CONTROL TECHNOLOGY CORPORATION

M3-61B DeviceNet Slave Module

M3-61B DeviceNet™ Slave Module

Blank

WARNING: Use of CTC Controllers and software is to be done only by experienced and qualified personnel who are responsible for the application and use of control equipment like the CTC controllers. These individuals must satisfy themselves that all necessary steps have been taken to assure that each application and use meets all performance and safety requirements, including any applicable laws, regulations, codes and/or standards. The information in this document is given as a general guide and all examples are for illustrative purposes only and are not intended for use in the actual application of CTC product. CTC products are not designed, sold, or marketed for use in any particular application or installation; this responsibility resides solely with the user. CTC does not assume any responsibility or liability, intellectual or otherwise for the use of CTC products.

The information in this document is current as of the following Hardware and Firmware revision levels. Some features may not be supported in earlier revisions. See www.ctc-control.com for the availability of firmware updates or contact CTC Technical Support.

Model Number	Hardware Revision	Firmware Revision
M3-61B	All Revisions	>= M361AV0102 >= BF5300V059046

TABLE OF CONTENTS

[1] OVERVIEW **5**

MODEL 5300 FIELDBUS MODULE ARCHITECTURE 5
M3-61B DEVICENET SLAVE MODULE 5
FRONT PANEL 6

[2] DEVICENET **7**

NETWORK OVERVIEW 7
TECHNICAL FEATURES OF DEVICENET 8
HMS ANYBUS-S 9
DEVICENET FEATURES 9

[3] INTERFACE BASICS **11**

BASIC ARCHITECTURE 11

[4] DEVICENET NETWORK SETUP OVERVIEW **13**

INSTALLATION 13
MASTER CONFIGURATION 14
CONTROLLER IO MAPPING 18
ADMINISTRATIVE SCREEN DEVICENET WINDOW 18

[5] EXPLICIT MESSAGING ACCESS **23**

ASSEMBLY OBJECT, CLASS 04H - INPUT DATA 23
ASSEMBLY OBJECT, CLASS 04H - OUTPUT DATA 25
PARAMETER DATA INPUT MAPPING OBJECT, CLASS B0H 27
PARAMETER DATA OUTPUT MAPPING OBJECT, CLASS B1H 28

[6] SPECIAL REGISTER FEATURES **30**

HIGH SPEED DUALPORT REGISTERS 30
HIGH SPEED DUALPORT REGISTER 48-50 30

CHAPTER 1

[1] Overview

The 5300 series programmable automation controllers can be simultaneously connected to one or more fieldbus networks. Modbus master and slave communications are built into the CPU module. Modbus master and slave communications are supported on both the serial COM ports as well as the Ethernet ports. Additional fieldbus networks are supported via Model 5300 Fieldbus Modules that plug into the 5300 backplane. CTC currently offers Model 5300 modules for the following fieldbus networks:

DeviceNet Master	M3-61A
DeviceNet Slave	M3-61B
EtherNet/IP Master	M3-61C
EtherNet/IP Slave	M3-61D

Additional fieldbus modules are under development for popular fieldbus networks such as Profibus, CANOpen and others. To check on the release status of modules other than those listed above, contact CTC sales.

Model 5300 Fieldbus Module Architecture

The CTC fieldbus modules contain two circuit cards. The first card is the universal fieldbus adapter, which handles all interfacing tasks between the Model 5300 controller and the second card, which is called the fieldbus interface adapter. The fieldbus interface adapter is developed by HMS. In adopting this architecture CTC teamed up with HMS (http://www.hms.se/) who is the industry leader in industrial networking cards. This allows CTC to provide a wide range of network interfaces. Additionally CTC benefits from HMS's large engineering staff which is focused on updating the fieldbus interfaces and making sure they are in compliance with the applicable ratings agencies.

M3-61B DeviceNet Slave Module

The M3-61B module provides DeviceNet Slave support for the 5300 series controller. This includes the ability to access all digital, analog IO, and up to 50 general registers.

Both polled and explicit messaging is supported with multiple DeviceNet Slave cards as well as Master (M3-61A)/Slave configurations.

Front Panel

LED – NS, Network Status, green - online, flashing green - no connection, red – link failure.
LED – MS, Module Status
LED – RS, Run Status, not used.

LED	LED Status	Description
Module status	Off	No power or not initialized
	Green	Module status is OK
	Flashing red	Minor fault
	Red	Major fault

Switch – 1, 2 baud rate (on = 1)

Switch – 3 to 8 MACID, 0 to 63 binary with switch 8 low bit.

Baudrate (kBit/sec)	DIP 1-2
125	0 0
250	0 1
500	1 0
Reserved	1 1

USB & COM are used for re-flash of firmware and future optional RS232 serial port.

LED 1-4 are reserved for future use.

CH – CAN High (terminating 120 ohm resistors required)
SD – Shield/Drain
CL – CAN Low

CHAPTER 2

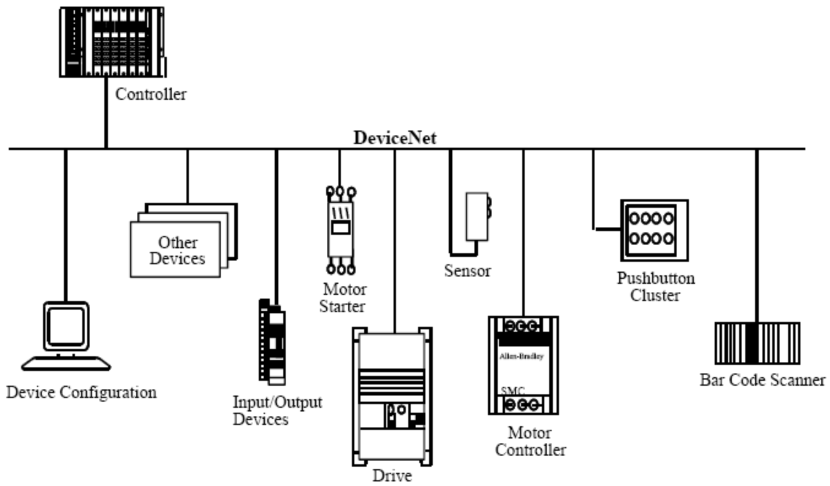
[2] DeviceNet

DeviceNet is a fieldbus system used for industrial automation, normally for the control of valves, sensors and I/O units and other automation equipment. The DeviceNet communication link is based on a broadcast oriented, communications protocol, the Controller Area Network (CAN). This protocol has fast I/O response and high reliability even for demanding applications.

DeviceNet has a user organization, the Open DeviceNet Vendor Association (ODVA), which assists members of matters concerning DeviceNet. HMS is a member of ODVA and also represented as a member of the DeviceNet Conformance SIG.

