# HOW TO MEASURE DEVICENET PERFORMANCE

By
James H. Gross

**This paper will explain how to measure performance on DeviceNet. Understanding the fundamental performance attributes of a DeviceNet network is basic to achieving network design goals, optimum performance and reliable operation. This paper will provide an overview of the methods, software and hardware used at Rockwell Automation to measure DeviceNet performance.**

**What is performance?**
For the purposes of this paper we will define performance as follows:

*Performance is that attribute of a network, which restricts the ability of a device to respond to or initiate a change in an I/O status or an explicit message.*

**Why measure performance?**
There are two very good reasons to measure performance. The first is to verify design goals. Such as new module performance, regression performance after a firmware change and network response. The second reason is to Analyze network problems. Common network problems are; slow response, Inconsistent performance and nodes dropping off the network.

**What is the goal?**
Therefore, it should be stated that the goal of this paper is to give you a basic understanding of how performance on DeviceNet maybe measured and how to analyze the information collected.

**Fundamental DeviceNet measurements.**

To be able to perform an analysis of DeviceNet there are three concepts that should be understood: One, CAN Signals - being able to view the DeviceNet message packets on a digital oscilloscope. Two, Understanding the difference between POLLED and COS(Change of State) messages. Three, The effects of message jitter.

**CAN signals on the DeviceNet**
Scope setup used to examine CAN Signal Levels. Using a dual channel digital oscilloscope connect and setup your scope as follows:
1. Connect channel A to the CAN-Hi signal line (white).
2. Connect channel B to the CAN-Lo signal line (blue).
3. Connect the scope signal ground to the CAN V- (black).
4. Set Channel A&B to 1 volt per division.
5. Set the scope for triggered sweep on channel A.
6. Adjust the channel A&B reference cursors to be on the same grid line.
7. Set the timebase to 50 usec.
8. Adjusted the trigger level to produce a stable display.
Figure 1 is a screen capture from a digital scope showing the CAN signals. Superimposed on the screen capture are the CAN Signal Level standards.

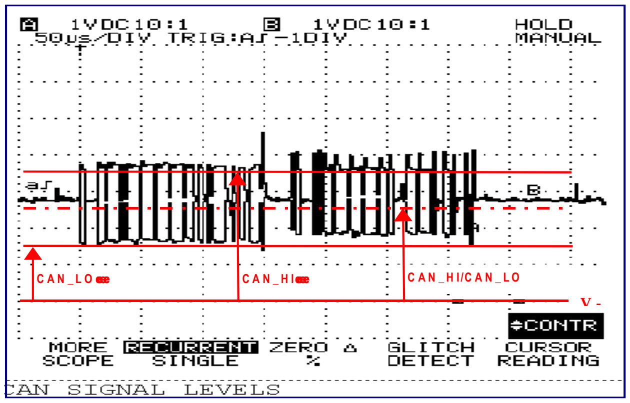


Figure 1. CAN Signals

**COS/POLLED messges.**
Figure 2 is the COS/Polled network setup.

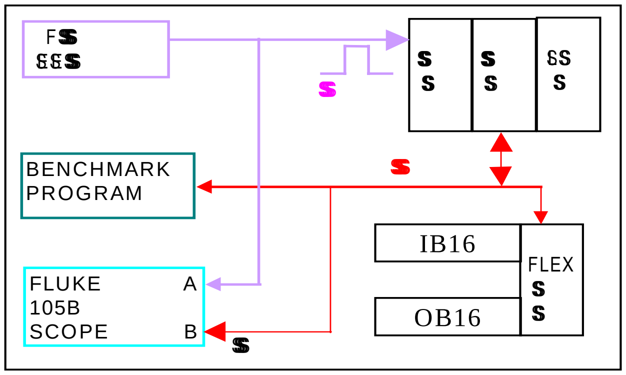
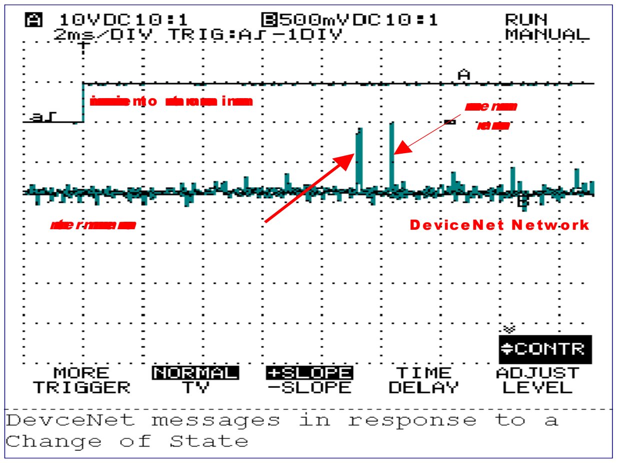


Figure 2. Network Configuration

The Flex ADN is placed in the scan list as either a COS or Polled device. An I/O COS is supplied by a function generator to the input module bit 0. Figure 3 shows the DeviceNet COS messages versus the input module bit 0. Messages are only generated when a COS occurs in the input data.



