	0x0	05	3	00 CB 00	...
15	0000:00'07"686820	0x700	3	0x4	00	7	05 0E 01 00 01 00 01	.....
16	0000:00'07"687279	0x605	3	0x0	05	4	00 8E 01 00	....
17	0000:00'07"687683	0x700	3	0x4	00	7	05 0E 01 00 01 00 02	.....
18	0000:00'07"688174	0x605	3	0x0	05	4	00 8E 0C 00	....
19	0000:00'07"688531	0x700	3	0x4	00	7	05 0E 01 00 01 00 03	.....
20	0000:00'07"689057	0x605	3	0x0	05	4	00 8E 01 00	....
21	0000:00'07"689978	0x700	3	0x4	00	8	85 00 10 05 00 02 00 09	.....
22	0000:00'07"690394	0x605	3	0x0	05	3	80 C0 00	...
23	0000:00'07"690648	0x700	3	0x4	00	4	85 81 4B 00	..K.
24	0000:00'07"691467	0x605	3	0x0	05	3	80 C1 00	...
25	0000:00'07"692182	0x605	3	0x0	05	4	00 90 4B 00	..K.
26	0000:00'07"692555	0x700	3	0x4	00	7	05 0E 05 00 02 00 07	.....
27	0000:00'07"693062	0x605	3	0x0	05	4	00 8E 12 00	....
28	0000:00'07"693972	0x700	3	0x4	00	7	05 0E 05 00 02 00 08	.....
29	0000:00'07"694981	0x605	3	0x0	05	4	00 8E 08 00	....
30	0000:00'07"695608	0x780	3	0x6	00	4	05 4C 05 00	.L..
31	0000:00'07"696057	0x745	3	0x5	05	2	00 CC	..
32	0000:00'07"697956	0x42D	2	0x5	05	8	00 00 00 00 00 00 00 00	.....
33	0000:00'07"699155	0x3C5	1	0xF	05	8	00 FC 00 00 00 00 00 00	.....
34	0000:00'07"699977	0x3C5	1	0xF	05	8	41 00 00 00 00 00 00 00	A.....
35	0000:00'07"700290	0x3C5	1	0xF	05	5	82 00 00 00 00	.....

Figure 2: Typical Start-up Trace of a UCMM-capable Slave (Node ID 7)

Figure 3 shows another typical startup sequence of a scanner and a slave with Quick Connect enabled on both sides, this time with a slave that is not UCMM-capable. Frames 1 through 8 show the typical sequence of UCMM open and Allocate requests issued by the scanner (node ID 0) as described above. As expected per spec, there is a wait time of about 1 second between each of these messages. Frame #9 is the first Duplicate Node ID Check request issued by this slave and the two frames that follow (#10 & #11) contain a UCMM open (#10) and an Allocate (#11) request. Since the slave shown in Figure 3 is not UCMM-capable, it responds to the Allocate request with frame #12. What follows then is a typical conversation between scanner and slave to setup and configure whatever connections have been agreed upon between the two. Frame 29 is the first frame containing I/O data, so this marks the end of the communication setup procedure. As can be observed from the time stamps, this slave took a mere 12 ms to become operational, partly because there are fewer messages required to setup communication with a non-UCMM slave.

Frame #	Time	Can Id	Grp	Msg ID	Mac ID	Size	Data	Ascii
1	0000:00'00"000000	0x43E	2	0x6	07	6	00 4B 03 01 01 00	.K....
2	0000:00'01"000839	0x43E	2	0x6	07	6	00 4B 03 01 01 00	.K....
3	0000:00'02"099471	0x780	3	0x6	00	4	07 4B 02 34	.K.4
4	0000:00'03"097516	0x780	3	0x6	00	4	07 4B 02 34	.K.4
5	0000:00'04"098337	0x43E	2	0x6	07	6	00 4B 03 01 01 00	.K....
6	0000:00'05"098807	0x43E	2	0x6	07	6	00 4B 03 01 01 00	.K....
7	0000:00'06"196086	0x780	3	0x6	00	4	07 4B 02 34	.K.4
8	0000:00'07"195644	0x780	3	0x6	00	4	07 4B 02 34	.K.4
9	0000:00'08"180306	0x43F	2	0x7	07	7	00 01 00 15 DC 09 C0	.....
10	0000:00'08"180597	0x780	3	0x6	00	4	07 4B 02 34	.K.4
11	0000:00'08"180823	0x43E	2	0x6	07	6	00 4B 03 01 01 00	.K....
12	0000:00'08"181029	0x43B	2	0x3	07	3	00 CB 03	...
13	0000:00'08"184194	0x43C	2	0x4	07	7	00 4B 03 00 01 02 00	.K.....
14	0000:00'08"184399	0x43B	2	0x3	07	3	00 CB 03	...
15	0000:00'08"184727	0x43C	2	0x4	07	6	00 0E 01 00 01 01	.....
16	0000:00'08"184941	0x43B	2	0x3	07	4	00 8E 01 00	....
17	0000:00'08"185276	0x43C	2	0x4	07	6	00 0E 01 00 01 02	.....
18	0000:00'08"185488	0x43B	2	0x3	07	4	00 8E 07 00	....
19	0000:00'08"186284	0x43C	2	0x4	07	6	00 0E 01 00 01 03	.....
20	0000:00'08"186495	0x43B	2	0x3	07	4	00 8E 4E 04	..N.
21	0000:00'08"186849	0x43C	2	0x4	07	7	00 10 05 00 01 0C 03	.....
22	0000:00'08"187027	0x43B	2	0x3	07	2	00 90	..
23	0000:00'08"187542	0x43C	2	0x4	07	8	00 10 05 00 02 09 4B 00	.....K.
24	0000:00'08"187829	0x43B	2	0x3	07	4	00 90 4B 00	..K.
25	0000:00'08"188290	0x43C	2	0x4	07	6	00 0E 05 00 02 07	.....
26	0000:00'08"188500	0x43B	2	0x3	07	4	00 8E 01 00	....
27	0000:00'08"188840	0x43C	2	0x4	07	6	00 0E 05 00 02 08	.....
28	0000:00'08"189050	0x43B	2	0x3	07	4	00 8E 01 00	....
29	0000:00'08"192032	0x43D	2	0x5	07	1	00	.
30	0000:00'08"192168	0x3C7	1	0xF	07	1	00	.