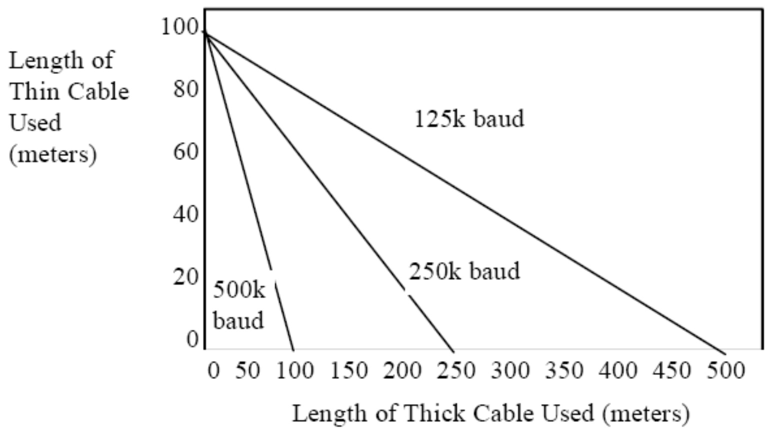
Network Overview

The physical media for the fieldbus is a shielded copper cable composed of one twisted pair and two cables for the external power supply. The baud rate can be changed between 125k, 250k and 500k bit/sec. Each node in the network is given a MAC ID, which is a number between 0 and 63 and is used to address the node.



Technical Features of DeviceNet

The maximum length of cable is dependent on the baud rate and DeviceNet cable that are used. Below is a diagram that shows the maximum allowed cable length in the network.



$L_{thick} + 5 \times L_{thin} = 500$ at 125Kbaud
$L_{thick} + 2.5 \times L_{thin} = 250$ at 250Kbaud
$L_{thick} + L_{thin} = 100$ at 500Kbaud

where L_{thick} is the length of thick cable and Lthin is the length of thin cable.

Summary: The Technical Features of DeviceNet

- DeviceNet specific cable (twisted pair)
- Access to intelligence present in low-level devices
- Master/Slave and Peer-to-Peer capabilities
- Trunkline-dropline configuration
- Support for up to 64 nodes
- Node removal without severing the network
- Simultaneous support for both network powered (sensors) and self powered (actuators) devices
- Use of sealed or open style connectors
- Protection from wiring errors
- Selectable data rates of 125k baud, 250k Baud, and 500k baud max. Trunk distance 500 meters and drop length 156 meters at 125k baud.
- Adjustable power configuration to meet individual application needs
- High current capability (up to 16 amps per supply)
- Operation with off-the-shelf power supplies
- Power taps that allow the connection of several power supplies from multiple vendors that comply with DeviceNet standards
- Built-in overload protection
- Power available along the bus; both signal and power lines contained in the trunkline
- Provisions for the typical request/response oriented network communications
- Provisions for the efficient movement of I/O data
- Fragmentation for moving larger bodies of information
- Duplicate MAC ID detection

HMS AnyBus-S

The M3-61B uses the HMS AnyBus-S DeviceNet interface module to ensure full compliance. As such, the module will appear on the network with the following parameters:

Description	Text string	Dec	Hex
Vendor ID	HMS Fieldbus Systems AB	90	0x005A
Product type	Communications adapter	12	0x000C
Product code	-	12	0x000C
Product name	AnyBus-S DeviceNet	-	-

The ANYBUS® S DeviceNet follows the DeviceNet standard that has been developed by ODVA. It is fully compatible with the DeviceNet specification rev. 2.0 Vol I and Vol II. The module operates according to the communication adapter profile (product type 12, see DeviceNet specification for more information). The module supports the I/O connections Bit strobe, Polled I/O, Change of state and Cyclic I/O data.

DeviceNet Features

Device Type:	Communication adapter	Slave:	Yes
Explicit peer-to-peer messaging:	Yes	I/O slave messaging:	
I/O peer-to-peer messaging:	No	Bit strobe	Yes
Configuration consistency value	Yes	Polling	Yes
Faulted node recovery:	No	Cyclic	Yes
Baud rates:	125K, 250K, 500K	Change of state (COS)	Yes

Blank

CHAPTER 3

[3] Interface Basics

The M3-61B must be configured prior to operation. This consists of setting the network speed and Master MACID via dip switches as well as setting up the proper network master configuration using a PC and device specific EDS (Electronic Data Sheet) files. The PC is then attached to the network and communicates with a DeviceNet Master, setting up the proper scan list of devices. A configurator is used to define how the data received from the remote M3-61B is transferred and how it is mapped into memory, hence assigned as I/O registers. In this chapter we will review the basic architecture of the M3-61B as it operates within the Model 5300 system. Then in Chapter 4 we will cover an actual example.

Basic Architecture

The M3-61B DeviceNet Slave operates asynchronously to the main Model 5300 controller, constantly updating IO information as it is collected from the controller backplane. In addition to its HMS AnyBus-S module there is a 60 MHZ ARM7 processor operating as an interface and high level controller. This processor handles the interface and mapping between the AnyBus-S DeviceNet data, and that observed by the controller, as well as all of the local register access. The AnyBus-S board handles the mapping of explicit messages to the IO area, as detailed in *Chapter 5: Explicit Messaging Access*.

There is a dual port memory device that exists between the Anybus-S and the ARM7 processor. Mapping of data by the DeviceNet Master and Configurator references this dual port memory. It is split into 3 sections. The first is for all Digital and Analog Inputs, data produced. The second is for all Digital and Analog Output access, data consumed. The third is for local register access, parameters. All Digital and Analog data are scanned while register access is by explicit message only.

Data is mapped into the Anybus-S dual port memory based upon the positioning of modules in the Model 5300 rack as well as the module type. First, all digital data is presented, followed by analog. Remaining space is consumed by registers with the total of I/O and registers limited to 512 bytes. As an example 1024 inputs would consume the first 128 bytes of memory, 32 analog inputs would then take up 32 * 4 bytes/analog, or

128 bytes, leaving 256 bytes for local registers. Up to 50 local registers are allowed, consuming 200 bytes. The remaining 56 bytes would not be used. If a larger number of analog I/O were present in the controller then the total number of local registers would be reduced to conform to the memory limitations. In some cases this could result in < 50. The telnet `get anybus info` command can be used to determine what is available.

When placing the M3-61B module in the controller rack, all local I/O modules to the left will be available for DeviceNet Slave access. Those modules to the right will remain local only and unavailable to the Master on the network, allowing the isolation of both local and public I/O on a DeviceNet network. This is also important given that if a Quickbuilder program changes an output value, the remote DeviceNet Master will not be aware of it. When writing with a DeviceNet Master, the existing previous data from an M3-61B is checked against the new data and only written if a change occurs. In a mixed Master/Slave environment (both cards installed in the Model 5300), only the local I/O is reported by the DeviceNet Slave. Remote I/O available to the Master card is ignored.