10VDC10:1 500mVDC10:1 RUN
2ms/DIV TRIG:A⌐-1DIV MANUAL
A
B
DeviceNet Network
CONTR
MORE TRIGGER
NORMAL TV
+SLOPE -SLOPE
TIME DELAY
ADJUST LEVEL
DevceNet messages in response to a Change of State

Figure 3. COS Messages

Figure 4 shows the DeviceNet POLLED messages versus the input module bit 0. Polled messages are generated constantly, independent of the changes in the input value.

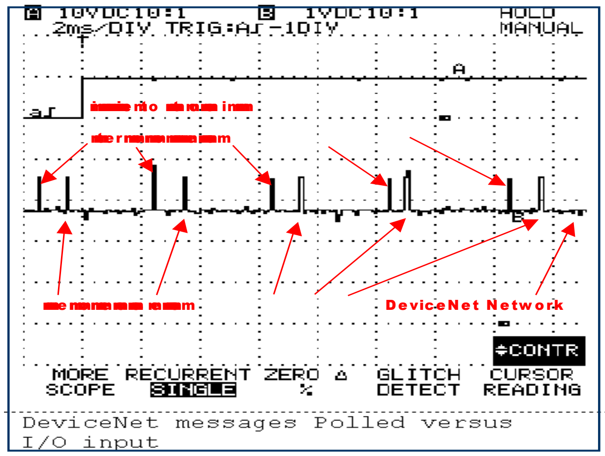


Figure 4. Polled Messages

**Effects of Message Jitter.**
The delay between the I/O input and the production of a DeviceNet message may not constant. Figure 5, shows that the delay between the application of a COS to the input module bit 0 and the actual generation of the DeviceNet messages is not constant. Message jitter may have several causes such as improper task priority allocation, excessive operating system latency, improper code design or incorrect processor speed selection. Message jitter should be minimized.

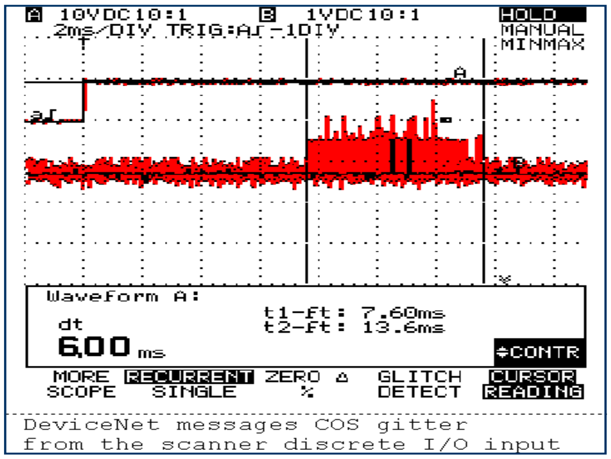


Figure 5. Message jitter

**A Software solution to measuring performance.**
Even a software solution requires some hardware and this paper assumes that a network interface card, necessary drivers and programming software is available. The interface to the card must be capable of informing the application that a valid CAN message has been received.
A software application to measure performance is attractive for several reasons. First, there is a minimum of intrusion on the network (the card may operate in passive mode). Second, a minimum of hardware is required, just a laptop and NIC. The third attractive feature is portability, movement from network to network is very easy.

**A Software solution: Network loading.**
Network loading (NETUSE) is a measure of the density of messages on the network in percent of available bandwidth. NETUSE is an important performance measure because the traffic density on the network will effect the ability of nodes to transmit and receive messages.

NETUSE Equations:

Maximum Bits possible = Sample Time * Baud Rate
% NETUSE = (Bits received / Maximum Bits possible) * 100

NETUSE Variables:

Set up number of messages to sample (max_messages).
Set up message counter (message_count).
Set up start sample time (start_scan_time)
Set up stop sample time (stop_scan_time).
Set up bit counter (bit_count).
Set up most messages possible (best_messages).
Set up Baud Rate (baud_rate).
Set up percent net usage (net_use).
Set up average network use (avg_net_use).
Set up average net use counter (avg_net_use_count).
Initialize bit_count to 0.
Initialize max_messages to 128.
Initilaize message_count to 0.

Initialize baud_rate to either 125KB, 250KB or 500KB.
Initialize avg_net_use_count to 0.

