EMBEDDED SYSTEMS ACADEMY

CANopen Hands-On Tutorial

An Introduction to CANopen

using

CANopen Magic ProDS Eval

Presented by Olaf Pfeiffer

EMBEDDED SYSTEMS ACADEMY

www.CANopenMagic.com

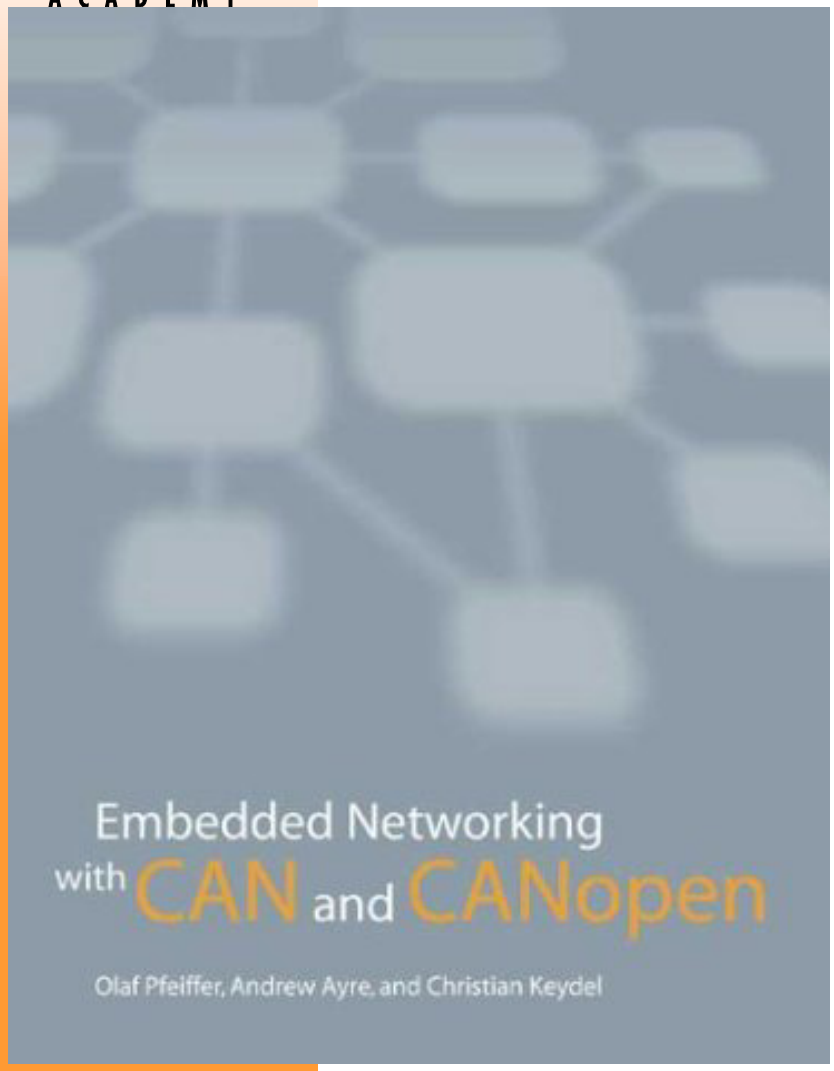
EMBEDDED SYSTEMS ACADEMY

Prerequisites

CANopen Magic ProDS Eval

- **All hands-on examples in this tutorial use the program**
  - CANopen Magic ProDS Eval
- **This tool allows simulation, configuration, analyzing and testing CANopen networks**
  - No hardware required
  - All CANopen communication is simulated
- **Download and install from**
  - www.canopenmagic.com

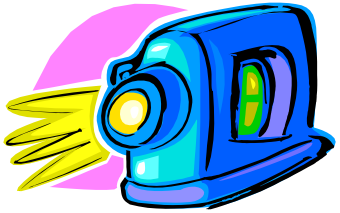
Our CANopen Book:
Embedded Networking with CAN and CANopen



- **Published by Annabooks / RTCBooks**
- **3 Parts**
  - Using CANopen
    Introductory level up to system integration
  - CANopen Engineering
    Developing CANopen nodes
  - CANopen Reference
    Quick access to all info required by integrators and developers
- **www.CANopenBook.com**

EMBEDDED SYSTEMS ACADEMY

Contents



Physical Settings

Network physical layer, message basics, communication bit rates

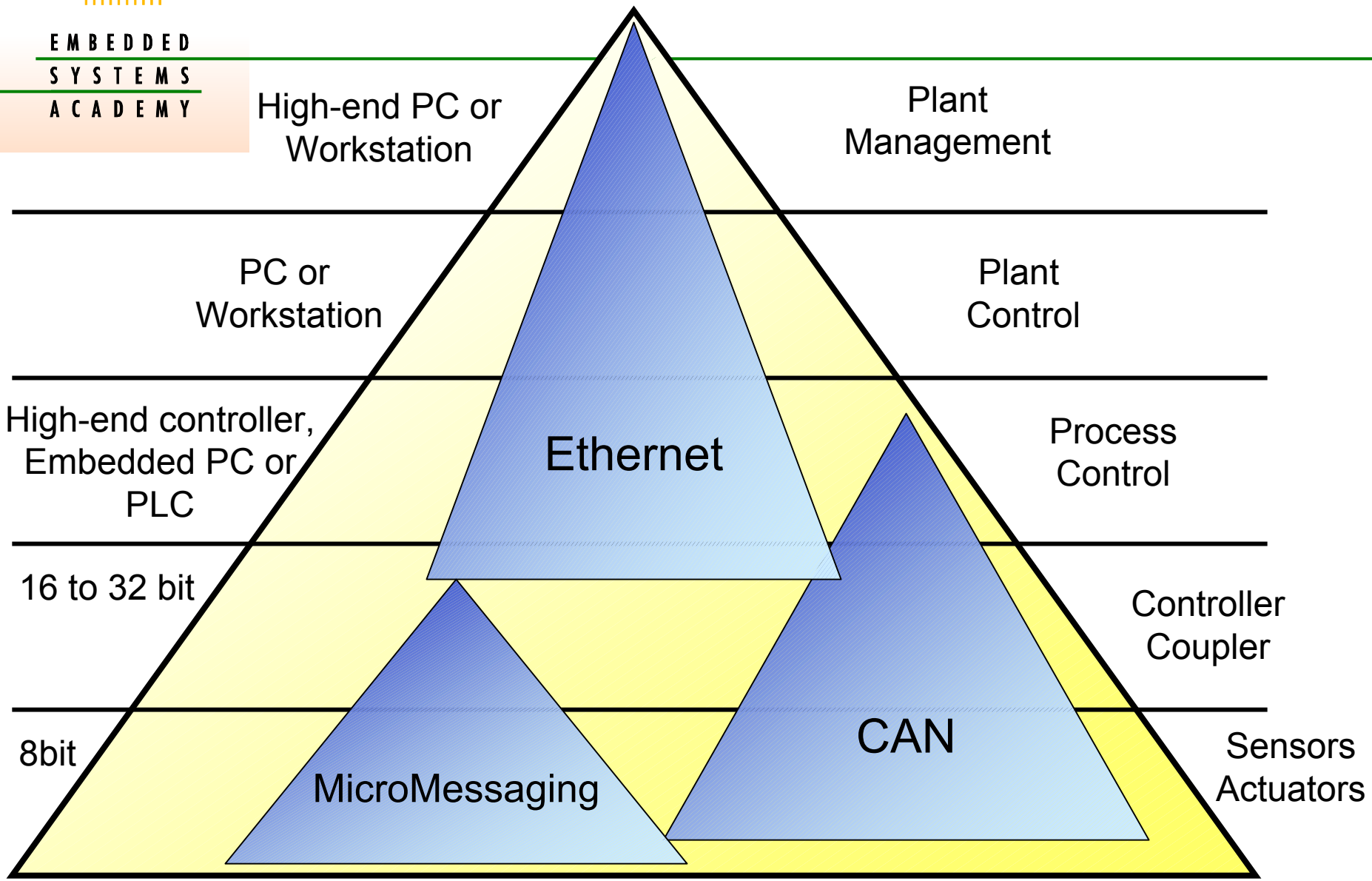
EMBEDDED SYSTEMS ACADEMY

CANopen is optimized for CAN

- **CANopen is 'open' to be used on a variety of networking technologies**
  - CANopen on Ethernet
    - www.Ethernet-Powerlink.com
  - CANopen on UART, I2C, LIN
    - www.MicroMessaging.com

- **However, it is optimized to be used on Controller Area Network (CAN)**
  - Using a maximum of 8 data bytes
  - Using message identifiers 0-1023

CANopen Suitable Physical Layers



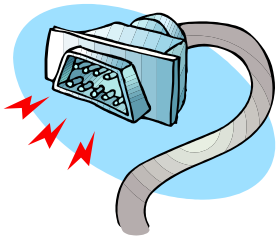
What's in a single message?

Identifier 11 bits	Data bits (0..8 Bytes)	Control and security 36 bits

- **In CAN, a single messages is kept short and only contains up to 8 data bytes**
- **Benefits:**
  - There can be many messages per second (rule over thumb: up to 10,000 per second at 1 Mbit, worst case of 20,000 per second)
  - No single message can occupy/block the network for a long time
  - Best for small sensors and actuators (I/O modules, encoder, push buttons, temperature,…)
- **Concept: send less data – more often**

EMBEDDED SYSTEMS ACADEMY

CANopen Network Speeds



- **When used on CAN, the specified networking bit rates are**
  - 10 kbps
  - 20 kbps
  - 50 kbps
  - 125 kbps
  - 250 kbps
  - 500 kbps
  - 800 kbps
  - 1000 kbps

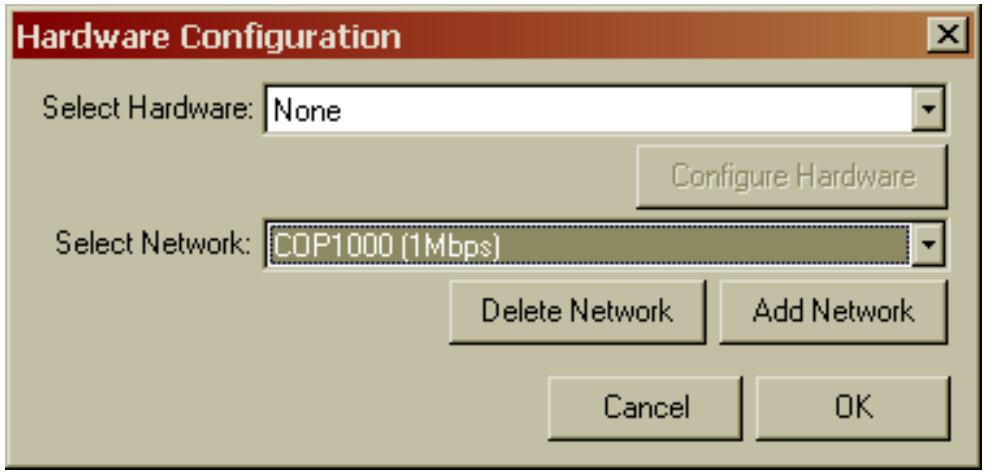
Each of these can be added to the hardware settings of CANopen Magic ProDS Eval

NOTE:
CANopen Magic ProDS Eval
Does not simulate network speed, available bandwidth only depends on the performance of the PC on which the simulation runs

Hands-On: Getting started

- **Start CANopen Magic ProDS Eval**

CANopen Magic ProDS Eval



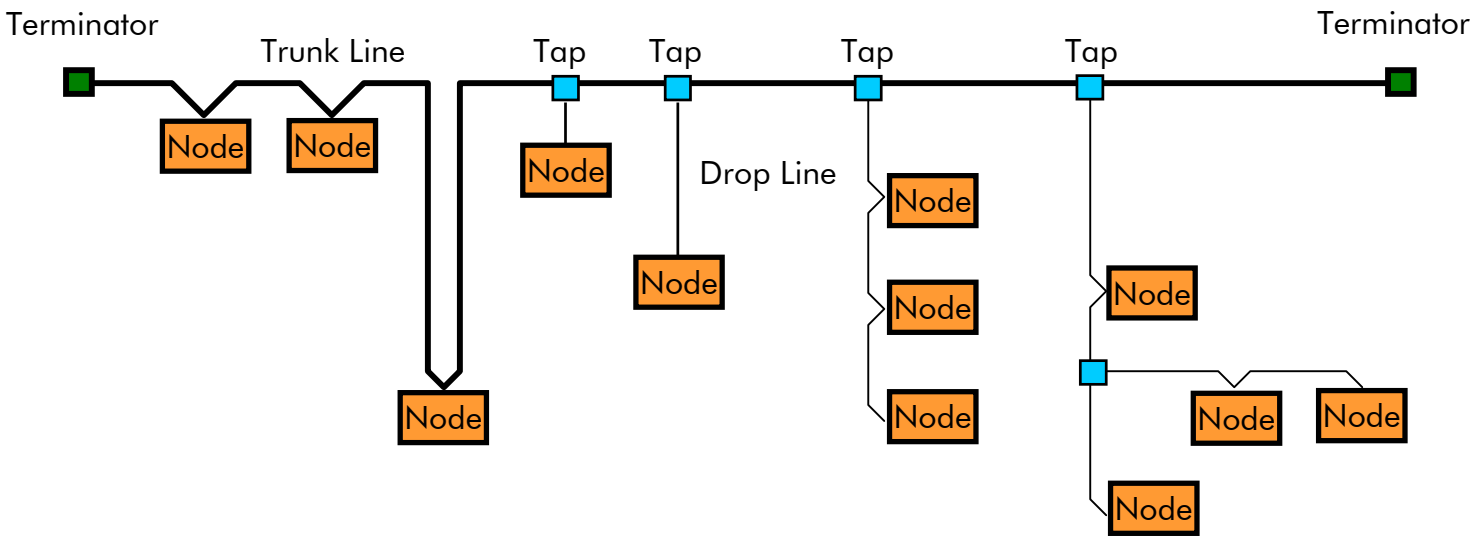
- **Select 'None' for the hardware and pick any network**
  - Networks are simulated "virtually" within the program. There is no "live" network traffic

Network Nodes

Unique Node IDs,
message IDs used by nodes
Default Connection Set

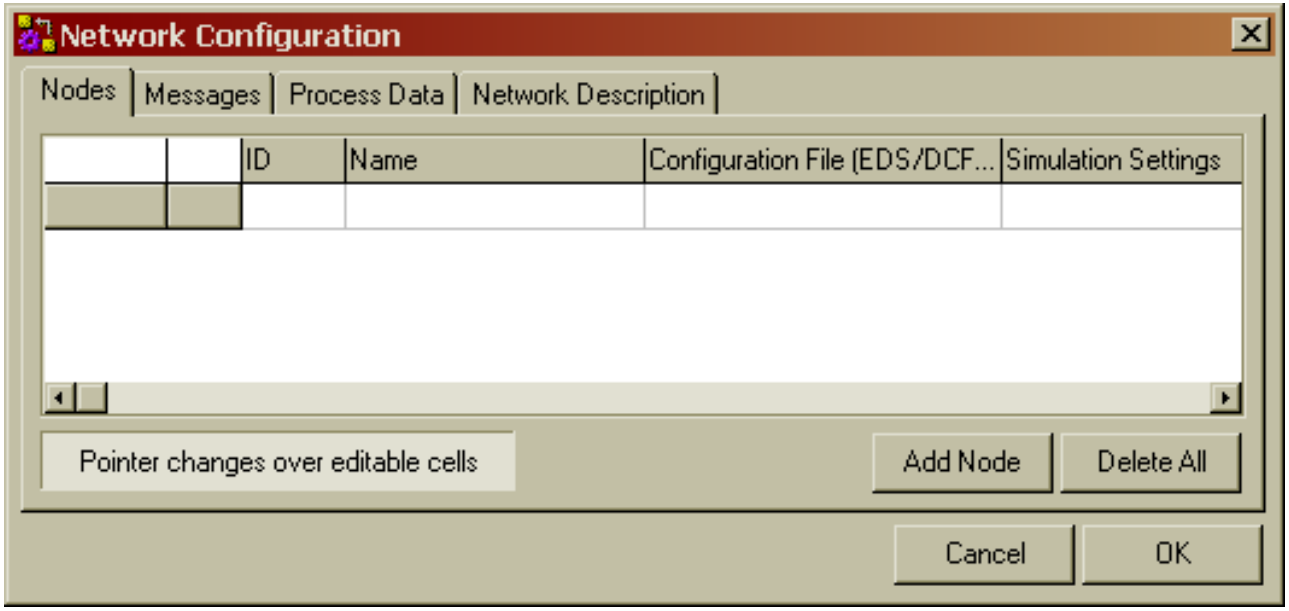
Layout with CAN physical layer

- **Main network trunk with termination resistors**
  - Drop lines only permissible if bit rate is 500kbps or below
- **Each node must have a unique node ID**
  - In the range of 1 to 127
- **Maximum length depends on network speed**
  - Example: about 250m with 250kbps



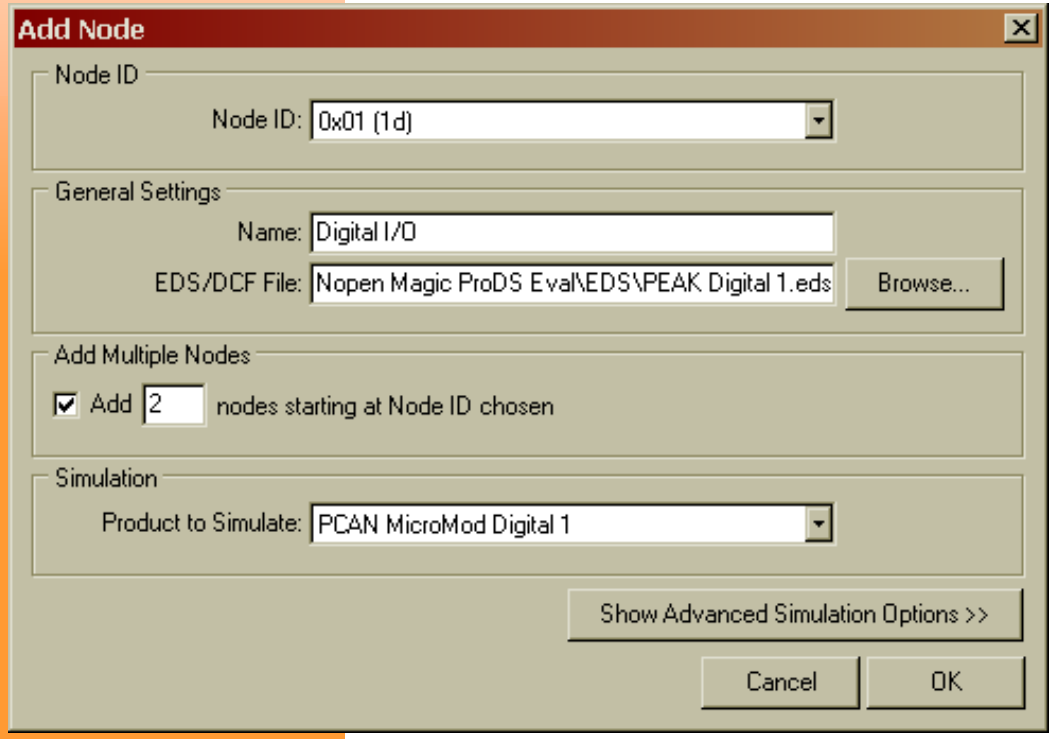
Hands-On: Network Configuration

- From the main menu, select Options – Configure Network
- Or press the Network Configuration tool button 



Hands-On:
Add nodes to the simulation

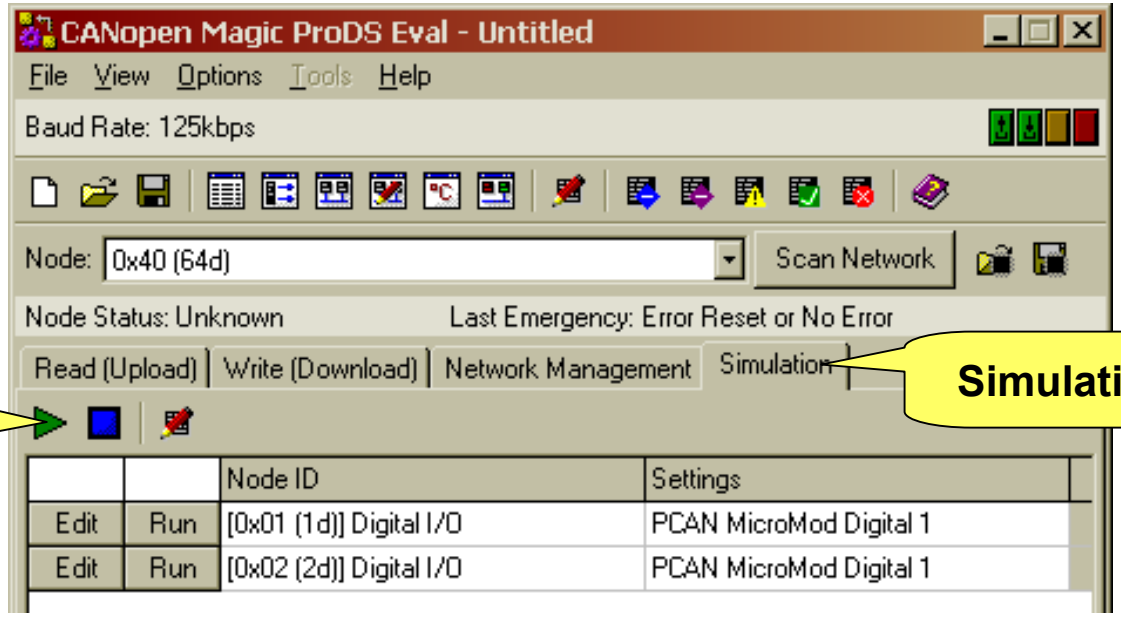
- In the Network Configuration window, click on the ‘Add Node” button
- Add two digital I/O nodes to the system



1. Node ID ‘1’
2. Name ‘Digital I/O’
3. EDS File ‘Peak Digital 1.eds’ from EDS directory
4. Add ‘2’ nodes
5. Choose the simulated product ‘PCAN MicroMod Digital 1’
6. Click the ‘OK’ button

Hands-On:
Run network simulation

- **Open Trace window, to see simulated network traffic**
  - From the main menu, select View – Trace
- **To run the simulation, go to the main window and select the simulation tab**
- **Click on 'Run' to run the simulation of individual nodes or on the green triangle to run all nodes**



Hands-On:
View first messages

- **Look at the Trace window**
- **Each node produced 2 messages**
  - Bootup (701h and 702h)
  - Emergency Clear (81h and 82h)

CAN Bus Trace - 0 messages/sec

Options

Time	Cnt	ID (hex)	Msg Type	Node	Details	Data (hex)
0:00:16:38:566.435	1	701	Bootup	[0x1 (1d)] Digital I/O		
0:00:16:38:566.464	1	081	Emergency	[0x1 (1d)] Digital I/O	Error Reset or No Error	00 00 00 00 00
0:00:16:39:020.763	1	702	Bootup	[0x2 (2d)] Digital I/O		
0:00:16:39:020.798	1	082	Emergency	[0x2 (2d)] Digital I/O	Error Reset or No Error	00 00 00 00 00

Message Identifier Assignment

- **The default message identifiers used by CANopen nodes are directly related to their node ID**
- **The node ID gets ‘embedded’ into the message identifier**
- **On CAN using an 11-bit message identifier, the node ID is in bits 0-7**
- **Bits 8-10 contain message type information**

- **Review the trace window:**
  - 700h base address is for ‘message type’ bootup
  - Add ‘1’ and ‘2’ to get the bootup messages for the nodes with the IDs 1 and 2

EMBEDDED SYSTEMS ACADEMY

Default usage of CAN message identifiers

- IDs not listed are free and may be used by system integrator
- IDs used for the PDOs and SDOs are derived by adding the "node ID"-1 to the "From" start address

With this scheme, the node ID gets inserted into bits 0 to 6 of the CAN message identifier

CAN ID		
From	To	Communication Objects
0h		NMT Service
80h		SYNC Message
81h	FFh	Emergency Messages
100h		Time Stamp Message
181h	1FFh	1st Transmit PDO
201h	27Fh	1st Receive PDO
281h	2FFh	2nd Transmit PDO
301h	37Fh	2nd Receive PDO
381h	3FFh	3rd Transmit PDO
401h	47Fh	3rd Receive PDO
481h	4FFh	4th Transmit PDO
501h	57Fh	4th Receive PDO
581h	5FFh	Transmit SDO
601h	67Fh	Receive SDO
701h	77Fh	NMT Error Control

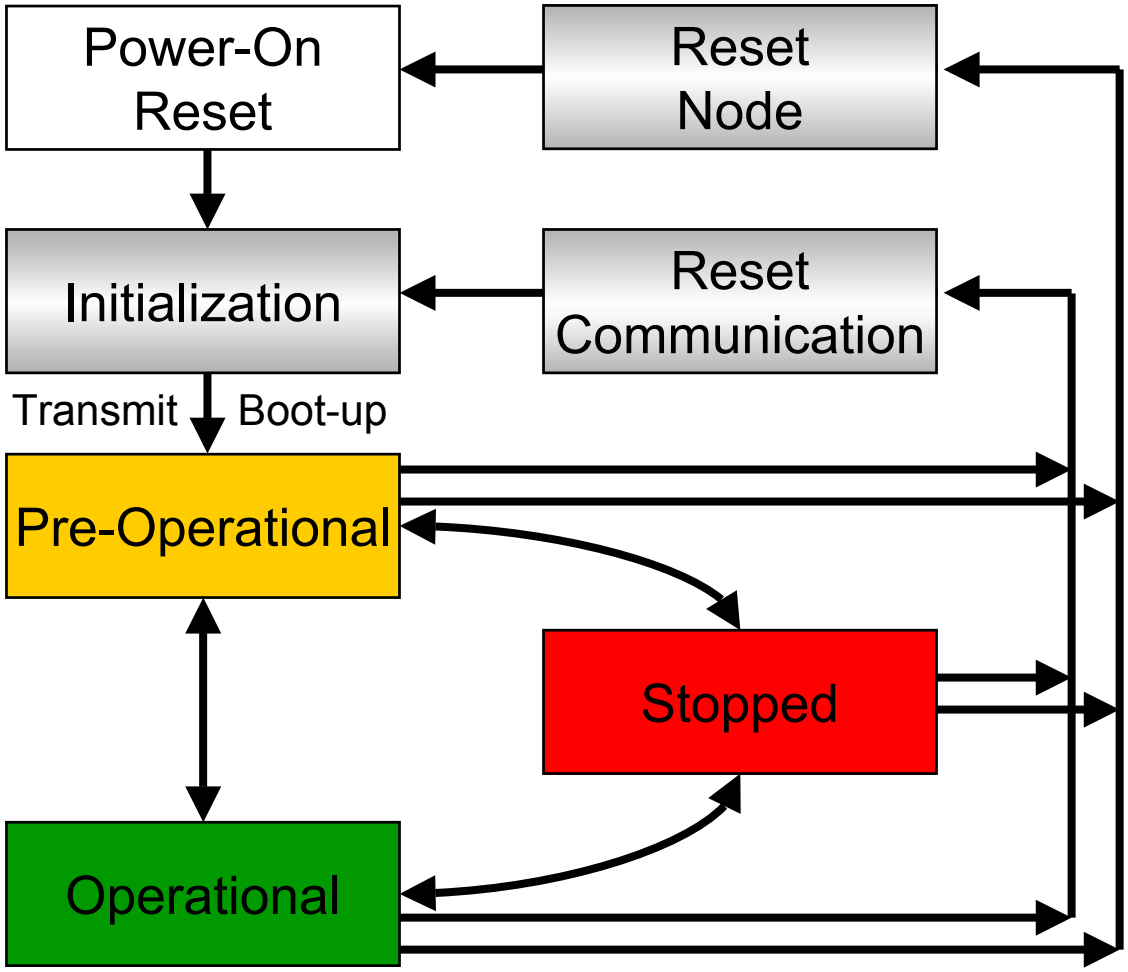
Each CANopen node MUST have a unique node ID in the range of 1 to 127

Boot-up, Heartbeat, Network Management (NMT)

Boot-up message,
heartbeat production, heartbeat consumption
NMT state machine, NMT Master Message

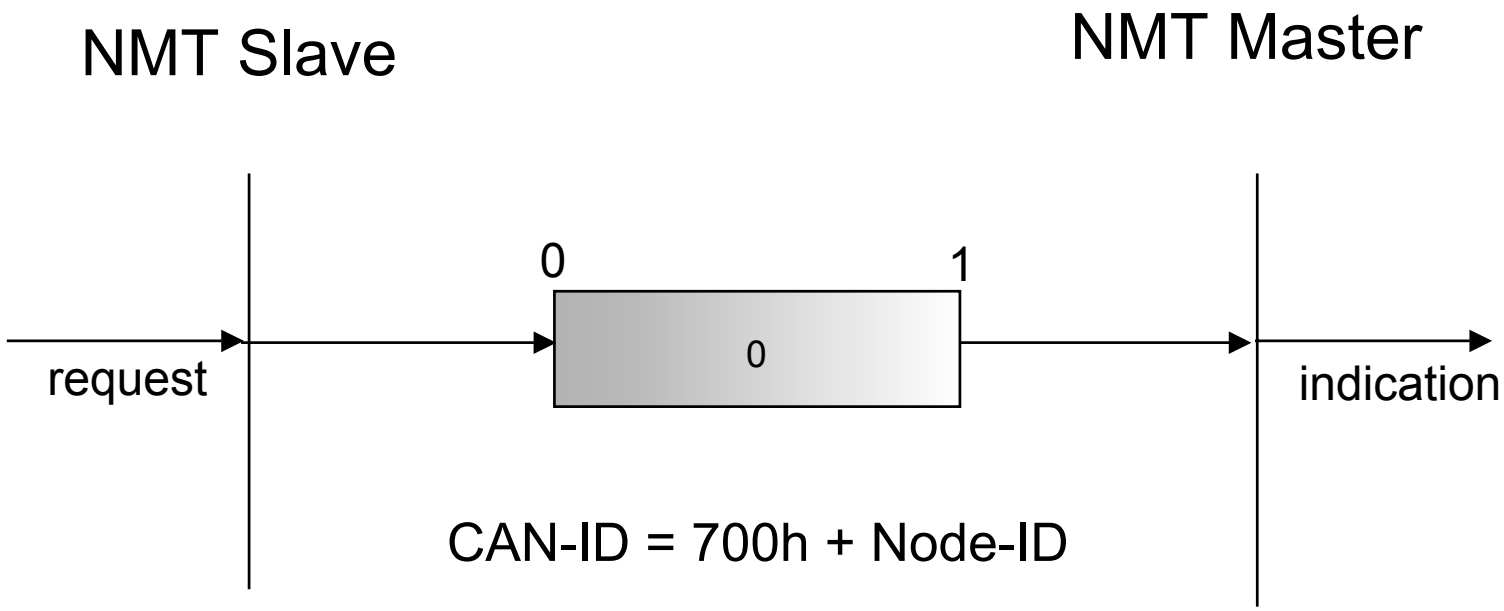
NMT Slave State Diagram

Transition from Initialization to Pre-Operational happens automatically at the end of the initialization

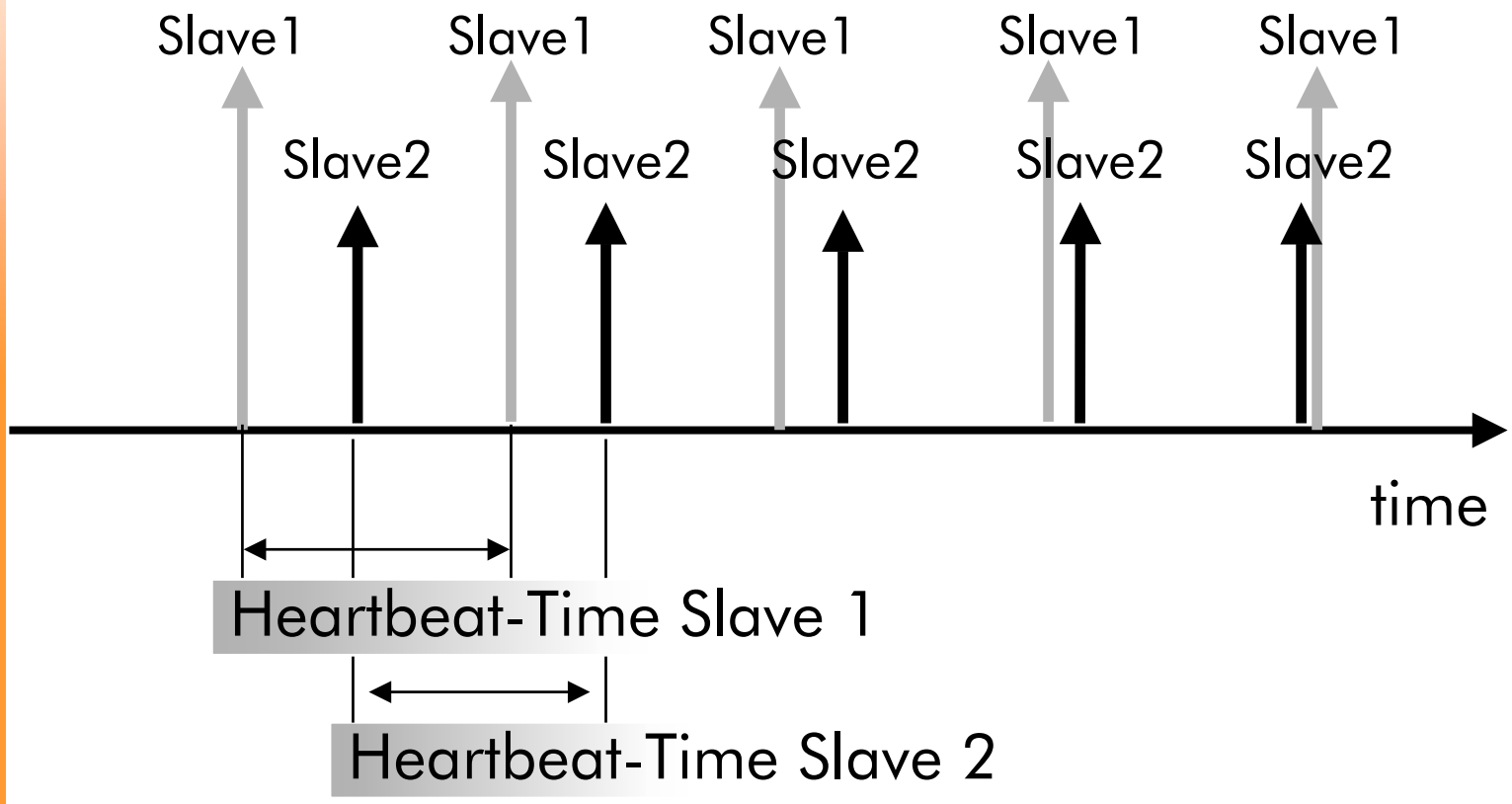


NMT Slave Boot-Up Protocol

- When going from state “Initializing” to “Pre-operational”, every CANopen slave node must send a boot-up message
- This informs NMT master, that the node is now available and is waiting for commands



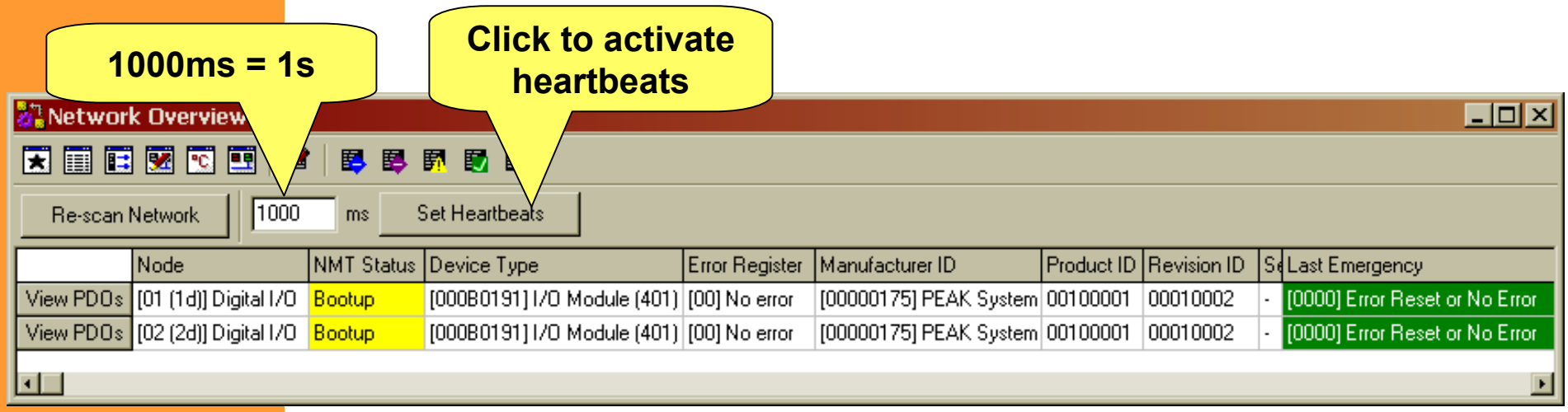
Heartbeat



CAN-ID = 700h + Node-ID

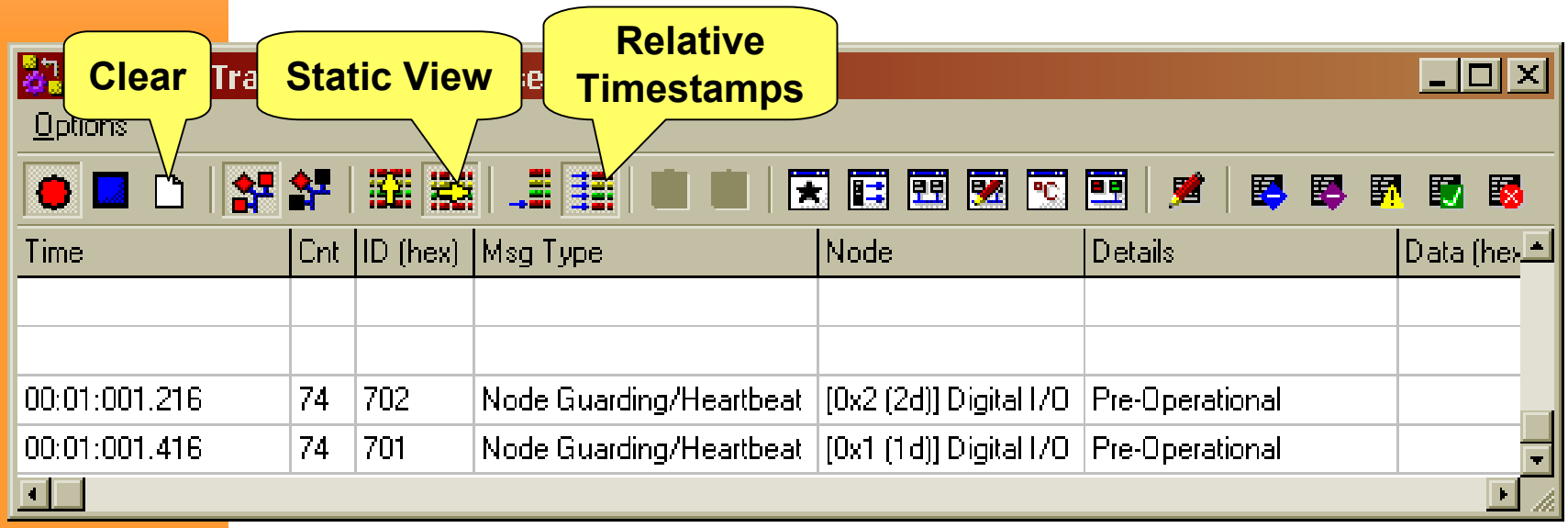
Hands-On:
Set heartbeat times

- **Open Network Overview window**
  - From the main menu, select View – Trace
- **CANopen Magic now actively scans for nodes connected to the network**
  - Nodes ‘1’ and ‘2’ are detected
  - Their identification is read and displayed
- **Set Heartbeats for all nodes to 1s**



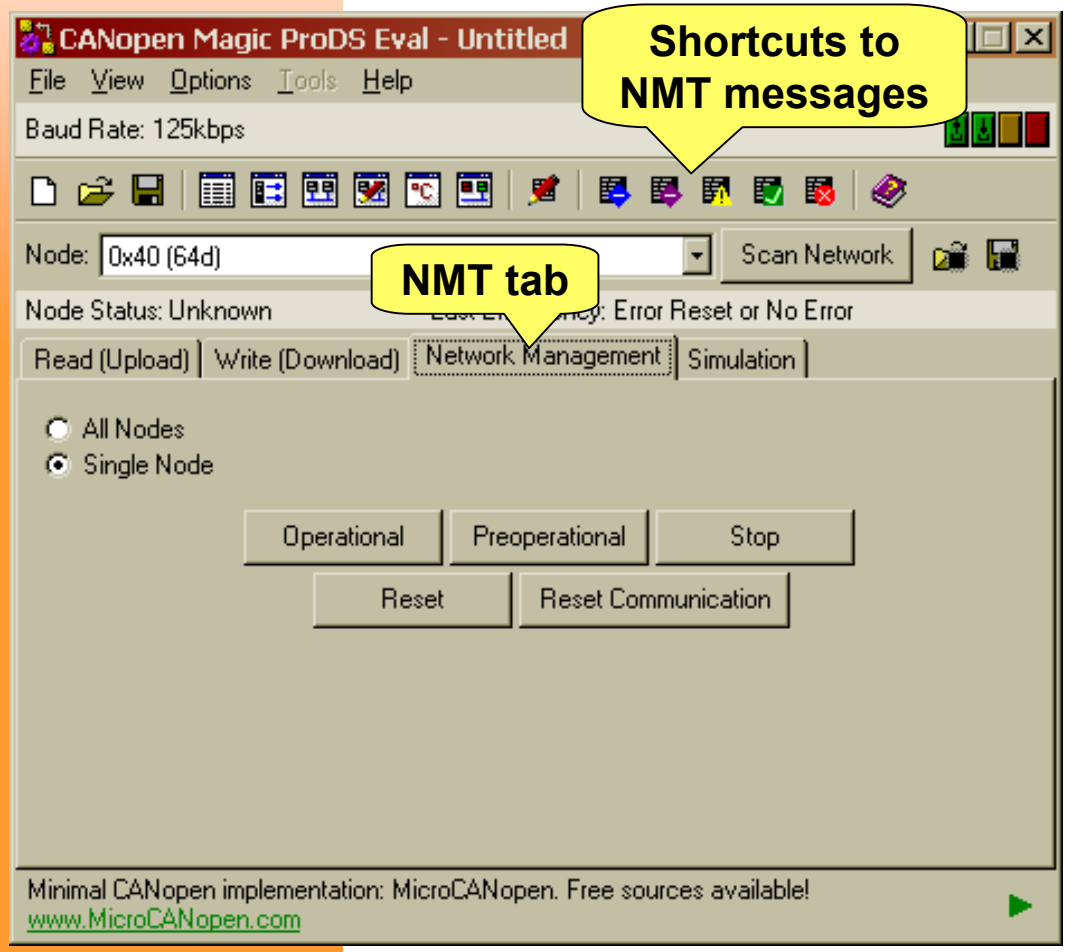
Hands-On:
Use Trace to verify heartbeat timing

- **Switch to the Trace Window**
  - Clear trace
  - Enable static trace view
  - Use relative timestamps
- **Periodically transmitted heartbeats are displayed**



Hands-On:
Send NMT Master Message

- **NMT (Network Management) Master Message**
  - Switches the NMT state of individual or all nodes



- **In main window go to the Network Management tab**
- **Here NMT Master messages can be generated addressed to individual nodes (the one currently selected) or all nodes**
- **Shortcuts to transmit 'NMT to all nodes' are available in the tool bar**

Object Dictionary

Organization of data communicated,
Electronic Data Sheets (EDS),
Service Data Objects (SDO)

EMBEDDED SYSTEMS ACADEMY

The Object Dictionary Concept

- **CANopen uses an Object Dictionary with a 16-bit index and an 8-bit sub-index**
  - Similar to a look-up table
  - Length of each entry is variable
- **Different areas (index ranges) in the table are reserved for certain purposes**
  - Data types
  - Communication profile
  - Device profile (includes process data)
- **A master or configuration tool can access any value in the object dictionary of the slaves**
  - Read or write access
  - Only if slave supports the specified entry

Mandatory object dictionary entries

- **The following object dictionary entries are all entries that are mandatory**
- **They must be supported by ALL nodes in order to be compliant to CANopen**

Index	SubIdx	Type	Description
1000h	0	UNSIGNED32	Device Type Information
1001h	0	UNSIGNED8	Error Register
1017h	0	UNSIGNED16	Heartbeat Time
1018h			Identity Object
	0	UNSIGNED8	= 4 (Number of sub-index entries)
	1	UNSIGNED32	Vendor ID
	2	UNSIGNED32	Product Code
	3	UNSIGNED32	Revision Number
	4	UNSIGNED32	Serial Number

EMBEDDED SYSTEMS ACADEMY

Mandatory object dictionary entries
Heartbeat

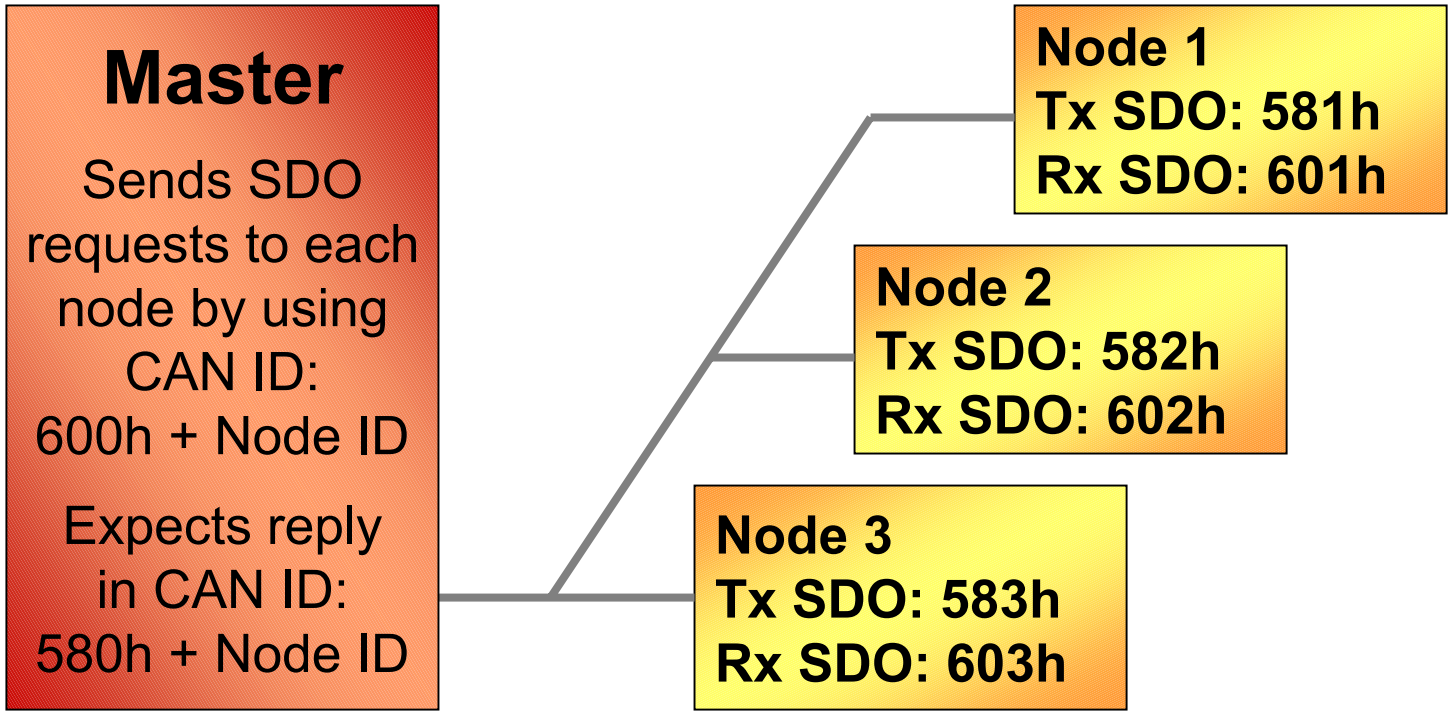
Index	SubIdx	Type	Description
1017h	0	UNSIGNED16	Heartbeat Time

- **In CANopen either “Node Guarding” or “Heartbeat” must be supported by a node to be CANopen conform**
- **As “Heartbeat” is the preferred, recommended method it is included in this listing of mandatory OD entries**
- **The heartbeat time is specified in milliseconds**
- **When this parameter is unequal zero, the node sends a heart beat message every “Heartbeat Time” milliseconds**

CANopen
SDO - Service Data Object Communication

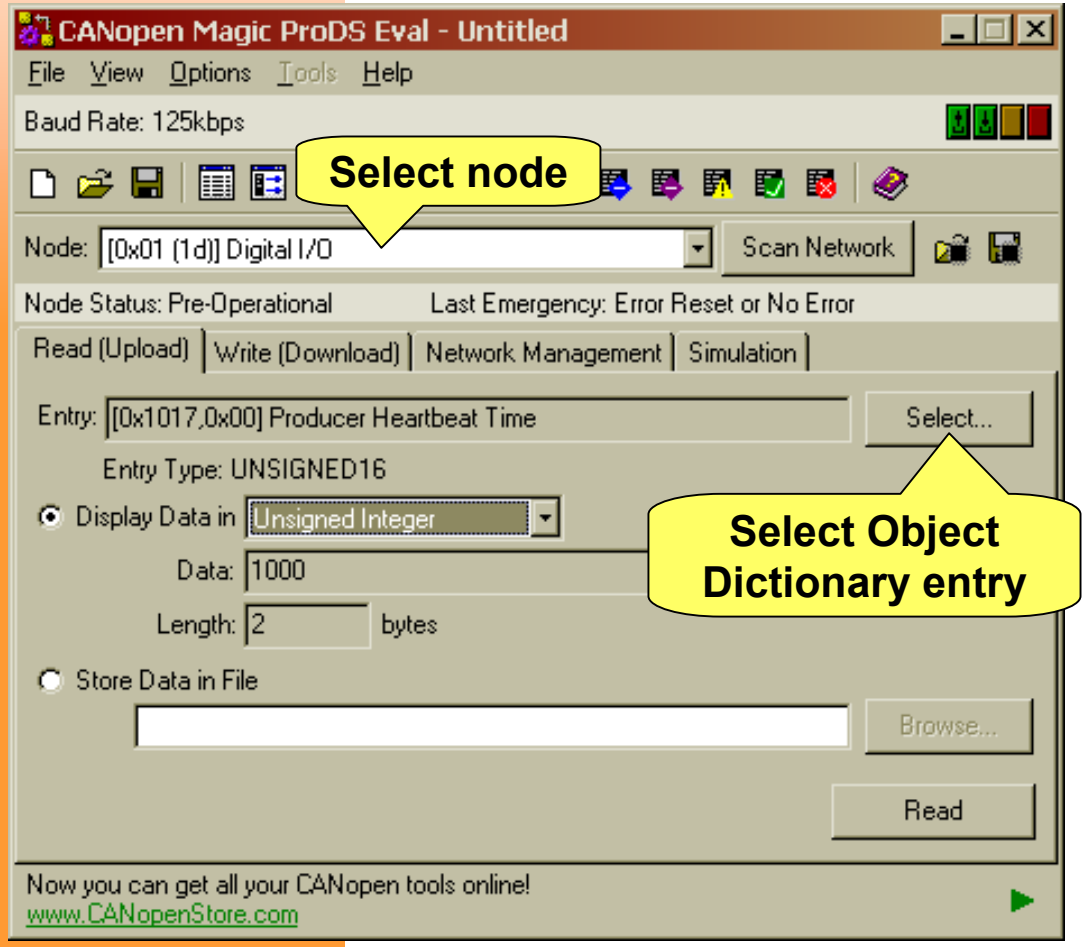
- **Used for Point-To-Point communication between a configuration tool or master/manager and the nodes**
- **Allows complete access to all entries in the Object Dictionary of a node**
  - Includes all process data
- **Confirmed communication mode**
  - Each request gets a response
- **Supports transfer of long data blocks such as upload or download of code blocks**
  - Up to 7 bytes per message, every message confirmed by receiver
  - Special “block transfer mode” allows transmitting blocks of messages with just one confirmation

Default CAN IDs used for SDO communication



Hands-On:
Reading Object Dictionary entries

- Object Dictionary entries can be accessed from the main window, choose Read or Write Tab



1. For this example, select the Read Tab
2. Select the node whose Object Dictionary should be accessed, here ‘1’
3. Select the Object Dictionary entry that should be accessed, here [1017h,00h]
4. Click on the Read button

- The message type used for the access is a SDO: Service Data Object

EMBEDDED SYSTEMS ACADEMY

Electronic Data Sheets (EDS)

- **Electronic Data Sheets specify the exact implementation of the Object Dictionary**
  - Specifies the implemented Object Dictionary entries for a specific device
  - The format of an EDS is similar to Windows .INI files to ensure machine readability

- **In order for a master and/or configuration tool to know the exact object dictionary entries supported by a node, it needs local access to the EDS for each node.**
  - A master running on an embedded microcontroller has that information hard-coded
  - A master or configuration tool with a file system would use and interpret the “real” EDS file

In the hands-on example we assigned EDS files to the nodes when we added them to the list of nodes

Process Data Objects (PDO) Communication Parameters

Message IDs used,
PDO Linking
message triggering options

PDO
Predefined Connection Set, Linking

- **Per default, each node has access to 8 PDOs, messages with process data in them**
  - 4 Transmit PDOs (TPDO)
  - 4 Receive PDOs (RPDO)
- **Per default, all transmit PDOs are received and handled ONLY by the master**
- **Per default, ONLY the master is allowed to use the CAN message IDs used for transmit PDOs**
  - So it's only the master who can send data to the nodes
- **With dynamic linking, PDOs are re-assigned**
  - Nodes can be configured to
    - Use specific CAN IDs for transmit PDOs
    - Listen to specific CAN IDs for receive PDOs

Hands-On:
Viewing the PDO Configuration

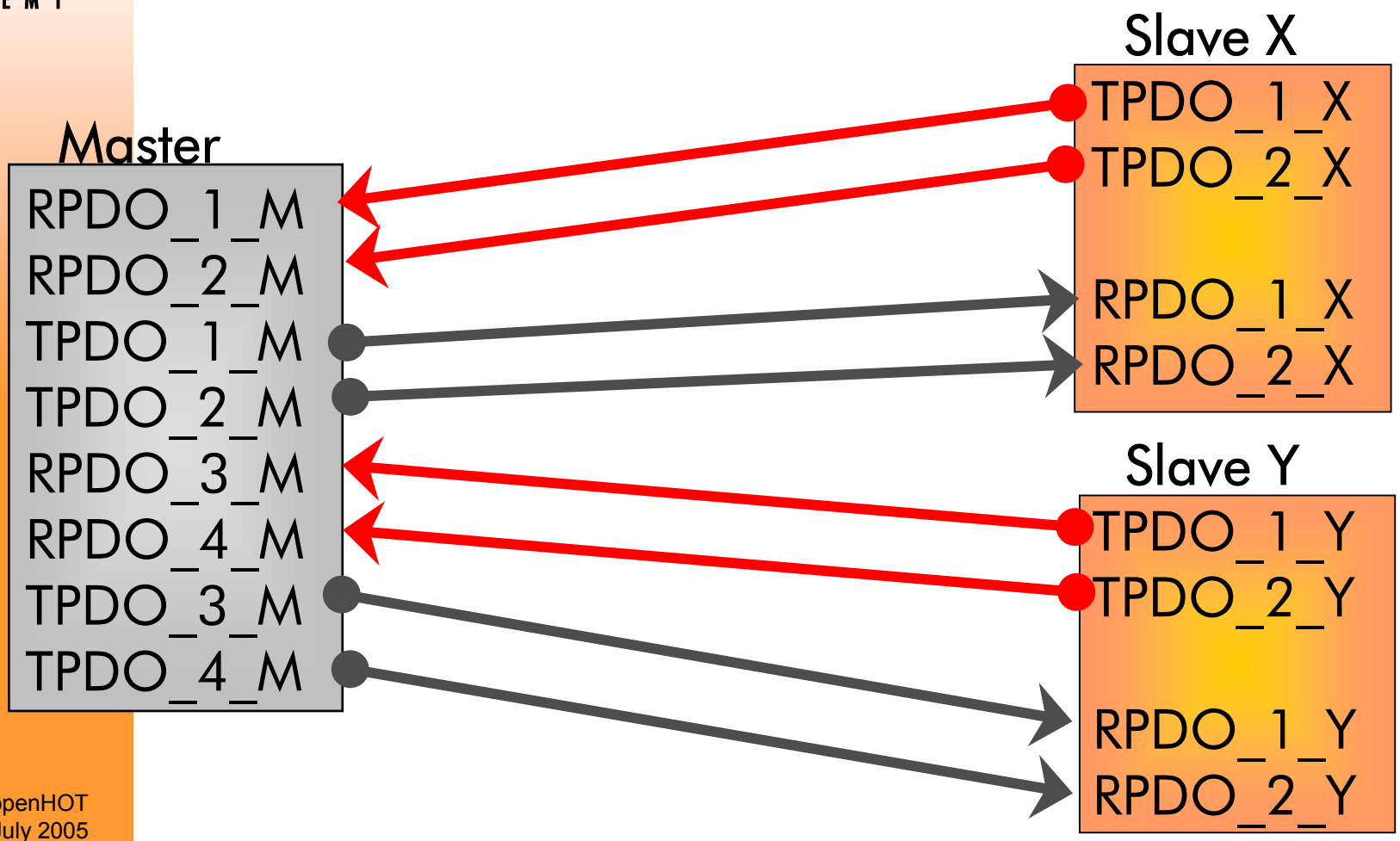
- **Switch to Network Overview window**
- **Click on 'View PDOs' button for node '1'**
- **Note the default CAN-IDs used**
  - They are from the Default Connection Set

PDO Configuration

Node: [0x01 (1d)] Digital I/O | Re-scan Node | Re-scan Network

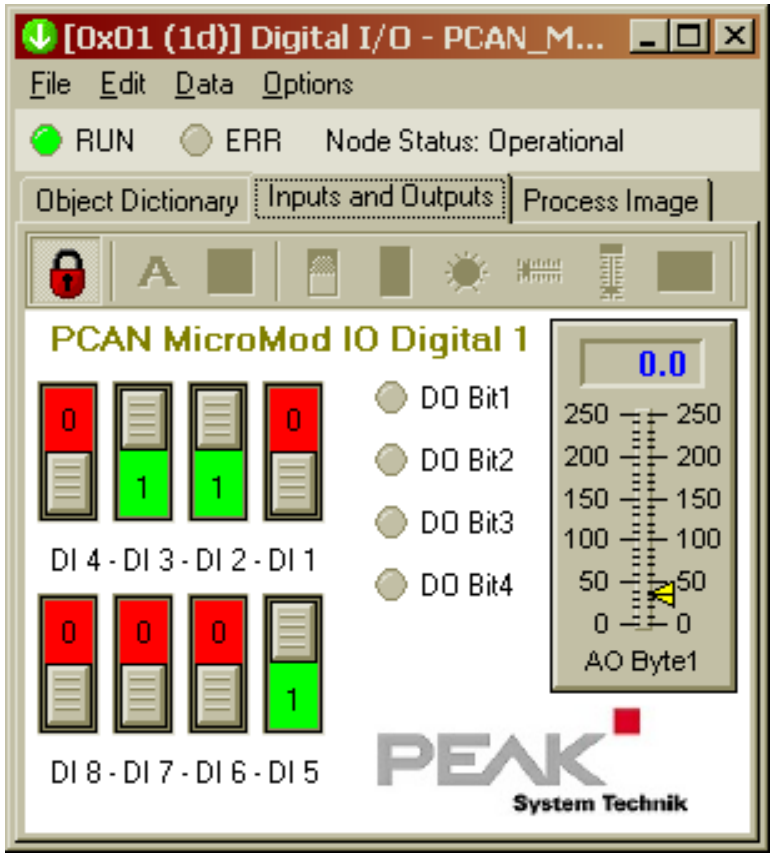
PDO	Enabled	CAN-ID	Name	RTR	Trans Type	Sync	Inhibit	Event	Map	Map 1 (hex)	Map 2 (hex)	Map 3 (
RPDO 1	☑	201			Device Profile	0	0	0	1	[6200,01,08]	0000,00,00	0000,0
RPDO 2	☑	301			Device Profile	0	0	0	1	[6411,01,10]	0000,00,00	0000,0
RPDO 3	☐	401			Device Profile	0	0	0	0	0000,00,00	0000,00,00	0000,0
RPDO 4	☐	501			Device Profile	0	0	0	0	0000,00,00	0000,00,00	0000,0
TPDO 1	☑	181		☐	Device Profile	0	0	0	1	[6000,01,08]	0000,00,00	0000,0
TPDO 2	☐	281		☐	Device Profile	0	0	0	0	0000,00,00	0000,00,00	0000,0
TPDO 3	☐	381		☐	Device Profile	0	0	0	0	0000,00,00	0000,00,00	0000,0
TPDO 4	☐	481		☐	Device Profile	0	0	0	0	0000,00,00	0000,00,00	0000,0

Predefined PDO Connections



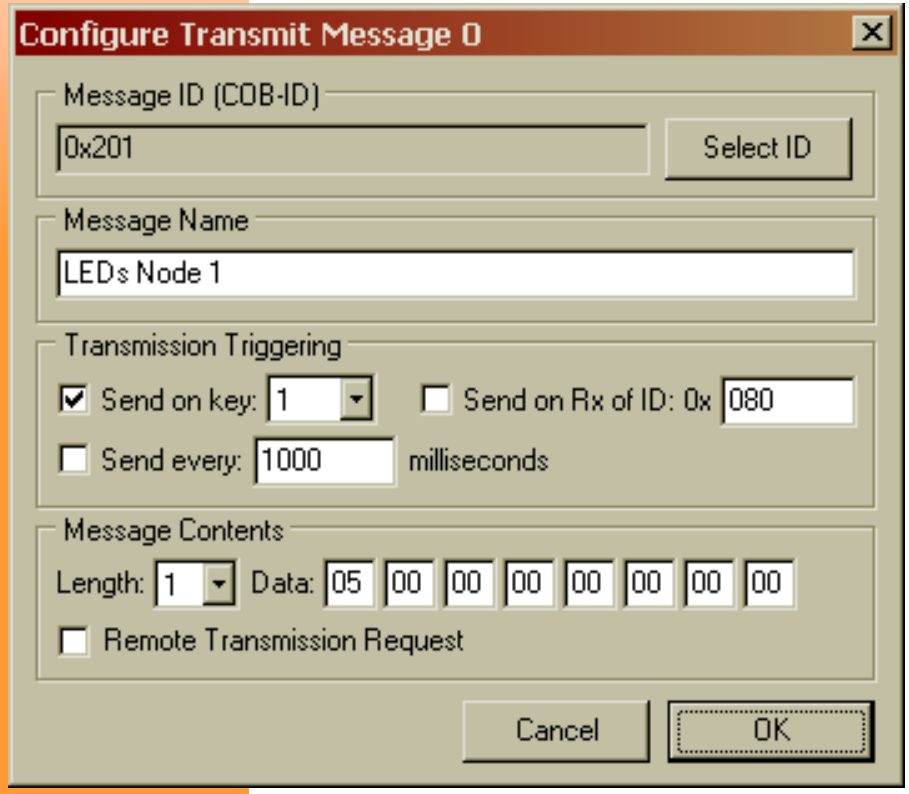
Hands-On:
Verify default TPDO linking

- **Reset all nodes**
- **View Trace window, clear contents**
- **Switch all nodes to operational**
- **Play with the digital switches on the I/O panel of node ‘1’**
- **In Trace window observe that each transition results in transmission of message ID 181h**
  - This is the default message ID for TPDO1 of node 1



Hands-On:
Verify default RPDO linking (1)

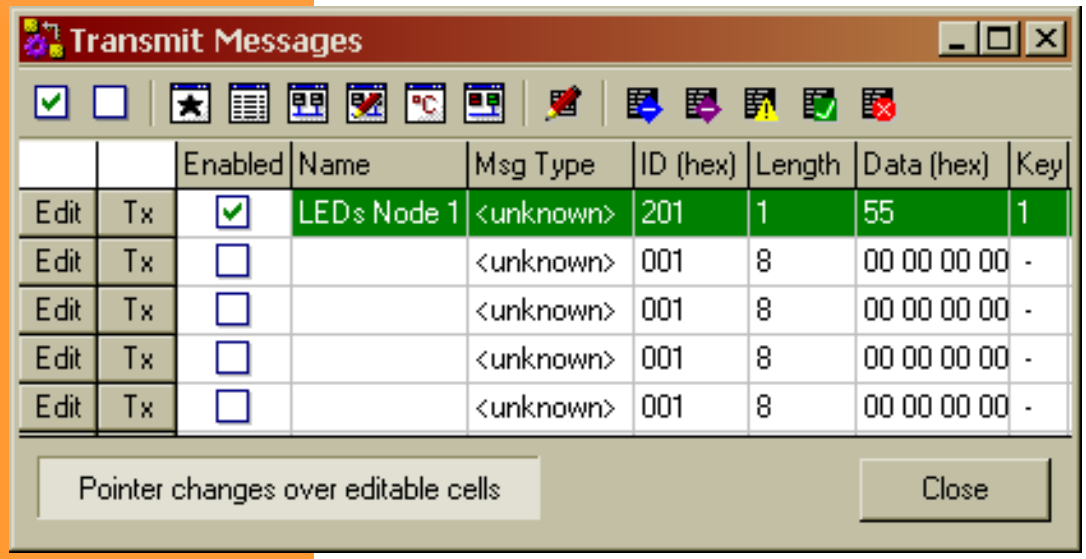
- **Open the 'Transmit List' window**
  - From the menu select View – Transmit List
- **Edit the first message by clicking on its Edit button**

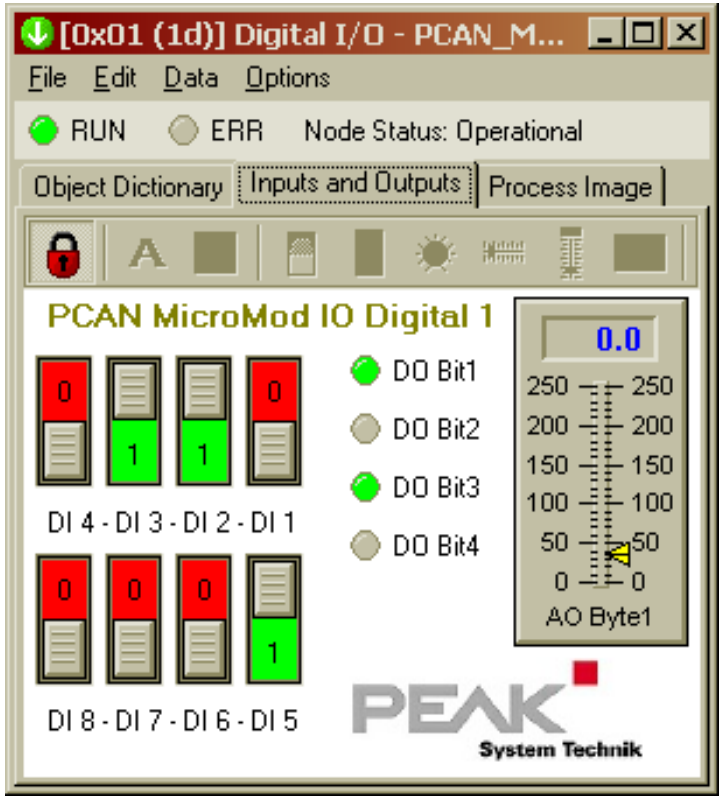


1. **Select ID for node '1', first RPDO (201h), the default message ID for RPDO1 of node 1**
2. **Enter a message name**
3. **Enter a trigger option for the message: key '1'**
4. **Define message length '1'**
5. **Enter data, for example '5'**
6. **Click 'OK'**

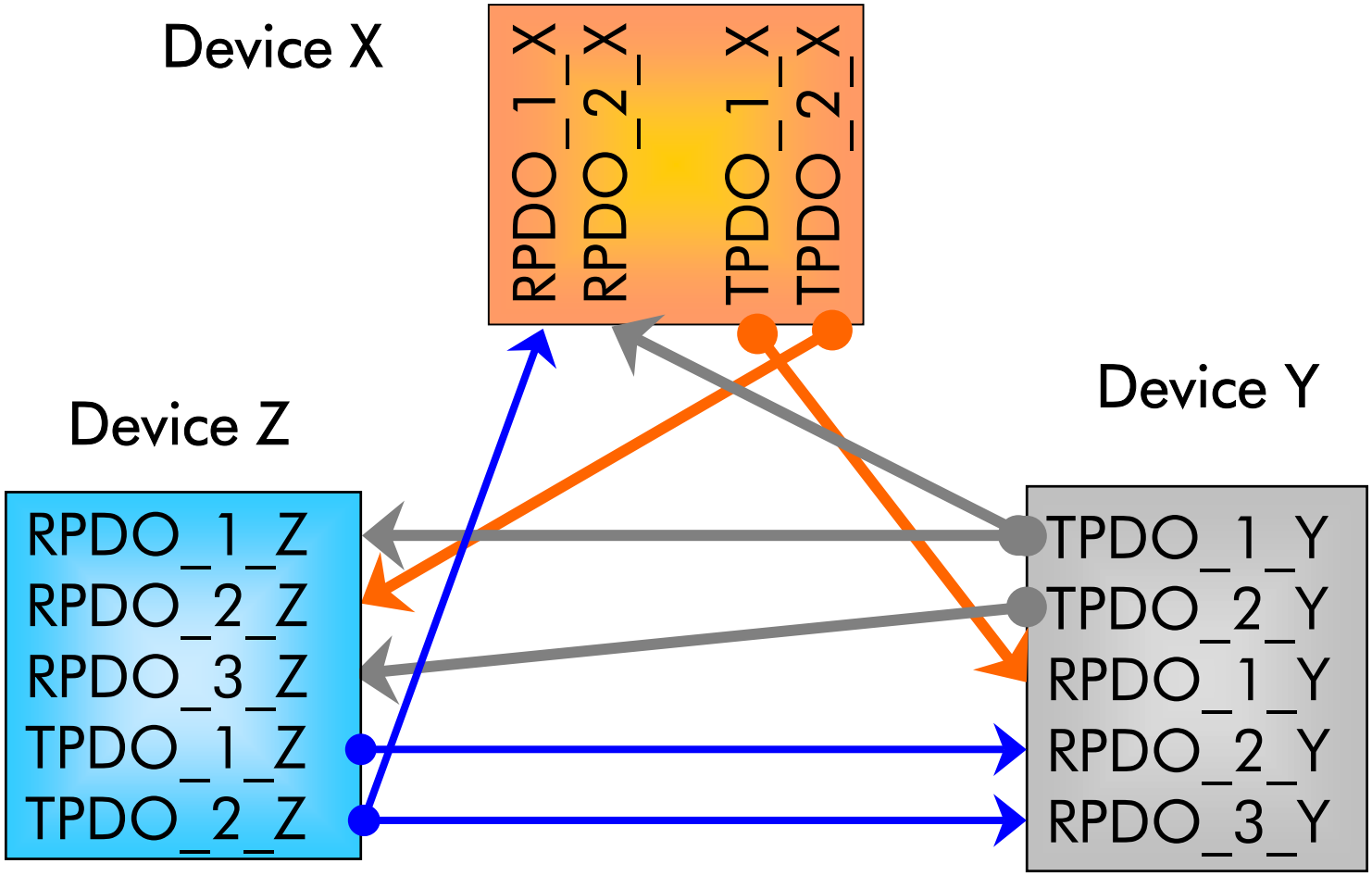
Hands-On:
Verify default RPDO linking (2)

- In ‘Transmit List’ window, ensure the message edited is enabled
- Press ‘1’ key or click on ‘Tx’ button to transmit the message
- Observe the Input / Output window of node ‘1’ now displaying the LED pattern



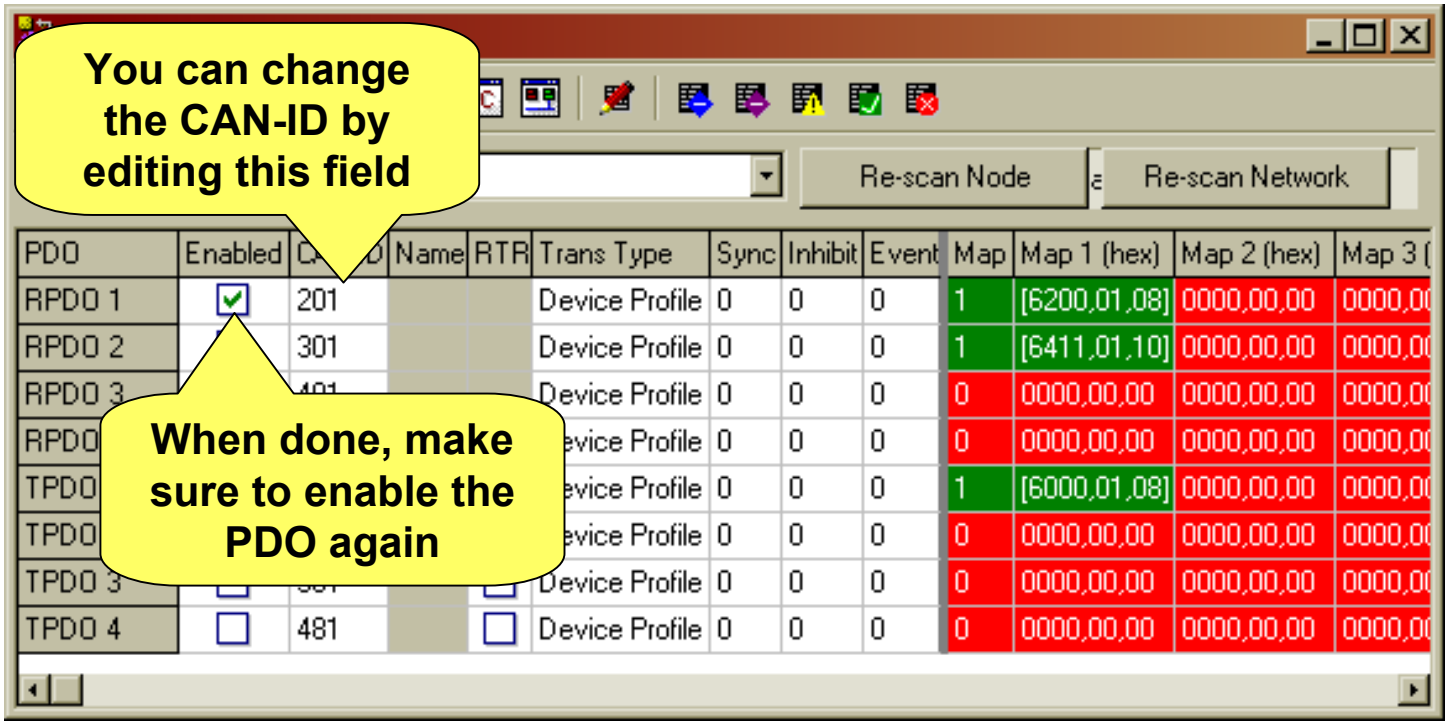


Dynamic PDO Linking
(supported by configuration tools)



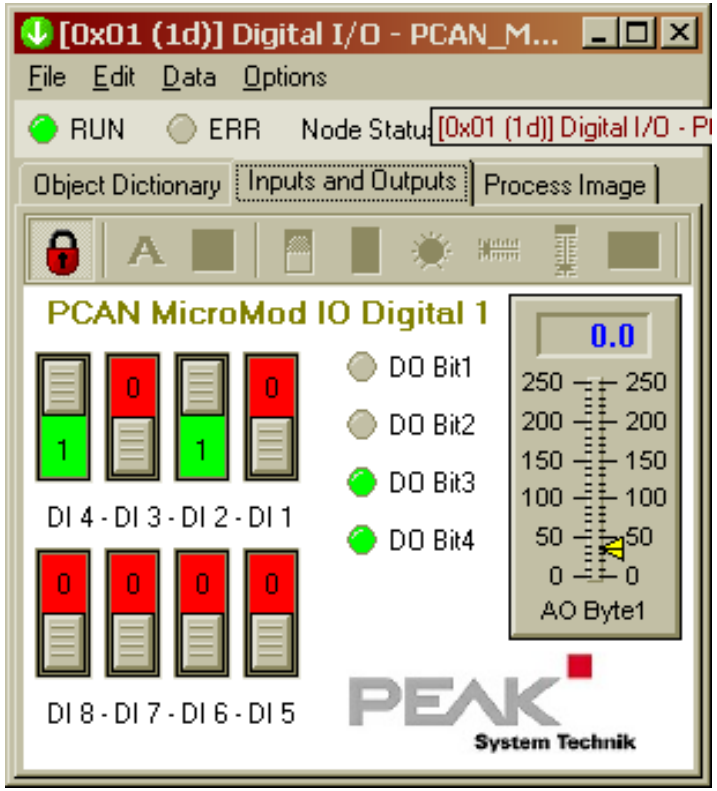
Hands-On: Changing the PDO Linking

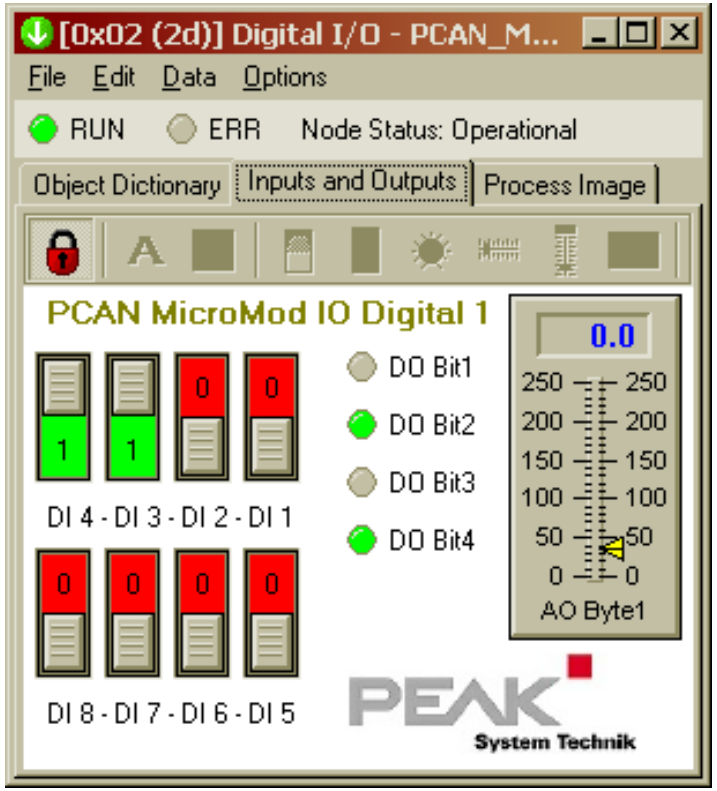
- **Switch to PDO Configuration window**
- **For node 1, change the CAN-ID for RPDO1**
  - From 201h to 182h (TPDO1 of node 2)
- **For node 2, change the CAN-ID for RPDO1**
  - From 202h to 181h (TPDO1 of node 1)



Hands-On:
Verify new PDO Linking

- **After a successful change of the PDO linking the two Input / Output display windows of nodes 1 and 2 are 'linked'**
- **Changing the input switches on one of them changes the LEDs on the other**





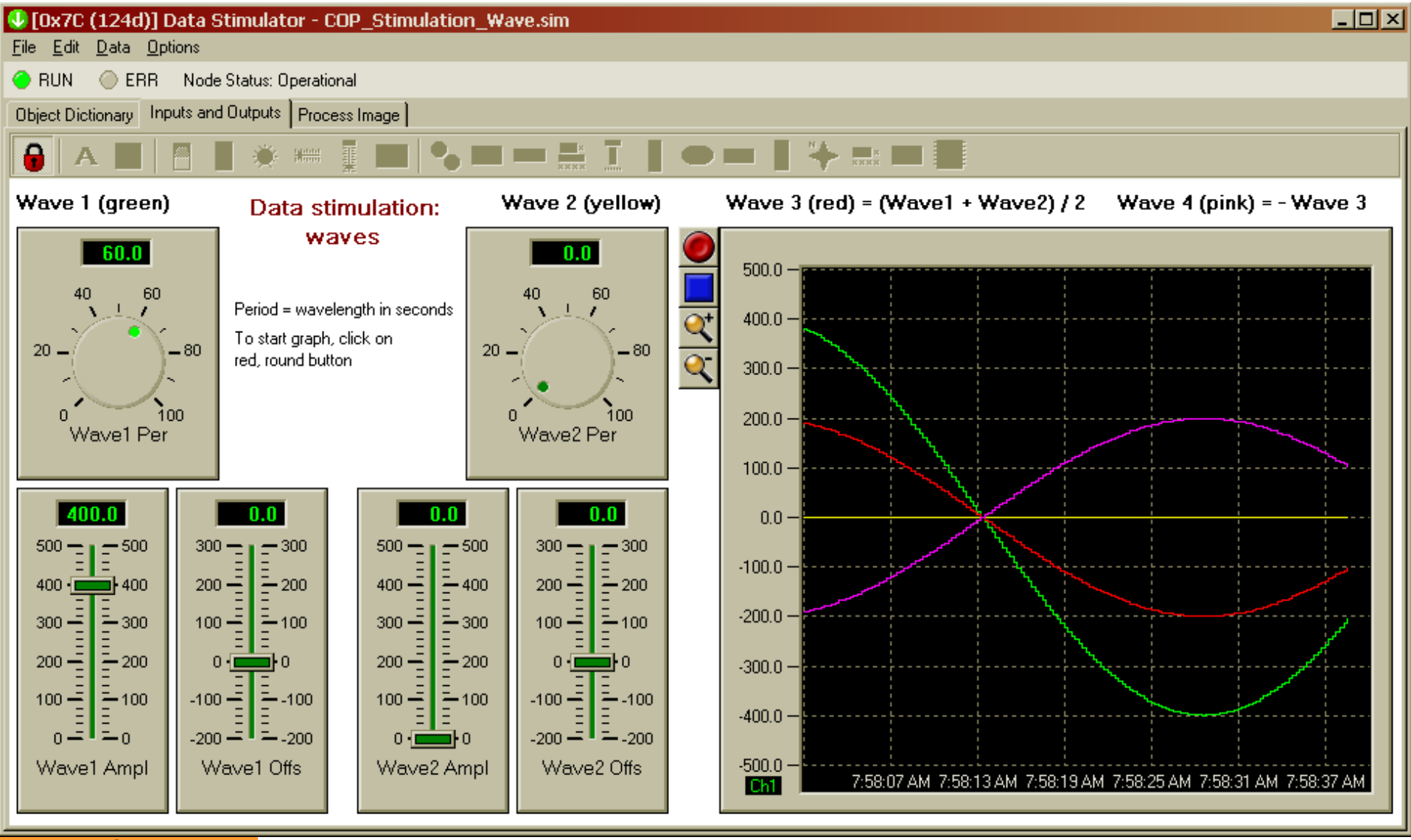
PDO – Communication Modes

- Several Communication Modes are supported:
  1. Event Driven
  2. Time Driven
  3. Synchronized polling (using SYNC message)
- An inhibit time ensures that a certain PDO does not get transmitted too often.
- The device profiles specifies which modes need to be supported by conforming nodes
- Default values can vary with manufacturers and are specified in the Electronic Data Sheets

Hands-On:
Preparation for PDO trigger examples

- **If you wish to save the current settings, go to the main menu and select File – Save Project**
- **From the menu, select File – Open Project**
  - Locate sub-directory ‘Projects/Data Stimulation’
    - Choose project ‘Data Stimulation’
- **This project only simulates one node, a node used for data stimulation**
- **In the simulation window, select ‘File – Open’**
  - Locate sub-directory ‘IO Files,
    - Choose file ‘COP_Stimulation_Wave.sim’
- **Press the red, round button to activate the graph**
- **Set ‘Wave1 Per’ to 60 (60s wavelength)**
- **Set ‘Wave1 Ampl’ to 400 (amplitude of 400)**

Example Waveform of Data Stimulation



1. Event Driven PDO

- **In event driven mode, a transmit PDO gets automatically transmitted on COS (Change Of State)**
  - If an input value changes, it is transmitted
  - To avoid frequent messages upon constant changes, an inhibit time can be specified
- **This mode minimizes bandwidth usage, as messages only get transmitted if an input changes**
- **CAUTION:**
  - If a receiving node misses the first message (maybe because it came into operational mode later), it does not know what the current value is until next COS – and that could potentially take “forever”
  - It’s hard to make a reliable prediction for the response time / real-time behavior of COS systems
  - If a PDO contains data from several channels, the inhibit time effects ALL channels

Hands-On:
Event Driven TPDO

- ❑ **Open the Trace window**
- ❑ **Set all nodes to operational**
- ❑ **Clear the Trace window**
- ❑ **Set trace display to static**
- ❑ **Use relative timestamps**

NOTE: The data stimulation nodes does not use the default TPDO CAN identifiers!

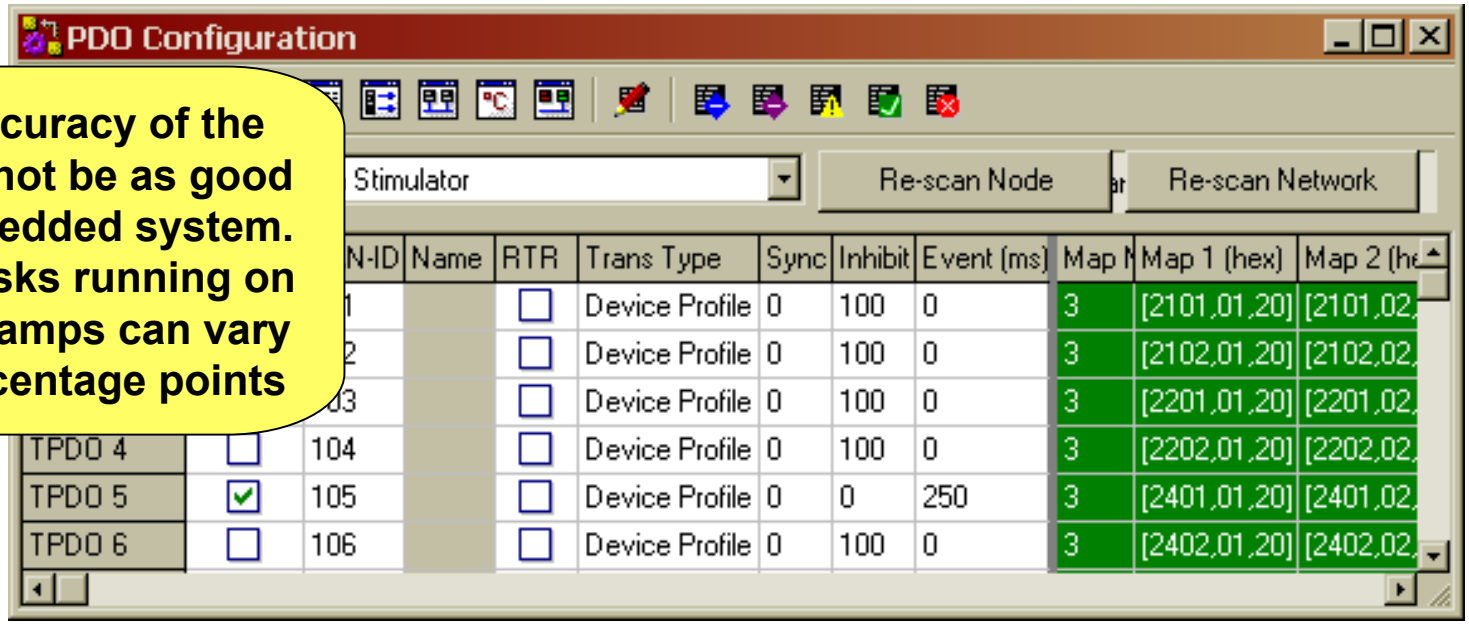
- ❑ **The three waveforms are transmitted in messages**
  - 105h, 107h, 108h, all event driven
- ❑ **Observe:**
  - In steep areas of the waveform, messages are transmitted more often
  - In flat areas of the waveform, messages are transmitted less often

2. Time Driven PDO

- **In time driven mode, a transmit PDO gets automatically transmitted on a fixed timer period basis**
  - For instance transmitting the data all x ms
- **This mode makes the bandwidth usage easily predictable and computable**
  - We can calculate exactly how much bandwidth is used by this PDO
- **CAUTION:**
  - If the data does not change often, a lot of bandwidth is wasted, as the same data is sent over and over again
  - Timers from different nodes are not synchronized
    - If 10 nodes all transmit in intervals of 50 ms, there is no way to predict the exact time slot when each message gets send

Hands-On:
Time Driven TPDO

- ❑ **Open the PDO Configuration window**
- ❑ **Select Node 7Ch, Data Stimulator**
- ❑ **For TPDO5, CAN-ID 105h, change**
  - Inhibit time to zero, Event time to 250
- ❑ **Observe in Trace Window:**
  - Message 105h is now transmitted every 250ms



		N-ID	Name	RTR	Trans Type	Sync	Inhibit	Event (ms)	Map N	Map 1 (hex)	Map 2 (he
		1		☐	Device Profile	0	100	0	3	[2101,01,20]	[2101,02,
		2		☐	Device Profile	0	100	0	3	[2102,01,20]	[2102,02,
		03		☐	Device Profile	0	100	0	3	[2201,01,20]	[2201,02,
TPDO 4	☐	104		☐	Device Profile	0	100	0	3	[2202,01,20]	[2202,02,
TPDO 5	☑	105		☐	Device Profile	0	0	250	3	[2401,01,20]	[2401,02,
TPDO 6	☐	106		☐	Device Profile	0	100	0	3	[2402,01,20]	[2402,02,

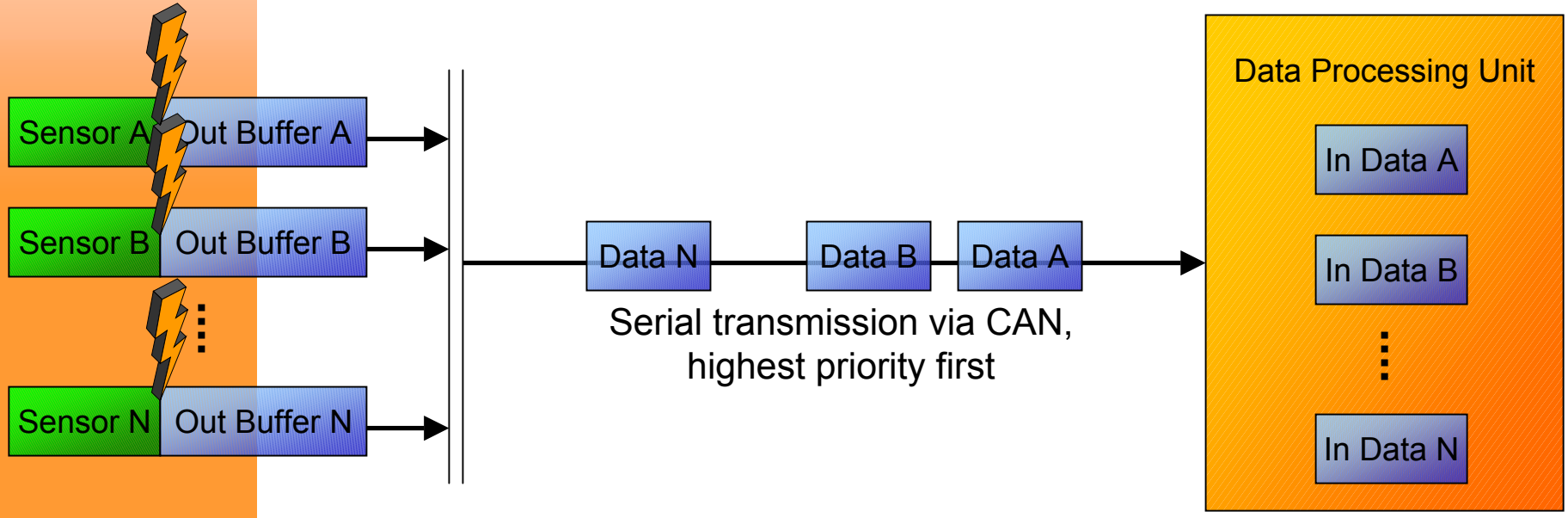
NOTE: The accuracy of the simulation cannot be as good as with an embedded system. Due to other tasks running on your PC timestamps can vary by several percentage points

3. SYNC PDO polling

- **In SYNC mode, a transmit PDO only gets transmitted AFTER a SYNC message was detected on the bus**
  - This implements a global, synchronized polling of all nodes at the same time
- **Primarily implemented for motion control systems, this feature also helps to make the overall system more predictable and computable**
  - The SYNC producer sends the SYNC message at fixed interval time slices
- **CAUTION:**
  - Each SYNC message causes bursts of data from all participating nodes

SYNC – Synchronized Communication Motion Control Systems - Sensors

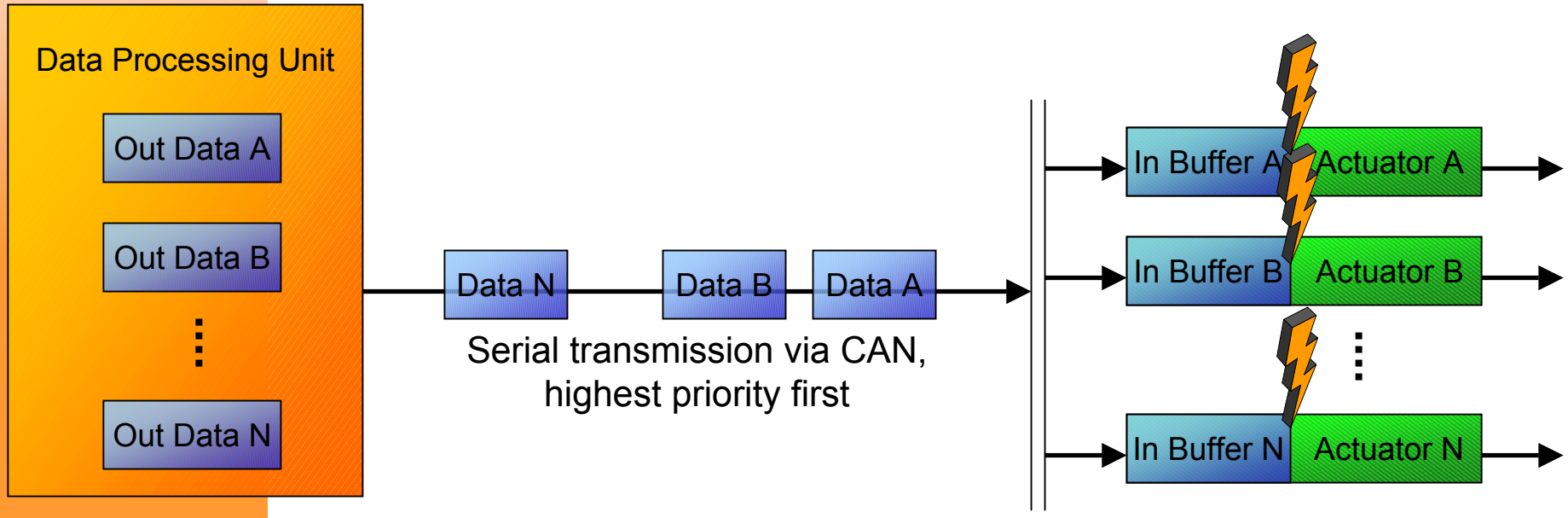
- **Avoiding jitter effects**
  - Sensors update their internal output buffer constantly
  - Sensors transmit data after receiving a high priority sync (or strobe) message



- Although data is transmitted message by message, the processing unit receives a set of data originating from the same moment in time

SYNC – Synchronized Communication Motion Control Systems - Actuators

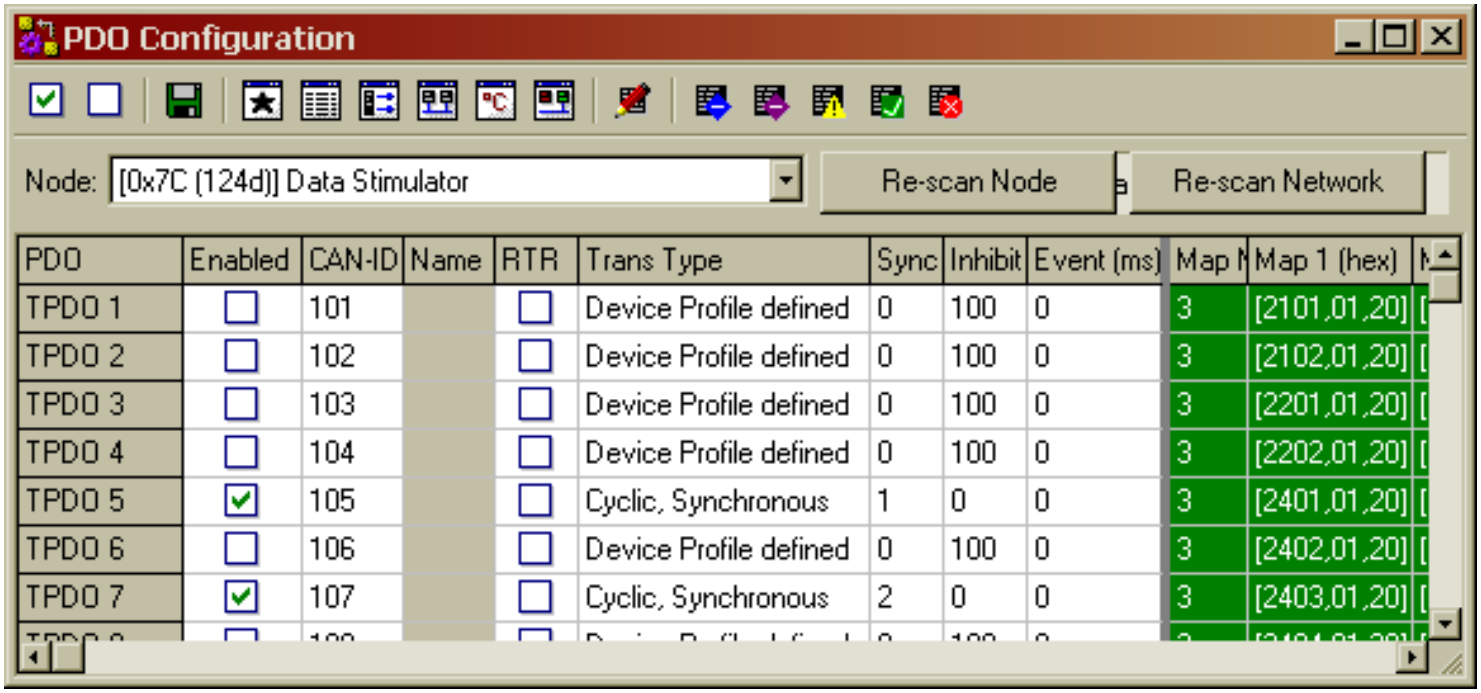
- **Avoiding jitter effects**
  - Actuators receive incoming messages to buffer



- Only when a sync / strobe message is received, do the actuators apply the previously received values

Hands-On: Synchronized TPDO (1)

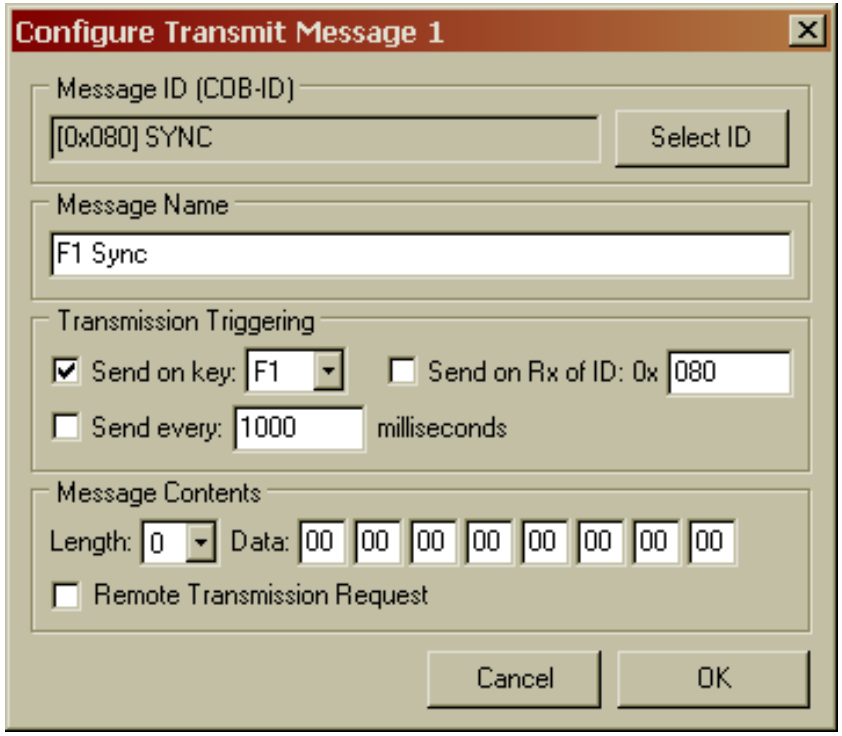
- **Switch to PDO Configuration window**
- **For TPDO5 and TPDO7, change**
  - Inhibit time to zero, Event time to zero
  - Sync value to 1 (TPDO5) and 2 (TPDO7)



PDO	Enabled	CAN-ID	Name	RTR	Trans Type	Sync	Inhibit	Event (ms)	Map N	Map 1 (hex)
TPDO 1	☐	101		☐	Device Profile defined	0	100	0	3	[2101,01,20]
TPDO 2	☐	102		☐	Device Profile defined	0	100	0	3	[2102,01,20]
TPDO 3	☐	103		☐	Device Profile defined	0	100	0	3	[2201,01,20]
TPDO 4	☐	104		☐	Device Profile defined	0	100	0	3	[2202,01,20]
TPDO 5	☑	105		☐	Cyclic, Synchronous	1	0	0	3	[2401,01,20]
TPDO 6	☐	106		☐	Device Profile defined	0	100	0	3	[2402,01,20]
TPDO 7	☑	107		☐	Cyclic, Synchronous	2	0	0	3	[2403,01,20]

Hands-On: Synchronized TPDO (2)

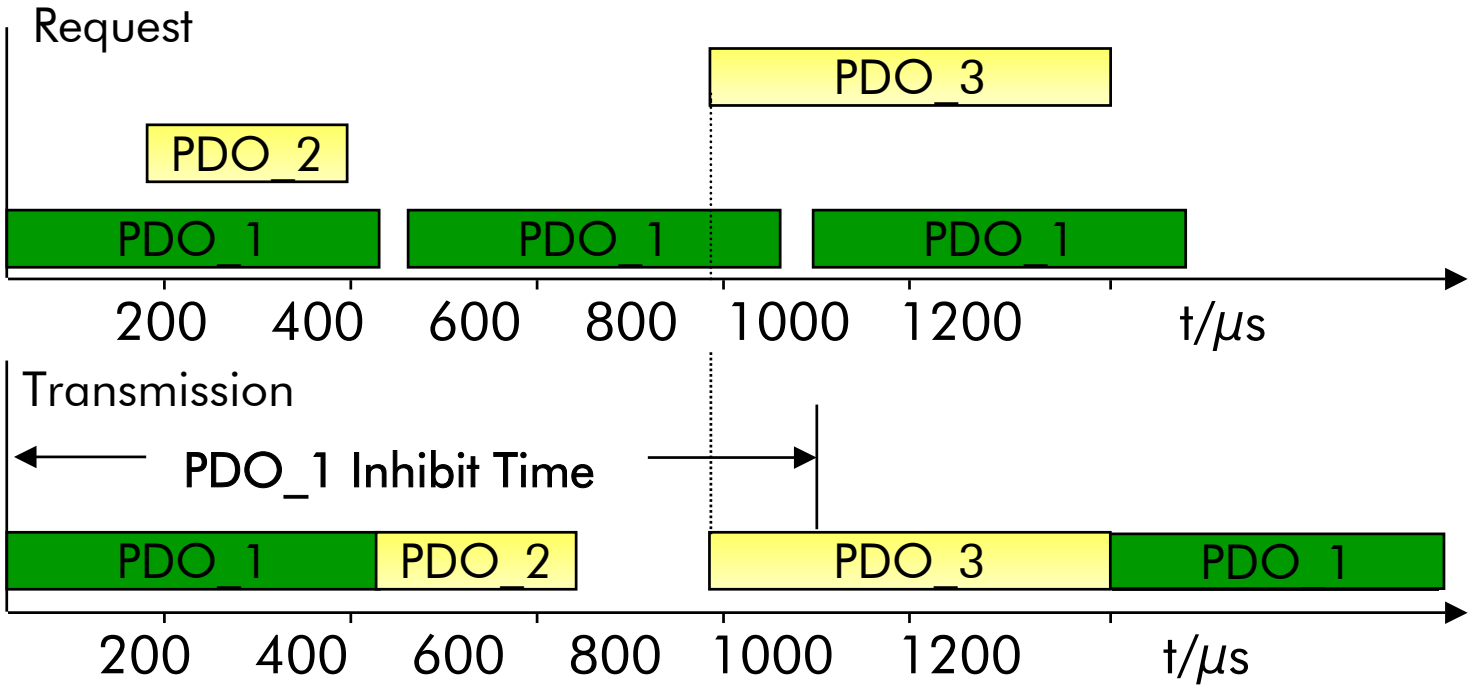
- **Open the 'Transmit List' window**
  - In main window menu, select 'View – Transmit List'
- **Edit a message, for example second row**
  1. Select CAN ID '80h'
  2. Send on key 'F1'
  3. Length: '0'
  4. Click 'OK'
- **In 'Transmit Messages' window, enable the message**
- **Switch back to Trace**



Observe: with each 'F1' key press the SYNC message is transmitted.
Message 105h comes after every SYNC.
Message 107h comes after every other SYNC.

PDO Parameter “Inhibit Time”

- **For each transmit PDO an inhibit time can be specified**
  - Implements a timeout in which a previously send PDO may not be transmitted again, until timer expires
  - Avoids high bus loads caused by the same PDO transmitted over and over



Hands-On: Inhibit Time

- **Switch to PDO Configuration window**
- **Default settings for Data Stimulator**
  - Inhibit Time of 10ms
  - NOTE:
    The Inhibit Time is specified in multiples of 100 microseconds
- **For TPDO5, change**
  - Sync: 0
  - Inhibit: 1000 (100ms)
  - Event 200 (200ms)
- **Observe in Trace window:**
  - In steep areas of the waveform, messages are transmitted more often, but never faster than 100ms
  - In flat areas of the waveform, messages are transmitted less often, but never less than 200ms

NOTE: The accuracy of the simulation cannot be as good as with an embedded system. Due to other tasks running on your PC timestamps can vary by several percentage points

Process Data Objects (PDO) Mapping Parameters

PDO contents,
which Object Dictionary entries are in a PDO

PDO – Process Data Object Mapping

- **Via entries in the Object Dictionary, the contents of PDOs can be mapped to any of the process data available via SDO read and write services**
  - Allows for a ”shortcut” access to the data without the need to start “lengthy” SDO communication
- **A PDO mapping entry can be seen as a specification of two pointers of a certain data type (determines length info for that pointer)**
  - The “OD pointer” specifies which object dictionary entry will be accessed with this map entry
  - The type info specifies how many bits are accessed (length of data entry)
  - The “PDO pointer” specifies where in the PDO message this data gets inserted
- **A default mapping is specified by device profiles**

Process Data Object Mapping: Customize PDO Contents

Index	SubIdx	Type	Description
6000h			Process data, digital inputs
	0	UNSIGNED8	= 2 (Number of sub-index entries)
	1	UNSIGNED8	8-bit digital input
	2	UNSIGNED8	8-bit digital input
6401h			Process data, analog inputs
	0	UNSIGNED8	= 2 (Number of sub-index entries)
	1	UNSIGNED16	16-bit analog input
	2	UNSIGNED16	16-bit analog input

TPDO1	D In 1	D In 2	A In 1		A In 2		Unused	
	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8

- **Multiple Object Dictionary entries can be combined into one PDO (transmitted using a CAN message)**
  - Total length cannot exceed 8 bytes
  - Unused bytes are not transmitted

Hands-On:
PDO Mapping Preparation

- **Examine available Object Dictionary entries from**
  - Documentation of a CANopen node…
  - Accessing the information from the EDS
- **Switch to the main window, Read tab**
  - Click on ‘Select’ to access the EDS browser
  - Browse for entries that you would like to map into one PDO, for example
    - [2101h,01h] Up counter 1, 32-bit value
    - [2201h,02h] Down counter A, 16-bit value
    - [2401h,02h] Wave 1, 16-bit value
  - Make sure the total of bits does not exceed 64, as that is the maximum payload of a single PDO
- **Now switch to PDO Configuration window**

Hands-On:
Change PDO Mapping

- **Select a PDO to change, for example TPDO3**
  - For Map 1, select [2101h,01h] Up counter 1
  - For Map 2, select [2201h,02h] Down counter A
  - For Map 3, select [2401h,02h] Wave 1
  - When done, make sure to set Map Num to ‘3’ and enable PDO

PDO Configuration

Node: [0x7C (124d)] Data Stimulator | Re-scan Node | Re-scan Network | ges over editable cells

PDO	Enabled	CAN	Name	RTR	Trans Type	Sync	Inhibit	Event	Map	Map 1 (hex)	Map 2 (hex)	Map 3 (hex)	Map
TPDO 1	☐	101		☐	Device Profile defined	0	100	0	3	[2101,01,20]	[2101,02,10]	[2101,03,08]	0000
TPDO 2	☐	102		☐	Device Profile defined	0	100	0	0	[2102,01,20]	[2102,02,10]	[2102,03,08]	0000
TPDO 3	☑	103		☐	Device Profile defined	0	100	0		[2101,01,20]	02,10]	[2401,02,10]	0000
TPDO 4		104		☐	Device Profile defined					02,			00

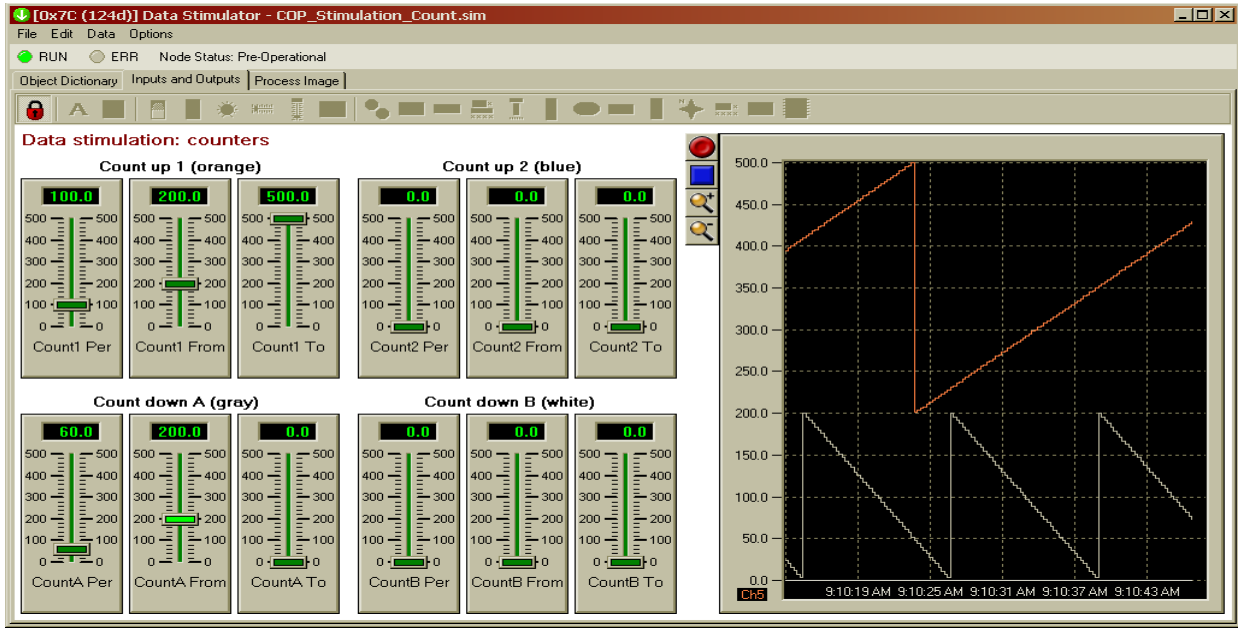
Last: ensure to enable

Change back to 3 when mapping completed

Map entries 1 to 3

Hands-On: Verify New Mapping (1)

- **Ensure the Data Stimulation node produces data for the counters and the waveforms**
  - In simulation window select menu ‘File – Open’
    - Select file ‘COP_Stimulation_Count.sim’
- **Change sliders for ‘Count Up 1’ and ‘Count Down A’**
  - Verify that there is action in the graph window
    - Click red, round button to enable graph



Hands-On: Verify New Mapping (2)

- **Switch to Trace window and observe data contents of message 103h, column data (hex)**
  - Note: CANopen uses Little Endian notation
    - Least significant byte comes first
  - First 4 bytes are the 32-bit value of the up counter
  - Next 2 bytes are the 16-bit value of the down counter
  - Last 2 bytes are the 16-bit value of the waveform

Device Configuration File

Saving and restoring node configurations

EMBEDDED SYSTEMS ACADEMY

Device Configuration Files (DCF)

- **A DCF file is derived from an EDS**
- **It contains the information about the exact settings for a particular node**
  - Instead of just specifying which baud rates are supported by a node in general (like in EDS)
  - A DCF specifies which baud rate a particular node is set to
- **Other typical information available via DCF**
  - Other communication parameters used for a node (if different from EDS)
  - Default settings for all IN and OUT parameters

Hands-On:
Generate DCF and restore DCF

- **All the current settings of the Data Stimulation node can be saved into a DCF**
- **In the main window make sure that the node you would like to generate a DCF for is currently selected**
- **From the menu select 'File – Generate DCF from Node'**
  - Specify a directory and file name for the file
- **Now reset the data stimulation node**
  - This erases all settings
- **From the menu select 'File – restore DCF to node'**
  - Pick the file just generated
  - After restoring, the data stimulation node is restored to the last saved configuration

Advanced Features

Process Data Visualization
Saving and restoring networks

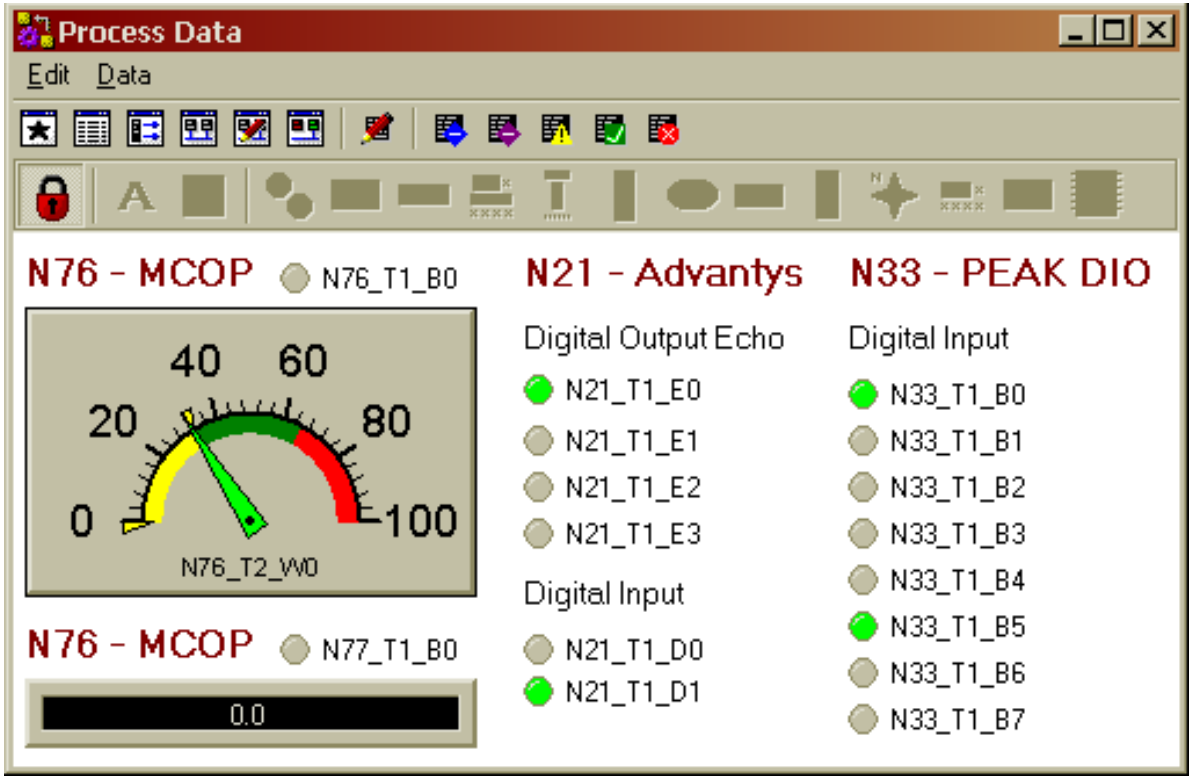
EMBEDDED SYSTEMS ACADEMY

Save and restore an entire network

- **CANopen Magic ProDS Eval can not only generate DCFs for individual nodes and restore them**
- **The configuration of ALL nodes can be stored into a single file, the Network Configuration File (NCF)**
  - This is an internal file format
  - Concatenation of multiple DCFs
- **With the NCF an entire network can be restored 'in one go'**
- **The NCF file format and restore mechanism is also supported by our CANopen PC Developers Kit**
  - Allows PC based applications to restore a network

Process Data Visualization

- The Process Data window allows the graphical display of any process data received in CAN messages
- For setup, the user must specify which data is located where in a CAN message



EMBEDDED SYSTEMS ACADEMY

Simulation of Custom Devices

- **Those versions of CANopen Magic ProDS that have access to a CAN interface also allow custom CANopen devices to be integrated into the simulation**

- **The Simulation Handler provides a CAN driver interface that can be used by CANopen source codes**
  - This has already been implemented and tested for
    - MicroCANopen (Plus)
    - CMX-CANopen
  - The free Borland-C compiler can be used

- **This allows CANopen nodes to be simulated before they are physically available**
  - Same CANopen code base can be used that will later go into the physical device

Related Web Pages

- **CANopen Magic**
  - www.CANopenMagic.com
- **Book recommendation**
  - www.CANopenBook.com
- **Embedded Systems Academy**
  - www.ESAcademy.com