An Example of potential conflict would be if the DeviceNet Master set an analog output to 1000, then the Quickbuilder program set the same analog output to 2000. If the Master wrote a 1000 again the analog output would not be updated since from an M1-61B perspective that was the previous value (knows nothing of the application program change). If a 1001 were written it would have been modified. The local registers can be used as a communications mechanism or flag to the Master should this be needed.

CHAPTER 4

[4] DeviceNet Network Setup Overview

This chapter provides a high level overview of the steps necessary to set up a DeviceNet network to properly communicate with the Model 5300 automation controller.

Installation

Hardware Installation:

1. M3-61B: Set the baud rate to match the network and set a unique MAC ID. Install the module into the Model 5300 rack. It must be installed in a slot after any M3-40 series modules, so keep M3-40 modules to the left of the M3-61B module.
2. DeviceNet Devices: Set the baud rate to match the network and set a unique ID.
3. Network cabling: Connect the M3-61B following proper DeviceNet network wiring standards. Note that there is no internal network terminator on the M3-61B module, so proper network termination is required.

After powering up the Model 5300 controller you may access the remote administrative screens via telnet and verify module installation. Below shows two M3-61 modules installed in slots 6 and 8: 6 is a slave, 8 is a master:

```
BlueFusion/>get versions

*Local 5300 Serial Number = 10086477
  DNS Name: CTC_BF_10086477  DHCP active: YES
  Group Name:
  IP Address = 12.40.53.9 MAC Address = 00C0CB99E84D
  Total:  DIN = 48  DOUT = 40  AIN = 16  AOUT = 16  MOTION = 0
  Base Firmware Revisions:
    Quickstep ARM9 Application                  V05.00.90R46
    Quickstep ARM9 Monitor                      V01.49 A
    CT webHMI Enabled
  Available Racks:
    1. 8 slots local.
    2. Not installed.
    3. Not installed.
    4. Not installed.
  Slot Firmware Revisions:
    R1.01(01). M3-20C - 5VDC SNK 16DI 16DO 8CNT       (00000000) V01.05
    R1.02(02). M3-20C - 5VDC SNK 16DI 16DO 8CNT       (00000000) V01.05
    R1.03(03). M3-31A - 16 AIN +/-10V                 (17000297) V01.02
    R1.04(04). M3-32B - 16 AOUT +/-10V                (17000274) V01.02
    R1.05(05). Empty                                  (00000000) V00.00
    R1.06(06). M3-61B - DeviceNet Slave               (00000000) V01.02
    R1.07(07). Empty                                  (00000000) V00.00
    R1.08(08). M3-61A - DeviceNet Master              (00000000) V01.03

*

BlueFusion/>
```

Note that the boards installed in slots 1 to 4 are public on the DeviceNet network with the following produced/consumed data:

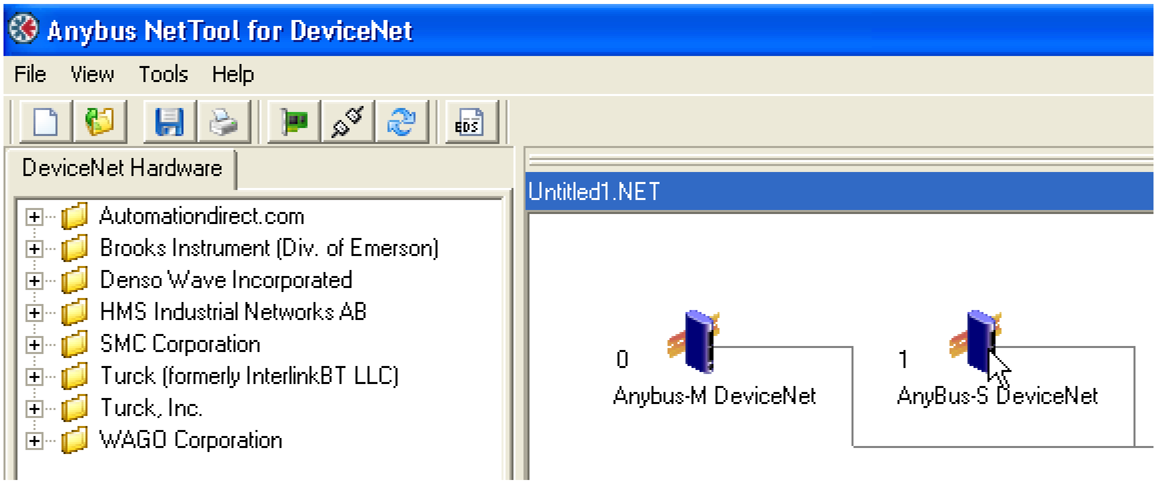
M3-20C: 2 bytes produced (16 inputs/8), 2 bytes (16 outputs/8) consumed.
M3-31A: 64 bytes produced (16 inputs * 4 bytes per value).
M3-32B: 64 bytes consumed (16 outputs * 4 bytes per value).

Total Produced = 68 bytes.
Total Consumed = 68 bytes.

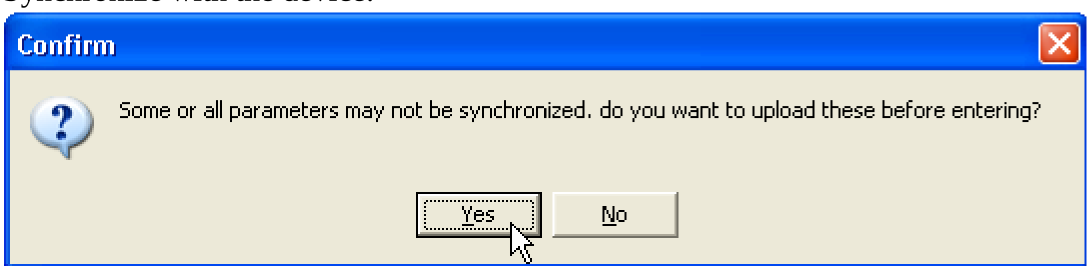
Master Configuration

Additional configuration information for the Model 5300 controller, with an M3-61B module is available in Appendix A of *Document No. 951-536101: M3-61A DeviceNet Master Module* applications guide. In summary, the proper RX/TX polled bytes can be derived using the calculation in the previous section and/or confirmed using NetTools and referencing the Model 5300 controller Anybus-S parameter area:

