The Road to an End-to-End Deterministic Ethernet

Tor Skeie, Svein Johannessen, and Øyvind Holmeide

No one now doubts the ascendancy of switched Ethernet in the automation world. Through use of the IEEE 802.1p traffic prioritisation standard it is now possible to predict worst case network latency. Although this represents a milestone towards deterministic Ethernet, there is still more to be resolved. A generic prioritisation mechanism may be the answer.

While office computers were getting networked left and right, traditional automation has been slower to embrace the revolution. One reason for the delay was that such systems usually depend on being able to sample input data at equally spaced intervals. The fieldbus concept was the first step on the road to networked automation systems. Usually marketed as a cost-saving device – cabling being the main outlay in any automation set-up – the various types offered different compromises between tight centralised control and network-type flexibility. If you wanted precise control over the data-sampling task, you either implemented a fieldbus with tight centralised control or cheated by using a separate wire for controlling the input data sampling.

Ethernet is by nature "democratic" – all nodes have an equal chance of accessing the network at a given point in time. Automation systems are "dictatorial" – some nodes are more important than others and want to have priority when accessing the network. Ethernet has no natural determinism. Automation systems want assurance, not probabilities.

Although the majority of these problems now have potential solutions, the end-nodes attached to the infrastructure also require predictable latency behaviour. Some 80-90% of the end-to-end latency occurs within the end-nodes when adhering to UDP/TCP/IP protocols. This strongly argues for a QoS mechanism to guarantee deterministic properties.

Even highways have queues

The non-deterministic behaviour of traditional switched Ethernet happens when packets from several input ports compete for the same output port. The contention might occur over low-priority traffic (node status reports, node firmware update, etc) which suggests the introduction of some sort of traffic rules. Without it, a situation may arise where the number of packets sent to an output port may exceed its bandwidth through output buffer overflow.

Even with a priority mechanism built into a system, there is still one queue mechanism left to deal with. Some standards propose to transfer real-time data by using a publish and subscribe approach (IEC 61850-9 springs to mind). A standard unmanaged Ethernet switch has to route these data packets to every drop link in the system. Since all those packets might have the same priority, they will be put in the same switch queue. The processing rate for this queue is equal to the bandwidth of the drop link between the switch and the node creating a drop link queue. Important data packets destined for one node may be delayed by traffic destined for another node. Putative traffic rules/mechanisms obviously need to deal with these multicast data packets.

Queues in the nodes

Introducing traffic rules in Ethernet switches will improve the worst-case latency across the network. However, there is usually only one single network task inside the node, and just a single hardware queue associated with the Ethernet controller. Since a network packet spends a large percentage of its total end-to-end time inside a node, internal packet prioritisation gives best control over the total transfer time.

In the transmit mode one typically could have a situation where multiple maximum size Ethernet packets eg, fragments of an FTP transfer, back up at the Ethernet driver level. In a standard implementation, real-time packets will be added to the end of the queue. Such behaviour will cause an unpredictable transmission delay.

The protocol stack implementation represents a possible system bottleneck in the receive mode. This is mainly due to the first-come first-served queue at the protocol multiplexer level (the level where the network packets are routed to the appropriate protocol handler software).

IEEE 802.1p has been introduced to alleviate the switch queue problem. Moreover, the standard specifies a layer-2 mechanism for giving mission-critical data preferential treatment over non-critical data[5]. The concept has been driven solely by the multimedia industry. It uses priority packets tagging and implementation of multiple queues within the network elements to discriminate packets. For tagging purposes IEEE 802.1Q defines an extra field for the Ethernet MAC header. This field is called Tag Control Info (TCI) field and is inserted as indicated by Fig 1. This field contains three priority bits, thus the standard defines eight different levels of priority.