### NETUSE Algorithm:

*Note: Entry is conditional on your Interface Card indicating a valid CAN message has been received and is available to be read.*

```
START:
Read CAN message from Interface card.
IF message_count is equal to zero THEN
    start_scan_time is assigned time stamp from Interface card.
END IF
Read CAN_data_Length from CAN message.
bit_count is assigned bit_count + 47 +(8 * CAN_data_length).
IF message_count is greater than or equal to max_messages THAN
    stop_scan_time is assigned time stamp from Interface card.
    IF stop_scan_time is greater than start_scan_time THAN
        time_to_scan is assigned the value stop_scan_time minus start_scan_time.
    ELSE
        time_to_scan is assigned the value start_scan_time minus stop_scan_time.
    END IF
    best_messages is assigned the value baud_rate times time_to_scan.
    net_use is assigned the value word_count divided by best_messages times 100.
    word_count is assigned the value of 0.
    message_count is assigned the value of 0.
    avg_net_use is assigned the value of avg_net_use plus net_use.
    Increment avg_net_use_count.
ELSE
    Increment message_count
END IF
IF avg_net_use_count is greater than or equal to 10 THAN
    avg_net_use is assigned the value of avg_net_use divided by avg_net_use_count.
    avg_net_use_count is assigned the value of 0.
    avg_net_use is assigned the value of 0.
END IF
Net Use data is now available and may displayed as desired.
END:
```

## A Software solution: Masters Produced Data Rate. (MPDR)

MPDR is the rate at which a master can produce I/O Poll/COS/Cyclic data, for Cyclic data MPDR is a verification of the cyclic rate, that is consumed by a slave. MPDR is an important network metric because the maximum rate that the master can produce data is directly related to how fast a change in I/O value can occur.

### MPDR Setup:

The Masters I/O Poll Command/Change of State/Cyclic message value must be constructed. The 11 bit identifier is weighted:

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** | **– bit** |
| **1** | **0** | **destination MAC ID** | | | | | | **1** | **0** | **1** | **- value** |

A value for the destination MAC ID(slave)is assumed to be entered.
The value of slave is assigned the value of slave, bit shifted 3 places left.
The value mpdr_data_id is assigned the value of 0x400 bitwise or'd with the value slave bit shifted 3 places left.
The value mpdr_data_id is assigned the value of mpdr_data_id bitwise or'd with the value 5.
This maybe expressed as: mpdr_data_id = 0x400|(slave<<3)|5.

### MPDR Variables:

Set up CAN Message holder structure(read_param).
Set up CAN Message Identity value (mpdr_data_id).
Set up number of messages to sample (max_messages).
Set up start sample time (m_prod_start_time).
Set up stop sample time (m_prod_stop_time).
Set up Master's produced time (m_prod_time).
Set up Master's produced minimum time (m_prod_min_time).
Set up Master's produced maximum time (m_prod_max_time).
Set up Master's produced message count (m_prod_count).
Set up Master's produced message average time (m_prod_avg_time).
Set up Mater's produced total messages time (m_prod_total_time).
Initialize max_messages to 128.
Initialize m_prod_count to 0.
Initialize m_prod_avg_time to 0.
Initialize m_prod_total_time to 0.

### MPDR Algorthm:

*Note: Entry is conditional on your Interface Card indicating a valid CAN message has been received and is available to be read.*

```
START:
Read CAN message from Interface card.
read_param.Ident is assigned the value of the CAN Message Identifier field.
IF mpdr_data_id is equal to the value of read_param.Ident THAN
    IF m_prod_start_time is equal to zero THAN
        m_prod_start_time is assigned time stamp from Interface card.
    ELSE
        m_prod_stop_time is assigned time stamp from Interface card.
        IF m_prod_stop_time is less than m_prod_start_time THAN
            m_prod_time is assigned the value of m_prod_start_time minus m_prod_stop_time.
        ELSE
            m_prod_time is assigned the value of m_prod_stop_time minus m_prod_start_time.
        END IF
        m_prod_start_time is assigned time stamp from Interface card
        IF m_prod_time is less than m_prod_min_time THAN
            m_prod_min_time is assigned the value m_prod_time.
        END IF
        IF m_prod_time is greater than m_prod_max_time THAN
            m_prod_max_time is assigned the value m_prod_time.
        END IF
        IF m_prod_count is greater than or equal to max_messages THAN
            m_prod_avg_time is assigned the value m_prod_total_time divided by m_prod_count.
            m_prod_count is assigned the value 0.
            m_prod_total_time is assigned to value 0.
        ELSE
            m_prod_total_time is assigned the value m_prod_total_time plus m_prod_Time.
            increment the value of m_prod_count.
        END IF
    END IF
END IF
Master Produced Data Rate is now available and may displayed as desired.
END:
```

## A Software solution: Slave Produced Data Rate. (SPDR)

SPDR is the rate at which a slave can produce Change of State/Cyclic data, for Cyclic data SPDR is a verification of the cyclic rate, that is consumed by another device usually a master. SPDR is an important network metric because the maximum rate that a slave device can produce data is directly related to how fast a change in I/O value can be sent to a master.

## SPDR Setup:

The Slave's I/O Change of State/Cyclic message value must be constructed. The 11 bit identifier is weighted:

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** | **– bit** |
| **0** | **1** | **1** | **0** | **1** | **---Source MAC ID---** | | | | | | **- value** |

A value for the Source MAC ID(slave)is assumed to be entered.
The value slave_data_id is assigned the value of 0x340 bitwise or'd with the value slave.
This maybe expressed as: slave_data_id = 0x340|slave.

