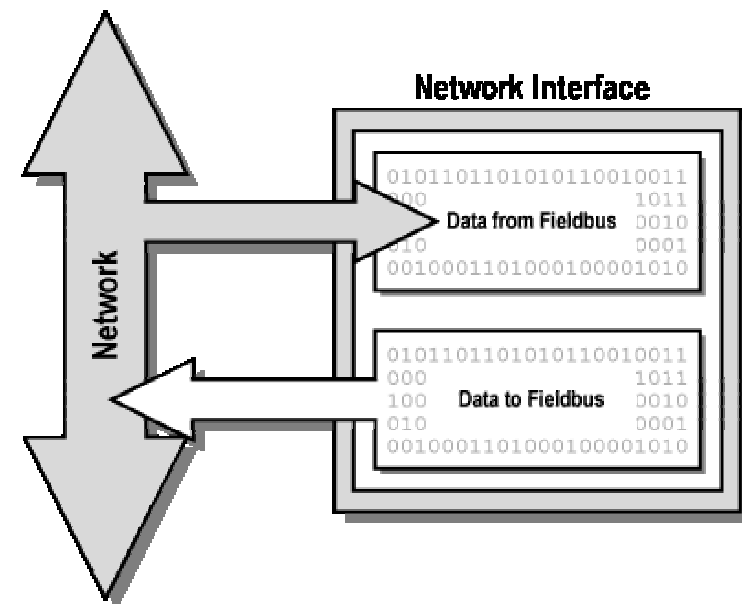




### *Anybus-CC Technical Introduction*

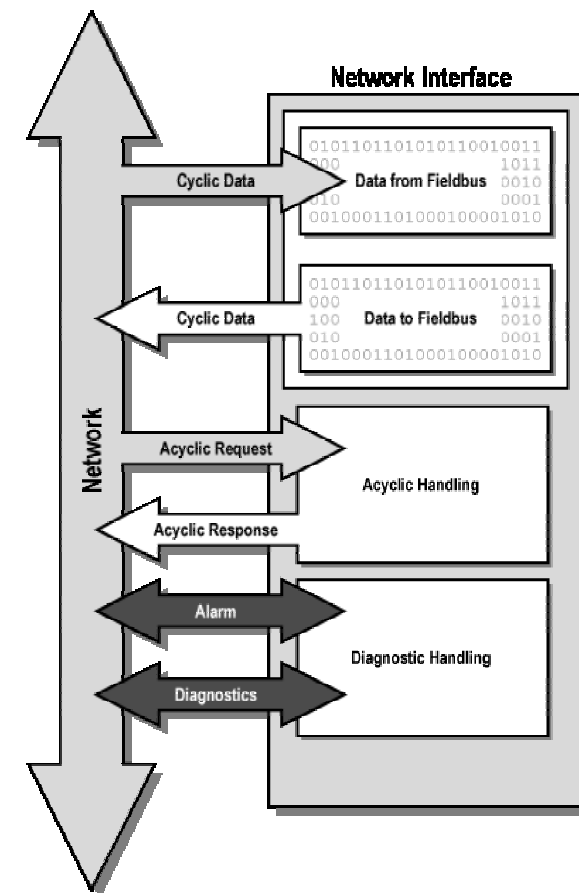
# Anybus Background

- From the beginning the functionality of an industrial network interface was primarily designed to exchange process data (I/O data).
- When Anybus was originally designed in 1994 this functionality was also what the Anybus modules offered.
- As the network specifications/functionalities developed parameter data capabilities, diagnostics and extended functionality these were added as an add-on function (message based interface)
- The main focus and demands of the original Anybus concept was to offer an open interchangeable industrial network interface. Tradeoffs in functionality (parameter support, diagnostics and events) were accepted to achieve this.



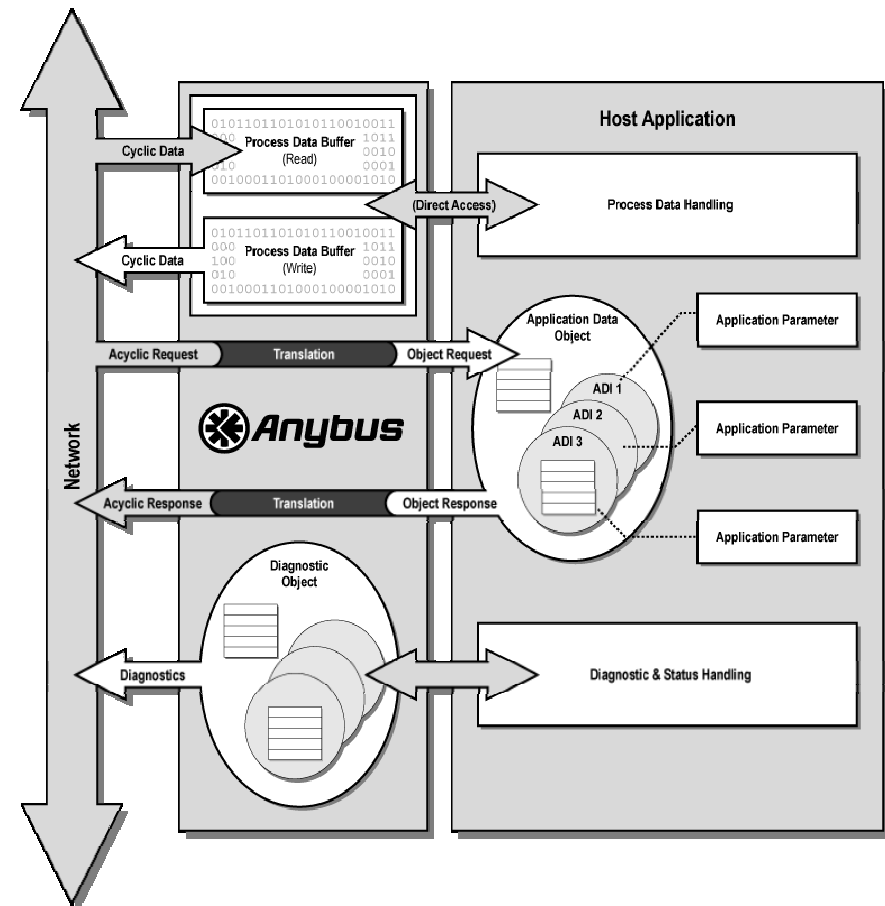
# Anybus Future

- Networking is an important technology for maintaining and controlling industrial devices
- The network interface must offer device specific features and functionality. The implementation differs fundamentally between different fieldbuses, but this is handled by dedicated objects in the Anybus module in a standardized way
- The network interface must be seen and work as a highly integrated component
- The Anybus-CC offers a standard Anybus API for
  - Process data
  - Parameter data
  - Device diagnostics
  - Network and system events



# Market Requirements on a product

- Make product's parameters and services visible to the network
  - Define product parameter list
  - Direct update via fieldbus on changes
  - Define associated parameter attributes
    - » Name, Value, Min, Max, Unit, etc.
- API functions (network independent)
  - Provide a cyclic (fast) interface mechanism for process data (I/O)
  - Provide an acyclic interface mechanism for parameter and services
  - Provide network status information to product
  - Implement correct product behavior for all states and state transitions of the network
  - Provide functions to report product exceptions and errors
    - » Trip, power-off, etc.
- When required, provide means for network specific functionality controlled by the product



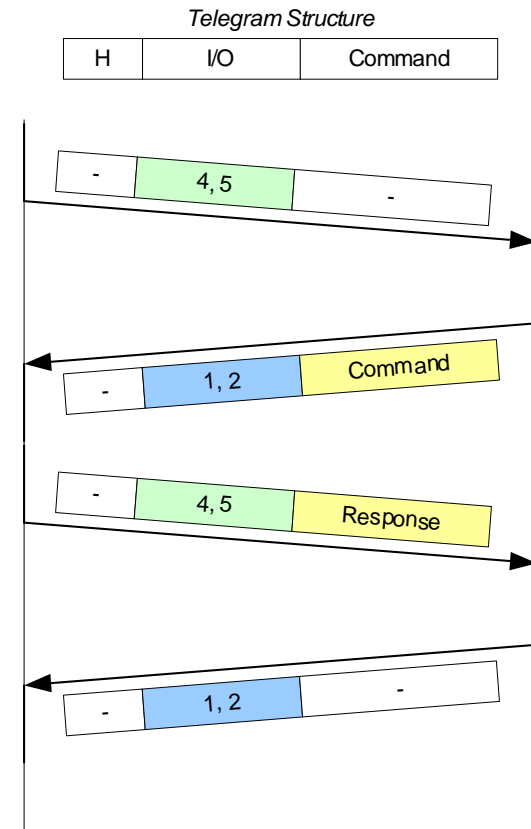
# ABCC Data Exchange Parameter data

- All parameters defined in the application product parameter list can be accessed from the network using acyclic/explicit services or directly via Network Management tools
- The Anybus-CC translates the network specific parameter access commands into a standard format
- The product performs the requested command and reports the result
  - ACK, not supported, value too big, etc
- Each parameter ADI (Application Data Instance) is described with a number of attributes

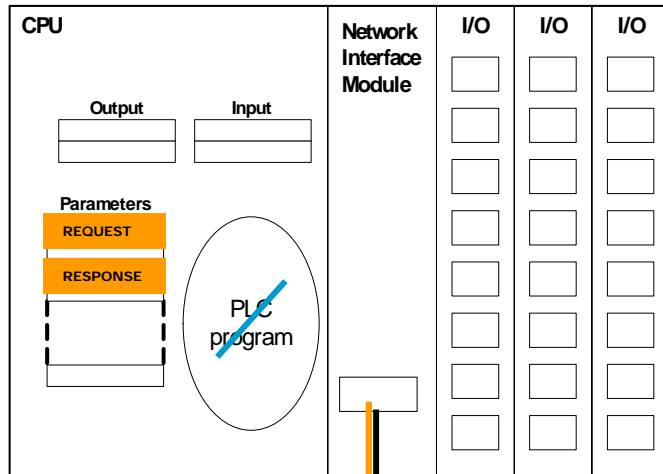
- Parameter number
- Name
- Data type
- Mapping
- Access
- Value
- Default
- Min
- Max
- Scale
- Etc.