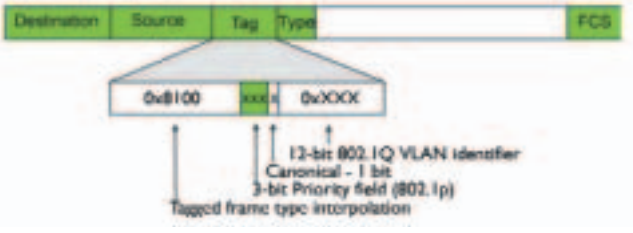


Fig 1 – MAC header (layer 2) with tag

Multicast distribution rules can reduce drop-link traffic. If you look closely at Fig 1, there is something called a "12 bit 802.1Q VLAN identifier". This is a mandatory part of each TCI field and can, when properly used, remove all unnecessary drop link traffic in an automation network based on publish and subscribe. Such an implementation consists of three steps:

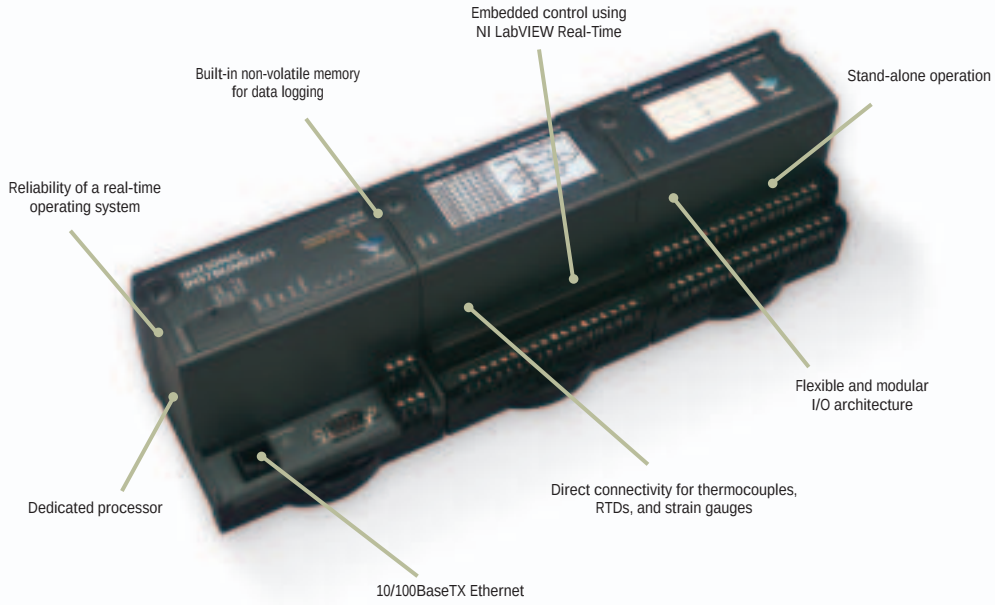
1. Map every multicast group to one specific VLAN identifier. This is, of course, an off-line activity.

2. At system start-up time, each of the nodes sends out one small packet for every VLAN (and thus multicast group) it wants to join. This packet – not a part of a complicated protocol stack – allows the network infrastructure to map VLAN identifiers to physical drop links. For simple data source nodes, these packets can be precompiled.

3. Whenever a multicast packet is to be transmitted, it is tagged with "real time priority" and the VLAN identifier corresponding to the multicast group.

These simple steps ensure that multicast packets do not block any unnecessary drop links.

It's What's Inside That Counts



FieldPoint™ distributed I/O – it's easy, reliable, and powered by LabVIEW™

When developing an embedded control application, the solution that offers ease of use determines whether a project is completed on schedule and within budget. National Instruments integrates LabVIEW Real-Time ease of use with NI FieldPoint distributed I/O for poweful embedded control.

FieldPoint 2010 Real-Time Controller

- Autodetect modules for easy configuration
- Connect directly with up to 144 I/O points
- Log 16 channels of temperature for a week

LabVIEW Real-Time Software

- Graphical programming
- Power and ease of use
- Embedded real-time development environment

ni.com/info

Download the FieldPoint white paper – visit our Web site and enter iycw0d.

NATIONAL INSTRUMENTS™

Austria 0662 45 79 90 0 • Belgium 02 757 00 20
Czech Republic 02 2423 5774 • Denmark 45 76 26 00
Finland 09 725 725 11 • France (0)1 48 14 24 24
Germany 089 741 31 30 • Greece 30 1 42 96 427
India 91 80 535 5406 • Israel 03 6393737 • Italy 02 413091
Netherlands 0348 433466 • Norway 32 27 73 00
Poland 48 22 3390 150 • Portugal 351 210 311 210
Russia 095 238 7139 • Slovenia 386 3 425 4200
Spain 91 640 0085 • Sweden 08 587 895 00
Switzerland 056 200 51 51 • UK 01635 523545 • info@ni.com

Circle 67

Priority packets, priority treatment

A network data packet spends a relatively small part of its application-to-application transfer time on the physical network. The actual percentage varies with the speed of the network and the performance of the node, but for a 100Mbps Ethernet, the average ranges between 20% and 0.1%. The implemented network priority has little influence on this period (but retains a large influence on the worst-case transfer time). To improve the average application-to-application transfer time, the concept of priority must be extended to include the protocol layers in both the sending and receiving end. To do this requires consideration of the following:

- Adjustable process priority for the protocol stack software;
- Ability to run several instances of the protocol software stack at different priority levels;
- Multiple transmit queues at the Ethernet driver level.

In most real-time operating systems, the protocol stack runs as a single thread in the context of some networking task. Adjusting the protocol stack software in line with packet priority ensures that the processing of high priority network packets does not get interrupted by less important administrative tasks. This needs to be sidestepped. We really want to process high priority messages before low priority ones. The perfect solution would be to suspend the processing of low priority network packets when a high priority message arrives. At first glance, there exists an ideal solution. It goes like this:

At compile time, decide on the task priority that should correspond to each packet priority and to an untagged packet. Create one task for each priority and put the task IDs in a table;

When a packet arrives, extract the packet priority, use it with the task table described above, send the packet to that task and send a signal to the task in order for it to start processing.

The problem here is that it presupposes that the network software is re-entrant, a condition seldom fulfilled. Rewriting the stack software to make it re-entrant is not hard, just tedious and time-consuming. Of course, it also means that you have to support the rewritten software in the future. Running multiple instances of the protocol software may be an elegant and efficient solution, but requires time and resources.

An alternative solution does not suspend the processing of lower priority packets, but selects the next packet to be processed from a set of criteria based on the packet priority.

Implementing multiple transmit queues

On a transmit request, a standard Ethernet driver takes the contents of a buffer, does some housekeeping, and transfers it to the Ethernet controller hardware. Such a driver is simple and "fair" (first come, first served), but may be unsuitable for an efficient priority implementation. One or more large low-priority packets, once they are scheduled for transmission, will delay high-priority packets for the time it takes to transfer the low-priority ones. If we want high priority packets to go to the head of the transmission queue, and the hardware does not support multiple queues, we must do some priority handling at the driver level.

The simplest solution is to use two queues: the hardware queue and a software queue. Low priority packets go into the software queue, high priority packets go straight into the hardware queue. From this point onward, there exist several algorithms addressing different needs. If the real-time requirements are moderate, move the first packet in the software queue to the hardware queue whenever the hardware queue is empty.

Alternatively, if the real-time requirements are strict, move the first packet in the software queue to the hardware queue whenever a high priority packet has been placed in the hardware queue.

Fig 2 shows how a non-optimised protocol stack would be implemented in a VxWorks RTOS environment (VxWorks is used for illustration purposes only).

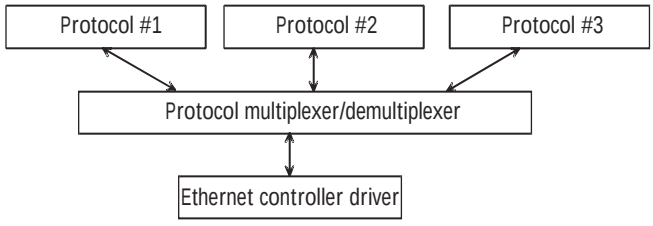


Fig 2. Example of a protocol stack implementation in VxWorks

Fig 3 shows an example of a stack implementation that would cure most of the queue problems of the standard implementation. The solution is to introduce a thin layer between the Ethernet controller driver and the protocol multiplexer. This layer implements multiple queues for both transmit and receive. Firstly, all transmit and receive packets are placed in a queue corresponding to their priority. Secondly, a selection algorithm selects the next packet to be passed to the next layer whenever the node resources are able to handle it.

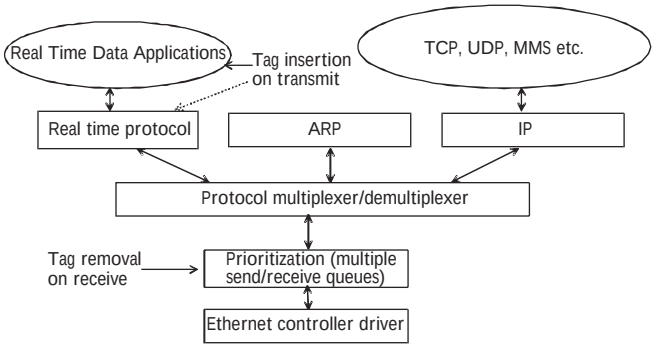


Fig 3. Optimised implementation of tagging in a protocol stack

Dock on and start.