Figure 3: Typical Start-up Trace of a non-UCMM-capable Slave (Node ID 5)

Application constraints for Quick Connect

Quick Connect is not supposed to be used in every application. There are good (safety) reasons for the “slow” Duplicate Node ID Check Algorithm and this normal procedure should only be given up when necessary. Quick Connect should only be used when:

- The application has individual devices or small groups of devices that must be disconnected and reconnected on a regular basis.
- The likelihood that these or other components in the overall system get modified is very small.
- There is a strong need to have fast start-up time, e.g. when the Quick Connect components are part of a changeable tool on a robot.

There are no physical or topological restraints for the use of Quick Connect except that such component must always be part of a drop line. The disconnect process must not lead to an interruption of the DeviceNet trunk line.

Setting up Quick Connect

As of today, Quick Connect can only be set up online. From Version 4.00 onwards, RSNetWorx for DeviceNet, the Rockwell Automation DeviceNet network configuration software, contains a standalone tool that searches the network for Quick Connect enabled devices, reads the status of the Quick Connect enabling attribute and identifies the scanner(s) owning the slaves in the network.

Figure 4 shows the result of a scan of a network for Quick Connect. Two nodes (#5 & #7) are capable of supporting Quick Connect while the other three nodes do not support Quick Connect. Quick Connect can now be enabled in both the scanner (node #0) and the slaves (#5 & #7). It is important to note that Quick Connect in the scanner is enabled on a slave by slave basis. Slaves that shall not participate in

the Quick Connect process are not supposed to have this feature enabled in their scanner.

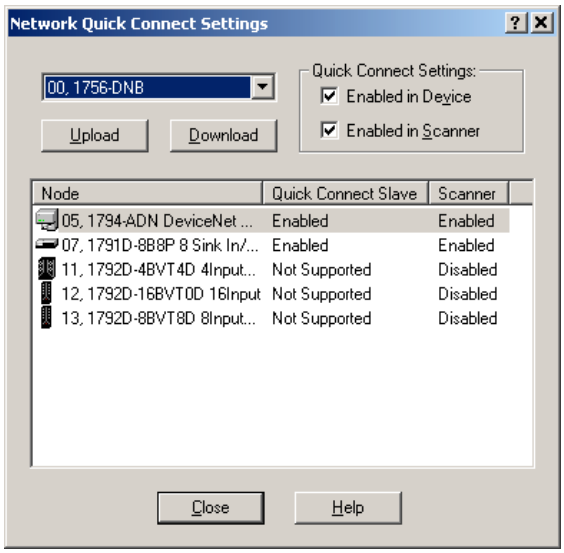

Figure 4: Quick Connect Tool within RSNetWorx for DeviceNet

Furthermore, to get the best results, Quick Connect must be enabled on both sides, slave and scanner. Enabling Quick Connect on one side only will give an improvement but not the full speed required for many applications. The traces shown in Figure 2 and Figure 3 were captured with the Quick Connect feature enabled on both sides.

Conclusions

Quick Connect is a versatile extension of the communication access mechanism in DeviceNet that allows a very fast reconnection of nodes to an existing DeviceNet system. The reconnect times, including self-test times of the reconnected nodes, typically lie below 1 second compared to 5 to 10 seconds without Quick Connect.

Author	Viktor Schiffer
Company	Rockwell Automation
Address	Düsselberger Str. 15
	42781 Haan
Phone	+49-2104-960-193
Fax	+49-2104-960-121
E-mail	vschiffer@ra.rockwell .com
Website	www.rockwellautomation.de

References

[1] THE CIP NETWORKS LIBRARY, Volume 3, DeviceNet Adaptation of CIP, Edition 1.5, Nov 2007, ODVA.