## SPDR Variables:

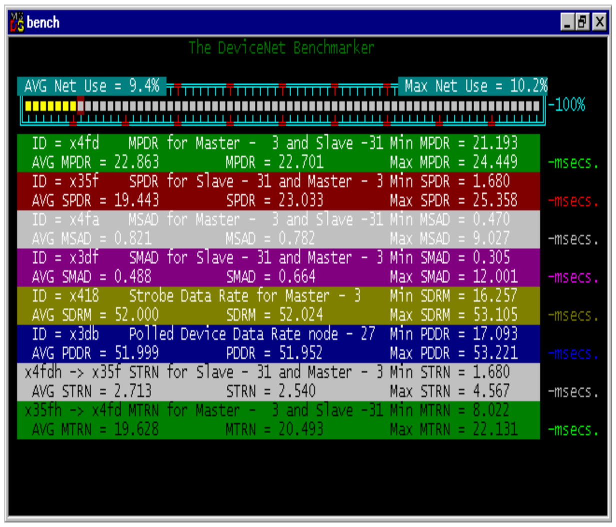
Set up CAN Message holder structure(read_param).
Set up CAN Message Identity value (slave_data_id).
Set up number of messages to sample (max_messages).
Set up start sample time (s_prod_start_time).
Set up stop sample time (s_prod_stop_time).
Set up Slave's produced time (s_prod_time).
Set up Slave's produced minimum time (s_prod_min_time).
Set up Slave's produced maximum time (s_prod_max_time).
Set up Slave's produced message count (s_prod_count).
Set up Slave's produced message average time(s_prod_avg_time).
Set up Slave's produced total messages time (s_prod_total_time).
Initialize max_messages to 128.
Initialize s_prod_count to 0.
Initialize s_prod_avg_time to 0.
Initialize s_prod_total_time to 0.

## SPDR Algorthm:

*Note: Entry is conditional on your Interface Card indicating a valid CAN message has been received and is available to be read.*

```
START:
Read CAN message from Interface card.
read_param.Ident is assigned the value of the CAN Message Identifier field.
IF slave_data_id is equal to the value of read_param.Ident THAN
IF s_prod_start_time is equal to zero THAN
s_prod_start_time is assigned time stamp from Interface card.
ELSE
s_prod_stop_time is assigned time stamp from Interface card.
IF s_prod_stop_time is less than s_prod_start_time THAN
s_prod_time is assigned the value of s_prod_start_time minus s_prod_stop_time.
ELSE
s_prod_time is assigned the value of s_prod_stop_time minus s_prod_start_time.
END IF
s_prod_start_time is assigned time stamp from Interface card.
IF s_prod_time is less than s_prod_min_time THAN
s_prod_min_time is assigned the value s_prod_time.
END IF
IF s_prod_time is greater than s_prod_max_time THAN
s_prod_max_time is assigned the value s_prod_time.
END IF
IF s_prod_count is greater than or equal to max_messages THAN
s_prod_avg_time is assigned the value s_prod_total_time divided by s_prod_count.
s_prod_count is assigned the value 0.
s_prod_total_time is assigned to value 0.
ELSE
s_prod_total_time is assigned the value s_prod_total_time plus s_prod_Time.
increment the value of s_prod_count.
END IF
END IF
END IF
Slave Produced Data Rate is now available and may displayed as desired.
END:
```

Figure 6. is a screen capture of how the data maybe displayed.



```
                    The DeviceNet Benchmarker

AVG Net Use = 9.4%                              Max Net Use = 10.2%
                                                                     -100%
  ID = x4fd     MPDR for Master -  3 and Slave -31 Min MPDR = 21.193
  AVG MPDR = 22.863         MPDR = 22.701          Max MPDR = 24.449  -msecs.
  ID = x35f     SPDR for Slave - 31 and Master - 3 Min SPDR = 1.680
  AVG SPDR = 19.443         SPDR = 23.033          Max SPDR = 25.358  -msecs.
  ID = x4fa     MSAD for Master -  3 and Slave -31 Min MSAD = 0.470
  AVG MSAD = 0.821          MSAD = 0.782           Max MSAD = 9.027   -msecs.
  ID = x3df     SMAD for Slave - 31 and Master - 3 Min SMAD = 0.305
  AVG SMAD = 0.488          SMAD = 0.664           Max SMAD = 12.001  -msecs.
  ID = x418     Strobe Data Rate for Master - 3    Min SDRM = 16.257
  AVG SDRM = 52.000         SDRM = 52.024          Max SDRM = 53.105  -msecs.
  ID = x3db     Polled Device Data Rate node - 27  Min PDDR = 17.093
  AVG PDDR = 51.999         PDDR = 51.952          Max PDDR = 53.221  -msecs.
x4fdh -> x35f STRN for Slave - 31 and Master - 3 Min STRN = 1.680
  AVG STRN = 2.713          STRN = 2.540           Max STRN = 4.567   -msecs.
x35fh -> x4fd MTRN for Master -  3 and Slave -31 Min MTRN = 8.022
  AVG MTRN = 19.628         MTRN = 20.493          Max MTRN = 22.131  -msecs.
```

Figure 6. Sample Data Display

## A Hardware solution: Direct I/O measurements

Although DeviceNet message performance is important in analyzing a network, the actual bit times are what really matters. Three bit timings will be investigated, output bit turn-on time, input bit turn-on time and wrap-around network response. A PLC with a DeviceNet scanner and Discreete I/O module will be used.