Attribute	ADI #1 <i>Attribute value</i>	ADI #2 <i>Attribute value</i>
ID:	1	2
Name:	"SPEED 1"	" MAX TORQUE"
Data Type:	word	word
Mapping:	Process_Read	Parameter
Access:	R/W	R/W
Current Value:	500	10
Default Value:	0	0
Min Value:	0	0
Max Value:	4000	20
Scale:	x10	x1
Unit:	Hz	Nm
Etc.:	-	-

Host Interface



# ABCC Data Exchange Parameter data

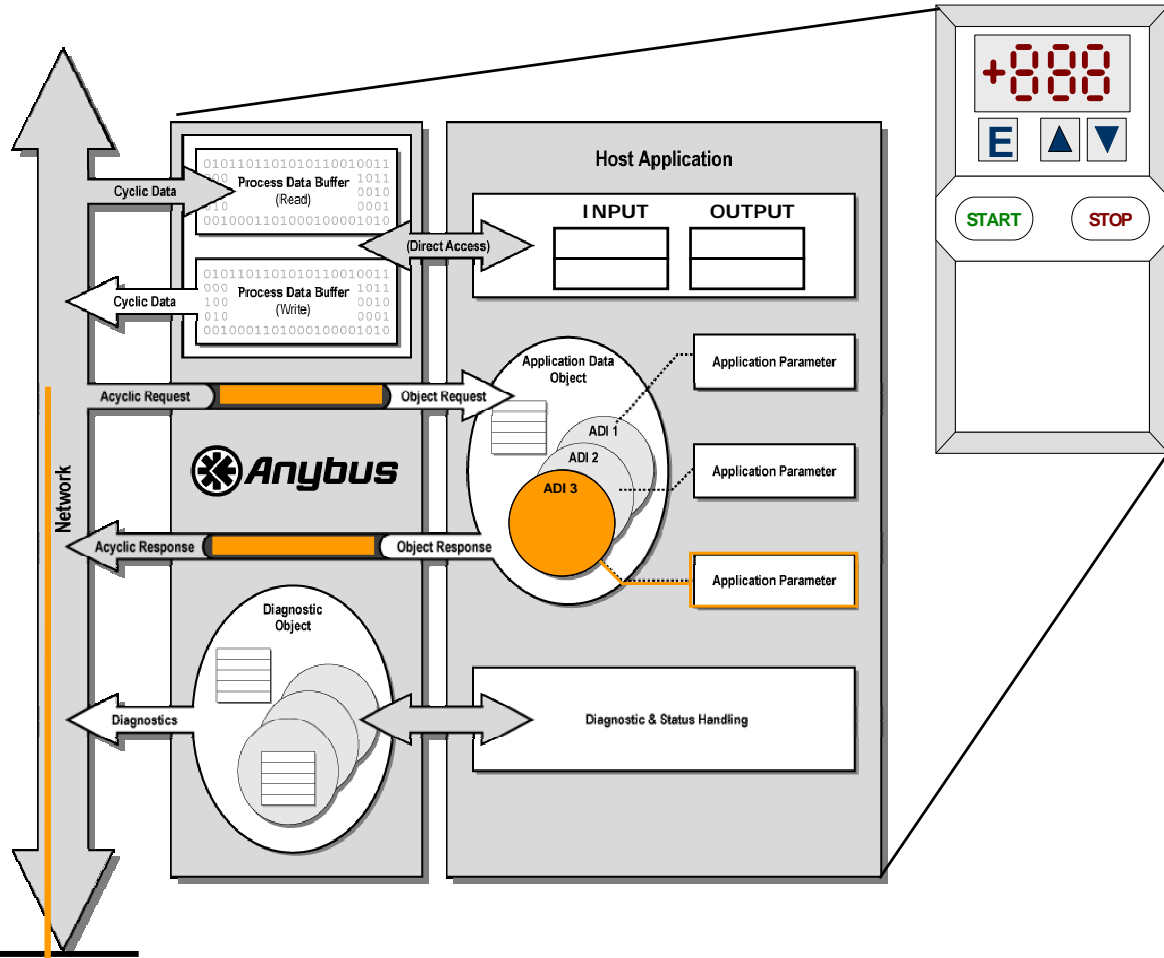


INDEX 0  
SLOT 2

INSTANCE 3  
ATTRIBUTE 5

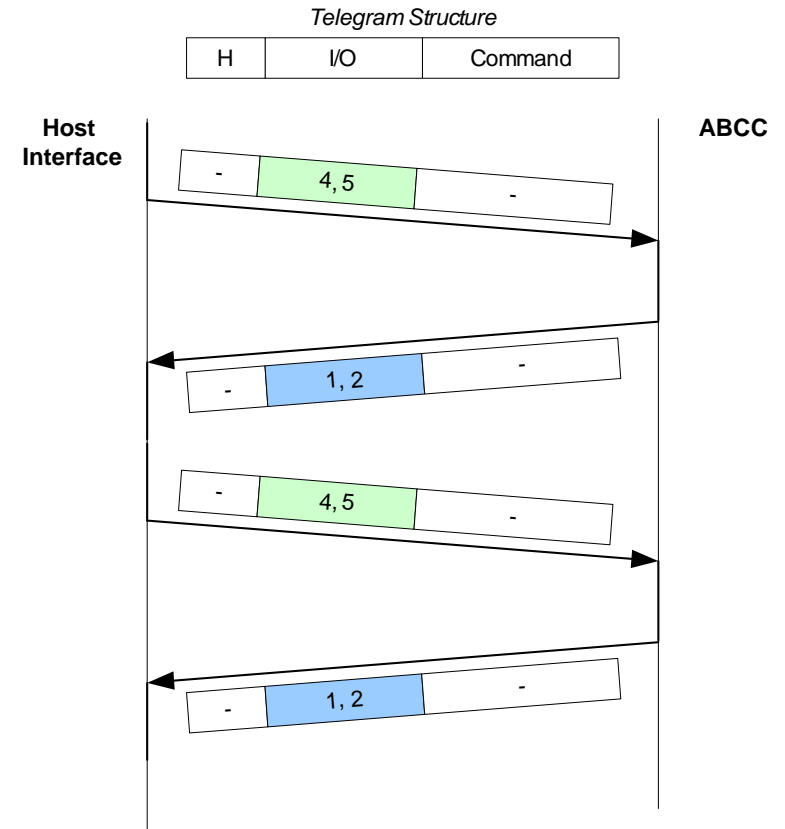
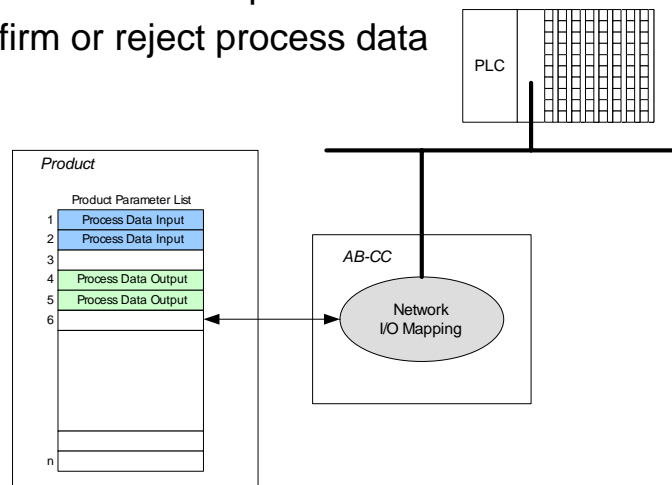
PROFI<sup>®</sup>  
BUS

DeviceNet.