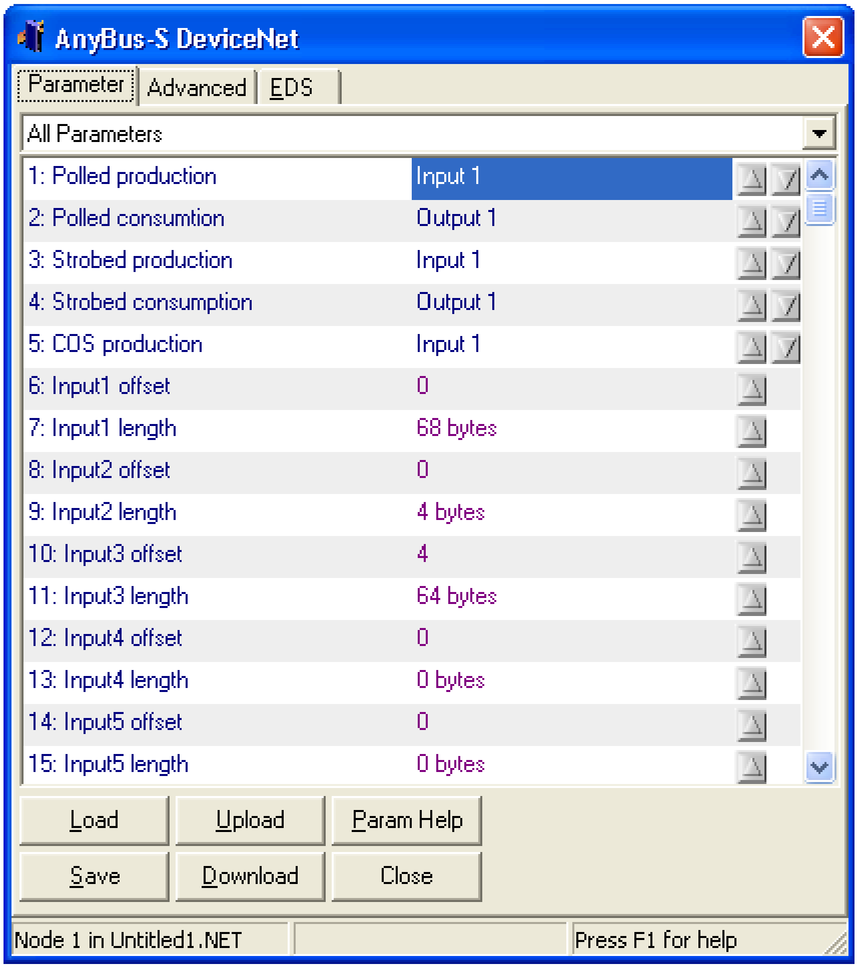
Double clicking from within NetTools:



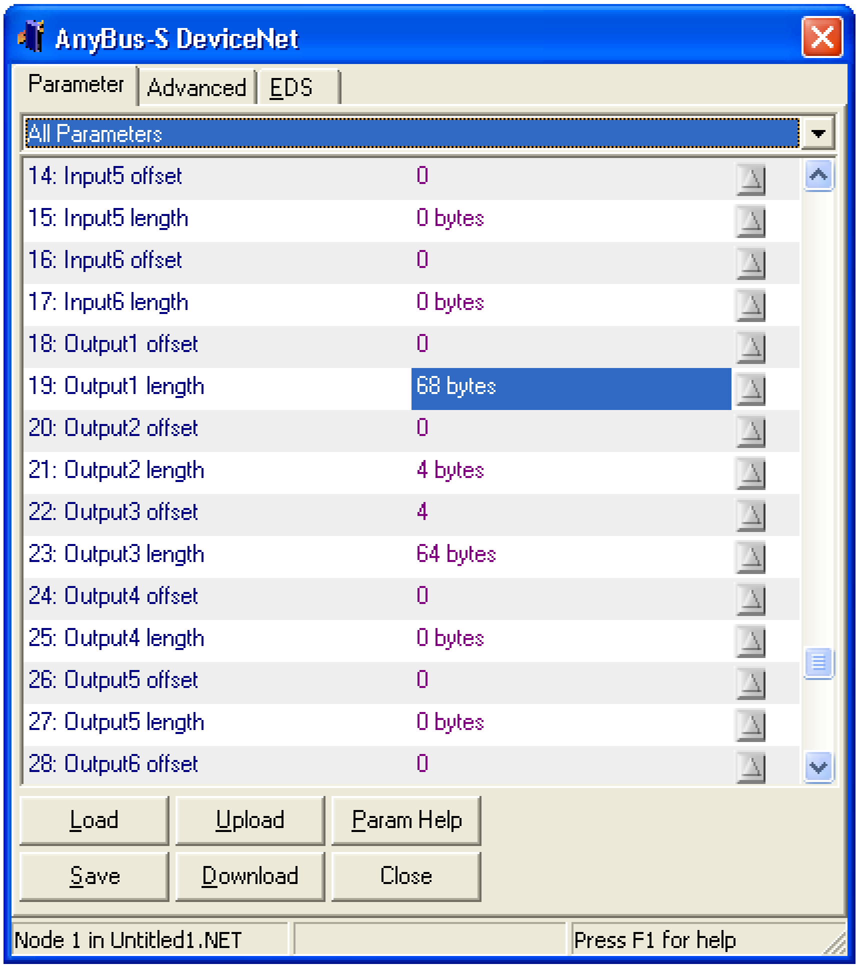
Synchronize with the device:



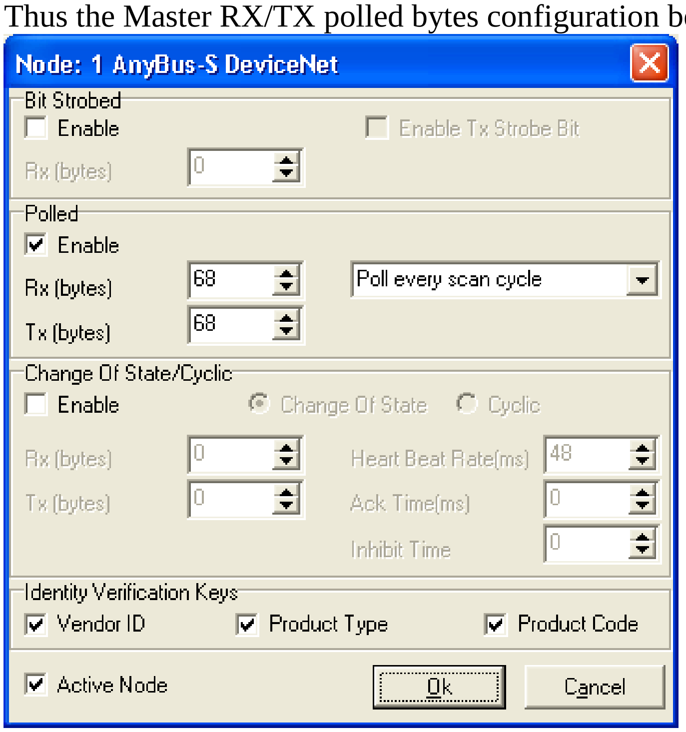
Reference 1 (polled production) and 2 (polled consumption). Note that #7 input 1 length is 68 bytes, this is RX or produced data:



Note that #19 output length is 68 bytes, this is TX or consumed data:



Thus the Master RX/TX polled bytes configuration becomes:



Controller IO Mapping

Chapter 5: Explicit Messaging Access discusses how I/O is mapped for explicit messaging. With regards to polling all digital is reported first, then analog:

RX bytes, produced:

```
struct
{
        unsigned char digitalIn[Number of digital inputs / 8];
        int analogIn[Number of analog inputs];
} IOInputs;
```

TX bytes, consumed:

```
struct
{
        unsigned char digitalOut[Number of digital outputs / 8];
        int analogOut[Number of analog outputs];
} IOOutputs;
```

Administrative Screen DeviceNet Window

The Model 5300 can be accessed via telnet or a serial port in order to access the standard Remote Administrative screen. From this screen general node status and version information can be obtained. The command to retrieve this information is `get anybus info`. Upon execution something similar to below will appear for each network module installed; in this example a slave is in slot 6 and a master in slot 8:

```
BlueFusion/>get anybus info
*** BOARD #1, Slot 6 ***
M3-61B DEVICENET SLAVE MODULE CONTROL INFO:
  XML Config Ver = 0x0000
  Avail Heap Mem = 464128
  Number Tags =    0x0000
  Active Exp Msgs= 0
  Boot Ver =       0x2301
  Mod Ver =        0x3601
  Fbus Ver =       0x3601
  Serial No =      0xFE230AA0
  VendorID =       0x0100
  Fbus Type =      0x2500
  LED[ ] =         0 1 0 1
  Module Status =  0x0302
M3-61B DEVICENET SLAVE MODULE FIELDBUS INFO:
  Identity Status = 0x0000
  Explicit Status = 0x00
  Polled Status   = 0x00
  Strobed Status  = 0x00
  COS Status      = 0x00
  Master Status   = 0x00
  Total DIN       = 16
  Total DOUT      = 16
  Total AIN       = 0
  Total AOUT      = 0
  Local Regs IN   = 50
  Local Regs OUT  = 50

*** BOARD #2, Slot 8 ***
M3-61A DEVICENET MASTER MODULE CONTROL INFO:
  XML Config Ver = 0x0204
  Avail Heap Mem = 462060
  Number Tags =    0x0000
  Active Exp Msgs= 0
  Boot Ver =       0x2301
  Mod Ver =        0x3101
  Fbus Ver =       0x0000
  Serial No =      0xE6340BA0
  VendorID =       0x0100
  Fbus Type =      0x2500
  LED[ ] =         0 1 1 1
  Module Status = 0x0102
M3-61A DEVICENET MASTER MODULE FIELDBUS INFO:
  Module Status = 00 06 04 00 00 4B 01 0C
  Node Active =   0000001C 00000000
  Node Faulted =  00000000 00000000
  Total DIN    =  24
  Total DOUT   =  24
  Total AIN    =  6
  Total AOUT   =  6
  Local Regs IN = 256
  Local Regs OUT= 256
  Node Status =   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00


BlueFusion/>_
```

The Master Module information is available in *Document No. 951-536101: M3-61A DeviceNet Master Module* applications guide. The following information only references the M3-61B in slot 6:

CONTROL INFO:

XML Config Ver – This is the XML module configuration file version that is currently installed. 0x0000 means there is no XML configuration file installed. By default none is needed for the standard configuration.

Avail Heap Mem – This is the amount of memory available for DeviceNet operation. It is referenced for diagnostic purposes to ensure explicit messages are returning memory to the system properly.

Number Tags – The number of tags defined in the XML configuration file. Not presently used on the M3-61B module.

Active Exp Msgs – Not used on the M3-61B module.

Control LED [ ] - Byte array of the 4 large LED's on the front panel. The important ones are the second and last one. Both must be 1's (on) for proper operation.
The second LED will be a 0 if there is no master polling. NS is LED 2, MS is LED 4 in the array. Off = 0, Green = 1, Red = 2.

LED	**State**	**Meaning**
Network Status (NS)	Off	Not powered, not on line
	Green	On line, one or more connections established
	Green, flashing	On line, no connections established
	Red	Critical link failure
	Red, flashing	One or more connections timed out
	Alternating Red/Green	Device self-test in progress
Module Status (MS)	Off	No power
	Green	Normal operation
	Green, flashing	Auto baud in progress
	Red	Major fault
	Red, flashing	Minor fault
	Alternating Red/Green	Device self-test in progress

FIELDBUS INFO:

Identity Status – This reflects attribute #5 of the Identity Object.

bit(s)	Name
0	Module Owned (A master/scanner has allocated the module)
1	(reserved)
2	Configured (always set to zero)
3	(reserved)
4... 7	Extended Device Status: Value: Meaning: 0000b Unknown 0010b Faulted I/O Connection (not implemented) 0011b No I/O connection established 0100b Non-volatile configuration bad (not implemented) 0110b Connection in Run mode 0111b Connection in Idle mode (other) (reserved)
8	Set for minor recoverable faults
9	Set for minor unrecoverable faults
10	Set for major recoverable faults
11	Set for major unrecoverable faults
12... 15	(reserved)

Explicit Status – This reflects attribute #1 in instance #1 of the Connection Object.

#	Access	Name	Type	Value
1	Get	State	USINT	Value: Meaning: 0 Non existent 1 Configuring 3 Established 4 Timeout 5 Deferred delete

Polled Status – This register reflects attribute #1 in instance #2 of the Connection Object.

#	Access	Name	Type	Value
1	Get	State	USINT	Value: Meaning: 0 Non existent 1 Configuring 3 Established 4 Timeout

Strobed Status – This reflects attribute #1 in instance #3 of the Connection Object.

#	Access	Name	Type	Value
1	Get	State	USINT	Value: Meaning: 0 Non existent 1 Configuring 3 Established 4 Timeout

COS Status – This reflects attribute #1 in instance #4 of the Connection Object.

#	Access	Name	Type	Value
1	Get	State	USINT	Value: Meaning: 0 Non existent 1 Configuring 3 Established 4 Timeout

Master Status – This indicates the current master scanner state:

0x00 – No information available, probably no Master connection.
0x01 – Master in RUN state
0x02 – Master in IDLE state

Total DIN – Total number of digital inputs that this card reports on the network.

Total DOUT – Total number of digital outputs that this card reports on the network.

Total AIN – Total number of analog inputs that this card reports on the network.

Total AOUT – Total number of analog outputs that this card reports on the network.