## Output Bit Turn-on time:

Output Bit Turn-on time is the time measured from the application of a change to the I/O Output Data Table of the processor till the slave device's output bit is turned on. A ladder program generates a 30 ms square wave output to a DeviceNet device. The output bit in the scanners discrete data table is mirrored to an output bit in the discrete I/O module. The setup for this test is shown in Figure 7. The results of this test are displayed in figure 8. Which is a capture from the digital scope. One use of this test is to see the effect of adding nodes to the scan list.

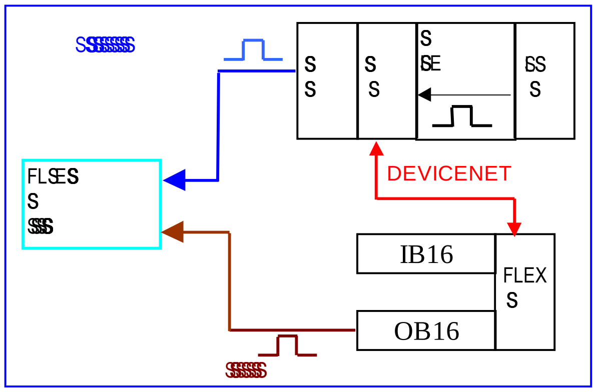
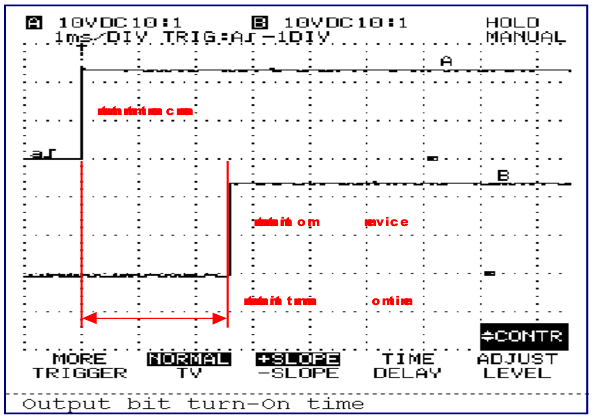
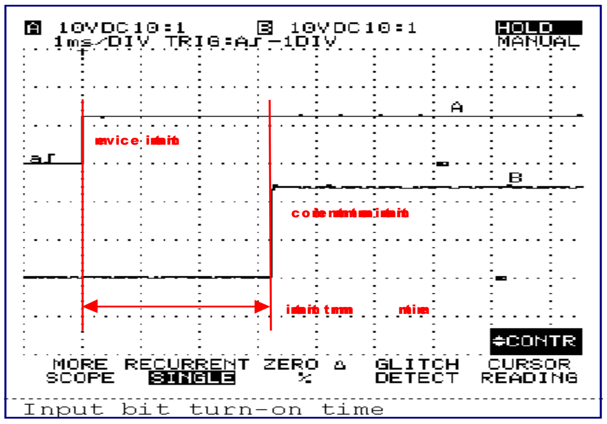


Figure 7. Output Bit Turn-on Time setup



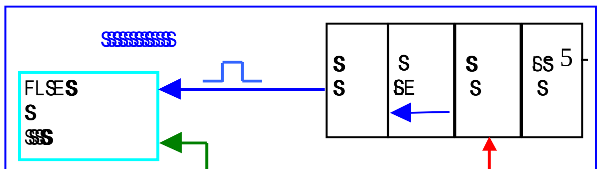
I Figure 8. Output Bit Turn-on Time results

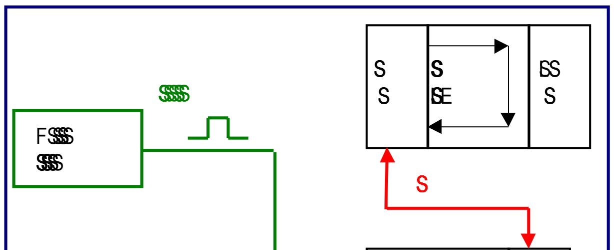
The time measured from the application of a change to an input bit on a device till that I/O change is detected in the processor and the bit placed in the Input Data Table. A ladder program moves the input bit from the scanner's input data table to an output bit in the discrete data table of an output module. The setup for this test is shown in Figure 9. The results of this test are displayed in figure 10. Which is a capture from the digital scope. One use of this test is to see how quickly a device may put data into the processor following an I/O change.