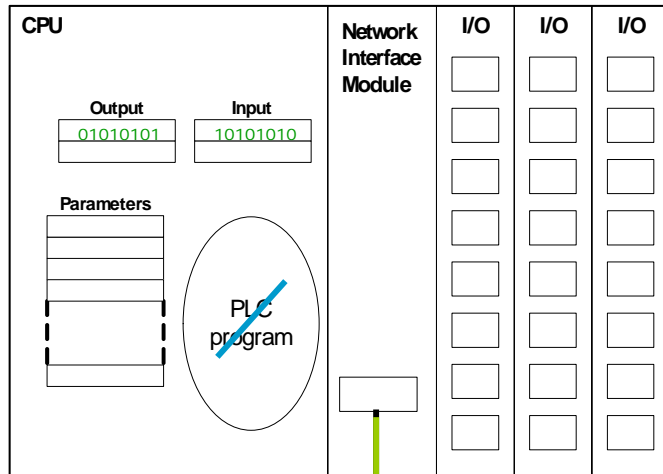


# ABCC Data Exchange Process data

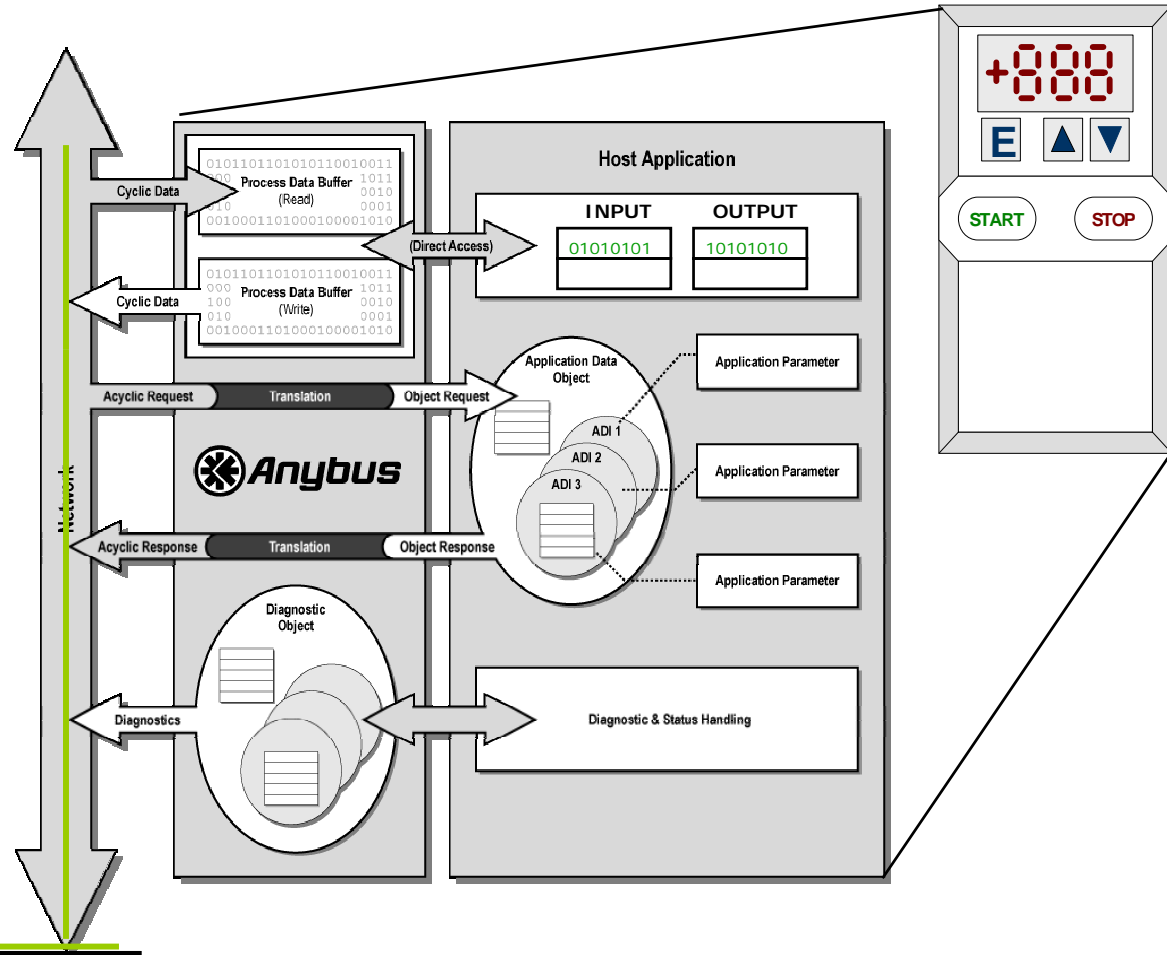
- In addition to Parameter Data, ADI's can also be assigned to exchange its data cyclically (process data).
- During initialization the application product indicates which of the defined product parameters that shall be used as process data input and output (I/O)
- The Anybus-CC will map these parameters as network process data (I/O)
- The process data parameter values are updated each cycle
- The data is unacknowledged
  - Industrial networks does not provide means to confirm or reject process data



# ABCC Data Exchange Process data

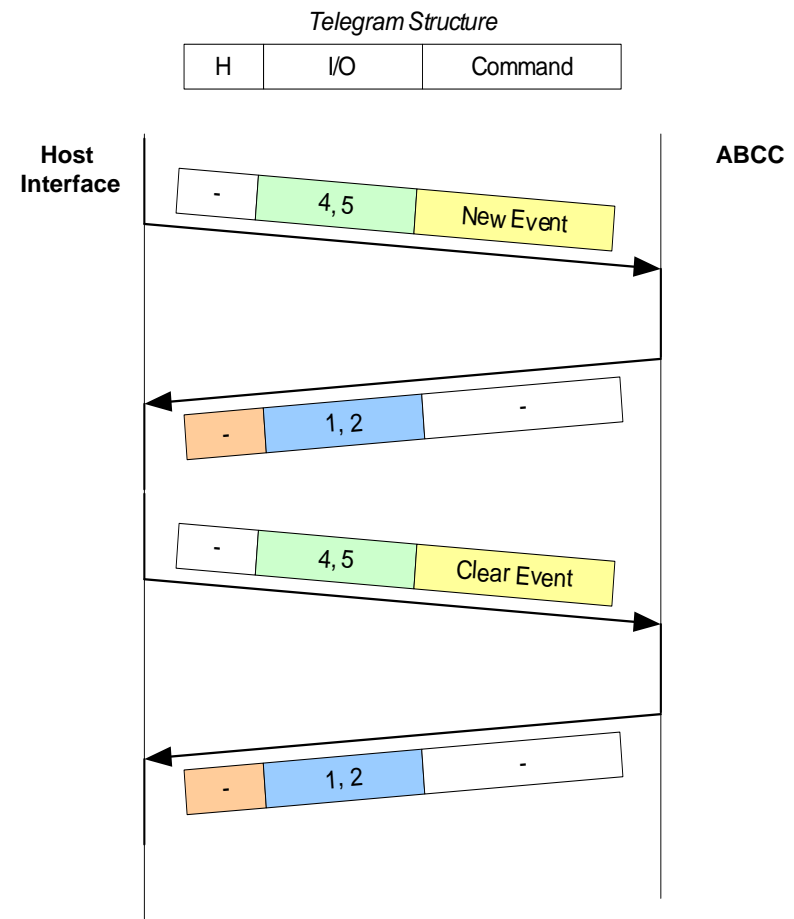


Process data is handled in the same way for all networks

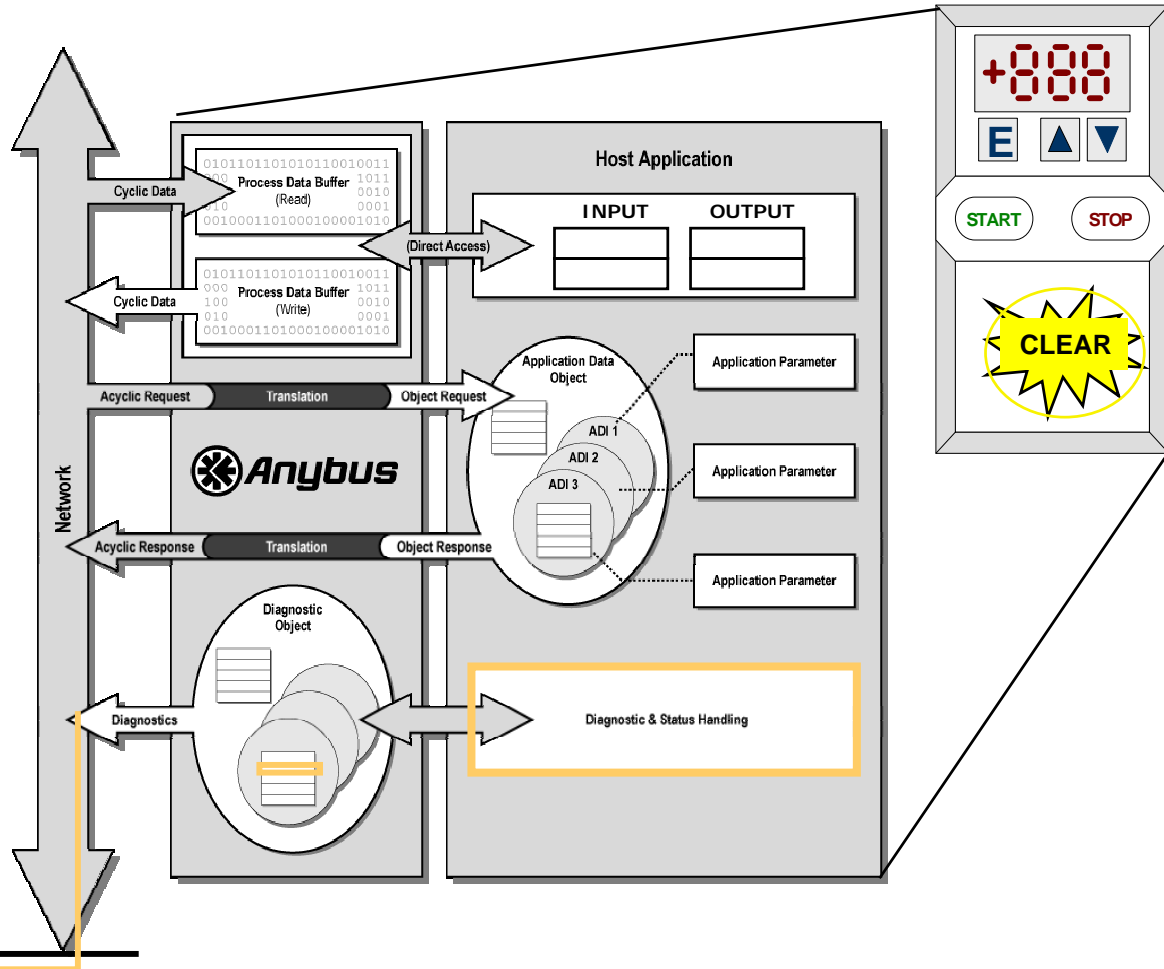
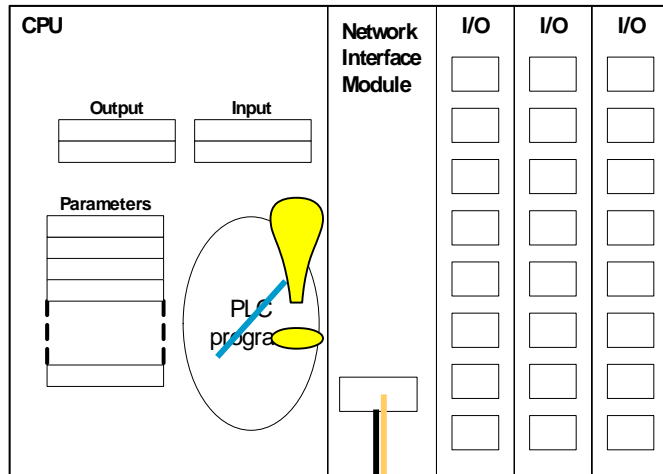


# Product Status and Diagnostics

- During operation the product needs to be able to report operational events to the network
- The Anybus-CC provides a diagnostic interface to handle operational events
  - Trip, power off, open wire, etc.
- The diagnostic function defines a number of standard events as well as product specific events
- The events are translated to the network specific format by the Anybus-CC
- The event is cleared by the Host

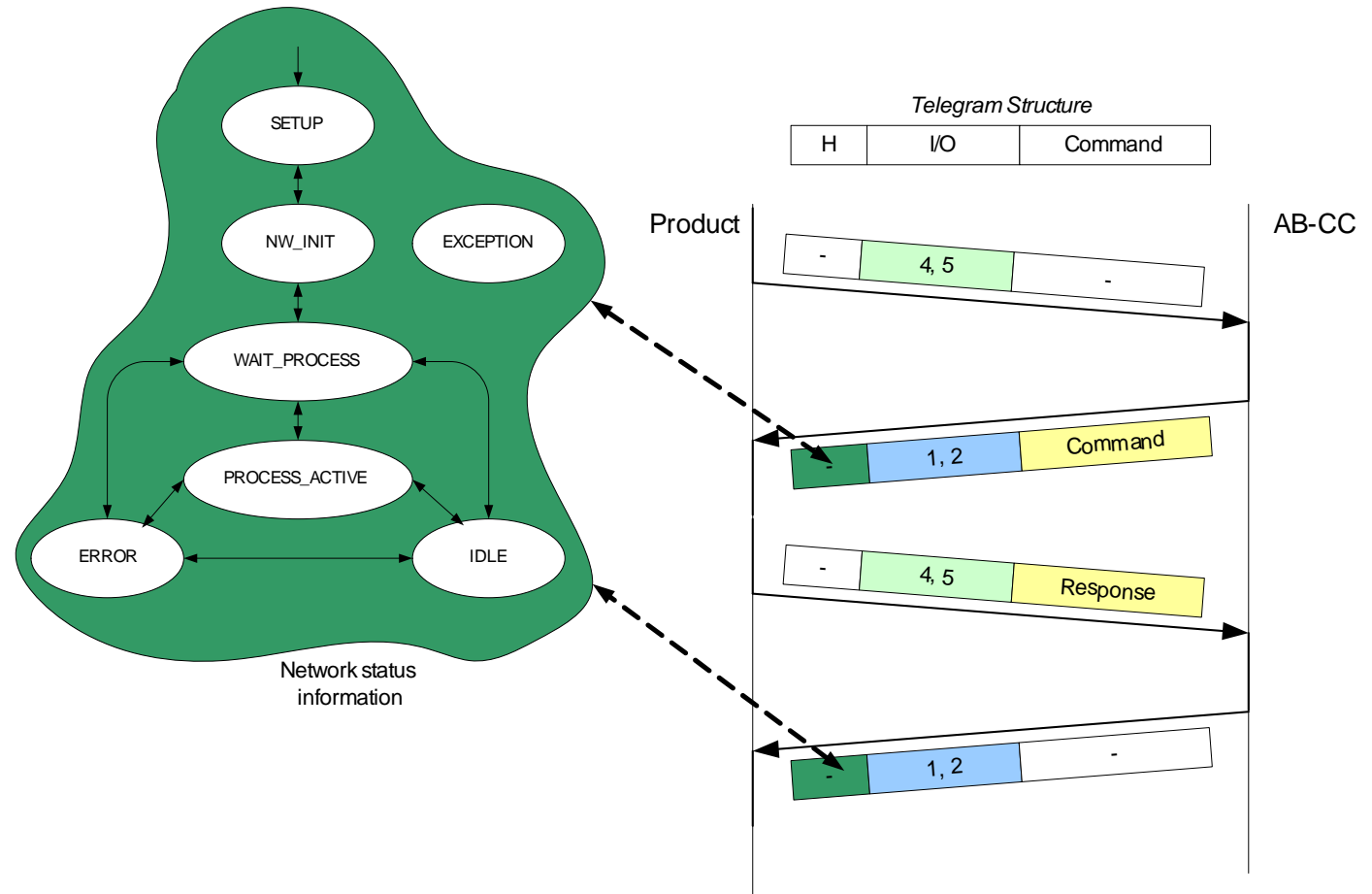


# Anybus-CC Diagnostics

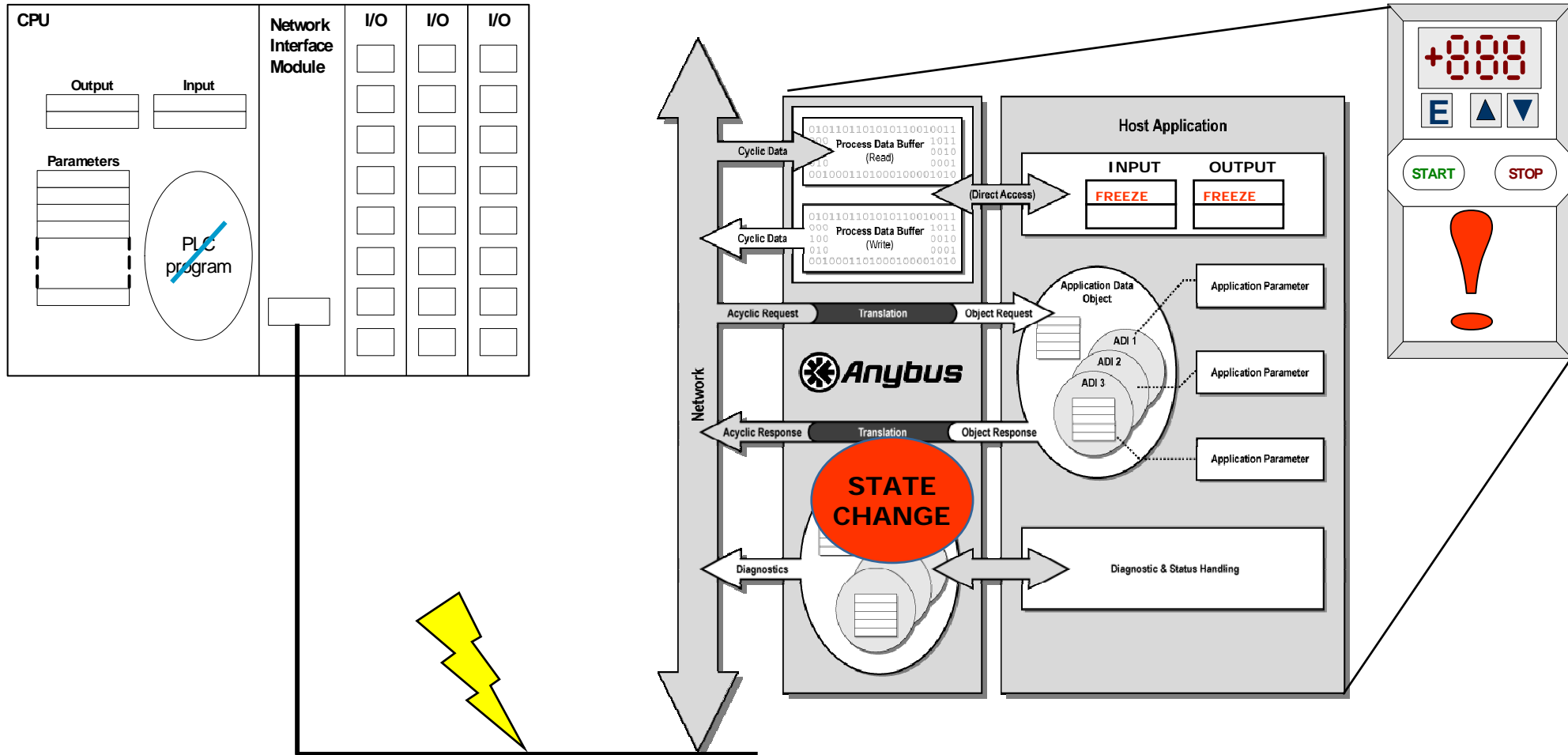


# Network Status

- The product must also be able to respond to network and system events for correct operation
- The Anybus-CC includes information about the actual network status within each transaction
  - ON-line/OFF-line, Idle, IO traffic, Error, etc.
- This information makes it possible to implement correct product behavior for all network states
- The status information is represented as a state machine

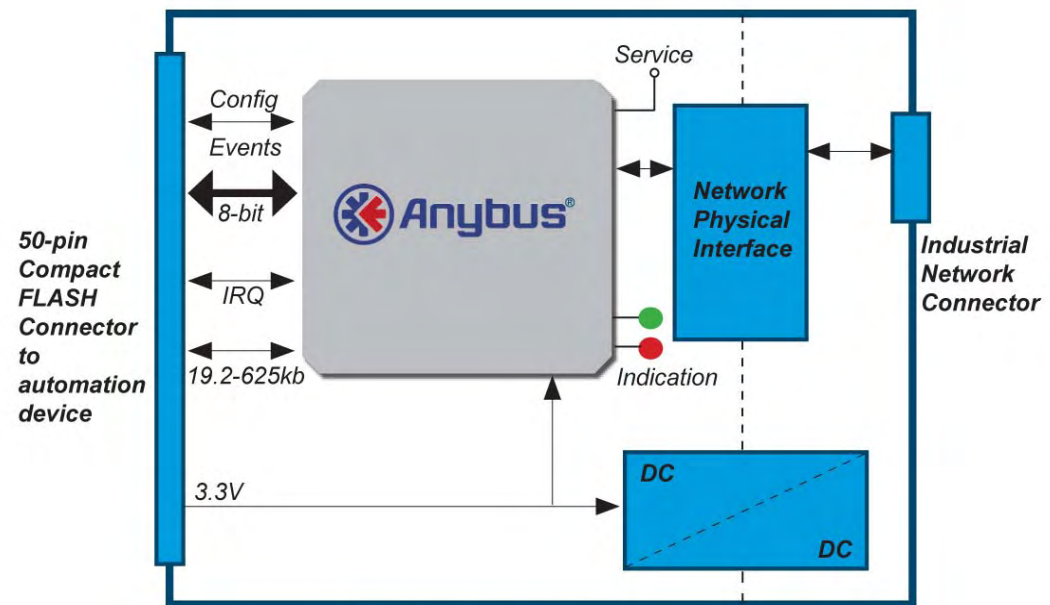


# Anybus-CC Status



# Application Interface and Hardware

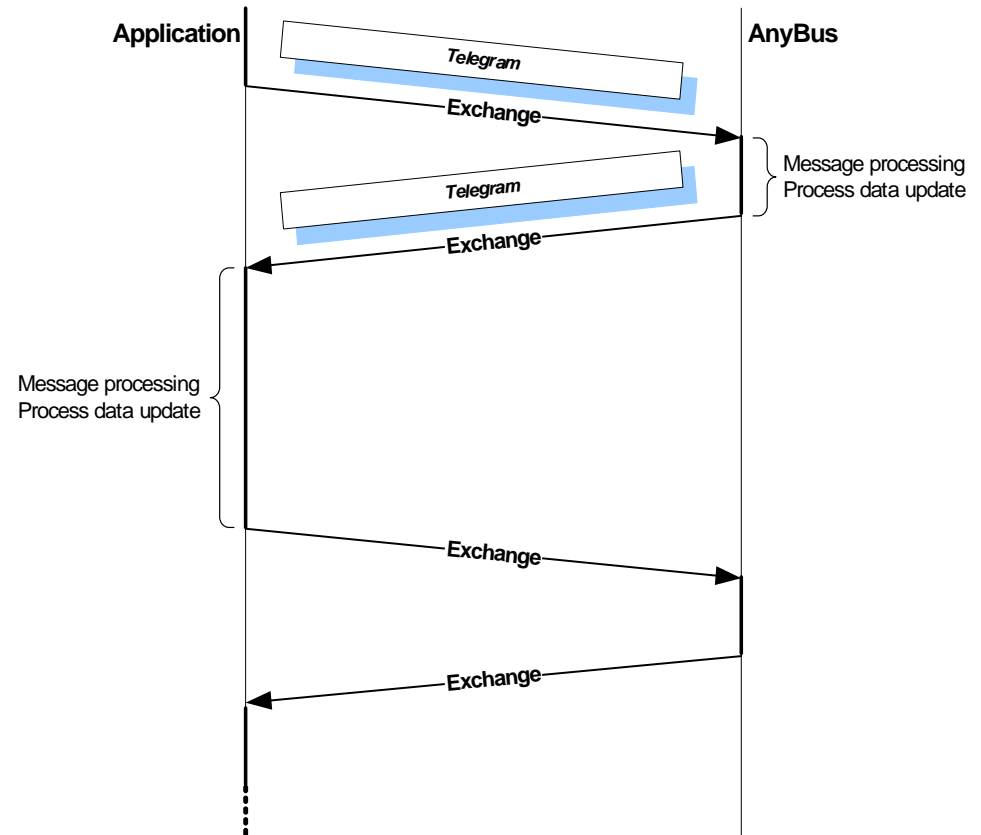
- Based on HMS newly developed 32/16-bit Risc microprocessor with built-in network controllers
- Parallel interface
  - 2kB (8-bit) Dual Ported RAM interface (DPRAM)
  - 30ns access
  - Interrupt or polled
- Serial interface
  - Standard asynchronous UART interface (n,8,1)
  - Configurable baud rates 19.2kbps – 625kbps
  - Fixed frame size for efficient DMA usage



Anybus-CC Overview

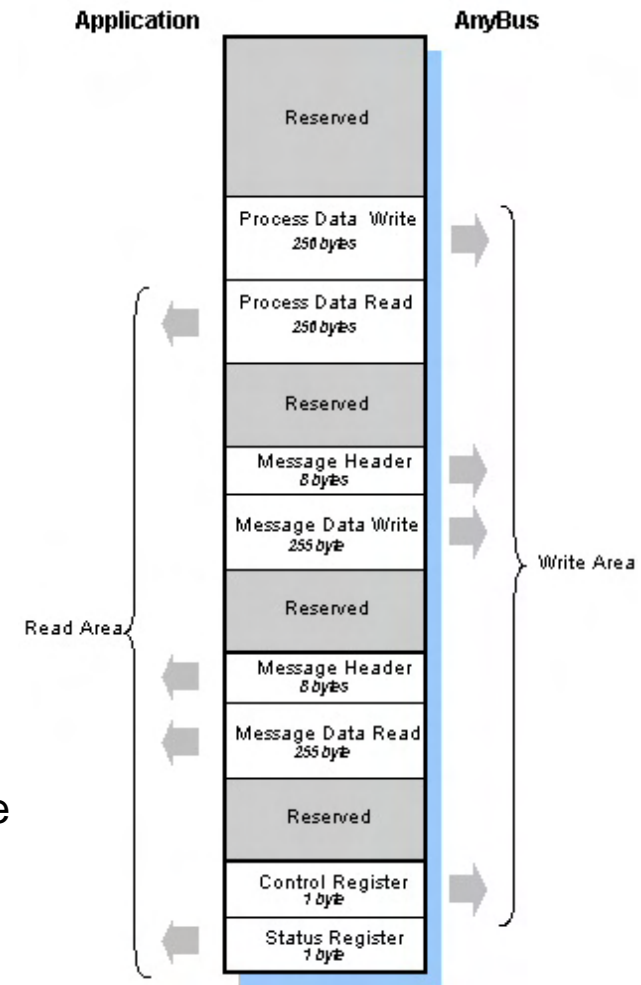
# How to interface

- The application interface is based on telegram messaging and the serial and parallel interface uses the same telegram structure
- Maximum amount of Process data is 256 byte and 255 byte of message data
- The telegrams are exchanged in half duplex and the application controls the exchange
- The Anybus-CC will respond to the telegrams with a minimum delay



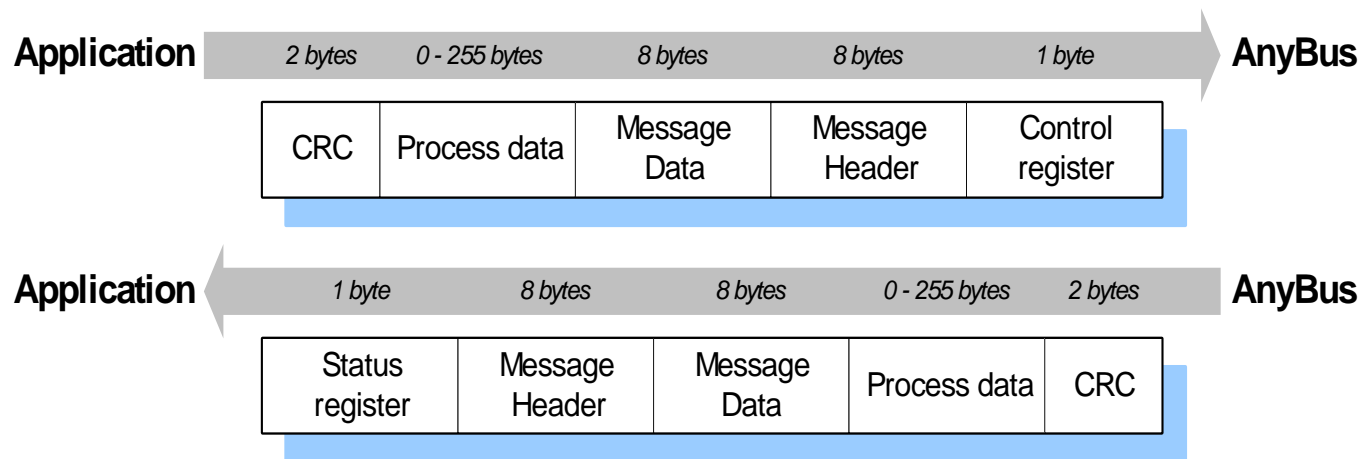
# Parallel Interface

- **The telegram structure is mapped directly on the 2kB DPRAM**
  - No additional information e.g. CRC is used
- **Communication Control** – contains control and status information for the data exchange and status
  - **Control Register** - Used by the application to control the communication with the Anybus
  - **Status Register** - Used by the Anybus to control the communication with the application and to indicate module and network status
- **Message Header** - Contains command/response information like data length, command type, object, instance, attribute as well as error indications. There can be several outstanding messages in each direction simultaneously
- **Message Data** - Data Associated with the command/response (if any) 0 – 255 bytes
- **Process Data** – Each telegram contains the network process data (I/O) 0 – 256 bytes

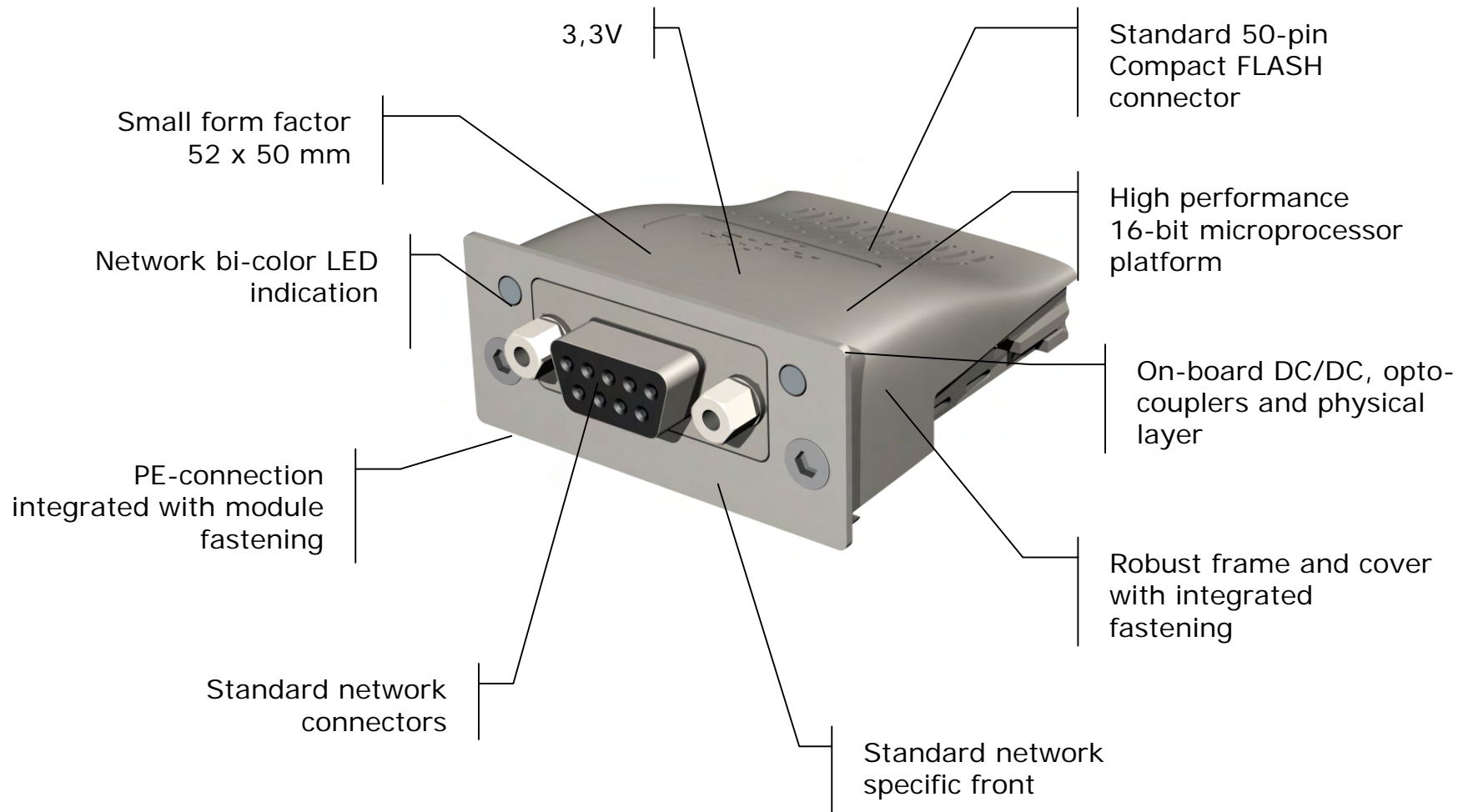


# Serial Interface

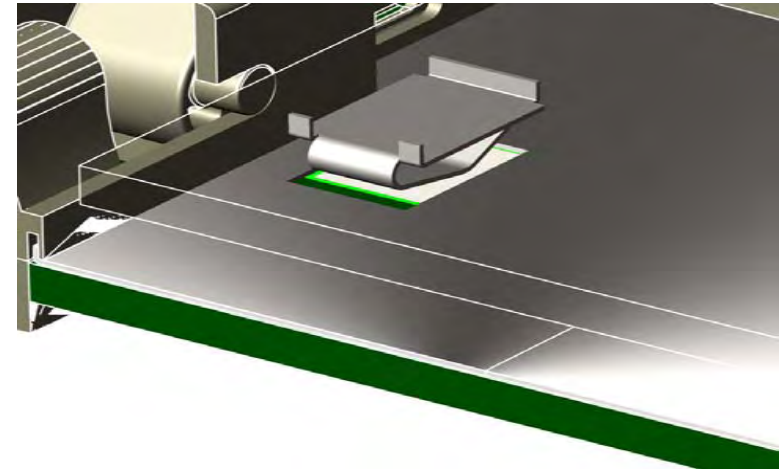
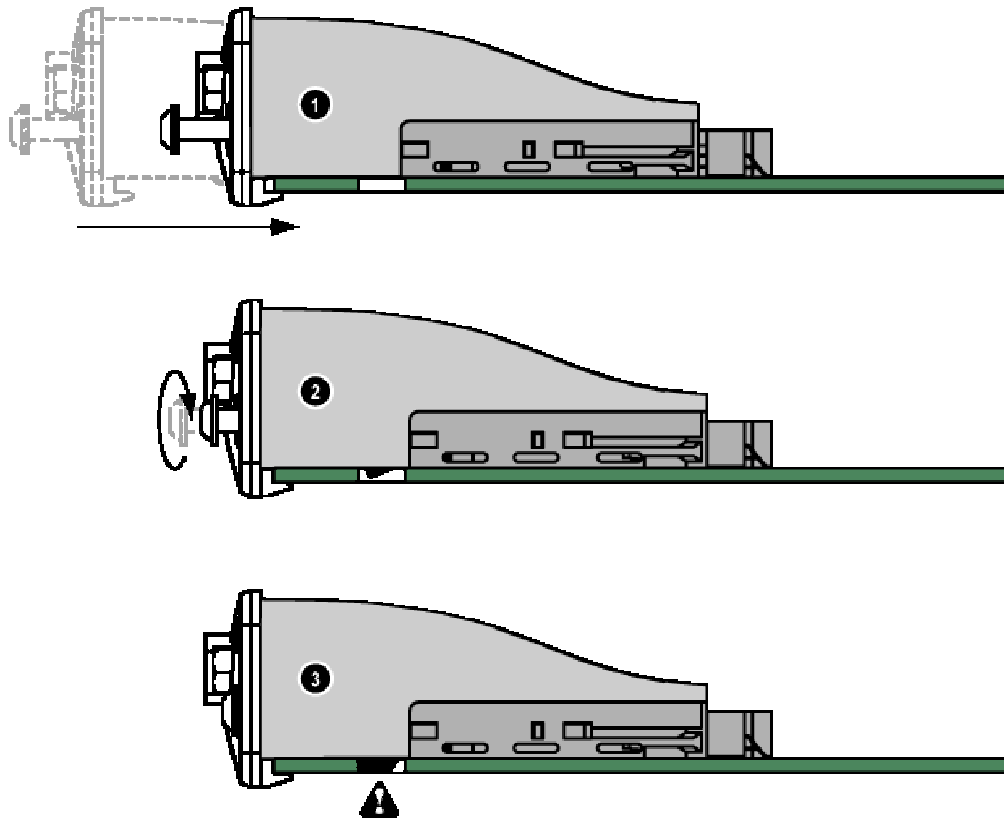
- The telegram structure is mapped directly on the serial frame
- A 16-bit CRC check sum is added for transmission error check
- The frame has a fixed length for DMA usage
  - In order to limit the overhead the message data is limited to 8 bytes in each telegram
  - A fragmentation protocol is used to for sending message frames (9 – 255 bytes)



# Anybus CompactCom Hardware



# Installation and PE connection

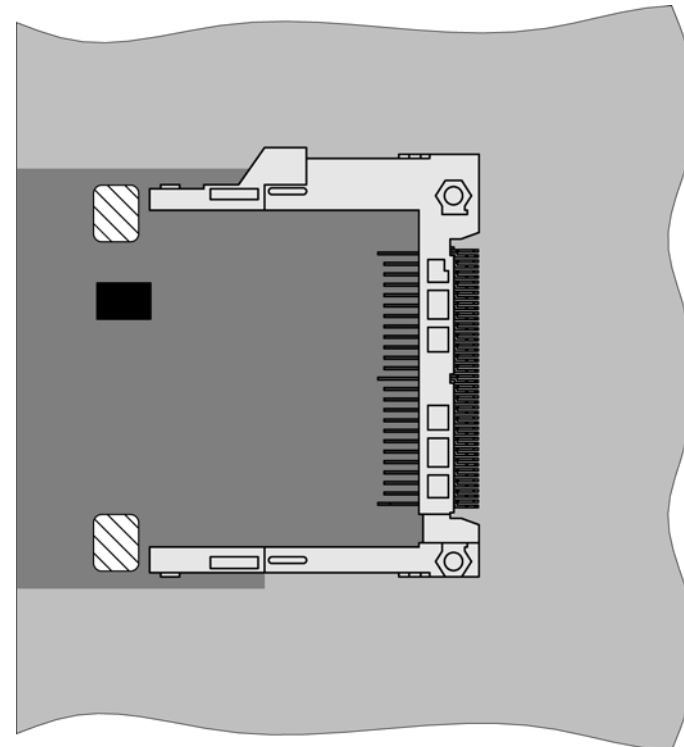


# Host Application PCB

Robust frame and cover with integrated fastening

Standard 50-pin Compact FLASH connector

PE-connection integrated with module fastening



- Reserved Area
- PE Area
- Fastening Support Holes
- Host Application PCB

- Beside the Compact Flash connector the application PCB need to have two fastening holes and a PE area
- No other components can be mounted in the reserved area

# Anybus CompactCom Passive modules

The Anybus CompactCom concept also provides other interface options like the Passive modules

- Passive modules only provide the low level physical layer of communications
- Can be used to implement direct access to configuration and monitoring port of the customer application
- Possible to implement proprietary protocols via RS232, RS485 or USB
- Future extensions include Ethernet Device Server, Bluetooth and other wireless technologies

Anybus-CC RS232 and 485



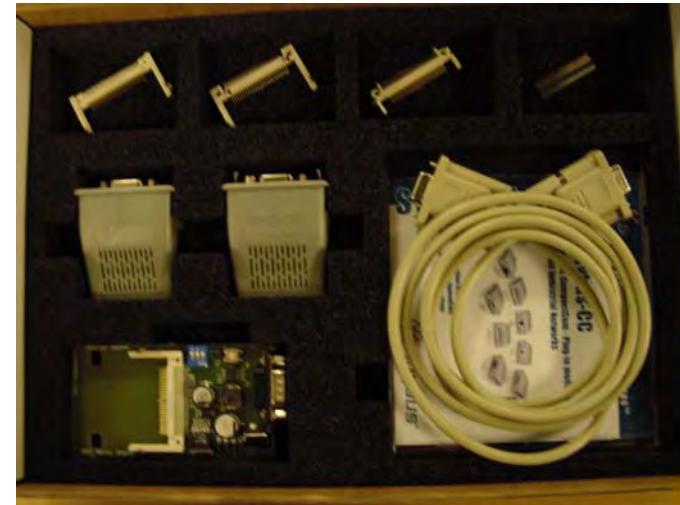
Anybus-CC USB card



# Anybus CompactCom Starterkit

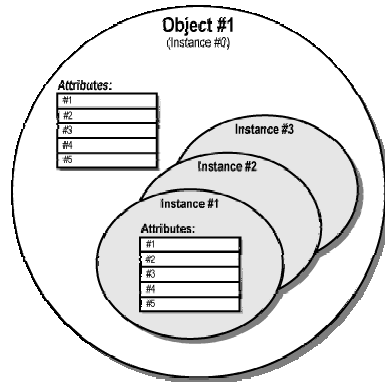
Contains all material required to make an in-design:

- 2 pieces of Anybus CompactCom modules (for selected fieldbus or passive modules)
- Example of CompactFlash connectors
- Anybus CompactCom mechanical drawings
- All manuals for software, hardware and fieldbuses
- Serial hardware adapter for connection to PC. The adapter can be used for test setup in PC environment and for firmware download to the ABCC
- Anybus CompactCom application driver software on CD

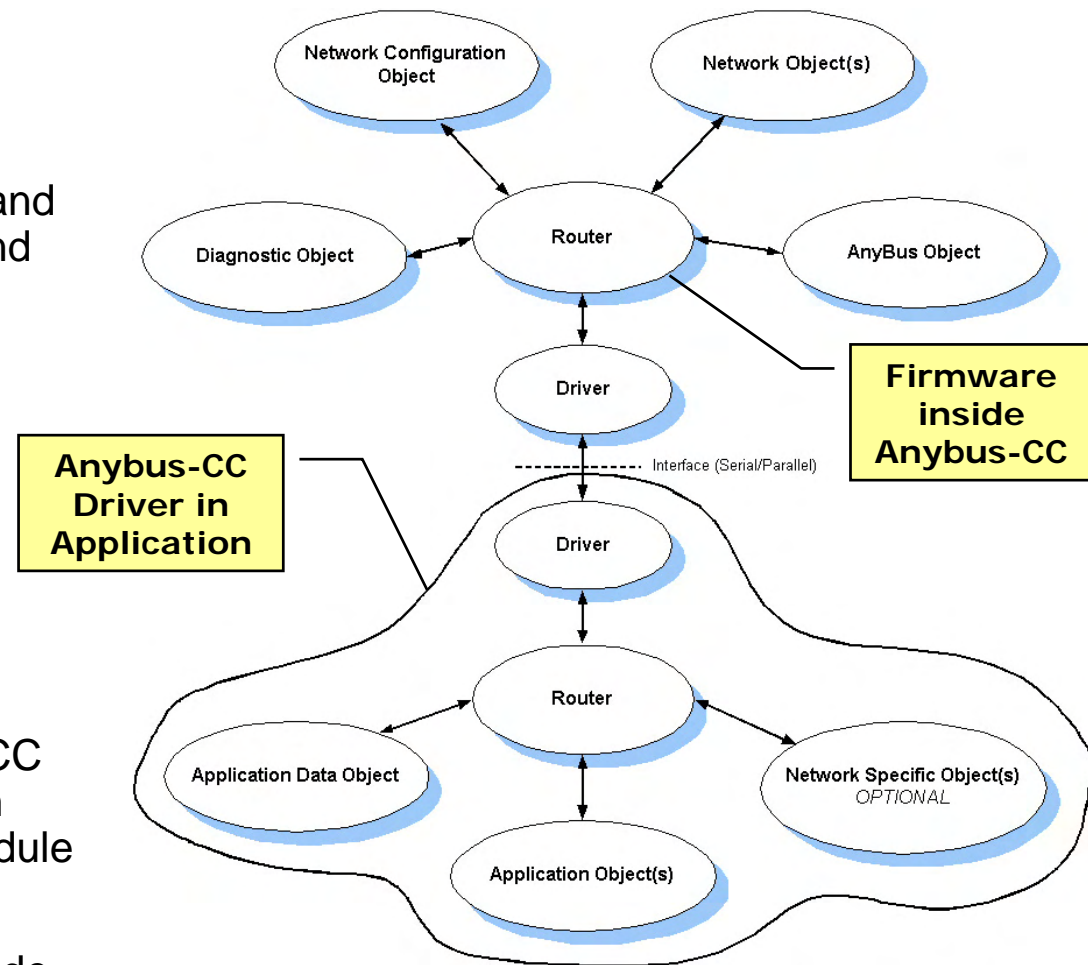


# Anybus-CC Object Model

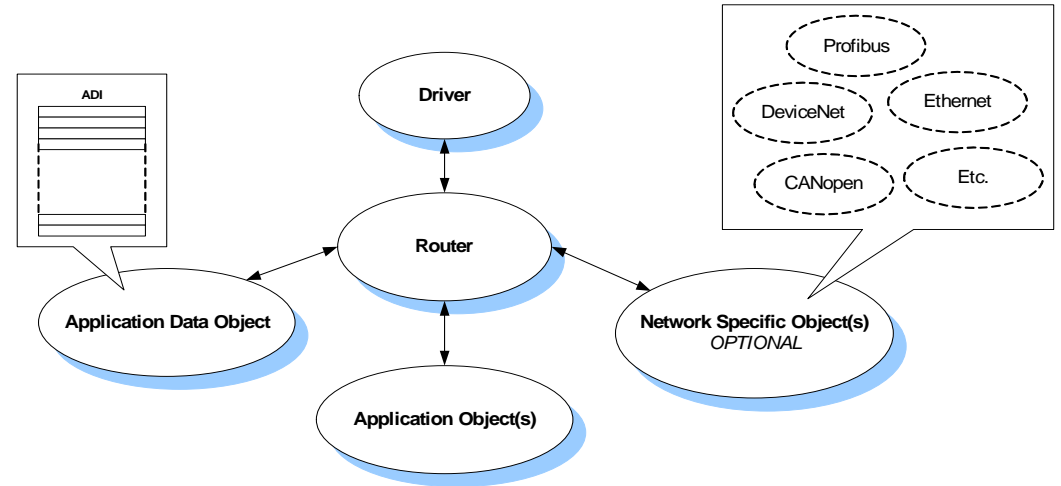
- The Anybus-CC API, functionality and software structure is described using an abstract object model
- The object model provides a well known and straight forward structure for designers and implementers



- All functions and features in the Anybus-CC is described using the object model which provides a good understanding of the module internal structure
- HMS provides device drivers in source code for fast integration



# Anybus CompactCom Application Driver



- ABCC Application Driver - working as glue between the ABCC physical interface and the application
- Providing a full driver between the application main software and the Anybus-CC module
- Including examples of how to set up Parameters/ADI's in the Application Data Object, Network Specific Objects and Application objects

# Anybus CompactCom Application Driver

## Items handled by the Application Driver

### Serial mode:

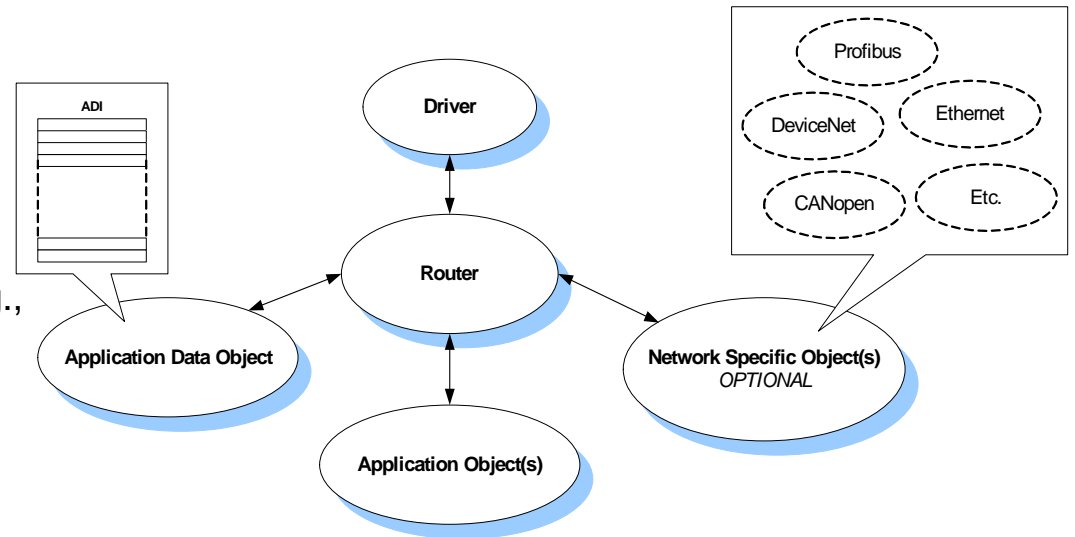
- Fragmentation.
- CRC-calculation.
- Interpretation of each telegram (Control / Status reg., Process Data vs. Messages).
- Retransmissions.
- Time-outs.
- Transmission mode change (Without PD or not).

### Parallel mode:

- Handshake register (Control / Status registers).
- Interpretation of each telegram (Process Data vs. Messages).
- Time-outs (Watchdog time-out).

### Both modes:

- Always handle Process Data mapping.
- Check if process data is new.

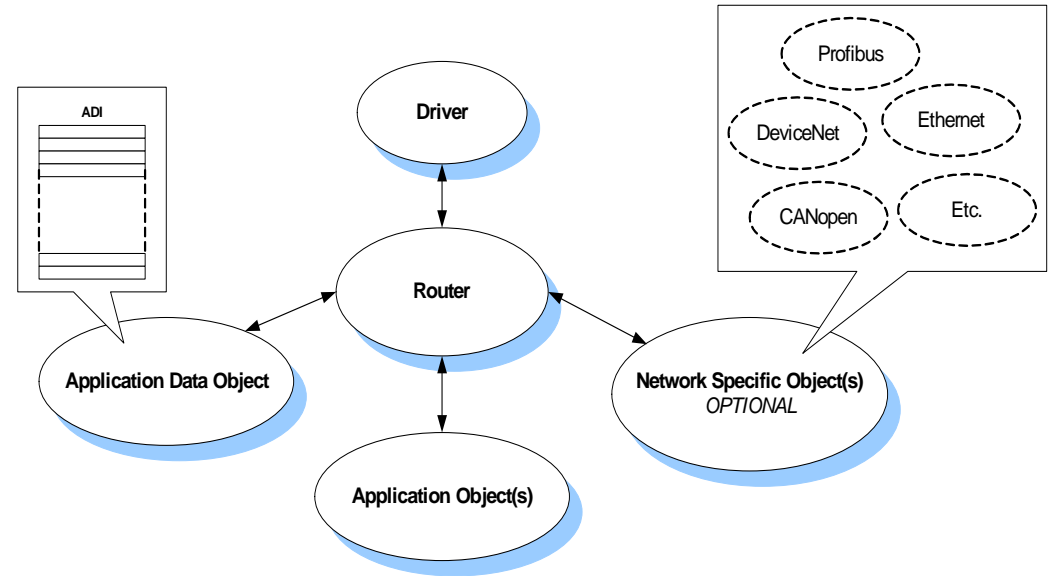


# Anybus CompactCom Application Driver

## Required application action:

In addition to porting hardware specific functions to the local platform, such as memory or serial hardware access, the application needs to follow the following guidelines.

- Call the timer system of the driver cyclically with a pre-defined interval, preferably 1ms.
- Specify the length of the read and write process data.
- List all ADI's that shall be mapped to Process data.
- Create Application objects and handle all object related actions.
- Handle object specific non cyclic requests. This includes ADI's (Application Data Instances), diagnostics and Anybus information.



# Anybus CompactCom Functional Summary

- Optimal ANY functionality i.e. the possibility for an application to change between different networks and still use the same Anybus driver
  - All network and protocol specific functions are handled by the Anybus as default (connections, data exchange, events, etc.)
  - The data handling functions supported by the Anybus lets the application control the network functionality i.e. optimal functionality for each application
  - Complete application parameter/data handling
  - The network specific objects enables the application to take full advantage of network specific data and services when needed
  - Standardized behavior and diagnostics independent of network type