Total REGIN – Total number of local registers that a remote DeviceNet Master may read from. It is also the last mapped parameter input attribute, Class B0h.

Total REGOUT – Total number of local registers that a remote DeviceNet Master may write to. It is also the last mapped parameter output attribute, Class B1h.

.

CHAPTER 5

[5] Explicit Messaging Access

All M3-61B data may be accessed on demand by using explicit messages. There are some unique features when using explicit messages, for example being able to access only the digital IO, analog IO, or individual registers. When scanning, all IO is accessed as a block. In general the M3-61B follows the CIP Object implementation defined by the HMS Industrial Networks Anybus-S module:

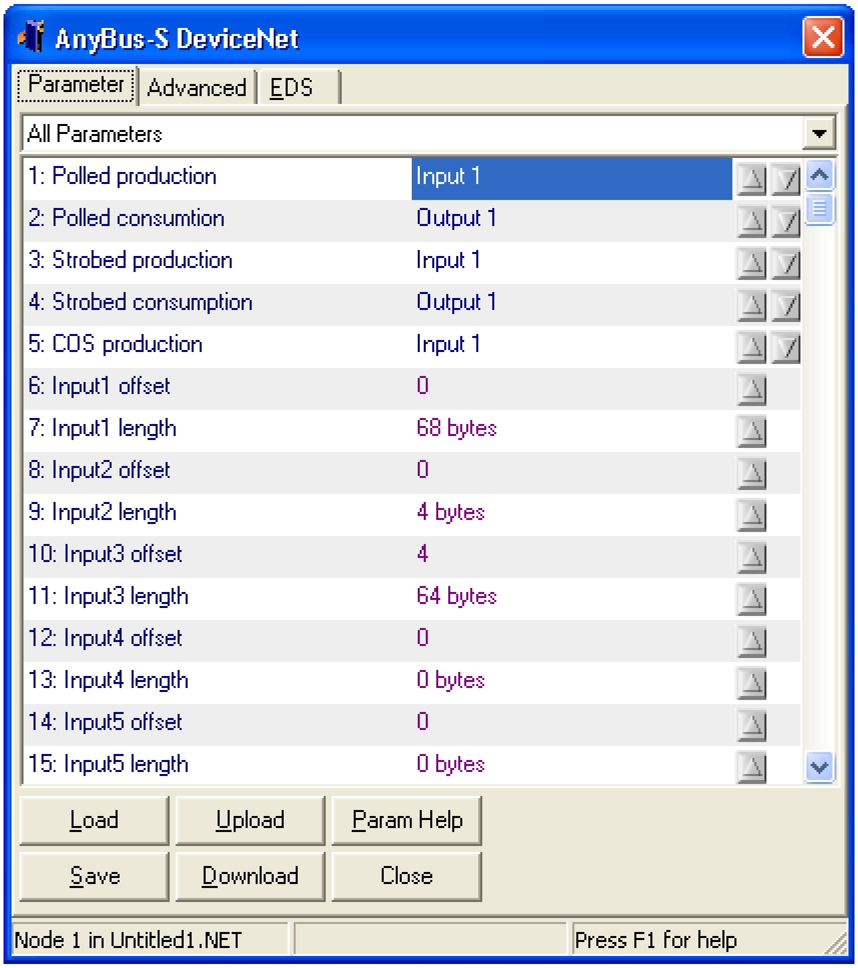
http://www.anybus.com/upload/72-8125-ABS-DEV_Fieldbus_Appendix_2_08.pdf

The following sections detail the areas that have been customized for the M3-61B.

Assembly Object, Class 04h - Input Data

The Assembly Object can be used to read or write blocks of digital and analog IO. Instances 64h to 66h are for input data (produced).

Referencing the M3-61B parameters in NetTools you can derive the number of bytes returned for each of the Instances defined below.



Instance 64h – Input 1 (#6/7)

Attribute	Access	Name	Type	Description
3	Get	Digital/ Analog Inputs	Structure of Digital USINT and Analog DINT (signed 32 bit)	Model 5300 Digital and Analog Input data

Structure construct for Instance 64h data is as follows:

```
struct
{
        unsigned char digitalIn[Number of digital inputs / 8];
        int analogIn[Number of analog inputs];
} IOInputs;
```

Instance 65h – Input 2 (#8/9)

Attribute	Access	Name	Type	Description
3	Get	Digital Inputs	Array of Digital USINT	Model 5300 Digital Input data

Array construct for Instance 65h data is as follows:

unsigned char digitalIn[Number of digital inputs / 8];

Instance 66h – Input 3 (#10/11)

Attribute	Access	Name	Type	Description
3	Get	Analog Inputs	Array of Analog DINT (signed 32 bit)	Model 5300 Analog Input data

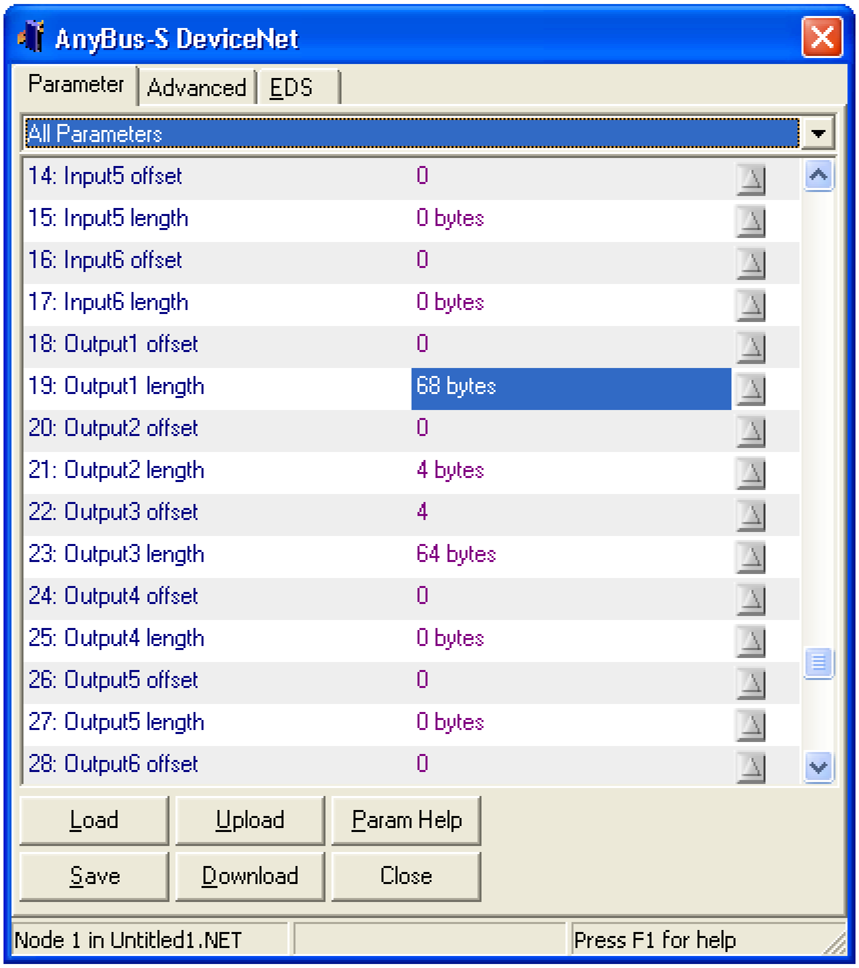
Array construct for Instance 66h data is as follows:

int analogIn[Number of analog inputs];

Assembly Object, Class 04h - Output Data

The Assembly Object can be used to read or write blocks of digital and analog IO. Instances 96h to 98h are for output data (consumed).