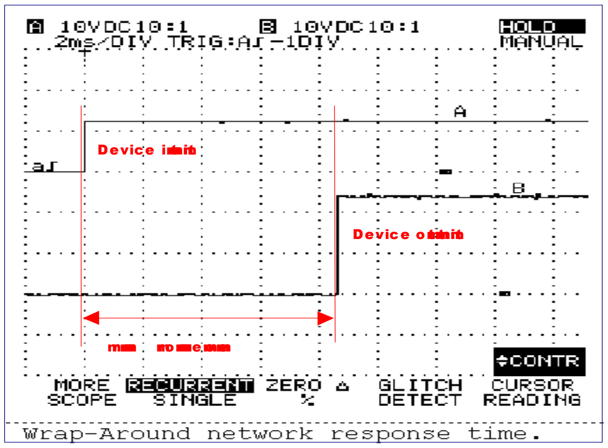


V Figure 10. Input Bit Turn-on time results

The time measured from the application of a change to an input bit on a device till that bit change is observed at the output bit of a network I/O device. Wrap around of the I/O bit change takes place in the processor. The processor moves the Input Data Table change to The Output Data Table. The Wrap-Around response includes all network and processor latencies such as, ladder scan time, backplane scanner and I/O device response times and any interscan delays that maybe set. Figure 11 is the setup for measuring the Wrap-Around response. Figure 12. Shows the results of the measurements.







Wrap-Around network response time.

Figure 12. Wrap-Around Network Response results

## PC Based DeviceNet Scanners

PC based scanner Output Bit Turn-on Time:

The use of PC based DeviceNet scanners present special problems when attempting to perform benchmarks. The main problem is the generation of I/O data. In most cases some type of 'soft' PLC is used. Another problem is finding a way to mirror the I/O data bit out to the "real world".

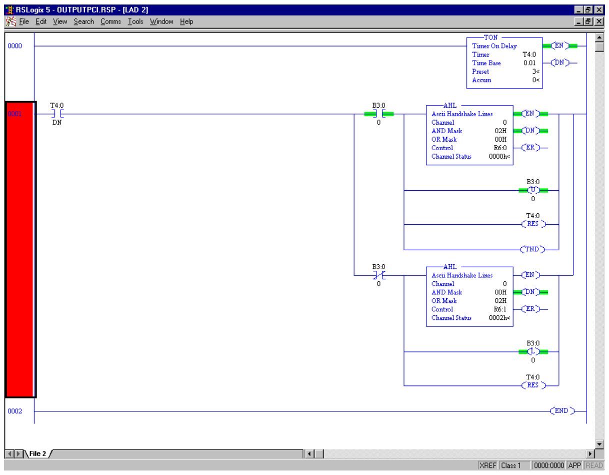

RSLogix 5 - OUTPUTPCI.RSP - [LAD 2]
File Edit View Search Comms Tools Window Help
TON
Timer On Delay
AHL
Ascii Handshake Lines
EN
DN
ER
END
File 2

Figure 13. "CTS" Ladder Program

The ladder program shown in Figure 13, will generate a change to the scanner's data table and mirror that I/O change to the serial port's CTS bit, actuated by the AHL instruction. The configuration for this test is shown in Figure 14. The results are shown in Figure 15.

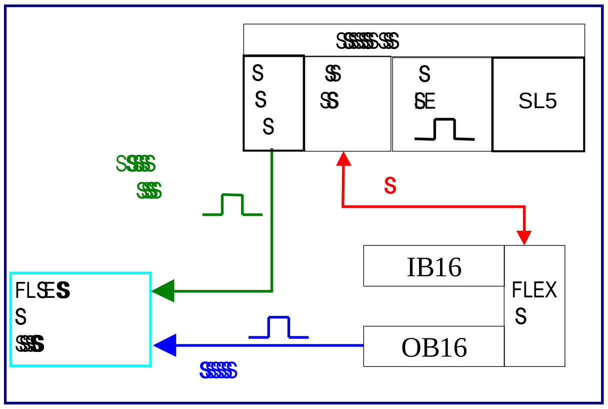
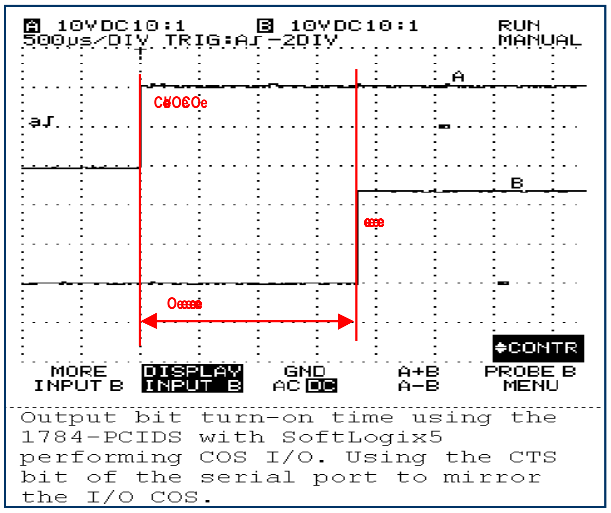