With Industrial Ethernet.

www.siemens.com/
hanover-fair
April 07–12, 2003, Hall 9, Booth A72

simatic net

E20001-F240-P820-X-7600

Microautomation on a new path: SIMATIC S7-200 communication using Industrial Ethernet! For future-oriented solutions. With vast cost benefits through the use of existing Ethernet structures, fast and convenient configuring, programming and remote maintenance from a central location via LANs, open data exchange with PC applications using OPC and much, much more.
SIMATIC NET makes it possible with the CP 243-1 communications processor and creates the connection to the company network with the Electrical Lean Switches. Tip: All this is available in an attractive set.
SIMATIC NET – Networking for Industry.

SIEMENS

www.siemens.com/industrial-ethernet

Circle 66

Testing a prioritised Ethernet

Measuring the total transmission delay from one application to another through a network is no easy matter. The principle itself is fairly simple. It comprises the generating of a message at the application level in the node to be tested (the "master test node"), time stamping it, sending it to a companion test node (the "slave test node") which immediately returns the message and time stamping the returned message in the master test node. The difference between the two time stamps represents the total message round trip delay. It comprises:

The time spent in the master node transmit protocol stack and transmit buffers;
The network delay from the master test node to the slave test node;
The time spent in the slave node receive buffers and protocol stack;
The time spent at the slave node application level creating the return message;
The time spent in the slave node transmit protocol stack and transmit buffers;
The network delay from the slave test node to the master test node;
The time spent in the master node receive buffers and protocol stack.

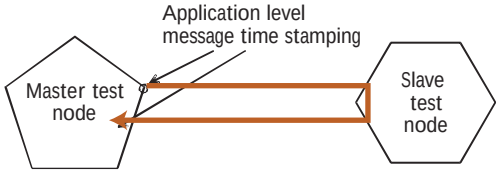


Fig 4. Measuring application-to-application transmission delay

Since we are testing a switched Ethernet set-up, the network delays can be further subdivided into:

a) Time spent across the transmitting node drop link;
b) Time spent from the switch input port to the switch output port;
c) Time spent in the switch output port buffers;
d) Time spent across the receiving node drop link.

For a given message size and a given switch, the delays *a, b* and *d* are independent of any priority tagging and can be calculated. Delay *c* depends on the traffic into the receiving node and the priority of the measurement message.

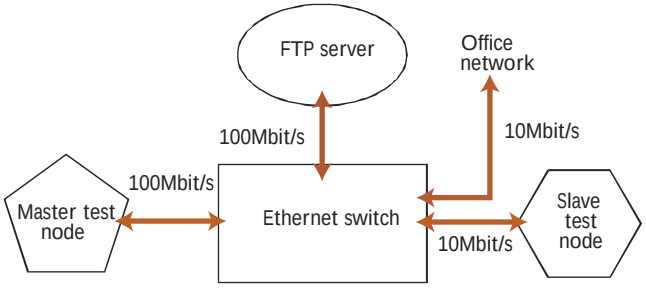


Fig 5. The complete measurement set-up

Fig 5 shows the measurement set-up. It contains the basic test elements (master node, slave node and network) plus an FTP server (may be used as a load generator) and a connection to a standard office network (used for controlling the test and downloading the results). Every measurement result set (in µs) is based on 200 individual measurements.

When running the tests, some unexpected annoyances surfaced and had to be dealt with. A 3Com SuperStack 3 switch 3300 SM was originally used in the test since it was the most up-to-date on the market with respect to the implementation of the GVRP and GMRP protocols. Unfortunately this switch was too smart for the test: it interpreted the tags correctly, but removed them before passing the packets to the node under test. Using an OnTime switch instead removed the problem. Also the first version of the test program used the default task priority. This had the unfortunate side effect of delaying the high priority packet generator whenever the load generator was busy. Running the test program at a very high priority got rid of that delay.

Baseline measurements

The raw test message was 160 bytes long (168 with preamble and CRC). The drop link transmission time for the test message took 13.44µs at 100Mbps drop link speed (10Mbps pro rata). Since the internal port-to-port delay in the switch is negligible, this meant the test message spent about 148µs on the drop links going from the master test node to the slave and the same amount back again. (296µs spent on the drop links for every roundtrip delay measurement). The baseline measurements (no FTP load) yielded a value of about 700µs for the roundtrip delay measurement with a standard deviation of about 14µs possibly arising from high priority processes pre-empting the CPU resources.

Measurements using untagged packets. Table 1 shows the round-trip delays for untagged packets in the presence of FTP receive and transmit loads (the bottom row shows the combined statistics). We see that the delay is much more influenced by the transmit load than the receive load. This was to be expected, since the Ethernet hardware transmit buffer may contain several full-size packets whenever a measurement packet is scheduled for transmission. This fact also shows up in the large standard deviation under transmit load.

Receive load				**Transmit load**			
Mean	Std	Max	Min	Mean	Std	Max	Min
749.2	10.8	780.1	724.8	1097.6	96.9	1423.7	742.9
752.8	11.5	781.8	729.6	1103.9	96.0	1481.5	733.8
763.3	28.8	863.7	723.2	1089.8	85.8	1287.8	742.2
754.1	12.2	786.9	726.2	1081.9	95.7	1418.7	720.5
754.9	17.5	863.7	723.2	1093.3	93.7	1481.5	720.5

Table 1. Measurements using untagged packets

Using tagged packets but without stack strategy. The results under receive load are more or less identical to the results using untagged packets. A possible explanation is that the receive load is far below the maximum switch capacity, enabling the measurement packets to go more or less directly to the measurement node whether or not the measurement packets are tagged.

The results under transmit load are also more or less identical to the results using untagged packets. A possible explanation is that the delays are due to lack of priority in the Ethernet hardware transmit buffer.

Measurements using transmit priority queuing. Table 2 shows the round-trip delays for priority tagged packets in the presence of FTP receive and transmit loads with internal transmit priority queuing (the bottom row is the combined statistics). We see a significant reduction in the delay under transmit load and an increase in the delay under receive load. The results from the powerful FTP server show the same tendencies, only more clearly (greater differences, smaller deviations between the results).

Receive load				**Transmit load**			
Mean	Std	Max	Min	Mean	Std	Max	Min
733.9	19.8	833.4	684.2	720.3	20.3	860.5	685.8
735.8	19.2	838.6	700.6	725.4	18.7	861.2	690.5
739.1	16.6	819.9	704.1	727.5	18.4	849.7	688.3
735.2	12.8	819.0	697.3	723.8	19.1	844.5	602.9
736.0	17.3	838.6	684.2	724.3	19.1	861.2	602.9

Table 2. Measurements using tagged packets with transmit priority queuing

MOXA

Total Solution for Industrial Device Networking

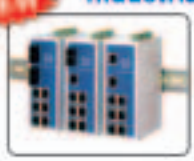
Want to know more about Serial-to-Ethernet Solution?

Go www.moxa.com to registering for downloading Moxa's free gift *"The Serial-to-Ethernet Guidebook: Solutions Utilizing Serial Device Server Technology"*

NPort Multiport Serial Device Servers

Industrial grade, 1/2/4/8/16 ports serial device server that can network-enable RS-232/422/485 devices located anywhere

Industrial Ethernet Switch



8-port intelligent Ethernet Switch with redundant and intelligent Ethernet capability

NPort Single-Port Serial Device Servers

Easy-to-use, RS-232/422/485 3-in-1 serial device server that can equip RS-232/422/485 devices with network capability instantly

Multiport Serial Boards

Cost-effective and high performance I/O solution with 2 to 32 RS-232/422/485 ports per board

Moxa Technologies GmbH
Tel: +49-2405-6627-66
info@moxatech.de

Moxa Technologies
Shanghai: +86-21-64666388/91
Beijing: +86-10-6872-3958/60/61
Shenzhen: +86-755-8368-4084/94
moxa@moxa.com.cn

Moxa Networking, Inc. (East)
Tel: +1-732-738-6280
Moxa Technologies, Inc. (West)
Tel: +1-626-961-2377
info@moxa.com

Moxa Technologies Co., Ltd.
Tel: +886-2-8919-1230
Fax: +886-2-8919-1231
info@moxa.com.tw

www.moxa.com

MOXA

Circle 65

Receive load				Transmit load			
Mean	Std	Max	Min	Mean	Std	Max	Min
712.1	17.3	860.7	683.7	710.6	18.8	864.4	682.1
716.3	20.1	867.7	681.9	715.7	17.1	871.4	684.5
717.1	17.9	859.3	687.5	715.0	17.3	859.4	686.9
714.8	17.6	872.0	687.5	715.7	17.9	863.2	687.3
715.1	18.3	872.0	681.9	714.8	17.8	871.4	682.1

Table 3. Measurements using tagged packets with receive and transmit priority queuing

Receive load				Transmit load			
Mean	Std	Max	Min	Mean	Std	Max	Min
766.9	141.9	1211.2	682.7	1091.7	87.7	1407.6	766.9
775.7	147.9	1359.5	687.6	1088.8	108.6	1430.0	775.7
714.5	12.6	750.3	684.0	1079.6	97.6	1427.9	714.5
776.5	148.7	1232.7	685.7	1067.9	97.4	1357.3	776.5
758.4	126.7	1359.5	682.7	1082.0	98.1	1430.0	758.4

Table 4. Measurements using tagged packets with receive priority queuing

Table 3 shows what happens when we use priority queuing in both receive and transmit situations (as usual, the bottom row is the combined statistics). We see a clear improvement under both receive and transmit loads.

Measurements using receive priority queuing,. Table 4, show the round-trip delays for priority tagged packets in the presence of FTP receive and transmit loads with internal receive priority queuing. The bottom row presents the combined statistics. We see the same improvements under receive load as in Table 3.

The receive load results deserve some comment. The large maximum values occur in one or two cases out of 250. This results in a slight change in the mean value, but on the other hand causing a large change in the standard deviation. The reason for these large values is currently an open issue; it might be that those measurement series have been considerably influenced (disturbed) by OS process handling, either in the embedded node or in the FTP server. However, it follows that the proposed receive priority queuing algorithm is not very effective when deployed as a standalone mechanism.

Traffic rules give predictability

There is enough here to conclude that the Ethernet switch and the droplinks do not constitute a bottleneck under FTP load at 100Mbps but rather the main communication delays occur inside the nodes. A suitable queue strategy for high priority packets assures that those will be processed before low priority packets. However the chosen switch must be configured not to remove the priority tagging information.

The introduction of the IEEE 802.1p traffic prioritisation standard does enable calculation of worst case latency through a switched Ethernet infrastructure. The assumption is that the real-time traffic pattern using the high priority queues is known, including the number of forwarding switches, and that the non-critical traffic uses lower priority present a solution for taking control of the latency in the end-nodes by introducing a QoS mechanism into the protocol stacks. However, a missing link on deterministic behaviour has been the ability to take control latency within the end-nodes potentially causing significant jitter. The introduction of a QoS mechanism into the protocol stacks does alleviate this which is borne out through real world experiments.

Tor Skeie , Svein Johannessen, and Øyvind Holmeide are with ABB Corporate Research

ADAM ....Connecting the eWorld

Advantech Industrial Networking guarantees the lowest cost of ownership for industrial GSM modem, Ethernet-to-Fiber, Ethernet Switch/Hub, Ethernet to RS- 232/422/485, and Serial Converter.

Industrial Ethernet Hub/Switch
ADAM-6510/6520/6521
Industrial Ethernet Hub/Switch

Ethernet to Fiber Media Converter
ADAM-6541/6542
Industrial Ethernet to Fiber Optic

Ethernet Data Gateway
ADAM-4570/4571/EDG-4504/4508/4516
Industrial Ethernet to RS-232/422/485

Serial Converter
ADAM-4520/4521
RS-232 to RS-422/485 Converter

ADAM-4541
Fiber Optic to RS-232/422/485 Converter

Wireless GSM Modem
ADAM-4581
GSM to RS-232/422/485 Wireless Data Gateway

Advantech Automation USA
Call Center Tel:887-294-8989
email: info@advantech.com
Advantech UK
Tel:44(0)1908304800
email: automation@advantech-uk.com

Please visit advantech website for more information
Advantech Industrial Automation
www.advantech.com/ia
Advantech Worldwild Contact
www.advantech.com/ia/cotact
Advantech
www.advantech.com

Advancing eAutomation

ADVANTECH

Circle 68