Referencing the M3-61B parameters in NetTools, you can derive the number of bytes required for each of the Instances defined on the following page:



Instance 96h – Output 1 (#18/19)

Attribute	Access	Name	Type	Description
3	Get/Set	Digital/ Analog Outputs	Structure of Digital USINT and Analog DINT (signed 32 bit)	Model 5300 Digital and Analog Output data

Structure construct for Instance 96h data is as follows:

```
struct
{
        unsigned char digitalOut[Number of digital outputs / 8];
        int analogOut[Number of analog outputs];
} IOOutputs;
```

Instance 97h – Output 2 (#20/21)

Attribute	Access	Name	Type	Description
3	Get/Set	Digital Outputs	Array of Digital USINT	Model 5300 Digital Output data

Array construct for Instance 65h data is as follows:
unsigned char digitalOut[Number of digital outputs / 8];

Instance 98h – Output (#22/23)

Attribute	Access	Name	Type	Description
3	Get/Set	Analog Outputs	Array of Analog DINT (signed 32 bit)	Model 5300 Analog Output data

Array construct for Instance 66h data is as follows:
int analogOut[Number of analog outputs];

Parameter Data Input Mapping Object, Class B0h

Parameter data input references the local registers that can be accessed both by the Model 5300 controller (variant array 36825) and by the DeviceNet network. Class B0h/Instance 01h is for reading the registers. Note that Attributes 48/49 allow for a block read of the registers. Attribute 50 is used to access the Model 5300 controller register access structure.

Instance 01h

Attribute	Access	Name	Type	Description
1	Get	Local Register 1	DINT (signed 32 bit)	Local Register
2 - 47	Get	Local Register 2-47	DINT (signed 32 bit)	Local Register
48	Get	Local Registers 1-25	Array of 25 DINT (signed 32 bit)	Local Register Block Low
49	Get	Local Registers 26-50	Array of 25 DINT (signed 32 bit)	Local Register Block High
50	Get	Controller register access structure	Structure block	Controller register access information

Attribute 50 Structure:

```
typedef struct __attribute__ ((__packed__))          // 3 ints, 6 bytes
{
        // REGISTER 48
        unsigned short regnum;   // Register to access
        unsigned char offset;    // Offset into local register array, with 0 first local register
        unsigned char reserved;
        // REGISTER 49
        unsigned short row;      // Row if neede,d (Variant only, else 0)
        unsigned short col;      // Column if needed, (Variant only, else 0)
```

```
    // REGISTER 50
    unsigned char qty;      // Number of registers to access, on completion bit 7 is set
    unsigned short type;    // Data type, VARIANT INTEGER, VARIANT FLOAT, VARIANT
                            // DOUBLE, BIT15 for write, BIT14 increment column
    unsigned char count;    // Incrementing counter for each operation, change of state causes
                            // execution
} REG_ACCESS;
```

Parameter Data Output Mapping Object, Class B1h

Parameter data output references the local registers that can be accessed both by the 5300 controller (variant array 36825) and by the DeviceNet network. Class B1h/Instance 01h is for writing the registers. Note that each Attribute 1 to 50, references an individual register for writing.

Instance 01h

Attribute	Access	Name	Type	Description
1	Get/Set	Local Register 1	DINT (signed 32 bit)	Local Register
2-49	Get/Set	Local Register 2-49	DINT (signed 32 bit)	Local Register
50	Get/Set	Controller register access structure	Structure block	Controller register access information

Note that writing the structure block to register 50 also changes registers 48 and 49.

Attribute 50 Structure:

```
typedef struct __attribute__ ((__packed__))          // 3 ints, 6 bytes
{
    // REGISTER 48
    unsigned short regnum;  // Register to access
    unsigned char offset;   // Offset into local register array, with 0 first local register
    unsigned char reserved;
    // REGISTER 49
    unsigned short row;     // Row if needed (Variant only, else 0)
    unsigned short col;     // Column if needed, (Variant only, else 0)
    // REGISTER 50
    unsigned char qty;      // Number of registers to access, on completion bit 7 is set
    unsigned short type;    // Data type, VARIANT INTEGER, VARIANT FLOAT, VARIANT
                            // DOUBLE, BIT15 for write, BIT14 increment column
    unsigned char count;    // Incrementing counter for each operation, change of state causes
                            // execution
} REG_ACCESS;
```

Attribute 50 is used to move registers either into (read) or out of (write) the DeviceNet Slave module. Thus a user would first use explicit messages to load the local registers as desired and then write to Attribute 50 with the appropriate information configured. Access to Class B0h, Instance 01h, Attribute 50 is done to polled bit 7 of the qty to verify completion. Also the count member should match that which was written to Class B1h. The count member must be incremented to initiate a transaction (causes change of state).

CHAPTER 6

[6] Special Register Features

As with the M3-61A DeviceNet Master Module, special registers are available on the M3-61B. This consists of a general storage area for shared DeviceNet remote mapped 32 bit data.

High Speed Dualport Registers

Up to 49 general purpose registers are available which are shared between the DeviceNet Network and the Model 5300 application program (register 50 is special purpose). From a DeviceNet network perspective these registers are accessed by explicit messages as parameter data, as discussed in *Chapter 5: Explicit Messaging Access*. From a Model 5300 application program perspective, this register area is accessed via Variant array 36825. This is a two dimensional array with the row reflecting the module number (0 = first), and the column being the register number (0 = first). Thus 36825[0][0] would be the first special register on the first DeviceNet module, 36825[0][1] would be the second, etc. Each Anybus Network board in the system represents the row number. Thus if the slave is the first board it would have a row index of 0.

The declared data type must match that of the application program, VARIANT_INTEGER (BIT0) or VARIANT_FLOAT (BIT3). Quickstep is limited to VARIANT_INTEGER. In most DeviceNet applications, only the integer type will be used, stored in little endian format.