Figure 14. PC Based Scanner – Output Bit Turn-on Time setup

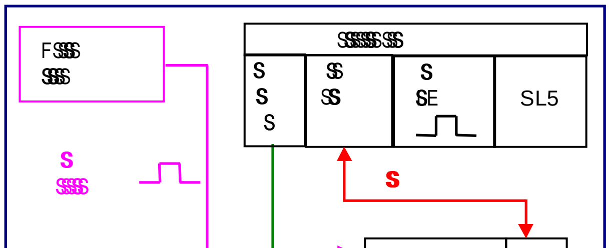


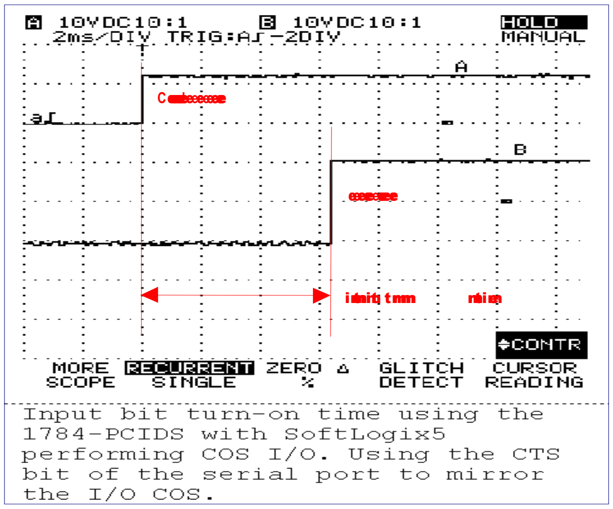
Output bit turn-on time using the 1784-PCIDS with SoftLogix5 performing COS I/O. Using the CTS bit of the serial port to mirror the I/O COS.

Figure 15. PC Based Scanner – Output Bit turn-on Time results

PC based scanner Input Bit Turn-on Time

The setup for Input bit turn-on Time is shown in Figure 16. Input Bit turn-on time is similar in measurement to the Output bit Turn-on Time except for the mirroring of the input data table bit to the CTS bit using the AHL instruction. Figure 17 shows the results of the PC based Input Bit turn-on Time measurement.





Input bit turn-on time using the 1784-PCIDS with SoftLogix5 performing COS I/O. Using the CTS bit of the serial port to mirror the I/O COS.

Figure 17. . PC Based Scanner – Input Bit turn-on Time results

**Real performance investigations that have been done.**

MPDR for COS.

An investigation was conducted to determine the maximum rate at which a master will produce messages to the slaves via DeviceNet without missing a COS. In this investigation, up to a 27 node network was tested. The investigation began with the scanner and a single node. Figure 18 shows the test setup. The subsequent nodes were attached to the network and added into the scan list. The failure of the master, that is when the master missed a state change, was determined to be 2x the recorded minimum value. The digital scope

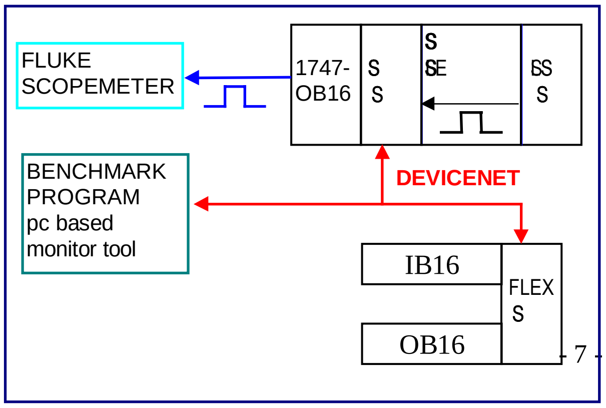


Figure 18. Setup for MPDR COS measurement

and the benchmarking program was used to determine the failure. The heartbeat of the nodes was set at 5000ms and there was no ack. All measurements were taken at node 3. An ladder program was created to be the pulse generator for the investigation. Figure 19 are the results of the test.

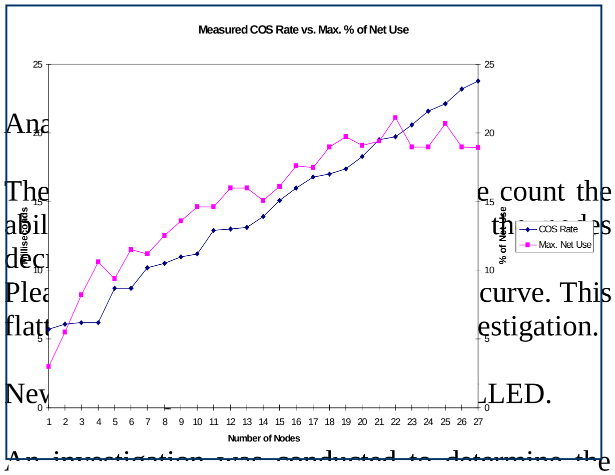


Figure 19. Results of MPDR COS test

Ana

The e count the abil th es dec Plea curve. This flatt estigation.

Nev LLED.

An investigation was conducted to determine the e slaves via DeviceNet using POLLED mode. In this investigation, up to a 62 node network was tested. The investigation began with the scanner and a single node. The test setup for MPDR was used. The subsequent nodes were attached to the network and added into the scan list. As this was a test to determine performance of a new product there was no failure criteria. Only the benchmarking program was used to measure the MPDR All measurements were taken at node 1. An ladder program was created to be the pulse generator for the investigation. Figure 20 shows the results of the test.
Analysis of results.

This chart (Figure 20) clearly shows that this scanner has some performance problems. Assuming that the code for the module has no software bottlenecks and there are no other external factors that are affecting performance, what is the cause? By looking at the NETUSE plot (bottom trace) it becomes clear that the scanner just can not get data on to network fast enough. Therefore the conclusion is that the scanner is hampered by an under powered processor.

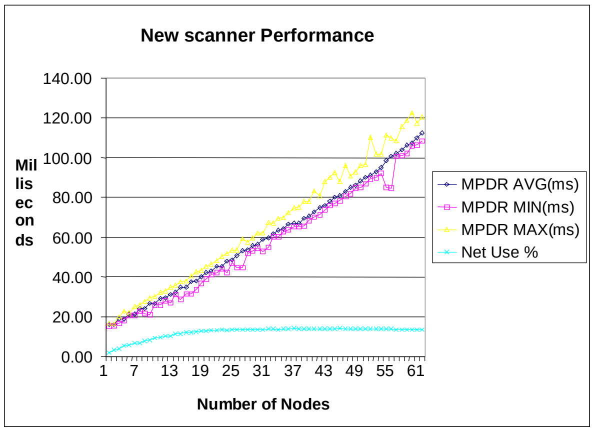


Figure 20. MPDR for a new product

Wrap around network response for COS.
Wrap around means that the processor copies the input discrete datatable to the output discrete datatable. The stimulus/response measurements for a wrapped DeviceNet network are composed of the time the network takes to respond to a COS at a node input, send the COS message to the scanner, process the data and return the information to the node's output. This network response time is the most complex and accounts for all latencies of a specific DeviceNet network. This investigation was conducted to determine the wrap around network response time for a pure COS network consisting from 1 to 27 nodes. Figure 21 shows the setup for the wrap around network response test. Figure 22 shows the results of that test.

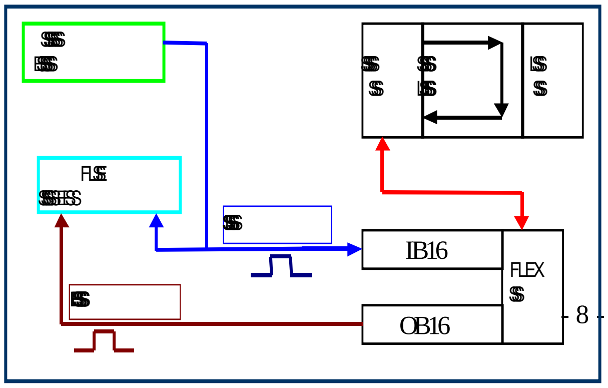


Figure 21. Setup for wrap around response test

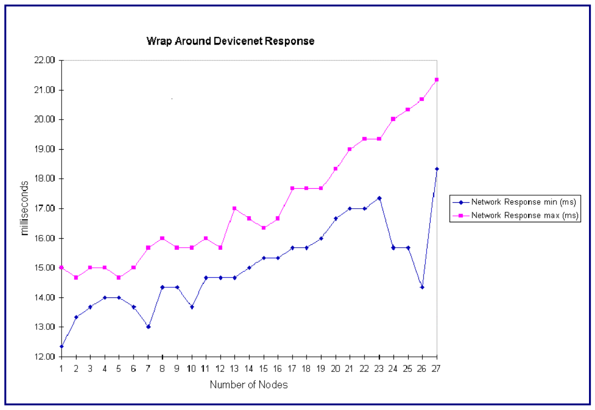


Figure 22. wrap around response test results

Analysis of results.

From the chart (Figure 22) the following conclusions can be drawn: That the devices will communicate no faster then the MIN network response value and that they will not be slower then the MAX network response value. That adding nodes gives a linear increase in response time.

Every network is different:

That every network is different is a fundamental principal of performance analysis. Every network presents unique configurations of devices, cable lengths, scanners, interface cards, node addressing, baud rate, Polled, Change of State/Cyclic, strobe operation and peer devices. However, experience with one network may allow general conclusions about other similar networks to be drawn.

**Summary:**

This paper has explained how to measure performance on DeviceNet. Shown the fundamental performance attributes of a DeviceNet network as basic to achieving network design goals, optimum performance and reliable operation. This paper has provided an overview of the methods, software and hardware used at Rockwell Automation to measure DeviceNet performance.

---

Rockwell Automation
1 Allen-Bradley Drive
Mayfield Heights, Ohio USA

(440)646-3943 voice
jhgross@ra.rockwell.com
www.cle.ab.com