High Speed Dualport Register 48-50

Local registers 48 to 50 can be used to form a structure that has special functionality when accessed remotely by DeviceNet (reference class B0/B1h in previous chapter). It can be used to move data in and out of registers 1 to 47, to/from the controller registers as a single or block access. Register 48 to 50 consume 12 bytes with the following structure, last byte (count) being a trigger to start the operation:

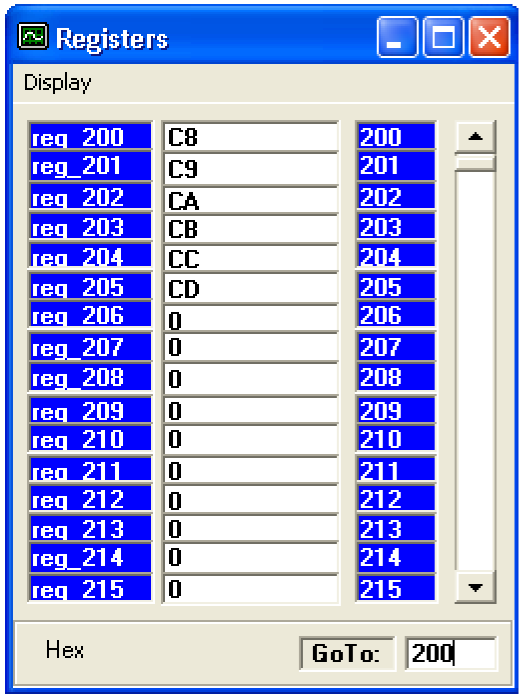
```
typedef struct __attribute__ ((__packed__))        // 3 ints, 6 bytes
{
        // REGISTER 48
        unsigned short regnum;  // Register to access
        unsigned char offset;   // Offset into local register array, with 0 first local register
        unsigned char reserved;
        // REGISTER 49
        unsigned short row;     // Row if needed (Variant only, else 0)
        unsigned short col;     // Column if needed (Variant only, else 0)
        // REGISTER 50
        unsigned char qty;      // Number of registers to access, on completion bit 7 is set
        unsigned short type;    // Data type, VARIANT INTEGER, VARIANT FLOAT, VARIANT
                                // DOUBLE, BIT15 for write, BIT14 increment column
        unsigned char count;    // Incrementing counter for each operation, change of state causes
                                // execution
} REG_ACCESS;
```

```
VARIANT_INTEGER   BIT0
VARIANT_FLOAT     BIT3
VARIANT_DOUBLE    BIT4
```

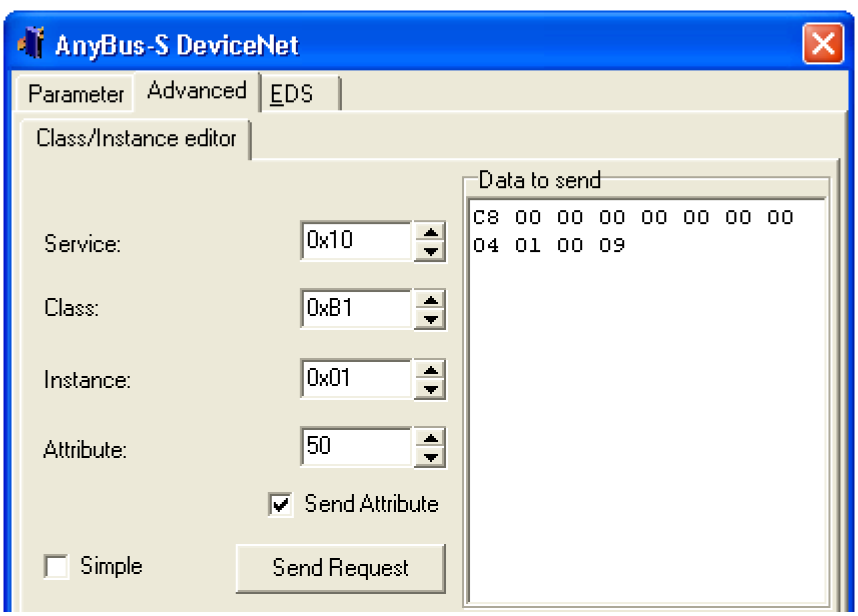
As an example, assume we want to read registers 200 to 203 into the local registers, starting with an offset of 0. This would be done by setting the structure as follows and writing to Class B1h, Instance 01h, Attribute 50:

regnum	– 200	// Controller register number would like to begin access on
offset	– 0	// 0 based, this being first local register
reserved	– 0	
row	– 0	// Not used, set to 0
col	– 0	// Not used, set to 0
qty	– 4	// 8, number of registers to move
type	– 1	// VARIANT_INTEGER, BIT15 clear for read, BIT14 clear since not Variant.
count	– 9	// Increment previous value, change triggers the reading of the // other registers (48/49). 9 is an arbitrary value, just has to be // different than what is currently in register 50.

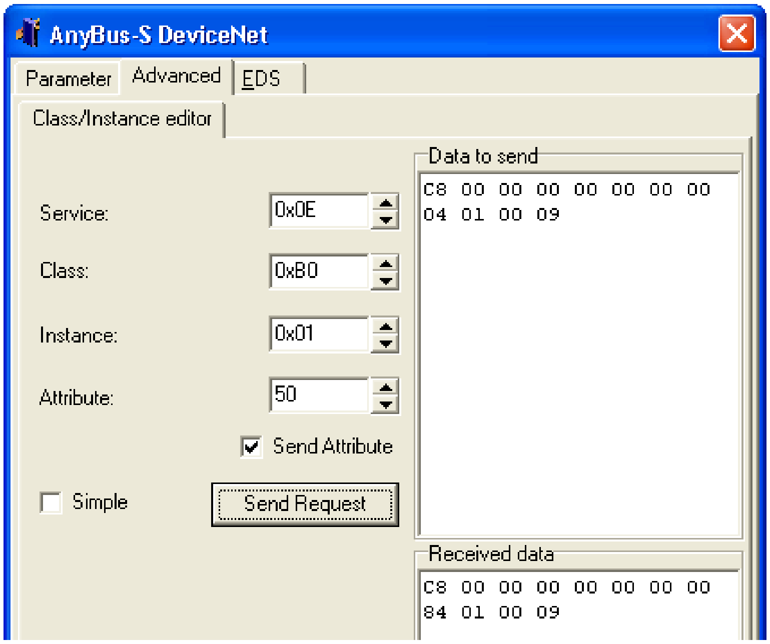
CTCMON can be used to verify what data is present in the registers to be read:



NetTools can then be used to create an explicit message with the structure data:



To check for completion verify that the ‘qty’ field has bit 7 set, 0x84 below:



To view the register results in the local registers use attribute 48 to read the lower 25 registers as a single block, note the values in the first 4 registers: