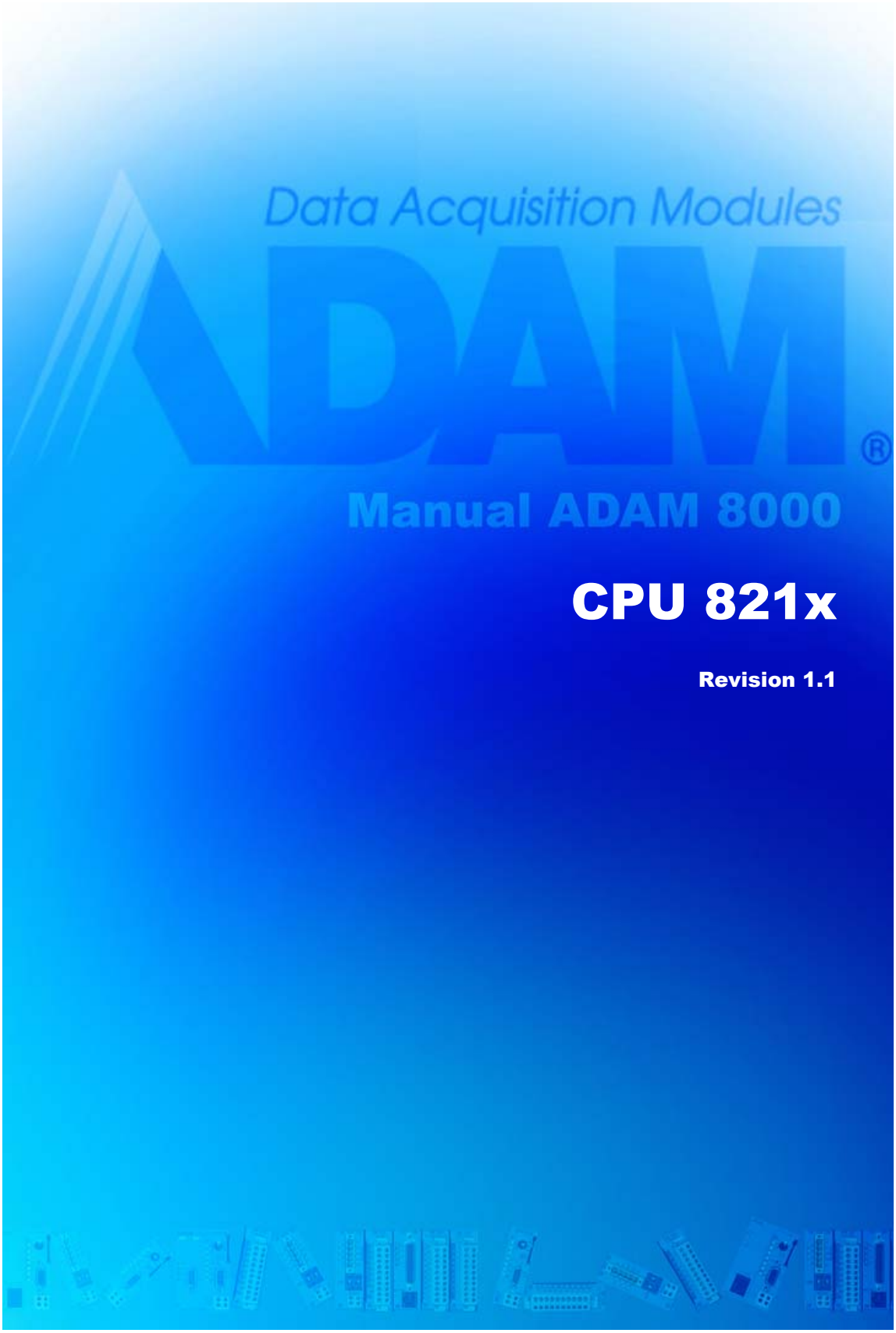

Data Acquisition Modules
ADAM®
Manual ADAM 8000
CPU 821x
Revision 1.1

The information in this manual is supplied without warranties. Information is subject to change without notice.

No. 1 Alley 20, Lane 2, Rueiguang Rd., Neihu District,
Taipei, Taiwan 114, R.O.C.

EMail: support@advantech.com
http://www.advantech.com

Disclaimer of liability

The contents of this manual was carefully examined to ensure that it conforms with the described hardware and software.
However, discrepancies can not be avoided. The specifications in this manual are examined regularly and corrections will be included in subsequent editions.
We gratefully accept suggestions for improvement.

Trademarks

ADAM® is a registered trademark of Advantech Co., Ltd.

STEP7® is a registered trademark of Siemens AG.

Any other trademarks referred to in the text are the trademarks of the respective owner and we acknowledge their registration.

About this manual

This manual describes the operation of the CPU 821x in the System ADAM 82xx from Advantech. The text provides details on the hardware, the programming and the functions integrated into the unit as well as Profibus and Ethernet applications.

Outline

Chapter 1: Principles

This introduction includes recommendations on the handling of the module as well as information about applications and implementation for CPU modules.

You may also read details about the mode of operation of the CPU 821x.

Chapter 2: Hardware description

Different versions of the CPU are available (CPU 821x, CPU 821xDP, CPU 821xNET). This chapter describes these versions in detail. In addition to the hardware description the chapter also includes a summary of the integrated circuits.

Chapter 3: Deployment of the CPU 821x

This chapter describes the deployment of the CPU section together with the peripheral modules of the System 8xxx that are installed on the same bus rail.

Chapter 4: Deployment of the CPU 821xNET

Applications, project design, functional description and programming of the CPU 821xNET.

Chapter 5: Deployment of the CPU 821xDPM

Content of this chapter is the deployment and project engineering of the CPU 821xDPM with integrated Profibus-DP master. Besides the description of the DP master operating modes you will also find information about the start-up behaviour and the commissioning.

Chapter 6: Deployment of the CPU 821xDP

Topic of this chapter is the CPU 821xDP (intelligent slave). You will find the description of the deployment, configuration and parameter definition for the CPU 821xDP under Profibus.

Chapter 7: Integrated SFCs

Here you find the description of the integrated ADAM-specific SFCs, like e.g. the SFCs for the CP communication.

Chapter 8: Command list

This chapter lists all available commands of the CPU in alphabetical order.

Contents

User considerations

Objective and contents

This manual describes the CPU 821x as well as all the versions of the product. It contains a description of the construction, project implementation and the application of the product.

The CPU 821x is compatible with all System ADAM 82xx components of Advantech.

Target audience

The manual is targeted at users who have a background in automation technology and PLC-programming.

Structure of the manual

This manual consists of 8 chapters. Every chapter provides the description of one specific topic.

Guide to the document

This manual provides the following guides:

- An overall table of contents at the beginning of the manual
- An overview of the topics for every chapter
- An index at the end of the manual.

Availability

The manual is available in:

- printed form on paper
- in electronic form as PDF-file (Adobe Acrobat Reader)

Icons Headings

Important passages in the text are highlighted by following icons and headings:

Danger!
Immediate or likely danger.
Personal injury is possible.

Attention!
Damages to property is likely if these warnings are not heeded.

Note!
Supplementary information and useful tips.

Safety information

Application specifications

The CPU 821x is constructed and manufactured for

- all System ADAM 82xx components
- communication and process control
- general control and automation tasks
- industrial applications
- operation within the environmental conditions specified in the technical data
- installation into a cubicle

Danger!

This device is not certified for applications in

- explosive environments (EX-zone)

Documentation

The manual must be available to all personnel in the

- project design department
- installation department
- commissioning
- operation

The following conditions must be met before using or commissioning the components described in this manual:

- Modification to the process control system should only be carried out when the system has been disconnected from power!
- Installation and modifications only by properly trained personnel
- The national rules and regulations of the respective country must be satisfied (installation, safety, EMC ...)

Disposal

National rules and regulations apply to the disposal of the unit!

Appendix

A Index

Chapter 1 Principles

Outline

This introduction contains references on the handling and information about application areas and usage of the CPU modules.

It also provides certain suggestions on the approach when programming the module and the CPU specifications that are important in this respect.

Below follows a description of:

- Safety information for users
- Construction and operation of the CPU 821x
- Programming principles

Contents

Safety information for users

Handling of electrostatically sensitive modules

ADAM modules make use of highly integrated components in MOS-technology. These components are extremely sensitive to over-voltages that may occur during electrostatic discharges.

The following symbol is attached to modules that can be destroyed by electrostatic discharges:

The symbol is located on the module, the module rack or on packing material and it indicates the presence of electrostatically sensitive equipment.

It is possible that electrostatically sensitive equipment is destroyed by energies and voltages that are far less than the human threshold of perception. These voltages may occur where persons do not discharge themselves before handling electrostatically sensitive modules and they may damage components thereby causing the module to become inoperable or unusable. Modules that have been damaged by electrostatic discharge are usually not detected immediately. The respective failure may become apparent after a period of operation.

Components damaged by electrostatic discharges can fail after a temperature change, mechanical shock or changes in the electrical load.

Only the consistent implementation of protective devices and meticulous attention to the applicable rules and regulations for handling the respective equipment is able to prevent failures of electrostatically sensitive modules.

Shipping of modules

Please ship the modules exclusively in the original packing material.

Measurements and alterations on electrostatically sensitive modules

When you are conducting measurements on electrostatically sensitive modules you should take the following precautions:

- Floating instruments must be discharged before use.
- Instruments must be grounded.

You should only use soldering irons with grounded tips when you are making modifications on electrostatically sensitive modules.

Attention!

Personnel and instruments should be grounded when working on electrostatically sensitive modules.

Hints for the deployment of the MPI interface

What is MP²I?

The MP²I jack combines 2 interfaces in 1:

- MP interface
- RS232 interface

Please regard that the RS232 functionality is only available by using the Green Cable from Advantech.

Deployment as MP interface

The MP interface provides the data transfer between CPUs and PCs. In a bus communication you may transfer programs and data between the CPUs interconnected via MPI.

Connecting a common MPI cable, the MPI jack supports the full MPI functionality.

Important notes for the deployment of MPI cables!

Deploying MPI cables at the ADAM CPUs from Advantech, you have to make sure that Pin 1 is not connected. This may cause transfer problems and in some cases damage the CPU!

Especially Profibus cables from Siemens, like e.g. the 6XV1 830-1CH30, must not be deployed at MP²I jack.

For damages caused by nonobservance of these notes and at improper deployment, ADAM does not take liability!

Deployment as RS232 interface only via "Green Cable"

For the serial data transfer from your PC, you normally need a MPI transducer. Fortunately you may also use the "Green Cable" from Advantech You can order this under the order no. ADAM-8950-0KB00.

The "Green Cable" supports a serial point-to-point connection for data transfer via the MP²I jack exclusively for ADAM 8xxx CPUs.

Please regard the hints for the deployment of the "Green Cable" on the following page.

Hints for the deployment of the Green Cable

What is the Green Cable?

The Green Cable is a green connection cable, manufactured exclusively for the deployment at ADAM System components.

The Green Cable is a programming and download cable for ADM CPUs 8xxx and ADAM fieldbus masters. The Green Cable from Advantech is available under the order no. ADAM-8950-0KB00.

The Green Cable allows you to:

- *transfer projects serial*
  Avoiding high hardware needs (MPI transducer, etc.) you may realize a serial point-to-point connection via the Green Cable and the MP^2I jack. This allows you to connect components to your ADAM CPU that are able to communicate serial via an MPI adapter like e.g. a visualization system.
- *execute firmware updates of the CPUs and fieldbus masters*
  Via the Green Cable and an upload application you may update the firmware of all recent CPUs 8xxx and certain fieldbus masters (see Note).

Important notes for the deployment of the Green Cable

Nonobservance of the following notes may cause damages on system components.

For damages caused by nonobservance of the following notes and at improper deployment, Advantech does not take liability!

Note to the application area

The Green Cable may exclusively deployed directly at the concerning jacks of the ADAM components (in between plugs are not permitted).

At this time, the following components support the Green Cable:

CPUs 8xxx and the fieldbus masters ADAM 8208-1xx01 from Advantech.

Note to the lengthening

The lengthening of the Green Cable with another Green Cable res. The combination with further MPI cables is not permitted and causes damages of the connected components!

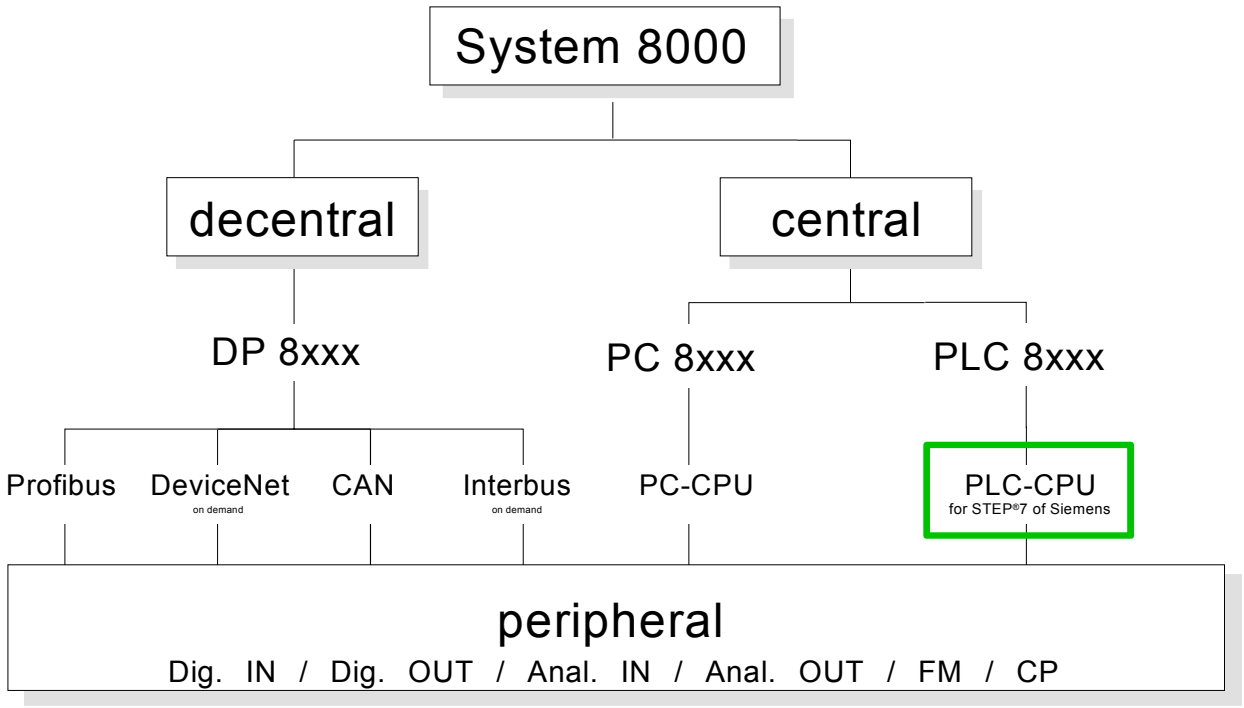
The Green Cable may only be lengthened with a 1:1 cable (all 9 Pins are connected 1:1).

Overview System 82xx

The System 82xx

The System 82xx is a modular automation system for low and middle range of performance that you may use either centralized or decentralized. The single modules are directly clipped to a 35mm DIN rail and are connected together with the help of special bus clips.

The following picture shows the range of performance of the System82xx:



Overview Manuals

The current manual describes the PLC-CPU family CPU 821x compatible to STEP®7 by Siemens.

The peripheral modules, PCs and decentralized peripheral equipment are to find in the manual HB97 "System82xx". This manual also contains hints for installation and commissioning of the System82xx.

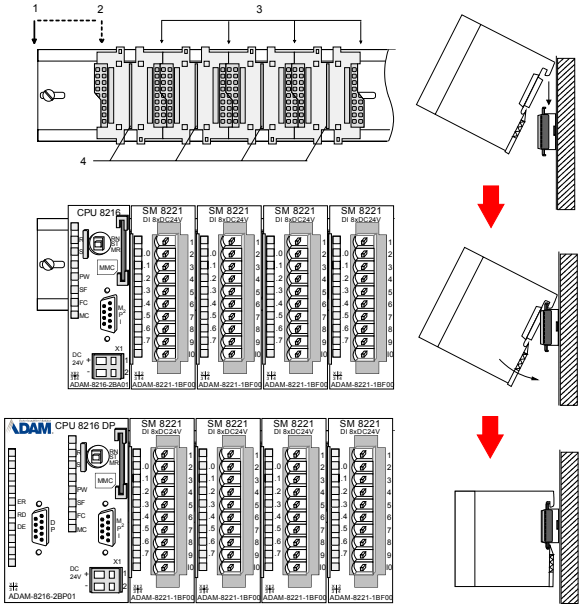
General description of the System 82xx

Structure / Dimensions

- Norm profile head rail 35mm
- Peripheral modules with labeling strip
- Measurements basic module:
  Single width: (HxWxD) in mm: 76x25.4x76; in inches: 3x1x3
  Double width: (HxWxD) in mm: 76x50.8x76; in inches: 3x2x3

Installation

Please note, that you have to plug in the CPU only at plug-in location 1 resp. 1 and 2 (if double width).



[1] CPU if double width
[2] CPU single width
[3] peripheral modules
[4] leading bars

Operating security

- Plug in via CageClamps at the front-facing connector, core cross-section 0.08...2.5mm^2 resp. 1.5 mm^2 (18pin plug)
- Total isolation of the cables during module changes
- Potential separation of all modules to the backplane bus
- EMC-proof ESD/Burst acc. EN61000-4-2/EN61000-4-4 to Level 3: 8kV/2.5kV
- Shock resistance acc. IEC 60068-2-6 / IEC 60068-2-27 (1G/12G)

Environmental conditions

- Operating temperature: 0... +55°C
- Storage temperature: -40... +85°C
- Relative humidity: 95% without condensation
- fan-less operation

Overview CPU 821x

Products

The ADAM CPU 821x is available in four different versions also differing in view:

- **CPU 821x** PLC-CPU
- **CPU 821xNET** PLC-CPU with Ethernet interface
- **CPU 821xDPM** PLC-CPU with Profibus-DP master
- **CPU 821xDP** PLC-CPU with Profibus-DP slave

Performance specification

Every CPU 821x is available in 3 CPU performance specifications: 214, 215 and 216. There is no visual difference between these units. The performance of the CPU 821x increases along with an increase of the number.

Data	CPU 8214	CPU 8215	CPU 8216
RAM	40kByte	80kByte	192kByte
Load memory	32kByte	64kByte	128kByte
Cycle time, bit/word operations	0.18µs / 0.78µs		
Bit memory/marker	2048Bit (M0.0...M255.7)		
Timer	128 (T0...T127)		
Counter	256 (Z0...Z255)		
Number of blocks FB / FC / DB	1024 / 1024 / 2047		
Total addressing space inputs / outputs	1024 / 1024 with 128Byte process image (PA) each		
PA Inputs	1024Bit (E0.0...E127.7)		
PA Outputs	1024Bit (A0.0...A127.7)		
Order data			
CPU 21x	ADAM-8214-1BA01	ADAM-8215-1BA01	ADAM-8216-1BA01
CPU 21xNET	ADAM-8214-2BT01	ADAM-8215-2BT01	ADAM-8216-2BT01
CPU 21xDPM	ADAM-8214-2BM01	ADAM-8215-2BM01	ADAM-8216-2BM01
CPU 21xDP	ADAM-8214-2BP01	ADAM-8215-2BP01	ADAM-8216-2BP01

Note!

If not especially specified, the information in this manual refers to all the CPUs of the ADAM CPU 821x family!

General information

The instruction set of the CPU 821x is compatible with STEP®7 of Siemens and is being configured by means of the Siemens STEP®7 manager. A large function library is included with the CPU.

The CPU employs a Multi Media Card (MMC) as external storage medium. The MMC is available from Advantech.

The project engineering of the Ethernet components takes place by means of the ADAM project configuration tool WinNCS. The data may be transferred into the CPU via MPI.

The CPU 821x has an integrated power supply that requires DC 24V via the front panel. It is protected against reverse polarity and short circuits.

Hints for project engineering

Outline

For the project engineering of the CPU 821x and the other System 82xx modules connected to the same bus, you use the hardware configurator from Siemens.

To address the directly plugged peripheral modules, you have to assign a special address in the CPU to every one.

The address allocation and the parameterization of the modules takes place in the STEP®7 manager from Siemens in form of a virtual Profibus system. For the Profibus interface is standardized software sided, the functionality is guaranteed by including a GSD-file into the STEP®7 manager from Siemens.

Transfer your project into the CPU via the MPI interface.

Preconditions

For the project engineering of your CPU 821x the following requirements have to be fulfilled:

- STEP®7 manager from Siemens is installed at your PC resp. PU.
- The GSD-file for the System 82xx is included to the hardware configurator
- serial connection to the CPU (e.g. via "Green Cable")

Note!

The configuration of the CPU requires a thorough knowledge of the Siemens STEP®7 manager and the hardware configurator!

Compatibility to STEP®7 manager from Siemens via GSD-file

The CPU 821x is configurable via the STEP7® manager from Siemens. This means the programming and parameterization of a System ADAM 82xx and the parameterization and project engineering under Profibus-DP.

The project engineering of a CPU 821x takes place via the STEP®7 manager from Siemens in form of a virtual Profibus system with the CPU 315-2DP as basic.

Due to the standardized Profibus interface ADAM is able to support the complete functionality of the System 82xx family in the STEP®7 manager from Siemens by including a GSD-file.

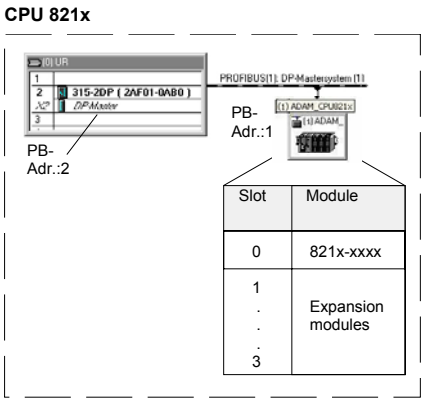
To be compatible with the STEP®7 configuration tool from Siemens, the following steps have to take place:

- **Project the Profibus-DP master system with CPU 315-2DP (6ES7 315-2AF01).**
- **Add Profibus slave with address 1.**
- **Include the CPU 821x at slot 0 of the slave system.**

Project engineering CPU 821x with central periphery

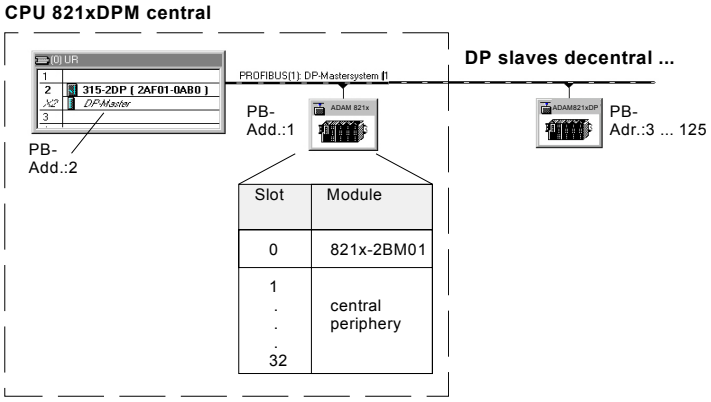
The following steps are necessary to project a CPU 821x in the hardware configurator from Siemens:

- Start the hardware configurator from Siemens
- Load the delivered ADAM GSD-file from Advantech
- Create a Profibus-DP master system with the CPU 315-2DP
- Include the slave system "ADAM_DP8000" from the hardware catalog in your master system. You find this slave system in the hardware catalog under *Profibus-DP* > *Additional field devices* > *I/O* > *ADAM*
- Assign the address 1 to your slave system, so that the CPU is able to recognize the system as central periphery system
- Add your modules to the slave system in the same order you have assembled them. Start with the CPU at plug-in location 0
- Include your System 82xx modules starting at plug-in location 1



Master projecting of the CPU 821xDPM

When projecting a CPU 821xDPM you include the central modules like described above. Slave systems that shall be connected to the master are added to the already existing master system:



Project engineering of the CPU 821xDP in a master system

When configuring a CPU 821xDP, the central plugged-in modules are parameterized like shown above.

Slave parameterization

As intelligent slave, the Profibus section maps its data areas into the memory area of the CPU 821xDP. The assignment of the areas is fixed via the propierties of the CPU 821xDP. These areas have to be provided with an according PLC program.

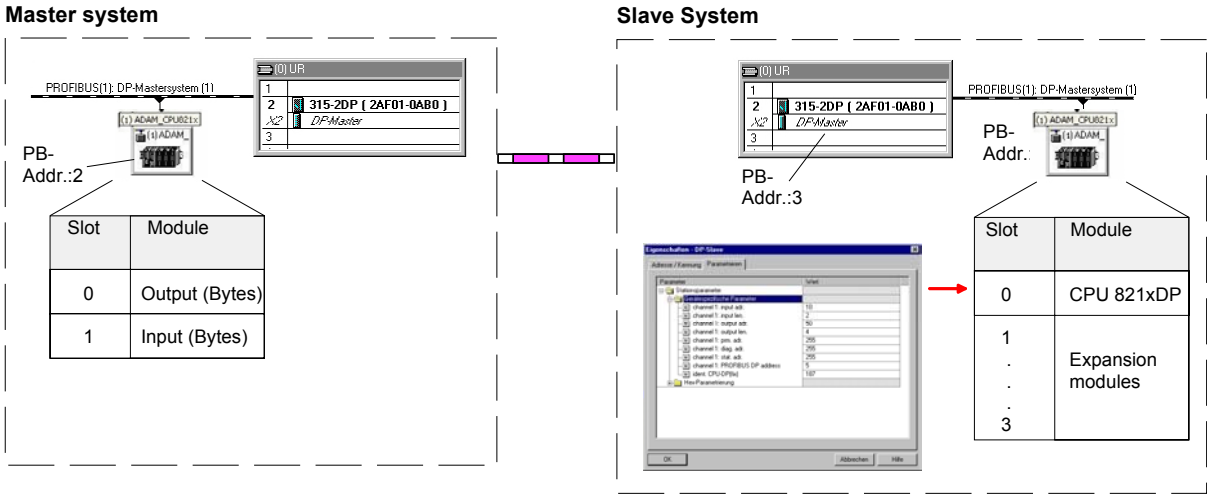
Attention!

The length values of input and output area have to be identical to the byte values of the master project engineering. Otherwise no Profibus communication is possible (slave failure).

Steps of project engineering in the DP master

- Configure the CPU with DP master system (address 2)
- Add Profibus slave 821xDP (GSD-file required)
- Assign Profibus input and output area starting with plug-in location 0

The following picture illustrates the project engineering:



Master projecting of the CPU 821xNET

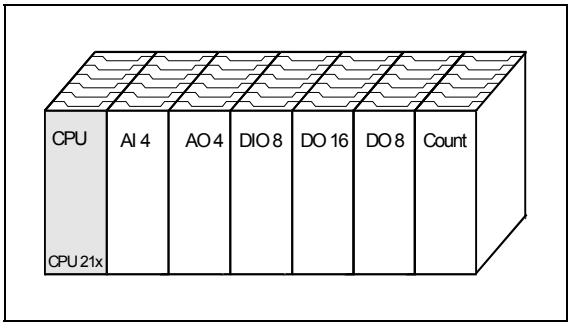
The project engineering of network connections via TCP/IP takes place via the configuration tool WinNCS from Advantech.

Application fields

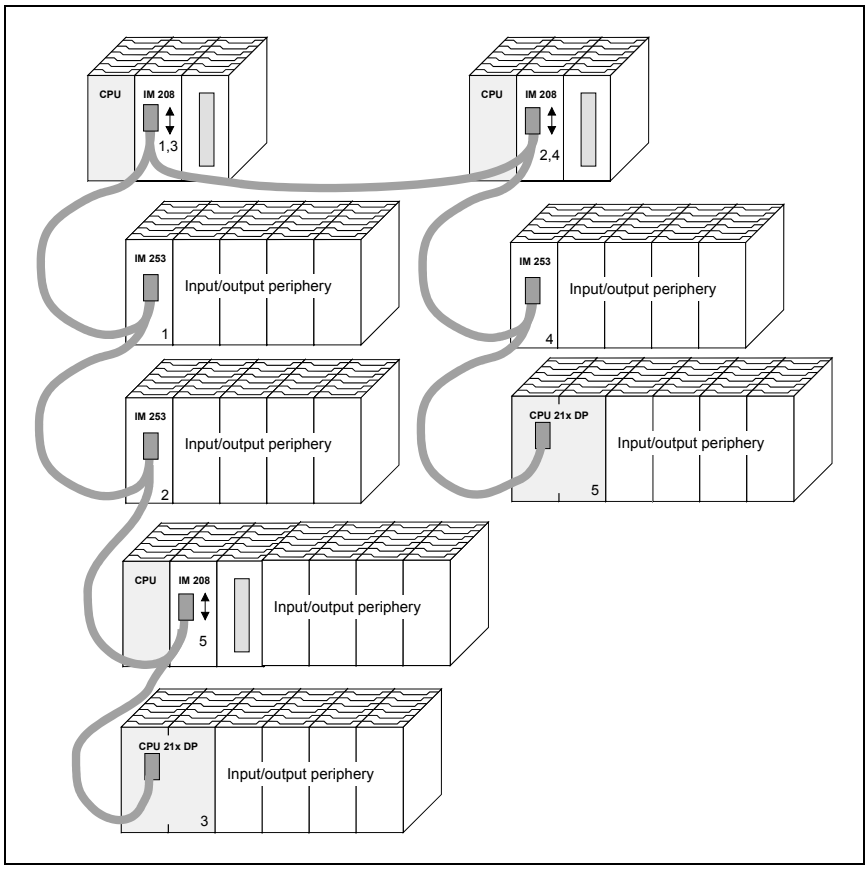
This series of CPU modules provides access to the peripheral modules of the ADAM System 8xxx. You can use a set of standard commands and programs to interrogate sensors and actuators. One single CPU may address a maximum of central 32 modules.

Application example

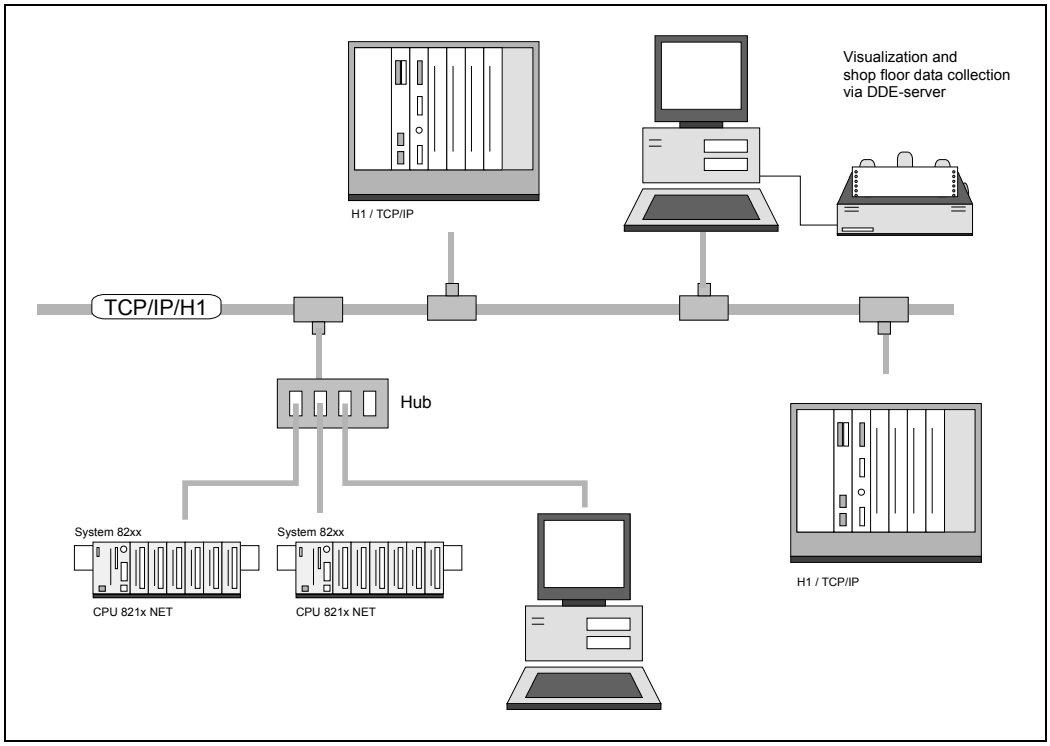
Compact centralized configuration



Decentralized configuration using Profibus



Application based on TCP/IP or H1



Features

The PLC-CPUs are employed as program executor.

They support the input/output modules and process the data from the function modules.

- Quick programming due to the compatibility with Siemens STEP®7.
- The compact construction requires less space.
- Enhanced flexibility provided by up to 32 function modules (DIG I/O, ANA I/O, SSI, pulse counter, communication modules, etc.).
- Additional function modules can be added quickly and easily by means of plug-in bus extension options.
- User-friendly maintenance using a PC via MPI.
- Optional Ethernet or Profibus interface.

Operating modes of a CPU

General

These CPUs are intended for small and medium sized applications and are supplied with an integrated 24V power supply. The CPU contains a standard processor with internal program memory. In combination with the System 82xx peripherals the unit provides a powerful solution for process automation applications within the System 82xx family.

A CPU supports the following modes of operation:

- cyclic processing
- timer processing
- alarm controlled processing
- priority based processing

Cyclic processing

Cyclic processing represents the major portion of all the processes that are executed in the CPU. Identical sequences of operations are repeated in a never ending cycle.

Timer processing

Where a process requires control signals at constant intervals you can initiate certain operations based upon a **timer**, e.g. not critical monitoring functions at one-second intervals.

Alarm controlled operation

If a process signal requires a quick response you would allocate this signal to an **alarm controlled** procedure. An alarm may activate a procedure in your program.

Priority based processing

The above processes are handled by the CPU in accordance with their **priority**. Since a timer or an alarm event requires a quick reaction the CPU will interrupt the cyclic processing when these high-priority events occur to react to the event. Cyclic processing will resume once the reaction has been processed. This means that cyclic processing has the lowest priority.

CPU 821x programs

Overview

The program that is present in every CPU is divided as follows:

- System routine
- User program

System routine

The system routine organizes all those functions and procedures of the CPU that are not related to a specific control application.

User program

This consists of all the functions that are required for the processing of a specific control application. The operating modules provide the interfaces to the system routines.

CPU 821x operands

Overview

The following operands are available for programming the CPU 821x:

- Process image and periphery
- Bit memory/marker
- Timers and counters
- Data blocks

Process image and periphery

The user program can quickly access the process image of the inputs and outputs PAA/PAE. You may manipulate the following types of data:

- individual bits
- bytes
- words
- double words

You may also gain direct access to peripheral modules via the bus from your user program. The following types of data are available:

- bytes
- words
- blocks

Bit memory

Bit memory is an area of memory that is accessible to the user program by means of certain operations. Bit memory is intended to store frequently used working data.

You may access the following types of data:

- individual bits
- bytes
- words
- double words

Timer and counter

With your program you may load a time cell with a value between 10ms and 9990s. As soon as the user program executes a start operation the value of this timer is decremented by the interval that you have specified until it reaches zero.

You may load counter cells with an initial value (max. 999) and increment or decrement this when required.

Data blocks

A data block contains constants or variables in form of bytes, words or double words. You may always access the current data block by means of operands.

You may access the following types of data:

- individual bits
- bytes
- words
- double words

Chapter 2 Hardware description

Outline

The CPUs 821x are available in different versions that are described in this chapter. In addition to the hardware description the chapter also contains installation and commissioning instructions and applications for the memory modules.

A summary of the integrated FBs and OBs and the technical data conclude the chapter.

The following section contains descriptions of:

- the components of the CPUs along with controls and displays
- the modules integrated into the CPU
- technical data

Contents

System overview

The ADAM CPU-821x family of products available from Advantech consists of 3 different models each with 4 versions:

- **CPU 821x** PLC-CPU
- **CPU 821xNET** PLC-CPU with Ethernet interface
- **CPU 821xDP** PLC-CPU with Profibus slave
- **CPU 821xDPM** PLC-CPU with Profibus master

All CPUs 821x are available in the versions 8214, 8215 and 8216.

CPU 821x

- Instruction set compatible with Siemens STEP®7
- MP-Interface for data transfer between PC and CPU
- Status LEDs for operating mode and diagnostics
- External memory card (MMC)
- "On board" memory

Order data CPU 821x

Type	Order number	Description
CPU 8214	ADAM-8214-1BA01	PLC CPU 8214 with 32KB of memory
CPU 8215	ADAM-8215-1BA01	PLC CPU 8215 with 64KB of memory
CPU 8216	ADAM-8216-1BA01	PLC CPU 8215 with 128KB of memory
MMC	ADAM-8953-0KX00	MMC storage module
Green Cable	ADAM-8950-0KB00	PG/AG download cable RS232/MPI, serial (only for usage at ADAMCPUs 8xxx)

CPU 821xNET

Identical to CPU 821x, additionally with:

- direct connection of twisted pair cable via RJ45 socket
- data throughput of up to 100 messages/sec
- bus load reduced by up to 20% due to simplified handshaking procedure
- drivers for different SCADA systems like zenOn, InTouch, etc.

Order data CPU 821xNET

Type	Order number	Description
CPU 8214NET	ADAM-8214-2BT01	PLC CPU 214 with Ethernet and 32KB of memory
CPU 8215NET	ADAM-8215-2BT01	PLC CPU 215 with Ethernet and 64KB of memory
CPU 8216NET	ADAM-8216-2BT01	PLC CPU 216 with Ethernet and 128KB of memory
MMC	ADAM-8953-0KX00	MMC storage module
Green Cable	ADAM-8950-0KB00	PG/AG download cable RS232/MPI, serial (only for usage at ADAM CPUs 8xxx)

CPU 821xDP

Identical to CPU 821x, additionally with

- integrated Profibus slave
- status LEDs for Profibus status and diagnostics

Order data CPU 821xDP

Type	Order number	Description
CPU 88214DP	ADAM-8214-2BP01	SPS CPU 214 with Profibus slave and 32KB of memory
CPU 8215DP	ADAM-8215-2BP01	SPS CPU 215 with Profibus slave and 64KB of memory
CPU 8216DP	ADAM-8216-2BP01	SPS CPU 216 with Profibus slave and 128KB of memory
MMC	ADAM-8953-0KX00	MMC storage module
Green Cable	ADAM-8950-0KB00	PG/AG download cable RS232/MPI, serial (only for usage at ADAM CPUs 8xxx)

CPU 821xDPM

Like CPU 821x, additionally with

- integrated Profibus-DP master
- Status-LEDs for Profibus status and diagnostics

Order data CPU 821xDPM

Type	Order number	Description
CPU 8214DPM	ADAM-8214-2BM01	PLC CPU 214 with Profibus-DP master and 32kByte memory
CPU 8215DPM	ADAM-8215-2BM01	PLC CPU 215 with Profibus-DP master and 64kByte memory
CPU 8216DPM	ADAM-8216-2BM01	PLC CPU 216 with Profibus-DP master and 128kByte memory
MMC	ADAM-8953-0KX00	MMC storage module
Green Cable	ADAM-8950-0KB00	PG/AG download cable RS232/MPI, serial (only for usage at ADAM CPUs 8xxx))

General

A CPU is an intelligent module. Your controlling programs are executed here. You are able to select one of three CPUs, depending on the performance required from your system. The higher the performance of the CPU, the more user memory is available.

The CPUs 821x are intended for small to medium applications and have an integrated 24V power supply. The CPUs contain a standard processor with internal program memory for the storage of user-programs. In addition, every CPU 821x is equipped with a socket for a memory module, which is located on the front.

Every CPU has an MPI-interface and is instruction set compatible with the Siemens STEP®7. The CPU series 214...216 have similar performance as the Siemens STEP®7 CPU series.

This series of CPUs provides access to the peripheral modules of the System 82xx. You may query sensors and control actuators by means of standardized commands and programs. The unit can address a maximum of 32 modules. You may parameterize the CPU via the integrated MPI-interface.

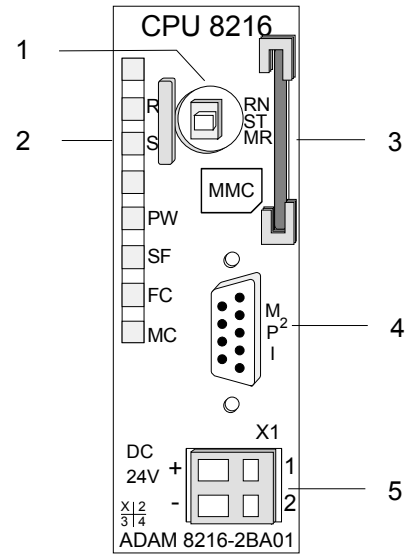
The remainder of this description refers to all CPUs of the CPU 821x family, since the CPUs 8214, 8215 and 8216 are functionally identical and their only difference is the memory size.

Properties

- Instruction set compatible to Siemens STEP7®
- Configuration by means of the Siemens STEP7® manager
- Integrated 24V power supply
- Total address range: 1024Byte inputs, 1024Byte outputs (128Byte process image each)
- 32...128kByte of work memory
- 40…192kByte of load memory
- Battery backed clock
- Memory-card socket
- MPI-interface
- Integrated V-Bus controller for controlling System 82xx peripherals
- User programs can be saved to the external memory card (MMC)
- 128 timer
- 64 counter
- 256Byte of bit memory

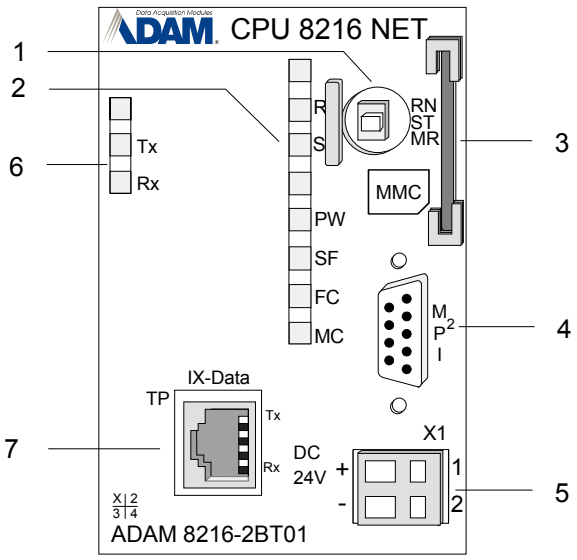
Structure

Front view CPU 821x



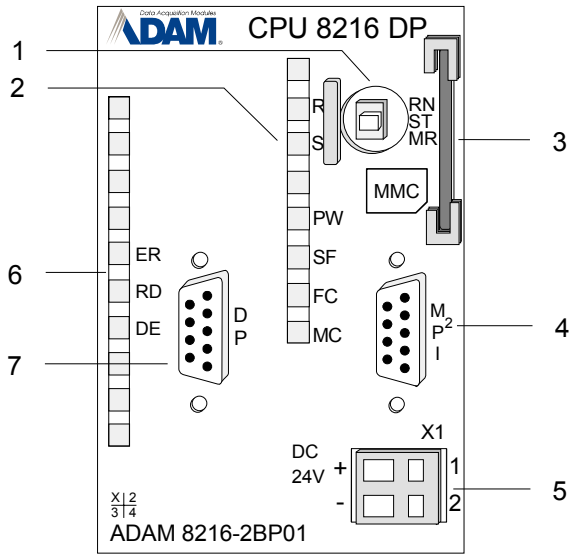
[1] RUN/STOP/OVERALL RESET operating mode selector switch
[2] Status indicator LEDs
[3] Socket for MMC memory card
[4] MP²I-interface
[5] Connector for 24V DC power supply

Front view CPU 821xNET



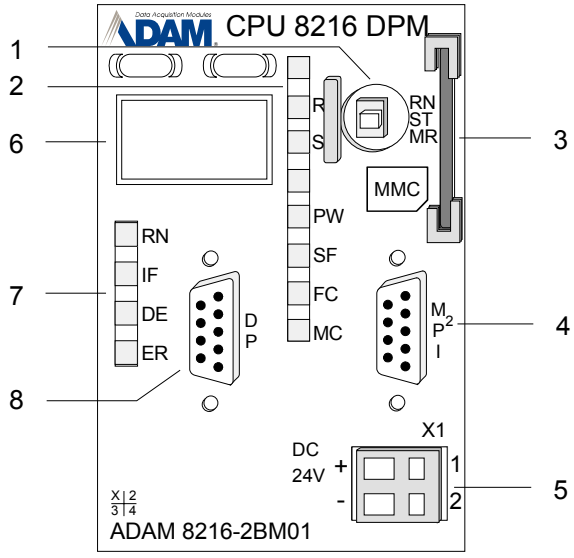
[1] RUN/STOP/OVERALL RESET operating mode selector switch
[2] Status indicator LEDs
[3] Socket for MMC memory card
[4] MP²I-interface
[5] Connector for 24V DC power supply
[6] Status indicator LEDs Ethernet
[7] Twisted Pair interface for Ethernet

Front view CPU 821xDP



[1] RUN/STOP/OVERALL RESET operating mode selector switch
[2] Status indicator LEDs
[3] Socket for MMC memory card
[4] MP²I-interface
[5] Connector for 24V DC power supply
[6] Profibus-DP slave status indicator LEDs
[7] Profibus interface

Front view CPU 821xDPM



[1] RUN/STOP/OVERALL RESET operating mode selector switch
[2] LEDs status indicator CPU
[3] Socket for MMC memory card
[4] MP²I-interface
[5] Connector for 24V DC power supply
[6] [1]LC-Display with keys
[7] LEDs status indicator Profibus-DP master
[8] Profibus interface

[1] in development

Components

CPU 821x

The components of the CPU 821x that are described here are components of all the CPUs presented in this manual.

LEDs

The CPUs 821x have a number of LEDs that are used to diagnose bus conditions and to display the status of a program. The table below describes the diagnostic LEDs and the according colors.

These LEDs are part of every CPU in this manual.

Name	Color	Description
PW	Yellow	Indicates CPU power on.
RN	Green	CPU status is RUN.
ST	Red	CPU status is STOP.
SF	Red	Is turned on if a system error is detected (hardware defect)
FC	Red	Is turned on when variables are forced (fixed).
MC	Yellow	This LED blinks when the MMC is accessed.

Function selector RN/ST/MR

You can select the operating mode STOP (ST) and RUN (RN) by means of the function selector. The CPU automatically executes the operating mode START-UP when the mode changes from STOP to RUN.

You may issue an overall reset by placing the switch in the Memory Reset (MR) position.

MMC socket memory card

You may install a ADAM MMC memory module in this slot as external storage device (Order No.: ADAM-8953-0KX00).

The access to the MMC takes always place after an overall reset.

Power supply

The CPU has an internal power supply. This is connected to an external supply voltage via two terminals located on the front of the unit.

The power supply requires DC 24V (20 ... 30V). In addition to the electronic circuitry of the CPU this supply voltage is used for the modules connected to the backplane bus.

The electronic circuitry of the CPU is not dc-insulated from the supply voltage. The power supply is protected against reverse polarity and short circuits.

Note!

Please ensure that the polarity of the supply voltage is correct.

Battery backup for clock and RAM

A rechargeable battery is installed on every CPU 821x to safeguard the contents of the RAM when power is removed. This battery is also used to buffer the internal clock.

The rechargeable battery is maintained by a charging circuit that receives its power from the internal power supply and that maintain the clock and RAM for a max. period of 30 days.

Attention!

The CPU will operate only if the battery is healthy.

The CPU will STOP when the battery is faulty. In this case the CPU should be checked. Please contact Advantech!

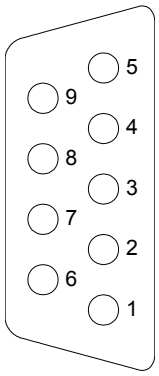
MP²I port

The MPI unit provides the link for the data transfer between the CPU and the PC. Via bus communication you are able to exchange programs and data between different CPUs that are linked over MPI.

For a serial exchange between the partners you normally need a special MPI-converter. But now you are also able to use the ADAM "Green Cable" (Order-No. ADAM-8950-0KB00), which allows you to establish a serial peer-to-peer connection over the MPI port.

Please regard the notes about the "Green Cable" in chapter 1!

The pin assignment of the MPI socket is as follows:



9pin socket

Pin	Assignment
1	reserved
2	GND
3	RS485_A
4	RS485_CTS
5	GND
6	Vcc
7	+DC 24V
8	RS485_B
9	RS485_RTS

CPU 821xNET

In addition to the components described in the section on the CPU 821x the CPU 821xNET module is provided with 2 further LEDs and an Ethernet interface located at the left-hand side of the module.

LEDs

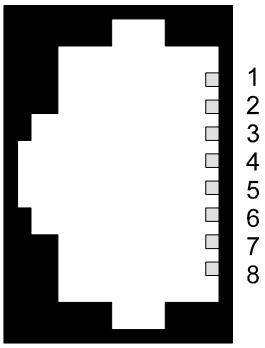
The LEDs are located on the left-hand side of the front panel and indicate communication activities.

The table below shows the color and the meaning of these LEDs.

Name	Color	Description
TxD	green	Transmit data
RxD	green	Receive data

Ethernet interface

An RJ45 socket provides the interface to the twisted pair cable, required for Ethernet. The pin assignment of this socket is as follows:

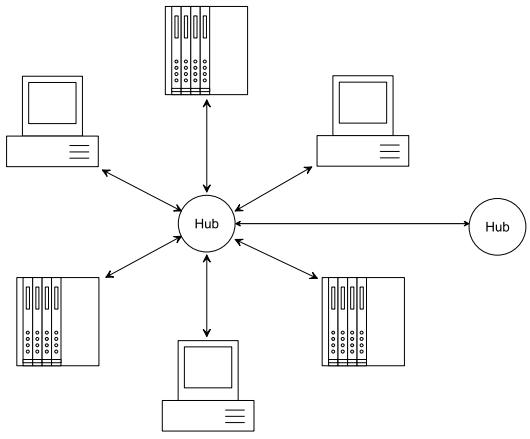


8pin RJ45 socket:

Pin	Signal
1	Transmit +
2	Transmit -
3	Receive +
4	-
5	-
6	Receive -
7	-
8	-

Star topology

A twisted pair network can only have a star topology. For this purpose a hub is required as the central node:



Note!

For more detailed information on twisted pair networks refer to chapter "Deployment of the CPU 821xNET".

CPU 821xDP

In addition to the components described in the section on the CPU 821x the CPU 821xDP module is provided with 3 more LEDs and a Profibus interface.

LEDs

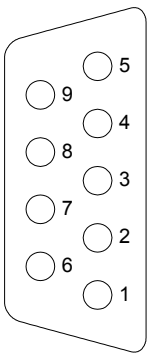
The LEDs are located in the left half of the front panel and they are used for diagnostic purposes. The following table shows the color and the significance of these LEDs.

Name	Color	Description
ER	red	On when an error is detected (Error). On when the CPU has been stopped. Flashes slowly (2Hz) at initialization error Flashes quickly (10Hz) when supply voltage falls below 18V. Flashes alternately with RD when the master configuration is bad (project configuration error). Flashes simultaneously with RD when parameterization is bad
RD	green	On when a data transfer is active via the V-bus. Flashes when the self-test result is positive (READY) and the initialization was successful.
DE	yellow	DE (Data exchange) indicates an active Profibus communication.

Profibus interface

The CPU 821xDP is connected to the Profibus system by means of a 9pin socket. The pin assignment of this interface is as shown:

9pin Profibus D-type socket:



Pin	Assignment
1	Screen
2	not used
3	RxD/TxD-P
4	CNTR-P
5	GND
6	5V (max. 70mA)
7	not used
8	RxD/TxD-N
9	not used

Note!

Refer to the chapter "Deployment of the CPU 821xDP" for details on the Profibus.

CPU 821xDPM

In addition to the components described in the section on the CPU 821x the CPU 821xDPM module is provided with 4 more LEDs and a Profibus interface.

LEDs

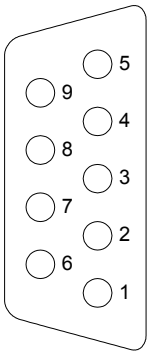
The LEDs are located in the left half of the front panel and they are used for diagnostic purposes. The following table shows the color and the significance of these LEDs.

Name	Color	Description
RN	green	If RN is the only LED that is on, then the master status is RUN. The slaves are being accessed and the outputs are 0 ("clear" state). If both RN+DE are on the status of the Master is "operate". It is communicating with the slaves.
IF	red	Initialization error for bad Profibus configurations
DE	yellow	DE (Data exchange) indicates Profibus communication activity.
ER	red	On when a slave has failed (ERROR).

Profibus interface

The CPU 821xDPM is connected to the Profibus system by means of a 9pin socket. The pin assignment of this interface is as shown:

9pin Profibus D-type socket:



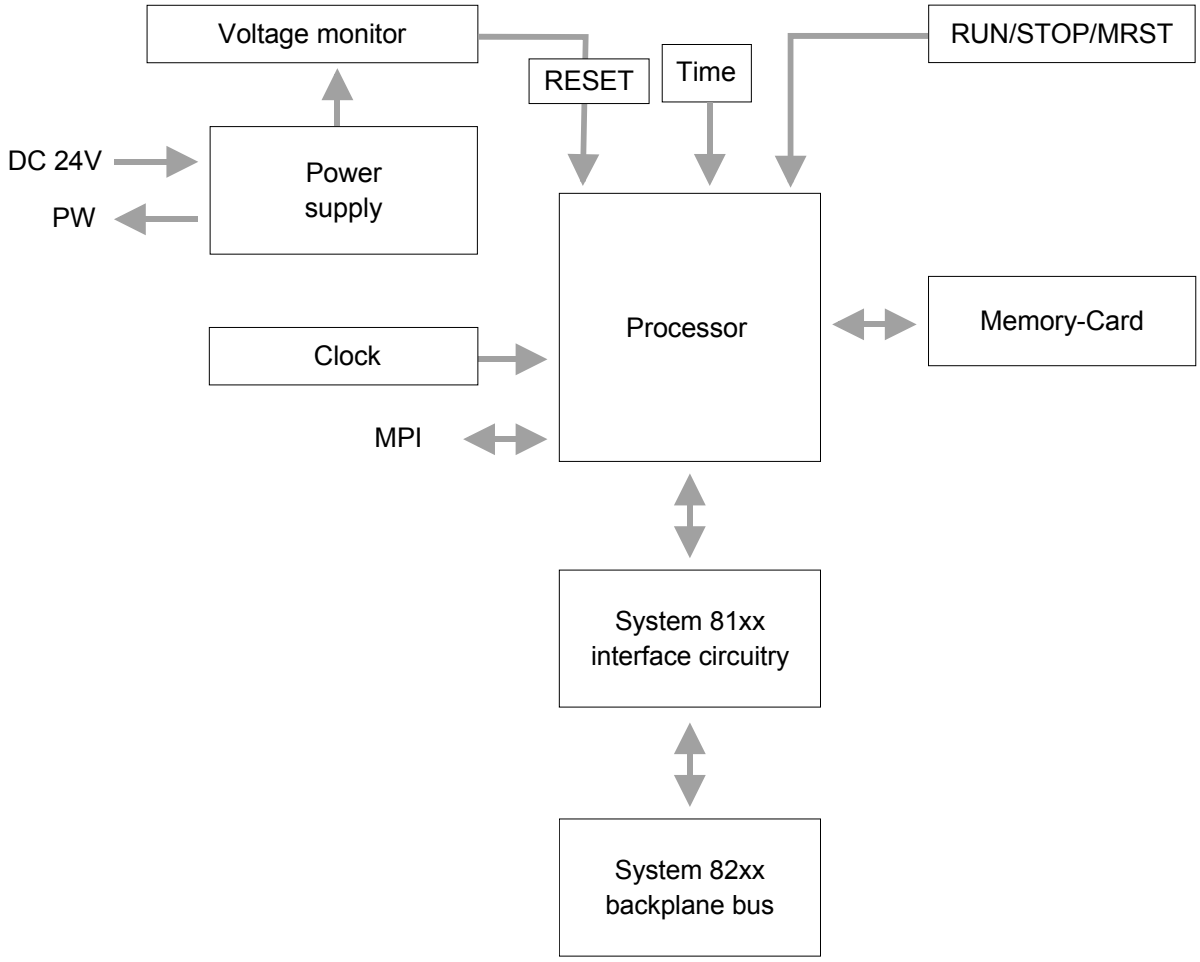
Pin	Assignment
1	Screen
2	not used
3	RxD/TxD-P
4	CNTR-P
5	GND
6	5V (max. 70mA)
7	not used
8	RxD/TxD-N
9	not used

Note!

Refer to the chapter "Deployment of the CPU 821xDPM" for details on the Profibus master.

Block diagram

The following block diagram shows the basic hardware construction of the CPU 821x modules:



Technical data

CPU 821x

General

Electrical data		ADAM-8214-1BA01 ... ADAM-8216-1BA01
Power supply		DC 24V
Current consumption		max. 1.5A
Dissipation power		max. 3.5W
Status indicators (LEDs)		by means of LEDs located on the front
Connections / interfaces		MP^2I-Slot for data communication
Clock, memory/clock backup		yes / Lithium accumulator, 30 days backup
Output current to backplane bus		max. 3A
Bit memory		2048Bit (M0.0…M255.7)
Timer		128 (T0...T127)
Counters		256 (Z0 ... Z255)
Number of Blocks	FB	1024 (FB0...FB1023)
	FC	1024 (FC0...FC1023)
	DB	2047 (DB1...DB2047)
Total addressing space input / output		1024/1024Byte, each 128 Byte for process image (PA)
PA Inputs		1024Bit (E0.0...E127.7)
PA Outputs		1024Bit (A0.0...A127.7)
Combination with peripheral modules		
max. no. of modules		32
max. digital I/O		32
max. analog I/O		16
Addressable inputs, outputs		1024 (digital), 128/128 (analog)
Dimensions and weight		
Dimensions (WxHxD) in mm		25.4x76x76
Weight		80g

Module specific

	CPU 8214	CPU 8215	CPU 8216
Work memory	32kByte	64kByte	128kByte
Load memory	40kByte	80kByte	192kByte
Cycle time bit operations	0.18µs	0.18µs	0.18µs
Cycle time word operations	0.78µs	0.78µs	0.78µs
Order-No.:	ADAM-8214-1BA01	ADAM-8215-1BA01	ADAM-8216-1BA01

CPU 821xNET

Electrical data	ADAM-8214-2BT01 ... ADAM-8216-2BT01
Power supply	via backplane bus
Current consumption	max. 380mA
Potential separation	≥ 500V AC
Status indicator (LEDs)	like CPU 821x additionally with LEDs for the Ethernet section
Connections/interfaces	like CPU 821x additionally with RJ45 socket for Twisted-Pair-Ethernet
Ethernet interface	
Connector	RJ45
Network topology	Star topology
Medium	Twisted pair
Transfer rate	10Mbit
Overall length	max. 100m per segment
Dimensions and weight	
Dimensions (WxHxD) in mm	50.8x76x76
Weight	150g

CPU 821xDPM

Electrical Data	ADAM-8214-2BM01 ... ADAM-8216-2BM01
Power supply	via backplane bus
Current consumption	max. 380mA
Potential separation	≥ 500V AC
Status monitoring (LEDs)	like CPU 821x additionally with LEDs for the Profibus section
Adapters/interfaces	like CPU 821x additionally with 9pin D-type socket (Profibus)
Profibus interface	
Connector	9pin D-type socket
Network topology	Linear bus, active bus termination at both ends
Medium	Screened and drilled twisted pair cable. Depending on environment screening may be omitted.
Transfer rate	9.6kBaud up to 12MBaud
Overall length	100m at 12 MBaud without repeater, up to 1000m with repeater
max. no. of stations	32 stations on every segment without repeater. Expandable to 126 stations with repeater.
Combination with peripheral modules	
max. number of slaves	125
max. number of input bytes	1024
max. number of output bytes	1024
Dimensions and Weight	
Dimensions (WxHxD) in mm	50.8x76x76
Weight	150g

CPU 821xDP

Electrical data	ADAM-8214-2BP01 ... ADAM-8216-2BP01
Power supply	via backplane bus
Current consumption	max. 380mA
Potential separation	≥ 500V AC
Status indicator (LEDs)	like CPU 21x additionally with LEDs for the Profibus section
Adapters/interfaces	like CPU 21x additionally with 9pin D-type socket (Profibus)
Profibus interface	
Connector	9pin D-type socket
Network topology	Linear bus, active bus termination at both ends
Medium	Screened and drilled twisted pair cable. Depending on environment screening may be omitted.
Transfer rate	9.6kBaud up to 12MBaud
Overall length	100m at 12 MBaud without repeater, up to 1000m with repeater
max. no. of stations	32 stations on every segment without repeater. Expandable to 126 stations with repeater.
Dimensions and weight	
Dimensions (WxHxD) in mm	50.8x76x76
Weight	150g

Chapter 3 Deployment of the CPU 821x

Outline

This chapter describes the deployment of the CPU 821x together with the peripheral modules of the System 82xx.

Besides commissioning and start-up behavior you will find here also a description of the project engineering, parameterization, operating modes and test functions.

This information also basically applies to the deployment of a bus resp. NET-CPU (CPU 821xDP, CPU 821xNET…).

More detailed information about the deployment of the bus resp. NET-CPUs is to find in the concerning chapters of this manual.

The following section contains a description of:

- Assembly and commissioning
- Principle of address allocation
- Project engineering and parameterization
- Deployment of MPI and MMC
- Operating modes and overall reset
- Usage of test functions

Contents

Note!

This information is valid for all the CPUs described in this manual since the backplane bus communication between the CPU and the peripheral modules is the same for all models of this CPU!

Start-up behavior

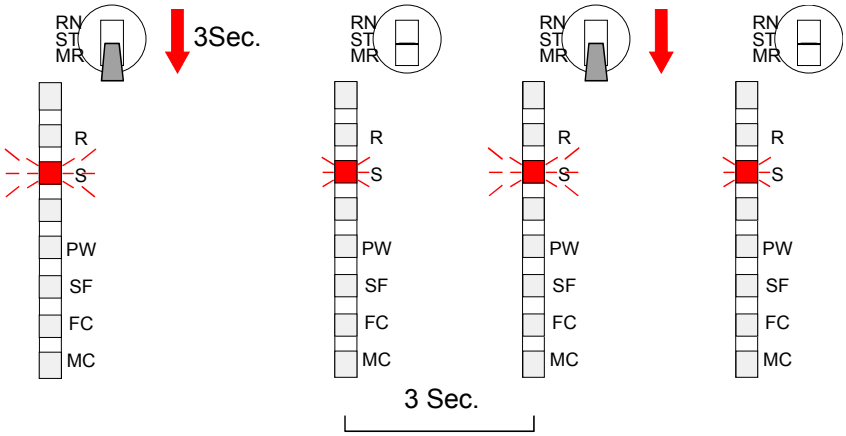
Turn on power supply

After turning on the power supply, the CPU switches to the operating mode that is fixed by the operating mode lever at the CPU.

Now you may transfer your project from your projecting tool into the CPU via MPI resp. plug-in a MMC containing your project and request an OVERALL_RESET.

OVERALL_RESET

The following picture shows the approach:



Note!

The transfer of the user application from the MMC into the CPU takes always place after an OVERALL RESET!

Start-up after delivery

After delivery the CPU is totally clear.

After a STOP→RUN transition, the CPU switches to RUN without application.

Start-up with valid data on the CPU

The CPU switches to RUN with the application located in the battery buffered RAM.

Start-up with empty battery

The battery is loaded directly via the integrated power supply and provides a buffer for up to 30 days. If this time is exceeded, the battery may be totally discharged and the battery buffered RAM is erased.

In this state the CPU starts an overall_reset. If a MMC is plugged in, the program of the MMC is transferred into the RAM.

Depending on the selected operating mode the CPU switches to RUN resp. stays in STOP.

This procedure is fixed in the diagnostic buffer with this entry: "Automatic start OVERALL_RESET (unbuffered POWER-ON)".

Address allocation

Automatic addressing

To provide specific addressing of the installed peripheral modules, certain addresses must be allocated in the CPU.

The CPU contains a peripheral area (addresses 0...1023) and a process image of the inputs and the outputs (for both each address 0...127).

When the CPU is initialized it automatically assigns peripheral addresses to the digital input/output modules starting from 0.

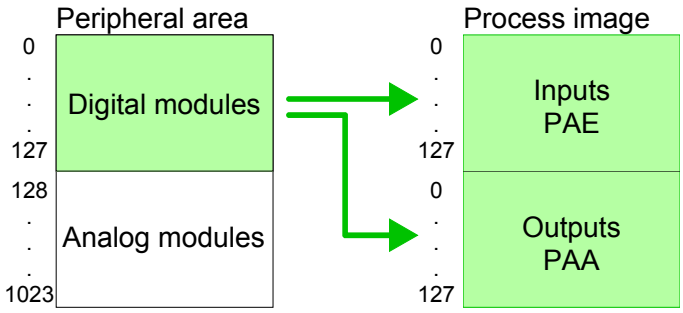
If there is no hardware projecting, analog modules are allocated to even addresses starting from address 256.

Signaling states in the process image

The signaling states of the lower addresses (0...127) are additionally saved in a special memory area called the *process image*.

The process image is divided into two parts:

- process image of the inputs (PAE)
- process image of the outputs (PAA)



The process image is updated automatically when a cycle has been completed.

Read/write access

You may access the modules by means of read or write operations on the peripheral bytes or on the process image.

Note!

Please remember that you may access different modules by means of read and write operations on the same address.

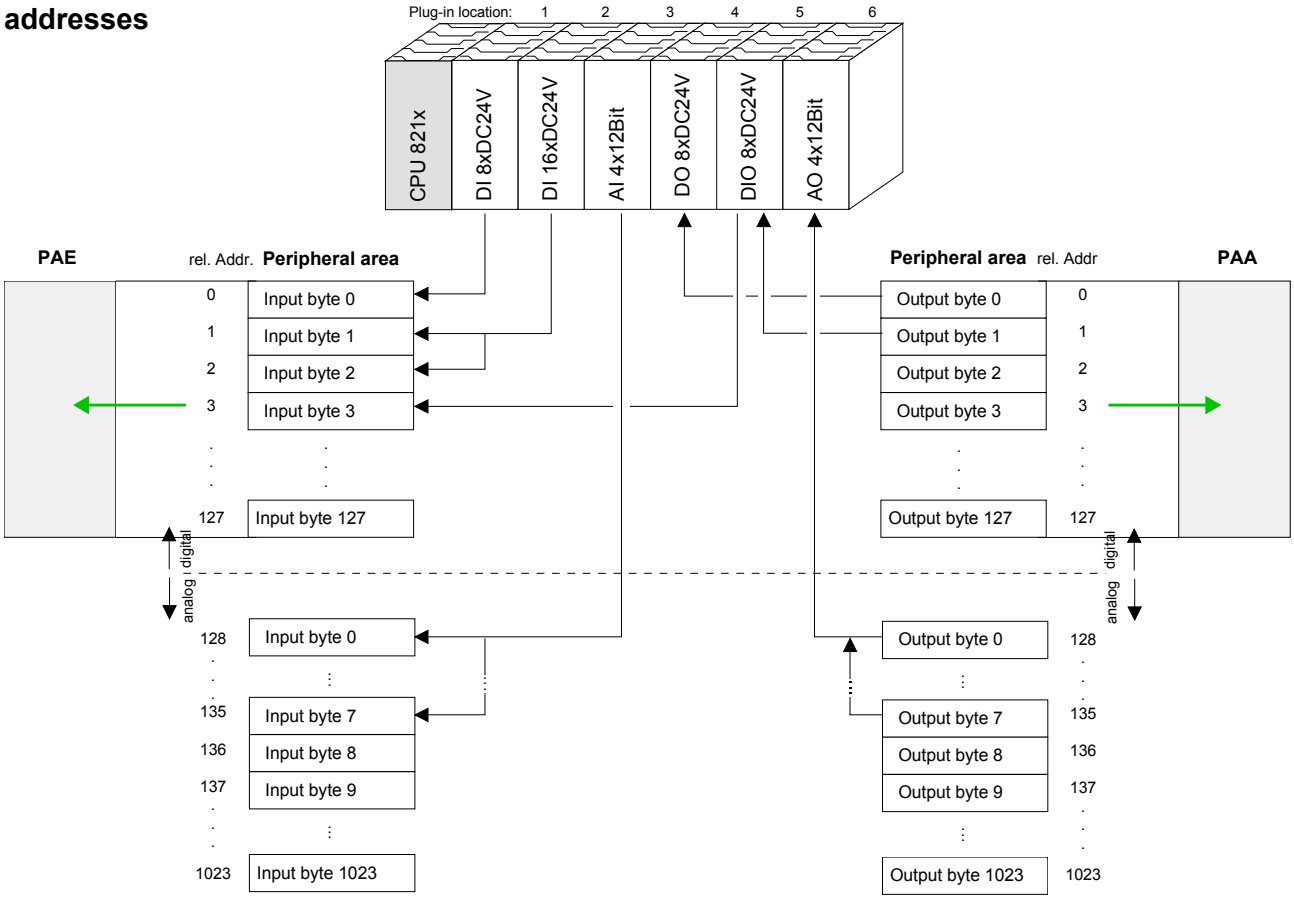
The addressing ranges of digital and analog modules are different when they are addressed automatically.

Digital modules: 0...127

Analog modules: 128...1023

Example for the automatic allocation of addresses

The following figure illustrates the automatic allocation of addresses:



Modifying allocated addresses by configuration

You may change the allocated addresses at any time by means of the Siemens STEP®7 manager. In this way you may also change the addresses of analog modules to the range covered by the process image (0...127) and address digital modules above 127.

The following pages describe the required preparations and the procedure for this type of configuration.

Preparations required for configuration

General

The following information always refers to modules that have been installed on the same bus adjacent to the CPU.

In order to address the installed peripheral modules individually, specific addresses in the CPU have to be assigned to them.

The allocation of addresses and the configuration of the installed modules is a function of the Siemens STEP®7 manager where the modules appear as a virtual Profibus system. For the Profibus interface is software standardized, we are able to guarantee the complete functionality of the System 82xx together with the Siemens STEP®7 manager by including a GSD-file (german: **G**eräte**S**tamm**D**atei = GSD-file).

The project is transferred serial to the CPU via the MPI-interface.

Requirements

The following requirements must be satisfied before starting the configuration of the modules:

- The Siemens STEP®7 manager must have been installed
- The GSD-file must have been included in the Siemens hardware configurator
- A serial connection to the CPU must have been established (e.g. with the "Green Cable" from Advantech)

Note!

The configuration of the CPU requires a thorough knowledge of the Siemens STEP®7 manager and the hardware configurator!

Installation of the Siemens hardware configurator

The hardware configurator is a component of the Siemens STEP®7 project configuration tool. A list of modules that may be configured by this tool can be obtained from the hardware catalog.

Before they are ready for usage, the System 82xx modules have to be included in the hardware catalog by means of the ADAM GSD-file.

Including the GSD- file

- Copy **all ADAM GSD-files** into your GSD-directory ... \siemens\step7\s7data\gsd
- Start the Siemens hardware configurator
- Close all projects
- Go to **Options** > *Install new GSD-file*
- Enter **all gsd files**

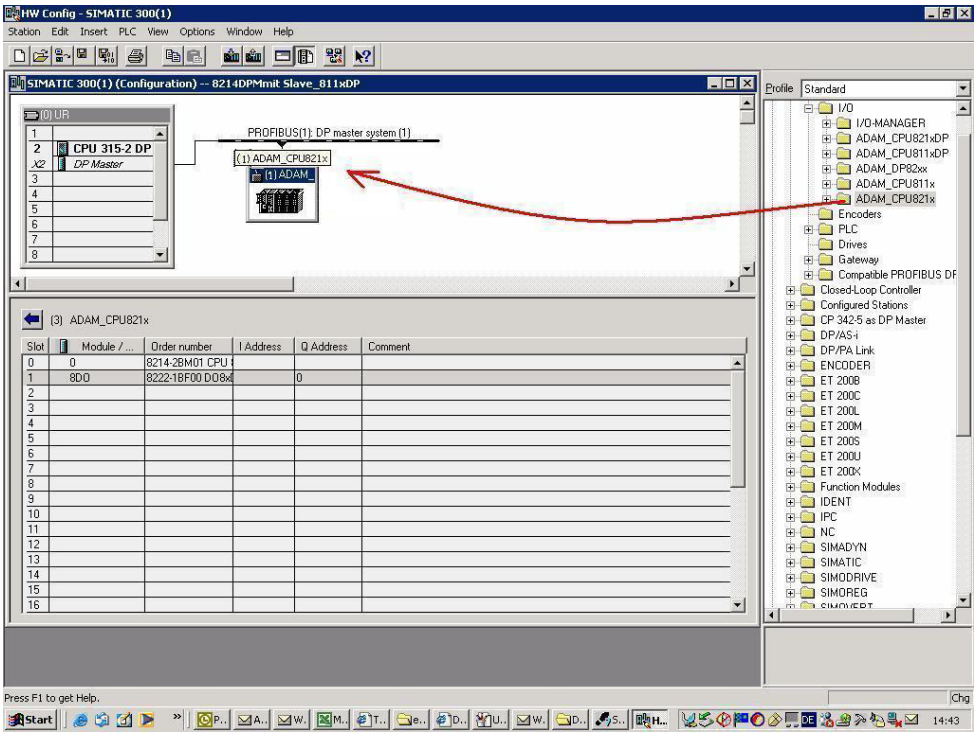
Now the modules of the ADAM System 82xx have been integrated into the hardware catalog and are available for configuration.

Configuration of directly installed System 82xx modules

Configuration as a virtual Profibus master system

The System 82xx modules located adjacent to the CPU on the V bus have to be configured as a virtual Profibus master system by means of the Siemens hardware configurator as described below:

- Create a new project.
- Insert the CPU 315-2DP (6ES7 315-2AF01-0AB0). You have to create a new Profibus subnet.
- Attach the System "ADAM_DP8000" to the subnet. The respective entries are located in the hardware catalog under *PROFIBUS DP* > *Additional Field Devices* > *IO* > *ADAM821x.* Assign Profibus address 1 to this slave.
- Place the ADAM CPU 821x that you want to deploy at plug-in location 0 of the configurator. **Plug-in location 0 is mandatory!**
- Include your System 82xx modules in the location sequence starting from plug-in location 1.
- Save your project.

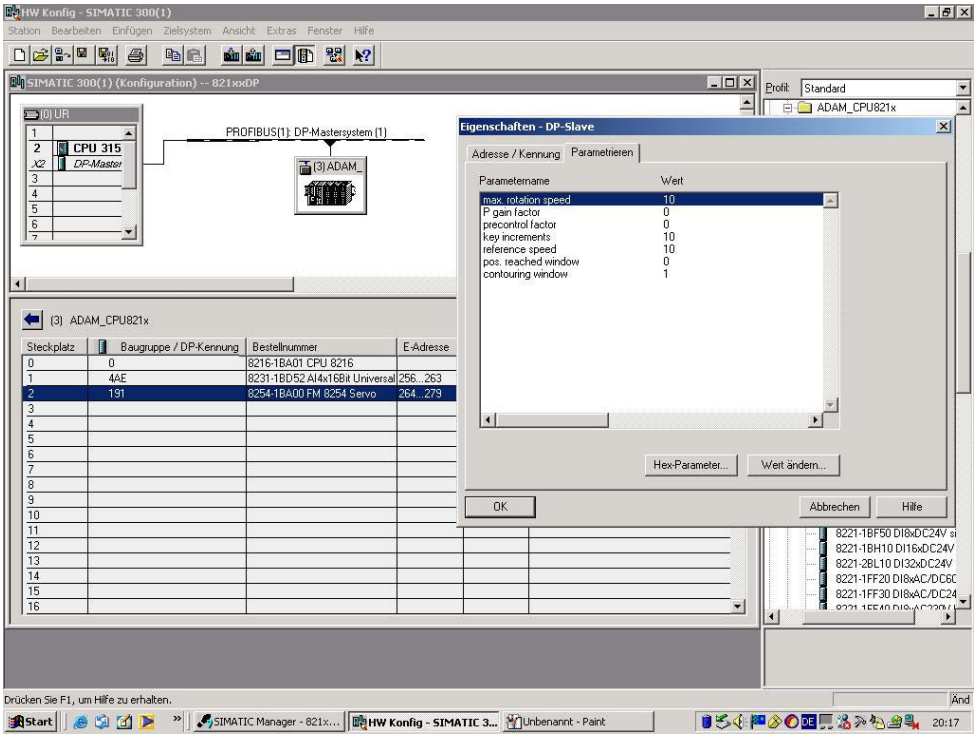


Configuration of modules

System 82xx modules may be supplied with up to 16Byte of configuration data from the CPU. The use of the Siemens STEP®7 manager provide the option to supply parameters to the configurable System 82xx modules at any time.

For this purpose you have to double-click the respective module in the plug-in diagram.

The following figure shows the configuration of the positioning module IM 8254:



Transferring the project

Data is transferred between the CPU and the PC via MPI. If your PU has no MPI functionality you may use the ADAM "Green Cable" to send the data serial by a peer-to-peer connection. The ADAM "Green Cable" has the OrderNo. ADAM-8950-0KB00 and may only be used with ADAM components System 8xxx.)

Please regard the notes about the "Green Cable" in chapter 1!

- Connect your PU to the CPU
- Transfer the project into the CPU by means of **PLC** > *Load into module* in your project configuration tool.
- Install an MMC and transfer the application program to the MMC by clicking on **PLC** > *Copy RAM to ROM*.
- During the write operation the MC-LED of the CPU blinks. For internal reasons the message signalizing completion of the write operation arrives too soon. The write operation is only complete when the LED has been extinguished.

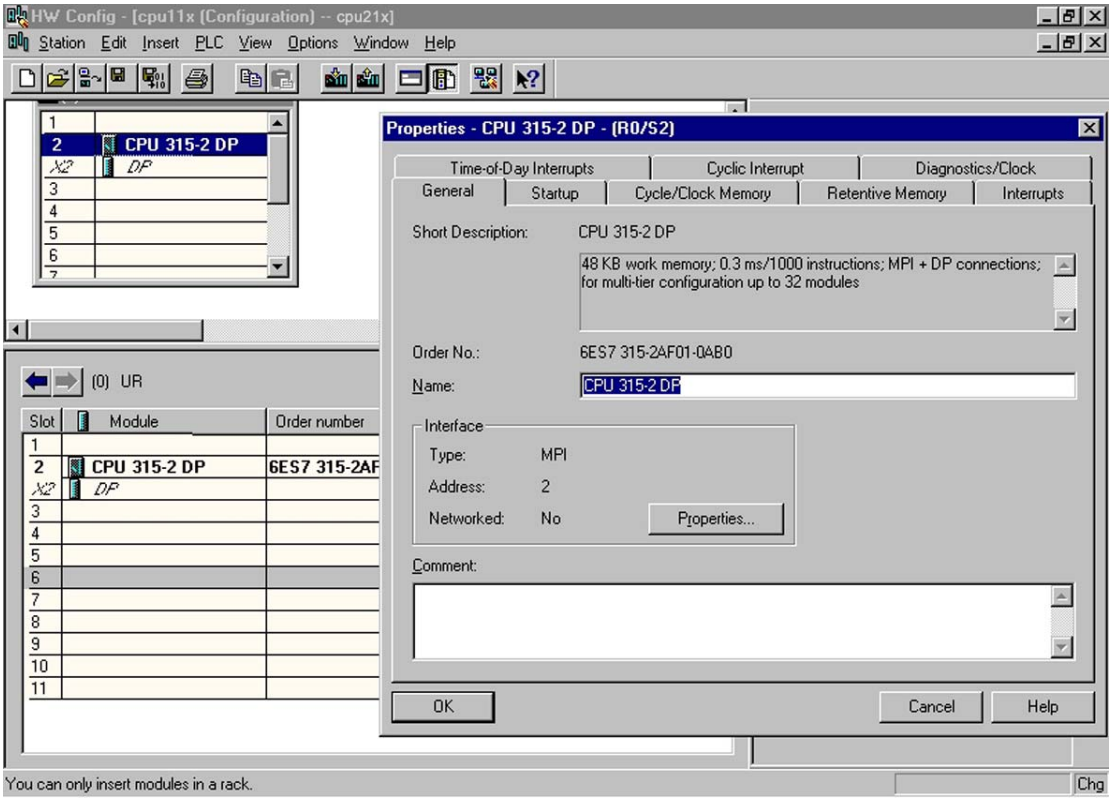
Configuration of the CPU parameters

Outline

The general parameters, concerning the CPU section of your CPU 821x, are to adjust in the hardware configurator from Siemens under the properties of the CPU 315-2DP.

Except of the Profibus parameters of the CPU 821xDP the **CPU parameterization** takes place in the **parameter dialog of the CPU 315-2DP**.

The parameterization of the Profibus section of the CPU 821xDP takes place via the parameterization dialog of the CPU 821x.



Parameterization CPU 821x under CPU 315-2DP

Per double-click on the CPU 315-2DP you reach the parameterization window of your CPU 821x. Via the different registers you may access all parameters of the CPU 315-2DP. The parameters are those of the CPU 821x from Advantech.

Please regard, that at this time not all the parameters are supported.

Parameterization of the Profibus section of CPU 821xDP

You may reach the parameterization window for the Profibus section of the CPU 821x by double-clicking on the added System 821x-CPU under ADAM_DP8000.

Via the different registers you may access all Profibus parameters of the CPU 821x.

More information is to find in the chapter "Deployment of the CPU 821xDP".

Supported parameters

The CPU 821x doesn't use all the parameters that may be defined in the Siemens STEP®7 manager.

The following parameters are currently employed by the CPU:

General:

MPI address of the CPU

maximum MPI address

Initialization:

Start-up, if the planned hardware does not match the actual hardware configuration

Remanence:

Number of bytes of bit memory starting at MB0

Number of S7-timer from T0

No. of S7-counters from Z0

Time alarm :

OB10: active
execution
start date
time-of-day

Prompter :

OB35: execution

Cycle / timing flags:

Cycle watching time

Load placed on cycle time due to communications

Timing flags with flag byte no.

Project transfer

Outline

There are two possibilities to transfer your project into the CPU:

- Transfer via MPI
- Transfer via MMC when using a MMC programmer

Transfer via MPI

The structure of a MPI network is principally the same than the one of a 1.5MBaud Profibus net. This means that the same rules are valid and for both networks you use the same components for building.

Per default the MPI net runs with 187kBaud.

Every bus participant identifies itself at the bus with an unique MPI address.

You link up the single participants via bus connectors and the Profibus bus cable.

Terminating resistor

A line has to be terminated with its ripple resistor. For this you switch on the terminating resistor at the first and the last participant of a network or a segment.

Please take care that those participants with the terminating resistor are supplied with power during start-up and operation.

Approach

- Connect your PU resp. PC with your CPU via MPI.

If your PU doesn't support MPI, you may use the ADAM "Green Cable" to establish a point-to-point connection.

The "Green Cable" has the order number ADAM-8950-0KB00 and may only be used with ADAM CPUs 8xxx.

- Configure the MPI-interface of your PC.
- Via **PLC** > *Load to module* you transfer your project into the CPU.
- If you want to save your project on MMC additionally, plug-in a MMC and transfer your user application via **PLC** > *Copy RAM to ROM*.

During the write process the MC-LED at the CPU is blinking. Due to the system, the completion of the write operation arrives too soon. It is only completed when the LED has been extinguished.

Configure MPI

Hints for configuring a MPI-interface are to find in the documentation of your programming software. At this place only the usage of the "Green Cable" from Advantech shall be shown, together with the programming tool from Siemens.

The "Green Cable" establishes a connection between the COM interface of the PC and the MP^{2}I-interface of the CPU.

Attention!

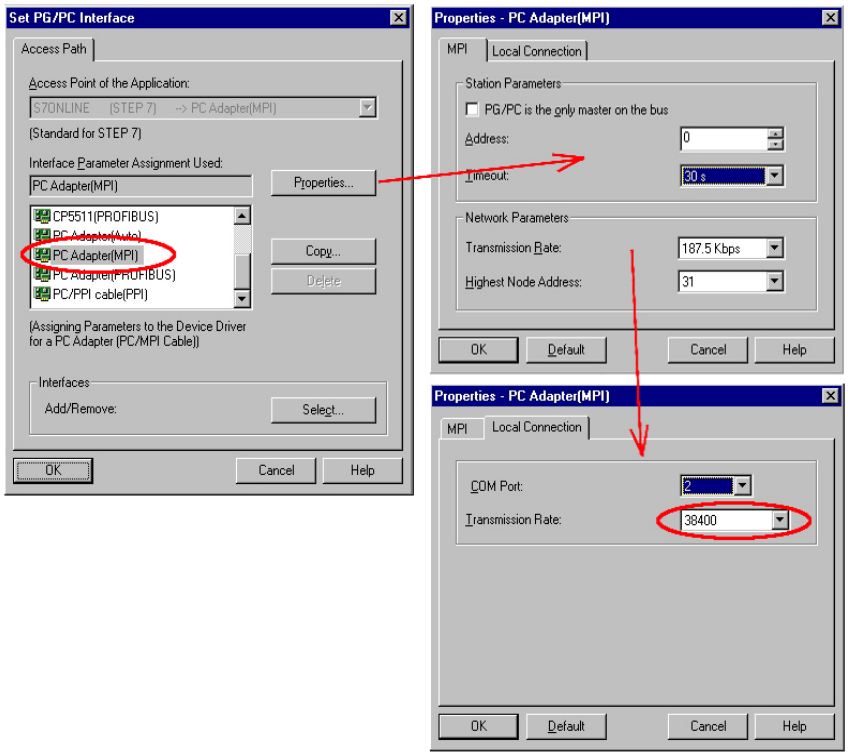
Please regard, that you may use the "Green Cable" exclusively at the MP^{2}I jacks of the Systems 8xxx from Advantech!

Please regard the hints for deploying the Green Cable and the MP^{2}I jack in chapter 1.

Approach

- Start the STEP®7 manager from Siemens
- Choose **Options** > *Configure PU/PC interface*

  → The following dialog window appears, where you may configure the used MPI-interface:



- Choose " PC Adapter (MPI) ", possibly you have to add this first.
- Click on < Properties >

  → In the following two dialog windows you may configure your PC adapter like shown in the picture.

Note!

Please make sure to adjust the transfer rate to 38400Baud when using the "Green Cable".

Usage of the MMC

As external storage medium the **M**ulti **M**edia **C**ard (MMC) is used (Order-No.: ADAM-8953-0KX00).

The reading of the MMC takes always place after an OVERALL RESET.

You may write on the MMC via a write command from the hardware configurator from Siemens or with a MMC reading device from Advantech (OrderNo.: ADAM-8950-0AD00). Thus it is possible to create your applications at the PC, copy them on MMC and transfer them into the CPU by plugging-in the MMC.

The MMC modules are delivered preformatted with the file system FAT16.

Needed files

There may exist several projects and subfolders on one MMC module. You just have to take care, that the recent project is stored in the root directory and has the name: **S7PROG.WLD**.

Transfer CPU → MMC

When the MMC has been installed, the write command stores the application program of the battery buffered RAM at the MMC.

The write command is controlled by means of the Siemens hardware configurator via **PLC** > *Copy RAM to ROM*.

During the write operation the yellow "MC"-LED of the CPU is blinking.

Transfer MMC → CPU

The transfer of the user application from the MMC to the CPU always takes place after an OVERALL_RESET. The blinking of the yellow "MC"-LED indicates the active transfer process.

If there is no valid user application on the MMC or if the transfer fails, an OVERALL_RESET of the CPU takes place and the "STOP"-LED blinks three times.

Note!

If the user application exceeds the user memory of the CPU, the content of the MMC is not transferred into the CPU.

If you initiate a write command and there is no MMC plugged in, an error message about insufficient memory occurs.

It is advisable to compress the application program before transferring it into the CPU, because this is not initialized automatically.

Operating modes

Outline

The CPU can be in one of 3 operating modes:

- STOP
- START-UP
- RUN

Certain conditions in the operating modes START-UP and RUN require a specific reaction from the system program. In this case the application interface is often provided by a call to an organization block that was included specifically for this event.

Operating mode STOP

- Processing of the application program has stopped.
- If the program was being processed, the values of counters, timers, flags and the contents of the process image are retained during the transition to the STOP mode.
- Outputs are inhibited, i.e. all digital outputs are disabled.
- RUN-LED off
- STOP-LED on

Operating mode START-UP

- During the transition from STOP to RUN a call is issued to the start-up organization block OB100. The length of this OB is not limited. The processing time for this OB is not monitored. The start-up OB may issue calls to other blocks.
- All digital outputs are disabled during the start-up, i.e. outputs are inhibited.
- RUN-LED blinks
- STOP-LED off

When the CPU has completed the start-up OB, it assumes the operating mode RUN.

Operating mode RUN

- The application program in OB1 is processed in a cycle. Under the control of alarms other program sections can be included in the cycle.
- All timer and counters being started by the program are active and the process image is updated with every cycle.
- The BASP-signal (outputs inhibited) is deactivated, i.e. all digital outputs are enabled.
- RUN-LED on
- STOP-LED off

Overall Reset

Outline

During the OVERALL_RESET the entire user memory (RAM) is erased. Data located in the memory card is not affected.

You have 2 options to initiate an OVERALL RESET:

- initiate the overall reset by means of the function selector switch
- initiate the overall reset by means of the Siemens STEP®7 manager

Note!

You should always issue an overall reset to your CPU before loading an application program into your CPU to ensure that all blocks have been cleared from the CPU.

Overall reset by means of the function selector

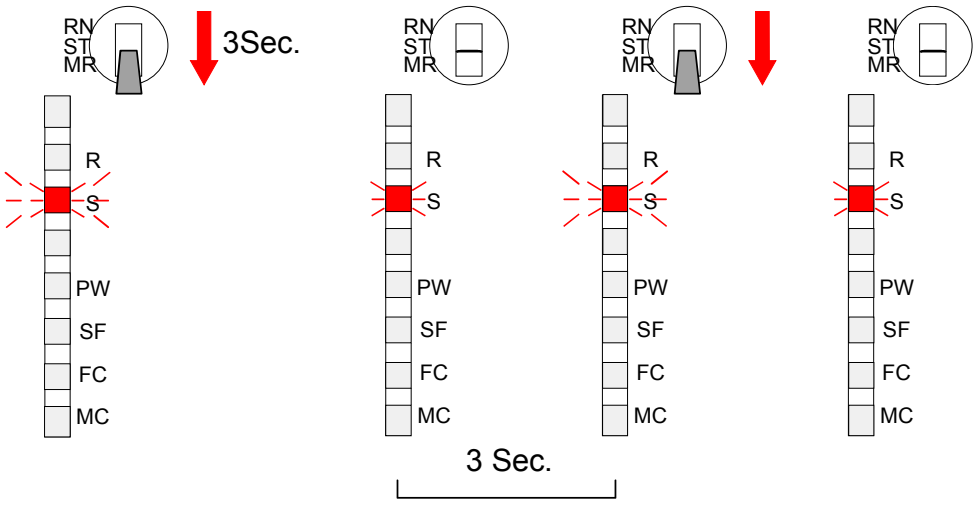
Condition

The operating mode of the CPU is STOP. Place the function selector on the CPU in position "ST" → the S-LED is on.

Overall reset

- Place the function selector in the position MR and hold it in this position for app. 3 seconds. → The S-LED changes from blinking to permanently on.
- Place the function selector in the position ST and switch it to MR and quickly back to ST within a period of less than 3 seconds.
  → The S-LED blinks (overall reset procedure).
- The overall reset has been completed when the S-LED is on permanently. → The S-LED is on.

The following figure illustrates the above procedure:



Automatic reload

At this point the CPU attempts to reload the parameters and the program from the memory card. → The lower LED (MC) blinks.

When the reload has been completed the LED is extinguished. The operating mode of the CPU will be STOP or RUN, depending on the position of the function selector.

Overall reset by means of the Siemens STEP®7 Manager

Condition

The operating mode of the CPU must be STOP.
You may place the CPU in STOP mode by the menu command **PLC** > *Operating mode*.

Overall reset

You may request the OVERALL_RESET by means of the menu command **PLC** > *Clear/Reset*.

In the dialog window you may place your CPU in STOP mode if this has not been done as yet and start the overall reset.

The S-LED blinks during the overall reset procedure.

When the S-LED is on permanently the overall reset procedure has been completed.

Automatic reload

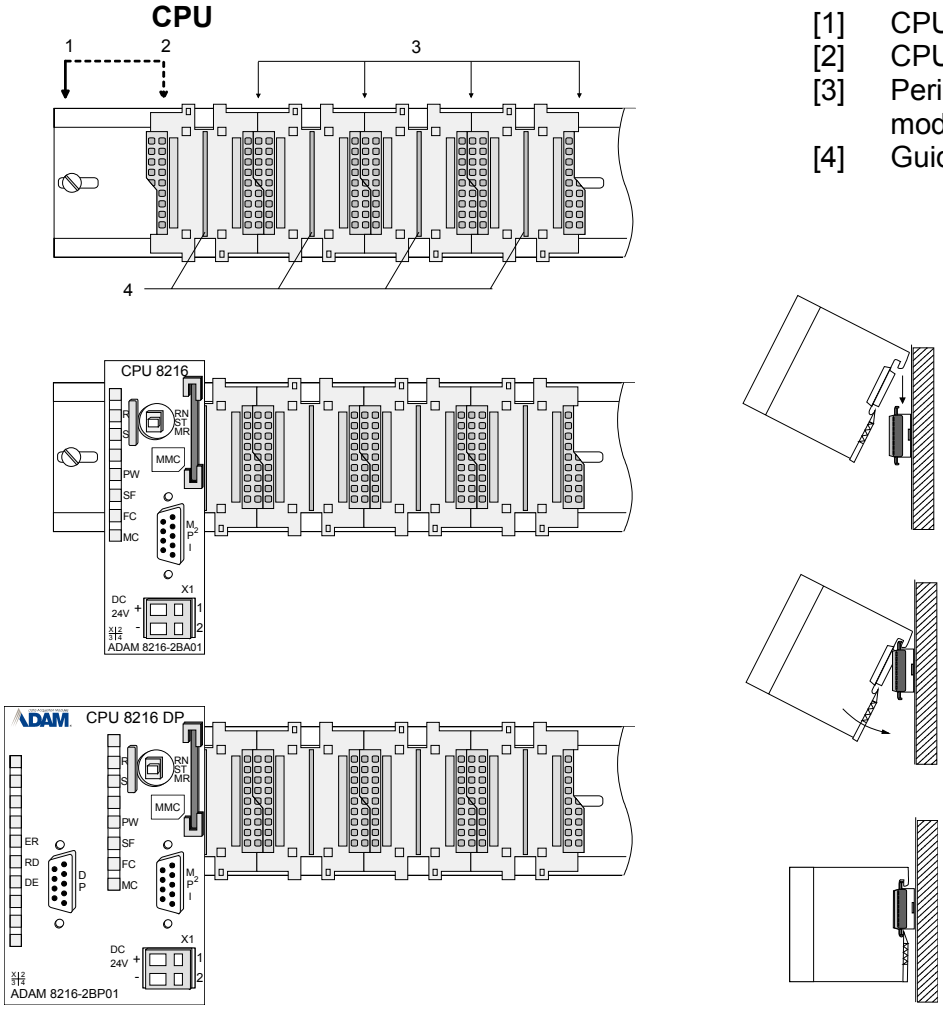
At this point the CPU attempts to reload the parameters and the program from the memory card. → The lower LED (without label) blinks.

When the reload has been completed, the LED is extinguished. The operating mode of the CPU will be STOP or RUN, depending on the position of the function selector.

Assembly

Attention!
It is mandatory to turn off the power supply before you insert or remove any modules!

Please note that the CPU may only be installed into plug-in location 1 or 2 (see figure below).



[1] CPU, double width
[2] CPU, single width
[3] Peripheral modules
[4] Guiding bars

For details on the assembly of System 82xx modules please refer to the System 82xx manual (Order-No.: ADAM-HB97).

Recalling version and performance information

Outline

As soon as you are online with your CPU 821x you have the possibility to recall the module state.

You may find the module state in the STEP®7 manager from Siemens under the **PLC** functions.

Approach

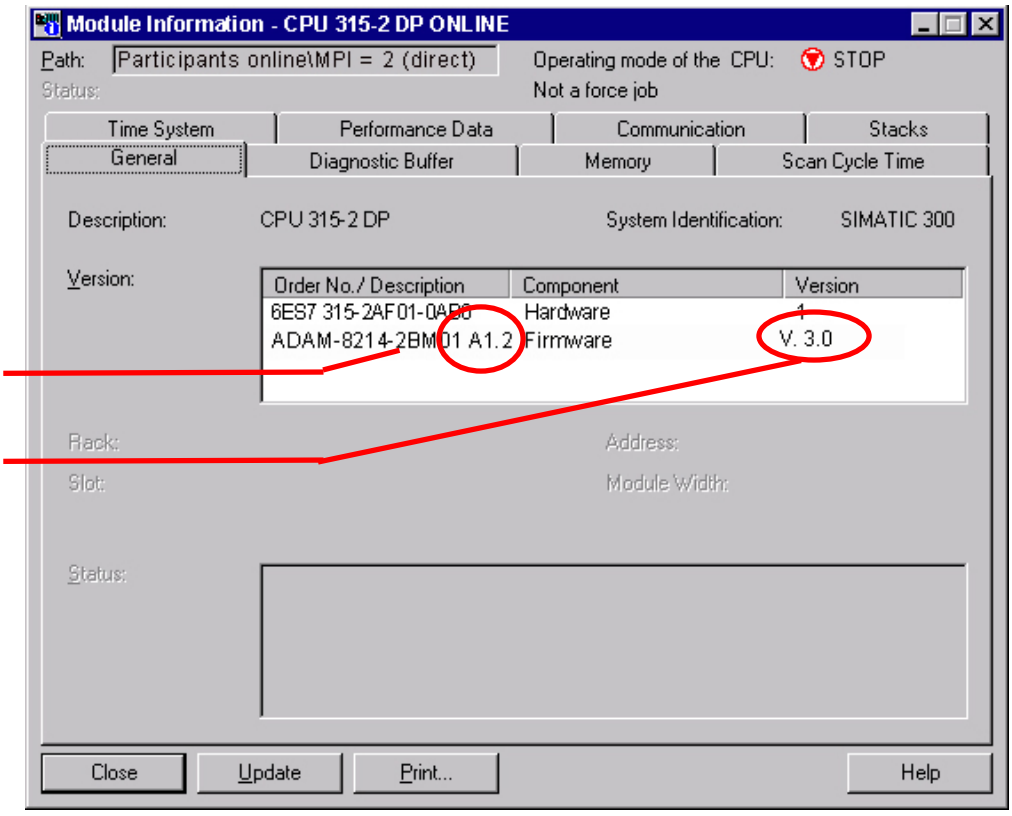
Open your project. Click on the system you want to examine.

Now open the dialog window about the module state via **PLC** > *Module information*.

Besides the possibility to recall numerous module states and performance information, you may find in the register "General" facts about the hardware version (also part releases) and the version of the firmware:

Hardware release

Firmware release



Note!

More detailed information to the deployment of the PLC functions are to find in the manual to the STEP®7 manager from Siemens.

Using test functions for the control and monitoring of variables

Outline

For troubleshooting purposes and to display the status of certain variables you may access certain test functions via the menu item **Test** of the Siemens STEP®7 manager.

The status of the operands and the VKE are displayed by means of the test function **Test** > *Status*.

You are able to modify and/or display the status of variables by means of the test function **PLC** > *Monitor/control variables*.

Test > *Status*

This test function displays the current signal state and the VKE of the different operands while the program is being executed.

It is also possible to enter corrections to the program.

Note!

When using the test function *Monitoring* the PLC has to be in RUN mode!

The processing of states may be interrupted by means of jump commands or by timer and process-related alarms. At the breakpoint the CPU stops collecting data for the status display and instead of the required data it only provides the PU with data containing the value 0.

For this reason jumps or time and process alarms may result in the value displayed during program execution remaining at 0 for the items below:

- the result of the logical operation VKE
- Status / AKKU 1
- AKKU 2
- Condition byte
- absolute memory address SAZ. In this case SAZ is followed by a "?".

The interruption of the processing of statuses does not change the execution of the program but it only shows that the data displayed is no longer valid after from the point where the interrupt occurred.

PLC >
Monitor/control variables

This test function returns the condition of a selected operand (inputs, outputs, flags, data word, counters or timer) at the end of program execution.

This information is obtained from the process image of the selected operands. During the "processing check" or in operating mode STOP the periphery is read directly from the inputs. Otherwise only the process image of the selected operands is displayed.

Control of outputs

It is possible to check the wiring and proper operation of output modules.

You may set outputs to any desired status with or without a control program. The process image is not modified but outputs are no longer inhibited.

Control of variables

The following variables may be modified:

E, A, M, T, Z, and D.

The process image of binary and digital operands is modified independently of the operating mode of the CPU 821x.

When the operating mode is RUN the program is executed with the modified process variable. When the program continues they may, however, be modified again without notification.

Process variables are controlled asynchronously with respect to the execution sequence of the program.

Chapter 4 Deployment of the CPU 821xNET

Outline

The following chapter describes applications of the CPU 821xNET and the H1 resp. TCP/IP communication procedure. It also contains an introduction to the configuration of the module by means of WinNCS along with a real-world example.

The chapter contains a description of:

- the principles of the twisted-pair network
- configuration by means of WinNCS along with an example
- a test program for TCP/IP connections

Contents

Principles

Network

A network provides a link between different stations that enables them to communicate with each other.

Network stations consist of PCs, IPCs, H1/TCP/IP adapters, etc.

Network stations are separated by a minimum distance and connected by means of a network cable. The combination of network stations and the network cable represent a complete segment.

All the segments of a network form the Ethernet (physics of a network).

Twisted Pair

In the early days of networking the Triaxial- (yellow cable) or thin Ethernet cable (Cheapernet) was used as communication medium. This has been superseded by the twisted-pair network cable due to its immunity to interference. The CPU 821xNET module has a twisted-pair connector.

The twisted-pair cable consists of 4 cores that are twisted together in pairs. Due to these twists this system is provides an increased level of immunity to electrical interference.

Where the coaxial Ethernet networks are based on a bus topology the twisted-pair network is based on a point-to-point scheme.

The network that may be established by means of this cable has a star topology. Every station is connected to the star coupler (hub/switch) by means of a separate cable. The hub/switch provides the interface to the Ethernet.

Star coupler (Hub)

The hub is the central element that is required to implement a twisted-pair Ethernet network.

It is the job of the hub to regenerate and to amplify the signals in both directions. At the same time it must have the facility to detect and process segment wide collisions and to relay this information. The hub is not accessible by means of a separate network address since it is not visible to the stations on the network.

A hub has provisions to interface to Ethernet or to another hub.

Access control

Ethernet supports the principle of random bus accesses: every station on the network accesses the bus independently as and when required. These accesses are coordinated by a CSMA/CD (Carrier Sense Multiple Access/Collision Detection) scheme: every station “listens” on the bus cable and receives communication messages that are addressed to it.

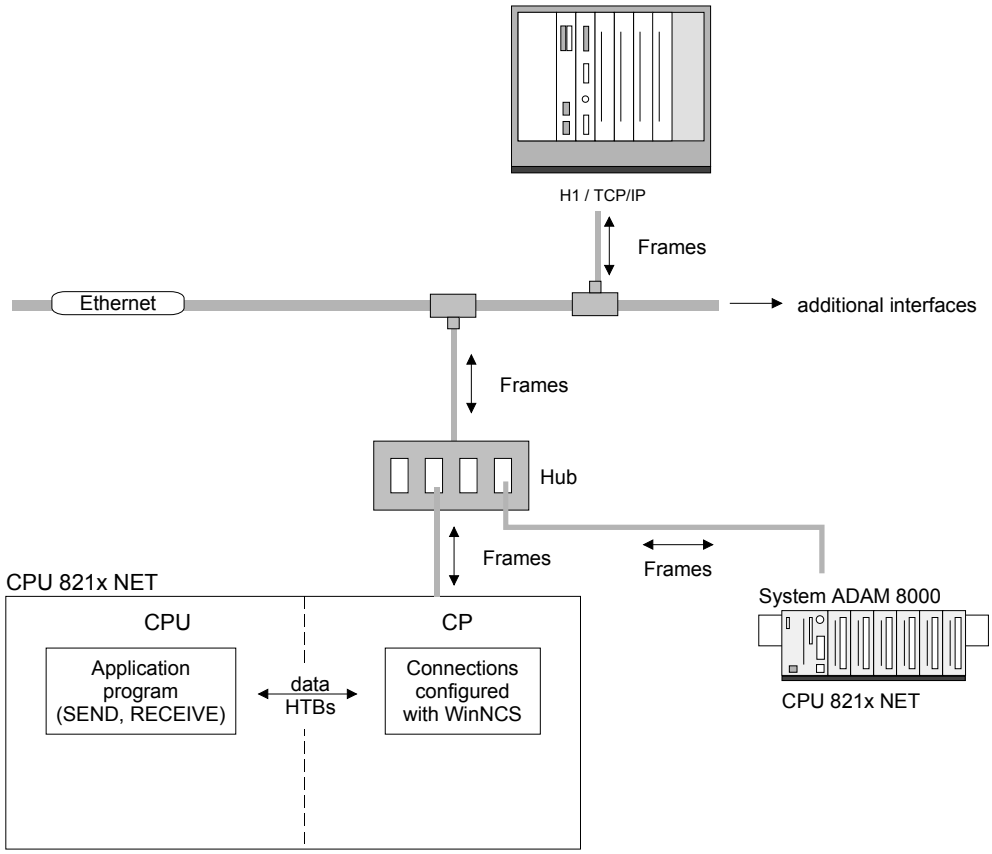
Stations will only initiate a transmission when the line is unoccupied. In the event that two participants should start transmitting simultaneously, they will detect this and stop transmitting to restart after a random delay time has expired.

Communication

The internal CP of the CPU 821xNET is directly connected to the CPU 821x by means of a Dual-Port-RAM. The Dual-Port-RAM is divided into 4 equal segments called page frames.

These 4 page frames are available at the CPU as standard CP interface. Data is exchanged by means of standard handler blocks (SEND and RECEIVE).

H1 and TCP/IP communication is controlled by means of connections that are defined with the ADAM configuration tool WinNCS and are directly transferred into the CPU via the twisted-pair connection.



H1 - Industrial Ethernet

H1, which is also referred to as "Industrial Ethernet", is a protocol that is based upon the Ethernet standard.

H1 information is exchanged between stations by means of H1-frames that are transferred via transport connections.

A transport connection is a logical link between two access points to the transport services on different stations. A transport connection is based upon addressing information that provides an exact description of the path between the two access points.

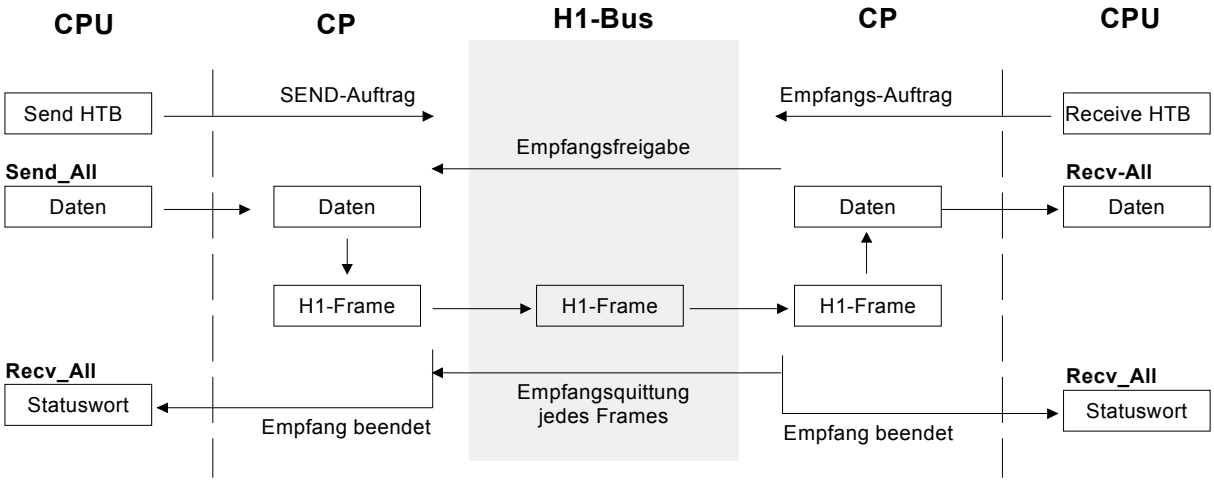
The following parameters describe a transport connection individually:

- *MAC address,* also referred to as Ethernet address, gives an unique definition of the access facility of a station.
- *TSAP Transport Service Access Points* identify access channels for the services of the transport protocol.

The CP prepares a data buffer and transfers the data into the data buffer by means of the background communication function SEND_ALL. Then the CP creates an H1-frame and transfers this to the partner station as soon as this has enabled reception. When the partner station has received the H1-frames, the CP receives an acknowledgment receipt. Next it uses the background communication function RECEIVE_ALL to transfer the status of the SEND-job into the respective indicator word.

This ensures error-free communications.

The following illustration shows the principle once again:



Note!

Due to the large quantity of acknowledgment receipts that are transferred via an H1 transport connection, the load on the network is appreciably higher under H1 than under TCP/IP, however, under TCP/IP the security of the data is reduced!

TCP/IP

TCP/IP protocols are available on all major systems. At the bottom end this applies to simple PCs, through to the typical mini computer up to main-frames (TCP/IP implementations also exist for IBM systems) and special processors like vector processors and parallel computers. For this reason TCP/IP is often used to assemble heterogeneous system pools.

TCP/IP can be employed to establish extensive open network solutions between the different business units of an enterprise.

For example, TCP/IP may be used for the following applications:

- centralized control and supervision of production plants,
- transfer of the state of production machines,
- management information,
- production statistics,
- the transfer of large quantities of data.

TCP and IP only provide support for two of the protocols required for a complete architecture. Programs like "FTP" and "Telnet" are available for the application layer of the PC.

The application layer of the CPU 821xNET-CP is defined by the application program using the standard handler blocks.

These application programs exchange data by means of the TCP or UDP protocols of the transportation layer. These themselves communicate with the IP protocol of the Internet layer.

IP

The main purpose of IP is to provide the addressing of data packets. This means that IP has the same function as an envelope has for a letter. The address is used by the network to determine the destination and to route the data packets accordingly.

The protocol divides the data into small portions since different networks use differently sized data packets.

A number is assigned to each packet. This is used to acknowledge reception and to reassemble the original data. To transfer these sequence numbers via the network TCP and IP is provided with a unique envelope where these numbers are recorded.

TCP

A packet of data is inserted into a TCP envelope. This is then inserted into an IP-envelope and transferred to the network. TCP provides for the secure transfer of data via network. TCP detects and corrects communication errors.

In this way TCP connections are relatively safe.

UDP provides a much faster communication link. However, it does not cater for missing data packets, nor does it check the sequence of the packets. UDP is an unsecured protocol.

TCP/IP services

OPEN / CONNECT

Opens a virtual connection to communication partner when the station is in active mode and waits for a connection from a communication partner in passive mode.

SEND

Transfers a data buffer to TCP for transmission to a communication partner.

RECEIVE

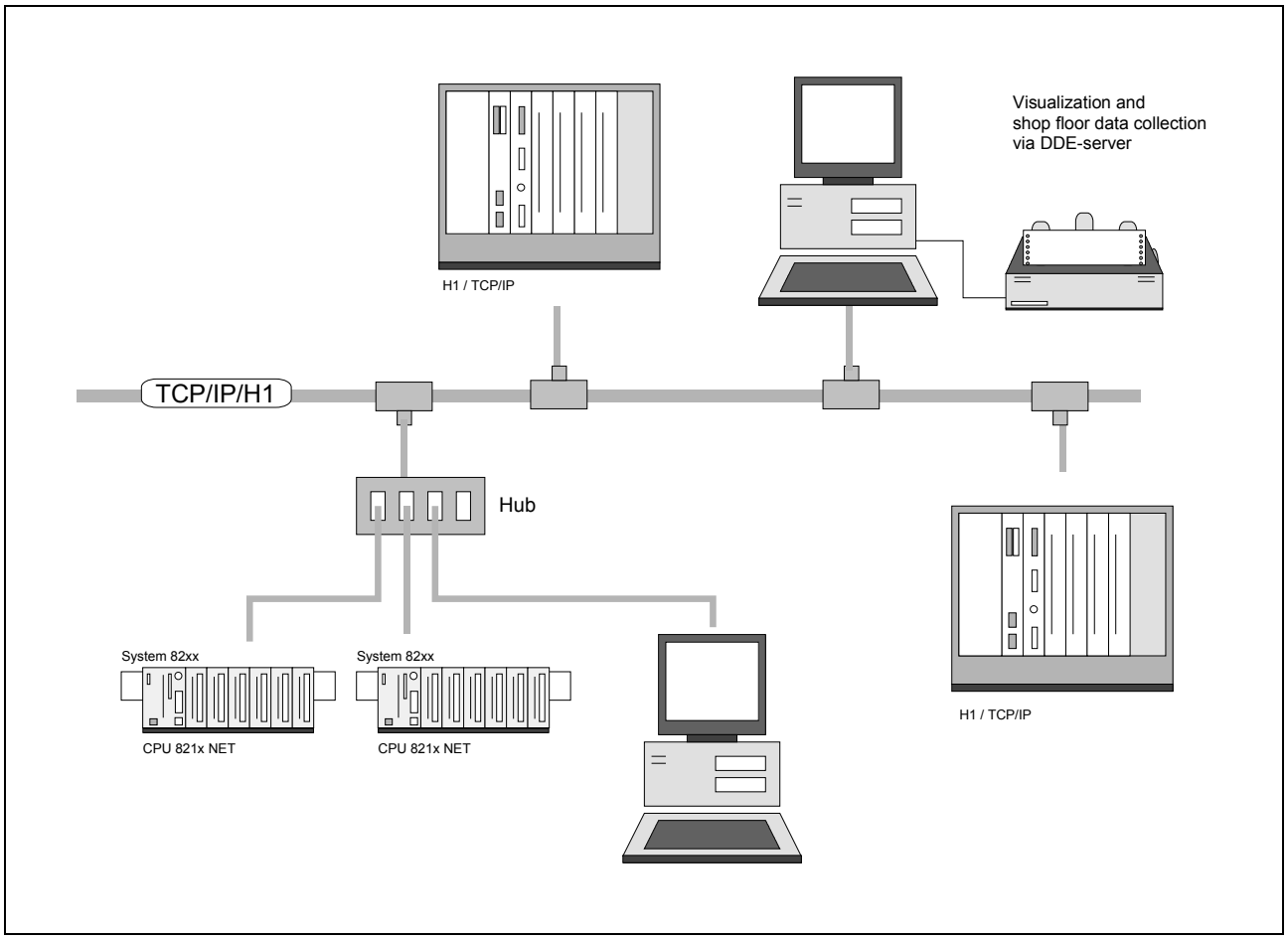
Receives data from a communication partner.

CLOSE

Terminates a virtual connection.

Example for H1 or TCP/IP application

Deployment under TCP/IP or H1

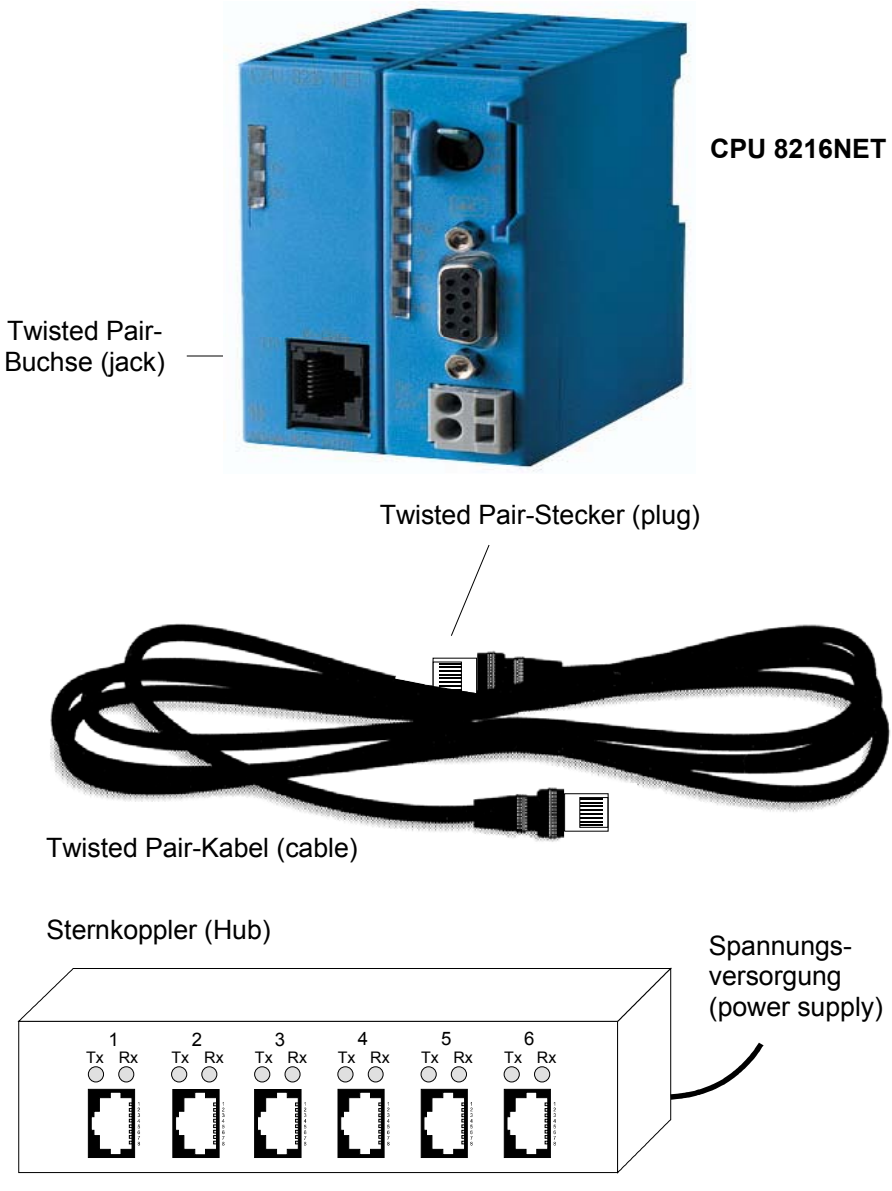


Network planning

Twisted-pair network hardware

A twisted-pair network can only be constructed as star topology. This requires a hub to connect the different stations.

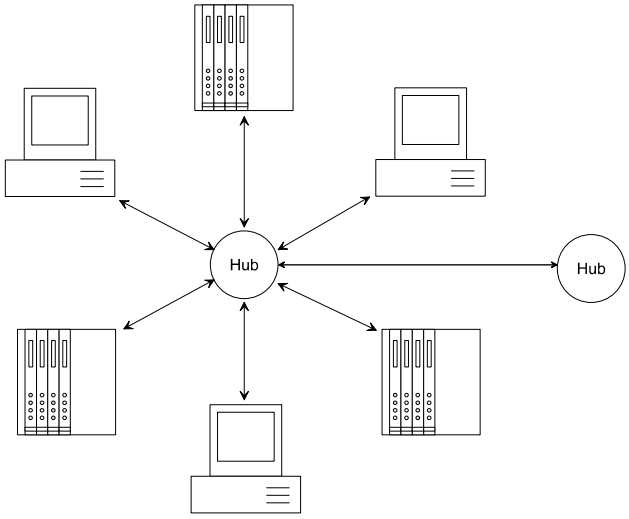
A twisted-pair cable has four cores, twisted together in pairs. The different cores have a diameter of 0.4 to 0.6 mm.



Restrictions

There are a few restrictions and rules that have to be taken into account using twisted-pair:

- Maximum number of hub elements per segment 2
- Maximum length of one segment 100m



Analyzing the network requirements

- What is the size of the area that has to be served by the network?
- How many network segments provide the best solution for the physical (space, interference related) conditions encountered on site?
- How many network stations (PLCs, IPCs, PCs, transceivers, bridges, if required) shall be connected to the cable?
- What is the distance between the different stations on the network?
- What is the expected “growth rate” and the expected number of connections that shall be catered for by the system?

Drawing a network diagram

Draw a diagram of the network. Identify every hardware item (i.e. station cable, hub). Observe the applicable rules and restrictions.

Measure the distance between all components to ensure that the maximum length is not exceeded.

Standards and norms

The main property of the bus structure is that there is only one single physical connection. The physical communication medium may be:

- one or more electrical cables (drilled cores),
- coaxial cables (triaxial cables),
- fiber optic cables.

The applicable rules and regulations have to be satisfied in order to establish reliable communications between the different stations.

These agreements define the form of the data protocol, the method of bus access and other principles that are important for reliable communications.

The ADAM CPU 821xNET was developed in accordance with the standards defined by ISO.

Standards and guidelines

International and national committees have defined the following standards and guidelines for networking technologies:

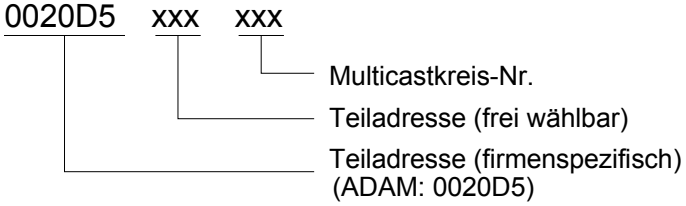
ANSI	American National Standards Institute The ANSI X3T9.5 standard currently defines the provisions for high-speed LANs (100 MB/s) based on fiber optic technology. (FDDI) Fiber Distributed Data Interface.
CCITT	Committee Consultative Internationale de Telephone et Telegraph. Amongst others, this advisory committee has produced the provisions for the connection of industrial networks (MAP) to office networks (TOP) on Wide Area Networks (WAN).
ECMA	European Computer Manufacturers Association. Has produced various MAP and TOP standards.
EIA	Electrical Industries Association (USA) This committee has issued standard definitions like RS-232 (V.24) and RS-511.
IEC	International Electrotechnical Commission. Defines certain special standards, e.g. for the Field Bus.
ISO	International Organization for Standardization. This association of national standards organizations developed the OSI-model (ISO/TC97/SC16). It provides the framework for the standardization of data communications. ISO standards are included in different national standards like for example UL and DIN.
IEEE	Institute of Electrical and Electronic Engineers (USA). The project-group 802 determines LAN-standards for transfer rates of 1 to 20 MB/s. IEEE standards often form the basis for ISO-standards, e.g. IEEE 802.3 = ISO 8802.3.

Ethernet and IP addresses

Structure of an Ethernet address

A station is addressed by means of the Ethernet address, which is also known as the MAC address. The Ethernet addresses of stations in a network must be unique.

Ethernet addresses consist of the following elements:



The Ethernet address has a length of 6Byte. The first 3Byte are defined by the manufacturer. These 3Byte are assigned by the IEEE-Committee. The last 3Byte may be defined as required.

The network administrator determines the Ethernet address.

The broadcast address (to transmit messages to all stations on the network) always is:

FFFFFFFFFFFFh

IP address structure

The IP address is a 32Bit address that must be unique within the network. The IP address consists of 4 numbers that are separated by a dot.

The structure of an IP address is as follows: **XXX.XXX.XXX.XXX**

Range: 000.000.000.000 to 255.255.255.255

The network administrator also defines IP addresses.

The broadcast address (transmit a message to all stations) is always:

255.255.255.255

Attention!

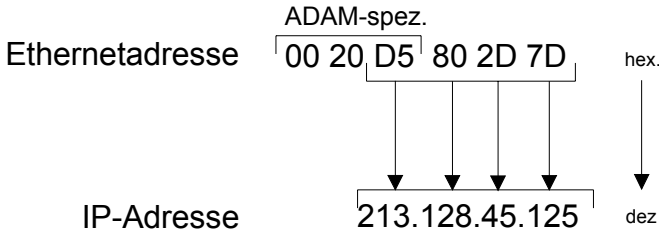
Certain IP addresses are restricted! These addresses are reserved for special services!

Initial address

When the CPU 824xNET is first turned on the module has a predefined initial Ethernet address.

This address is available from the label that has been attached to the side of the module.

This Ethernet address is only used when the module is first turned on, to calculate a unique IP address according to the following formula:



Note!

A relationship between the Ethernet address and the IP address only exists when the module is turned on for the first time.

You may always use the function CP-Init in WinNCS to assign a different address.

Attention!

The original Ethernet address may not be restored since it is not possible to perform an overall reset of the CP section.

Configuration of the CPU 821xNET

Outline

The configuration procedure for the CP section is identical to that of the CP 143 H1/TCP/IP ADAM module.

The configuration procedure for H1 and TCP/IP consists of two parts:

- **CP configuration** by means of WinNCS of Advantech (Ethernet connection).
- **PLC programming** by means of an application program (PLC connect).

CP-configuration with WinNCS

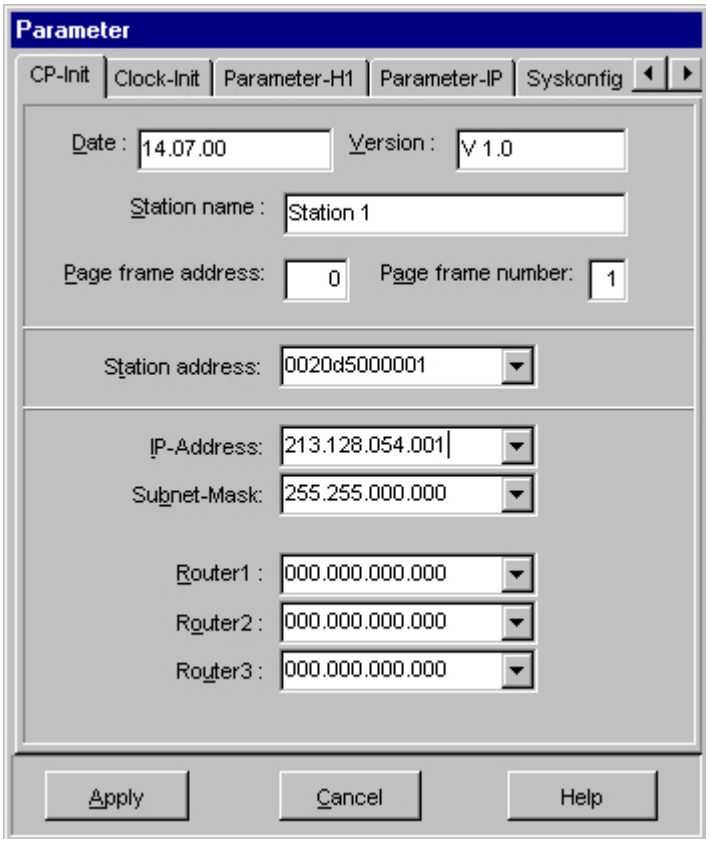
The CP section of the CPU 821xNET may only be configured with WinNCS from Advantech and consists of the following 3 parts:

- The initial CP configuration,
- configuration of connection modules,
- transfer configuration data into the CP.

Initial CP configuration

This is where the address and other identification parameters of a station are defined.

Under "Ethernet" you insert a new station into the network window and enter the configuration data for your station into the parameter window.



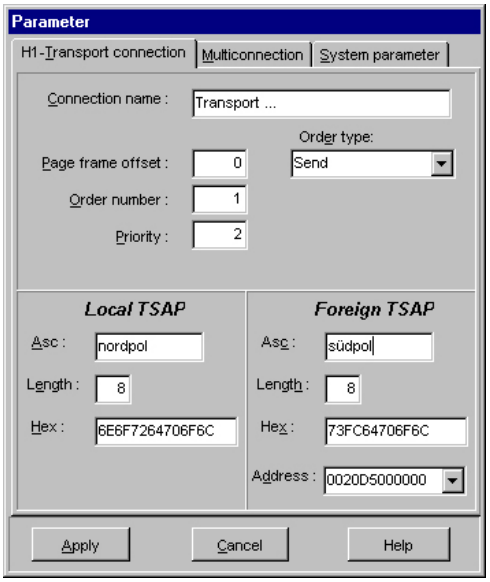
The basic CP configuration determines the behavior of your station on the network.

Configuration of a connection block

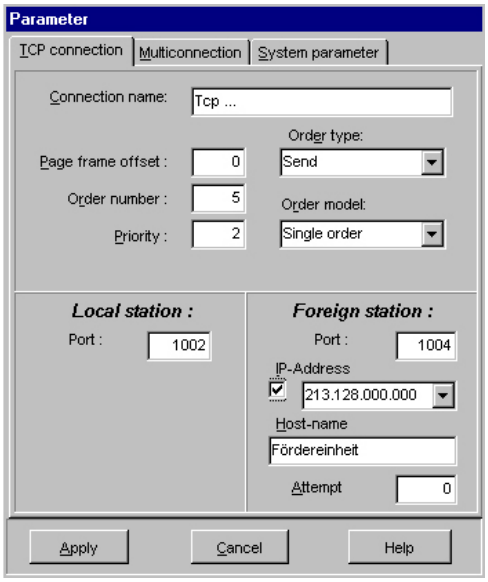
A connection block contains the remote parameters, i.e. parameters that are oriented towards the partner on the network, and local parameters, i.e. parameters that apply to the PLC program of a connection.

Depending on the selected protocol you may configure H1 or TCP/IP connections by selecting the symbol of the station and inserting/configuring the respective connection.

H1-Connection



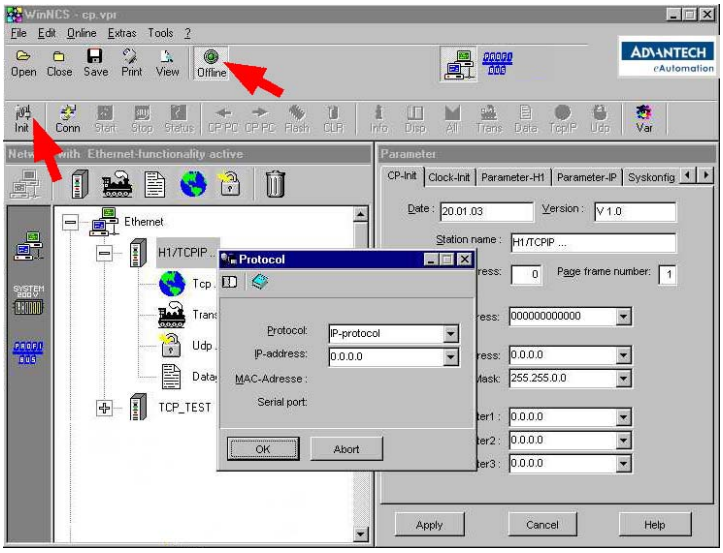
TCP/IP- Connection



Transferring the configuration to the CP

When all the required connections have been configured, you have to transfer the parameters to the CP. This operation is available from the "Module transfer functions" of WinNCS.

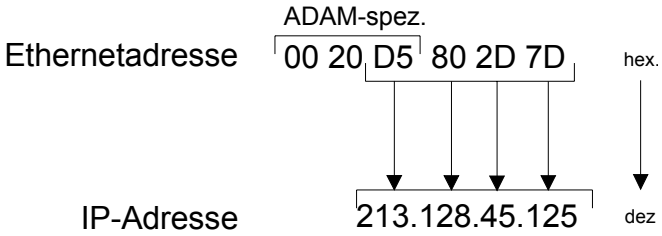
For transferring the configuration data, please activate the „online functions“ and click on the button INIT:



Like described before the CPU 821xNET is delivered with a predefined Ethernet address.

You will find this predefined address on the label at the side of the module.

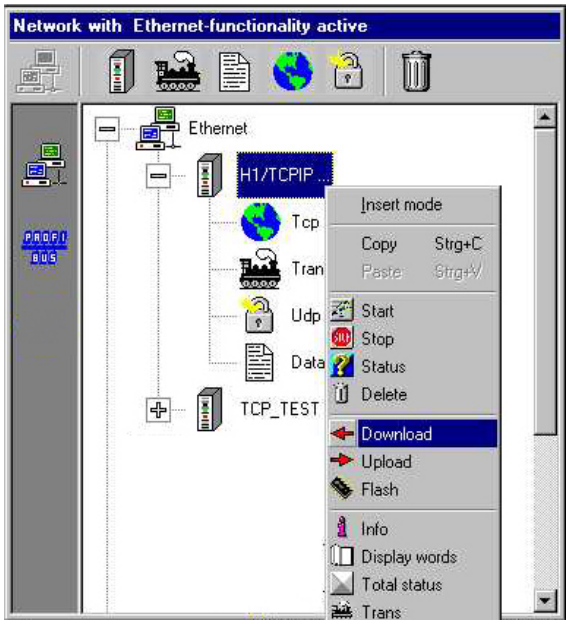
When using the module for the first time this default address is changed into an individual IP address following this rules:



Choose „IP protocol“ in the *Protocol*-window and type the determined IP address. Confirm your entry with [OK].

Now switch to the window *network* and click the according station. Use the right mouse button and choose „Download“.

Your project will now be directly transferred into the Flash-ROM of your CP.



PLC application programming

To enable the PLC to process connect requests, it requires an active PLC application program on the CPU. This uses the handler blocks (SEND, RECEIVE, ...) that are included in the CPU 821xNET amongst others.

The PLC-program also requires that a communication channel be specified first between the CPU and the CP ("synchronization"). This function is performed by the SYNCHRON block.

Transmission and reception is initiated by means of SEND and RECEIVE. A data transfer is initiated by means of SEND_ALL resp. RECEIVE_ALL.

Error messages will appear in the indicator word.

A description of the handler blocks is available in this manual in the chapter on "Integrated handler blocks".

Handler blocks

The following table lists the respective handler blocks:

Handler block		Description
SEND	SFC230	Transmit a job from the CPU to the CP.
SEND_ALL	SFC236	Initiate a file transfer between the CPU and the CP.
RECEIVE	SFC231	Reception of an order from the CP.
RECEIVE_ALL	SFC237	Initiate a file transfer between the CPU and the CP.
FETCH	SFC232	The FETCH block requests data. FETCH initiates a RECEIVE_ALL.
CONTROL	SFC233	The CONTROL block is used for a job-related status request, i.e. the ANZW of a specific job is updated.
CONTROL_ALL	SFC233 + ANR=0	Request for the present job
RESET	SFC234	The RESET block initiates a reset of the job for the specified connection.
RESET_ALL	SFC234 + ANR=0	RESET_ALL forces a full system reset of the CP.
SYNCHRON	SFC235	During start-up, SYNCHRON provides the synchronization between CPU and CP. At the same time the page frame is erased and the block size between PLC and CP is negotiated. Active data communications may only occur via synchronized page frames.

Synchronization

When a PLC starts-up, each of the configured interfaces of the CP has to be synchronized by means of SYNCHRON.

After power is turned on, the CPU 821xNET requires app. 15s for the boot-procedure. If the PLC should issue a request for synchronization during this time, an error is returned in the configuration error byte PAFE. This message is removed when the CP module has completed the boot process.

The timer in this block is initially set to 20s. Processing will be stopped if the synchronization is not completed properly within this period.

The following table shows the available block sizes.

Block size	CP block size in byte
0	Use the preset block size
1	16
2	32
3	64
4	128
5	256
6	512
255	512

Cycle

The sending and receiving blocks SEND and RECEIVE, which initiate the sending and receiving operations, must be configured in the cycle program OB1. The blocks SEND_ALL and RECEIVE_ALL perform the actual data-transfer.

Purely passive connections only require the components SEND_ALL or RECEIVE_ALL.

To protect the data transfer you should integrate various checkpoints that evaluate the indicator word.

Configuration examples

Outline and requirements

This chapter provides an introduction to use the bus systems H1 and TCP/IP for the System 82xx. This introduction is centered on the Advantech configuration software WinNCS.

The object of this chapter is to create a small communication system between two ADAM CPU 821xNET that provides a simple approach to the control of the communication processes.

Knowledge of the CP handler blocks is required. CP handler blocks are standard function blocks. These provide the options required to use the communication functions in the applications of the programmable logic controllers.

The minimum technical equipment required for the examples is as follows:

Hardware

- 2 CPU 821xNET,
- 1 PC or PU with twisted-pair Ethernet connection

Communication line

- 3 bus cables
- 1 hub

Software package

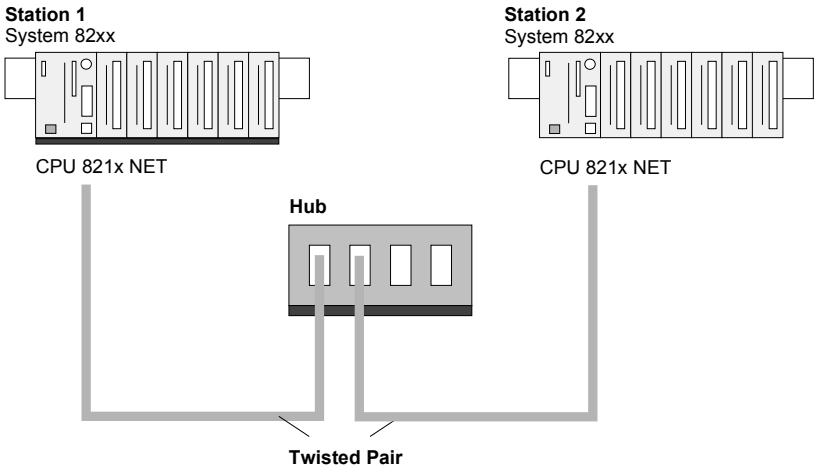
- Configuration software WinNCS from Advantech
- Siemens STEP®7 Manager programming package for CPU 821xNET

The implementation of the example is to program the two programmable logic controllers as well as configuring the communication processors by means of WinNCS.

Task

The introductory example is the application of a communication task, described below:

Structure



Purpose of the PLCs

Both of the CPUs run with the same PLC program, only the configuration of the CPs have to be adjusted.

Both stations are sending and receiving 16 data words per second.

- Data block DB11 transfers the data words DW0 to DW15 at an interval of 1s. Data word DW0 in DB11 is used as message counter. It is only incremented, if the preceding send command was processed correctly (completed without error). The remaining data words (DW1 to DW15) may be used for the transfer of user data.
- The receiving station stores the data in DB12 (DW 1 to DW 15).
- SEND is configured with job number A-No. = 11 and with a page frame offset SSNR = 0.
- RECEIVE is configured with job number A-No. = 1 and a page frame offset SSNR = 0.
- The source and destination parameters have to be configured directly.

At this point the purpose and the required settings have been outlined. The programs provide additional details of the configuration of the handler blocks. A detailed description of a suitable configuration of the CPs under control of H1 or TCP/IP is also included.

Configuration under WinNCS

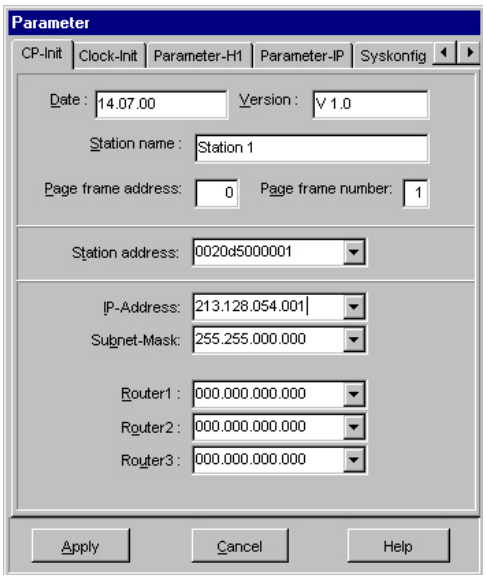
The two CPs are configured exclusively by means of WinNCS. Start WinNCS and create a project containing the function group "Ethernet_H1". The procedure is the same for both stations. It differs only in the parameters that have to be defined and is divided into the following 3 parts:

- Basic CP configuration,
- Configuration of connection blocks,
- Transfer of configuration data into the CP.

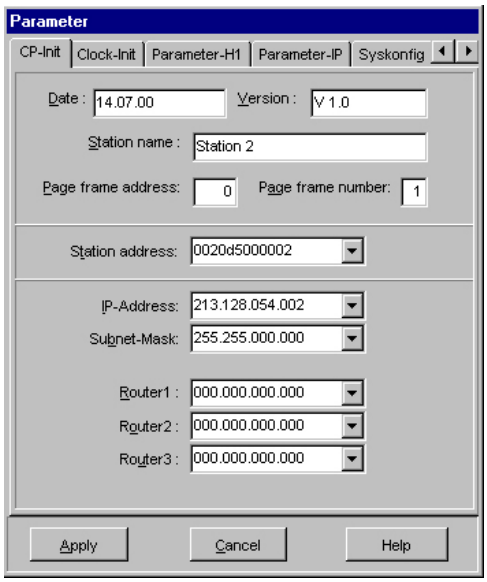
Basic CP configuration

Insert two stations and select the following settings:

Station 1



Station 2



Request the required station addresses from your system administrator.

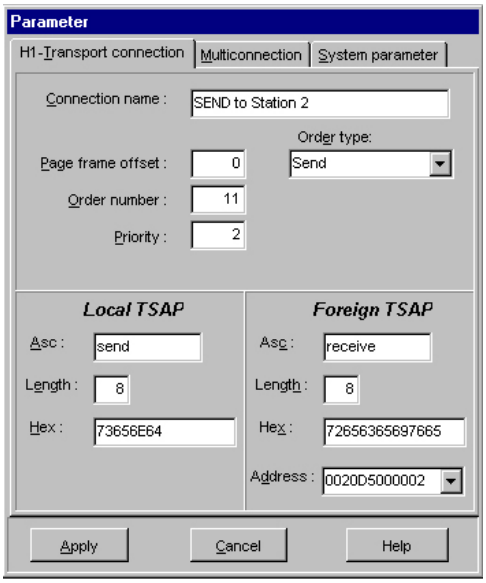
If necessary, you may enter additional settings into the configuration windows. Details are obtainable from your system administrator.

Connection block configuration

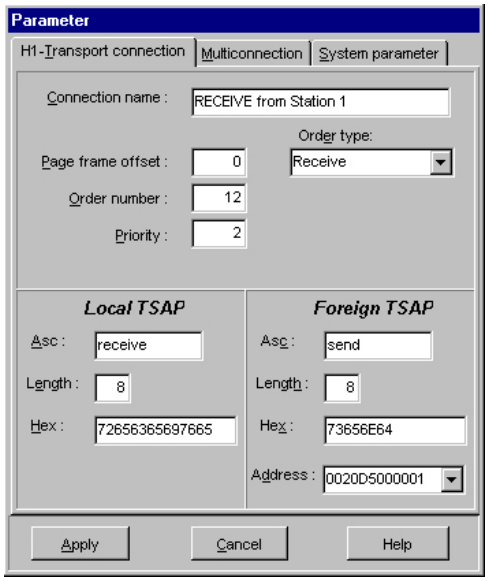
H1 connections

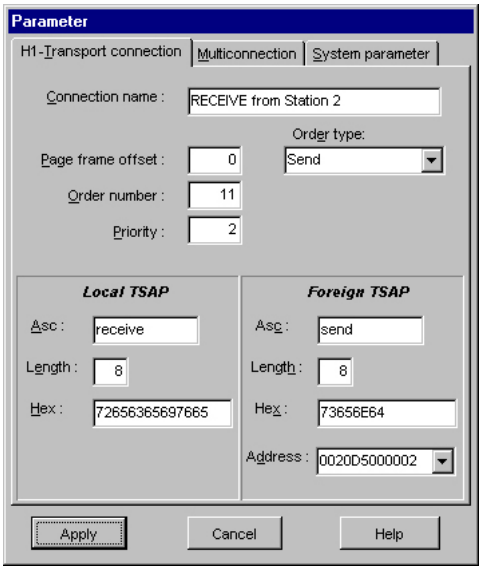
You configure your H1 connection by inserting an H1 transport connection below the stations by means of  and entering the following parameters for the stations:

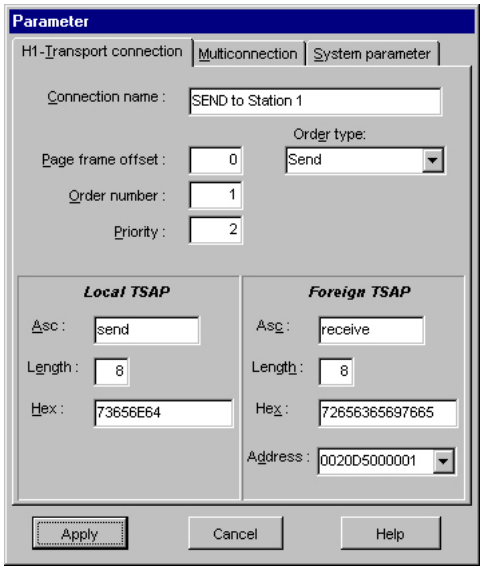
Station 1



Station 2





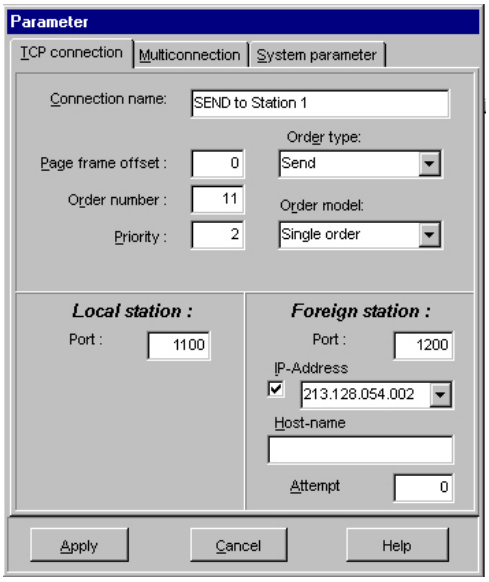


TCP/IP connections

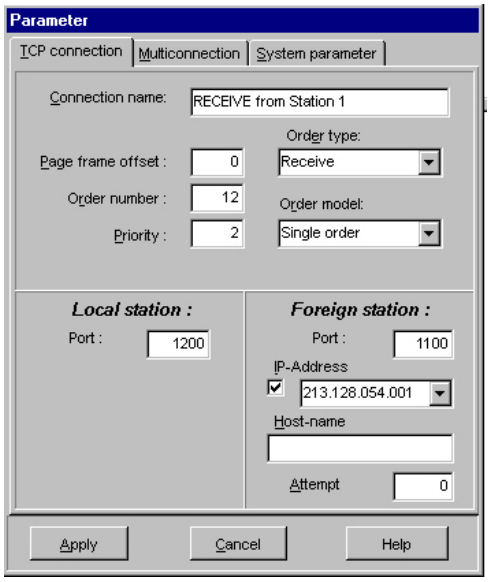
You configure your TCP/IP connection by inserting a TCP connection below the stations by means of  and entering the following parameters for the stations:

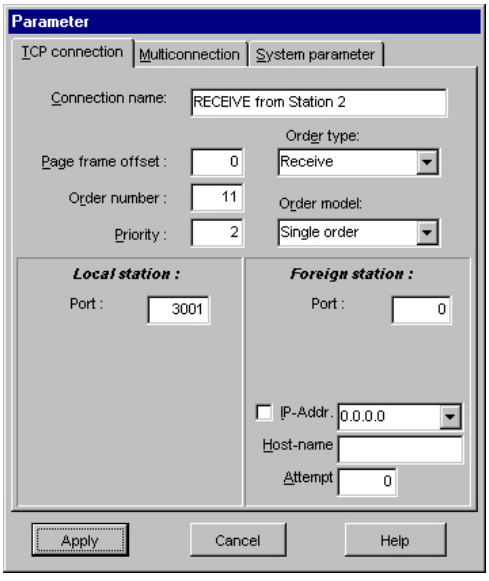
TCP/IP connections

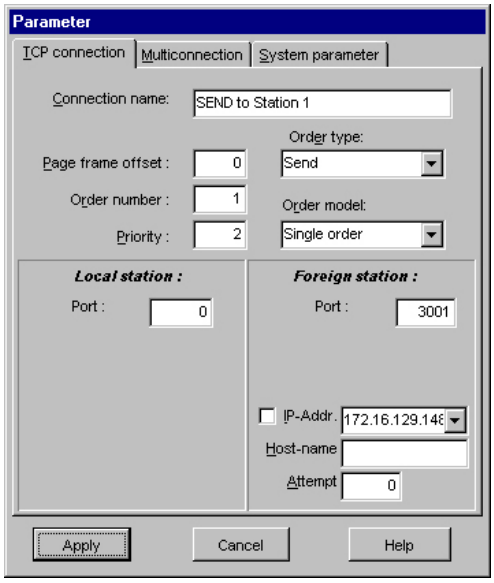
Station 1



Station 2



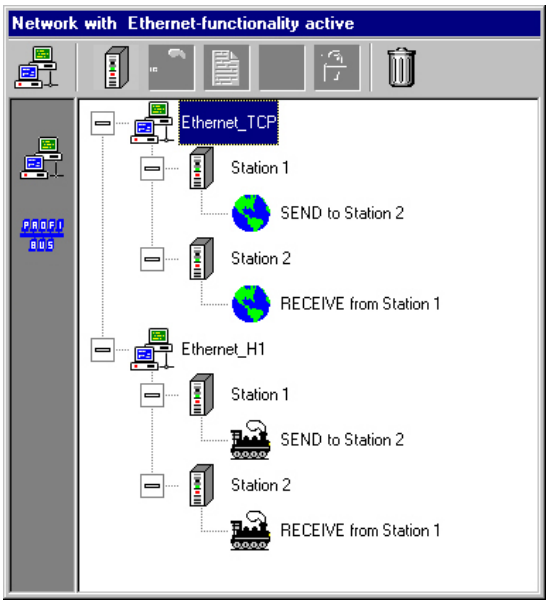




Please save your project!

Network window

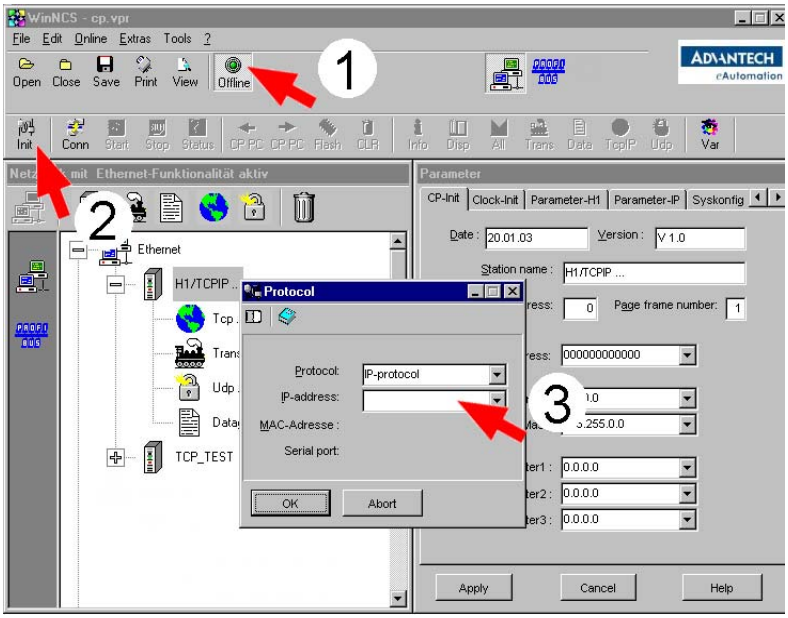
Your network window should have the following contents:



Transferring the configuration data into the CPUs

You can transfer your configuration online via the network into the respective CPUs. Create the system structure as shown above and start both CPUs.

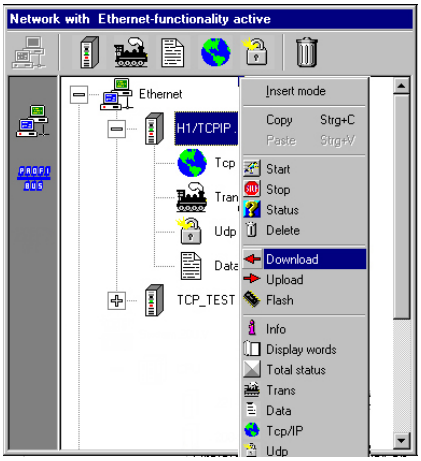
For the data transfer please activate the online functions and click on the button INIT:



Now choose „IP-protocol“ in the *Protocol* window and type the according IP address. Confirm with [OK].

Select the station to which you wish to transfer the configuration data in the *network* window of WinNCS and activate [Download].

Your project will now be transferred to the RAM of the CPU.

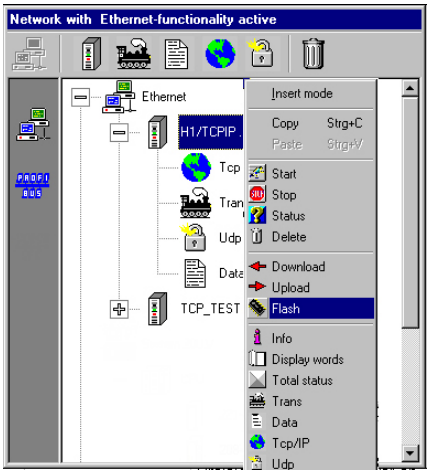


Attention!

After transferring the project into the CPU-RAM you should save the project additionally in the Flash-ROM. Otherwise the data will get lost at power off.

Therefore you are searching the according station in the *network* window. Click on the right mouse button and choose „Flash“.

The RAM data are now transferred to the Flash-ROM.



Repeat this procedure above for station 2 and don’t forget to save the project to Flash-ROM.

This concludes the configuration of the CP side. The following pages contain information on the programming for the PLC section.

PLC programs for the CPUs

The PLC programming in this example does not depend on the protocol and can therefore be used for H1 and TCP/IP.

This PLC program is used in both CPUs.

OB100 Interface Synchronization

Synchronization of the interfaces

In the start-up OB OB100 of the CPU the interface used on the CP has to be synchronized by means of the handler block SYNCHRON.

OB100 verifies that the synchronization procedure was completed without errors. If an error is detected, the error number is entered into MW100.

Operation block OB100:

```
OB100 :  "Complete Restart"
Commentary:

Network  1: Title:
Commentary:

      L     S5T#20S                    // Timer 20sec
      CLR                              // VKE 0
      SV    T      1
      SET                              // VKE 1
      SV    T      1                   // Timer Start

loop: CALL  "Synchron"                 // CP synchron
       SSNR:=0                         // Interface number
       BLGR:=6                         // Block size 6 : 512 byte
       PAFE:=MB200                     // Configuration error byte

      UN    M    200.0                 // no error - ready
      BEB
      U     T      1                   // Timer running
      SPB   loop                       // recall synchron
```

OB1 - Cycle
FC 1 - SEND
FC 2 - RECEIVE

The OB1 cycle controls the sending and receiving of the data. The initiation of transmission in station 1 is issued by a SEND handler block called in FC1. The partner station answers with RECEIVE (FC2). By means of SEND_ALL the data will be send and received by the partner with the command RECEIVE_ALL.

Cycle operation block OB1:

```
OB1 : Cycle
Send-All, Rec-All, Applikation

Network 1: All Function
Send- und Rec-All

      CALL  "Receive_All"              // takes data at CP
       SSNR:=0                         // is also used write passiv
       PAFE:=MB196                     // Parametrization error byte MB196
       ANZW:=MD180                     // Indication word:
//                                        showes the working order number

      CALL  "Send_All"                 // send data at CP
       SSNR:=0                         // is also used from Fetch passiv
       PAFE:=MB197                     // Parametrization error byte
       ANZW:=MD184                     // Indication word:
//                                        showes the working order number

Network 2: Title
SEND

      CALL  FC  1                      // Function call 1: sending


Network 3: Title
RECEIVE

      CALL  FC  2                      // Function call 2: receive
```

FC1 - SEND

```
FC1 : Send
Send function

Network 1 : Title
Time control sending

      CALL  "Control"                  //control send
       SSNR:=0                         //Interface number 0
       ANR :=1                         //Order number 1
       PAFE:=MB195                     //Parametrization error MB195
       ANZW:=MD174                     //Indication word MD174

      0     M    175.1                 //Job runs
      0     T      1                   //Timer runs
      BEB                            //end

      L     S5T#1S                   // Timer 1 sec
      CLR                            // vke 0
      SV    T      1
      SET                            // vke 1
      SV    T      1                 // Timer start

      CALL  "Send"                   // call send
       SSNR:=0                       // Interface number 0
       ANR :=1                       // Order number 1
       IND :=0                       // Indirect configuration
       QANF:=P#DB11.DBX 0.0 BYTE 16  // Data : db11, dw0, 16 byte
       PAFE:=MB196                   // Parametrization error MB 196
       ANZW:=MD174                   // Indicator word MD 174

      L     DB11.DBW    0            // Message counter
      +     1                        // DB11, DW0
      T     DB11.DBW    0            // increment message counter
```

FC2 - RECEIVE

```
FC2 : RECEIVE
RECEIVE funktion

Network 1 : Title
Cycle

      CALL  "Control"                  // SFC233
       SSNR:=0                         // Interface number 0
       ANR :=11                        // Order number 11
       PAFE:=MB195                     // Parametrization error MB 195
       ANZW:=MD184                     // Indicator word MD 184

      UN    M    185.0                 // if no handshake
      BEB                            // end


      CALL  "Receive"                  // SFC231
       SSNR:=0                         // Interface number 0
       ANR :=11                        // Order number 11
       IND :=0                         // Indirect configuration
       ZANF:=P#DB12.DBX0.0 BYTE 16
       PAFE:=MB195                     // Parametrization error MB 195
       ANZW:=MD184                     // Indicator word MD 184

      L     MB   188                   // Increment message counter
      +     1
      T     MB   188
```

Data blocks DB11, DB12

The frequency with which a SEND job is issued, depends on the time that was configured for the FC1 call. This timer is programmed for 1000ms in this example. For this the sample program initiates the SEND job at a rate of once every 1000ms.

Data word DW0 in data block DB11 is incremented ahead of every SEND call that actually transmits a message. This occurs in function block FC1. Altogether 16Byte of data are transferred. The partner receives the information and stores it in DB12.

Together with DB0 there is still 15Byte space for user data.

DB11 and DB12 are identical in design:

Address	Name	Type	Startvalue	Comment
0.0		STRUCT		
+0.0	STAT0	BYTE	B#16#0	
+1.0	STAT1	BYTE	B#16#0	
+2.0	STAT2	BYTE	B#16#0	
+3.0	STAT3	BYTE	B#16#0	
+4.0	STAT4	BYTE	B#16#0	
+5.0	STAT5	BYTE	B#16#0	
+6.0	STAT6	BYTE	B#16#0	
+7.0	STAT7	BYTE	B#16#0	
+8.0	STAT8	BYTE	B#16#0	
+9.0	STAT9	BYTE	B#16#0	
+10.0	STAT10	BYTE	B#16#0	
+11.0	STAT11	BYTE	B#16#0	
+12.0	STAT12	BYTE	B#16#0	
+13.0	STAT13	BYTE	B#16#0	
+14.0	STAT14	BYTE	B#16#0	
+15.0	STAT15	BYTE	B#16#0	
+16.0	STAT16	BYTE	B#16#0	
=18.0		END_STRUC		

Transfer project

The data transfer is realized via MPI. If your programming device has no MP-interface, you may also use the "Green Cable" (ADAM-8950-0KB00) from Advantech.

The "Green Cable" may only be used at the ADAM CPUs of the Systems 8xxx!

Please regard the notes about the "Green Cable" in chapter 1!

- Connect your PU with the CPU
- With **PLC** > L*oad to module* in your configuration tool you transfer your project into the CPU.
- Insert a Multi Media Card (MMC) into the according CPU-slot and transfer your application via **PLC** > *Copy RAM to ROM*.
- During the writing process the MC-LED on the CPU is blinking. Due to the system the dialog message of the successful writing process appears a little bit too soon. This process is not ready before the LED on the CPU is extinguished.
- Switch both CPUs to RUN.

Monitoring the data transfer in Siemens STEP®7 manager

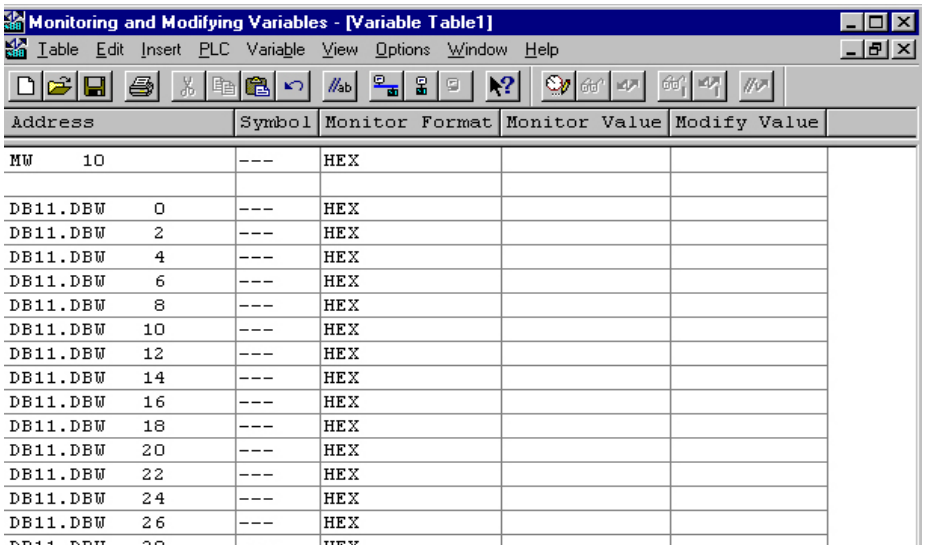
It is assumed that the CPs were programmed and that an overall reset was issued to the CPUs where the RUN/STOP switch has to be in STOP position.

You now load the above PLC programs into your CPUs. Start the programs by placing the respective RUN/STOP switch into the position RUN.

At this point, communication between the modules is established. This is indicated by the COMM-LEDs.

Start the Siemens STEP®7 manager and execute the following steps to monitor the transmitting job:

- **PLC** > *Monitor/Modify Value*
- In the "Operand" column you enter the respective data block number and the data word (DB11.DW0-15).
- Establish a connection and click "Monitor"  .



Entering user data

You may enter user data starting with DW1. Place the cursor on *Modify value* and enter the value you wish to transfer, e.g. W#16#1111.

The  button transfers the new value in every cycle and the  button initiates a single transfer.

Start-up behavior

Overview

When the power supply is turned on, the CPU and the CP execute the respective BIOS routine (hardware and driver initialization and memory test).

While the CPU detects the installed modules on the backplane bus and loads the application program, the CP starts the page frame administration routine.

After app. 15s the CP waits for the synchronization request from the CPU. In this condition data communication with the PLC is inhibited and it is only enabled after synchronization has taken place.

The boot time of the CPU 821xNET including the CP amounts to app. 18s.

Status after CP boot-up

With every status change from STOP to RUN as well as from RUN to STOP back to RUN, the CPU821xNET performs a cold-/warm reset. All connections are cleared and reestablished after the CP has rebooted.

Three different reasons can cause these status change requests:

- Resynchronization of a CP by the SYNCHRON-HTBs of the CPU (warm start) after it has already been synchronized,
- STOP/START-function of the configuration tool WinNCS (warm start),
- RESET_ALL (warm start).

System properties of the CPU 821xNET

Note

System properties of a CP should not be regarded as restrictions or equated with malfunctions. Certain functions can not be provided or are not desired when the overall system is taken into account.

General

The boot time of the CP section of the CPU 821xNET is app. 18 seconds. The integrated SYNCHRON block (delay time 30s) caters for this boot time.

Note only for H1

- Jobs with a priority 0/1 may transmit and/or receive a maximum quantity of data as defined by the SYNCHRON-HTB. Jobs of this priority are not blocked. This results in a maximum data transfer rate of 512Byte per job for a block size of 255 (refer also to block size).
- RECEIVE jobs that are mapped to the communication type "broadcast" cannot receive all data messages from a fast cyclic transmitter. Messages that have not been received are discarded.

Note only for TCP/IP

- The default length (-1, 0xFFFF) is not permitted for the ORG format length, i.e. the user has to define the exact length of the data that will be received.
- Jobs with a priority of 1 may send and/or receive a maximum quantity of data as defined by the SYNCHRON-HTB. Jobs of this priority are not blocked. This results in a maximum data transfer rate of 512Byte per job for a block size of 255 (see also block size).
- RECEIVE-jobs that mapped to communication type UDP cannot receive all data messages from a fast cyclic station. Messages that have not been received are discarded.

The TCP/IP protocol stack has a global buffer pool where the receive and transmit buffers are located. This is where system collisions can occur if:

- Data for a receive job is not collected. After a period of time a lack of resources will occur and the other connections will terminate themselves.
  It is only possible to re-establish proper communications when the receive buffers of this connection have been released (connection terminated) or when the data has been retrieved by means of RECEIVE.
- one or more cyclic stations place a load on a CP. When resource bottlenecks are encountered, the CP can also initiate the termination of connections.

- A station transmits two or more messages and the receiver did not have a chance to accept them. Then the reception of the unknown data type would cause collisions in the receiver. However, the CP prevents this. The PLC application requires a defined size for the reception of data and the wildcard length is not permitted. The size of the receive block of Prio1 - RECEIVE jobs is defined implicitly by the pre-defined block size (16, 32, 64, 128, 256, 512Byte).
- Advantech recommends the use of acknowledgment messages on the user level to ensure that data transfers are one hundred percent safe.
- Please regard, that the **Port 7777** is used by WinNCS for communication. This may not be occupied by other applications!

Communication links to foreign systems

ORG format

The organization format is the abbreviated description of a data source or a data destination in a PLC environment. The following table lists the available ORG formats.

In the case of READ and WRITE the ORG block is optional.

The ERW identifier is used for the addressing of data blocks. In this case the data block number is entered into this identifier. The start address and quantity provide the address for the memory area and they are stored in HIGH/LOW format (Motorola-formatted addresses)

Description	Type	Range
ORG identifier	BYTE	1..x
ERW identifier	BYTE	1..255
Start address	HILOWORD	0..y
Quantity	HILOWORD	1..z

The following table contains a list of available ORG formats:

ORG identifier 01h-04h

CPU area	DB	MB	EB	AB
ORG identifier	01h	02h	03h	04h
Description	Source/destination data from/into data block in main memory.	source/destination data from/into flag area.	Source/destination data from/into process image of the inputs (PAE).	source/destination data from/into process image of the outputs (PAA).
DBNR valid range:	DB from where the source data is retrieved or to where the destination data is transferred. 1...255	irrelevant	irrelevant	irrelevant
Start address Significance valid range:	DB-No. from where the data is retrieved or where data is saved. 0...2047	MB- No. from where the data is retrieved or where data is saved. 0...255	EB-No. from where the data is retrieved or where data is saved. 0...127	AB-No. from where the data is retrieved or where data is saved. 0...127
Quantity Significance valid range:	Length of the source/destination data - block in words. 1...2048	Length of the source/destination data block in words. 1...256	Length of the source/destination data block in words. 1...128	Length of the source/destination data block in words. 1...128

Structure of PLC header

For every READ and WRITE the CP generates PLC headers for request messages and for acknowledgment messages. Normally the length of these headers is 16Byte and they have the following structure:

for WRITE

Request message

System identifier	="S"
	="5"
Header length	=16d
Ident.OP code	=01
OP code length	=03
OP Code	**=03**
ORG block	=03
ORG block length	=08
ORG identifier	
DBNR	
Start address	H
	L
Length	H
	L
Dummy block	=FFh
Dummy block length	=02
64K data only if error no.=0	

Acknowledgment message

System identifier	="S"
	="5"
Header length	=16d
Ident.OP code	=01
OP Code length	=03
OP Code	**=04**
Ack. Block	=0Fh
Ack. Block length	=03
Error No.	=No.
Dummy block	=FFh
Dummy block length	=07
not used	

for READ

Request message

System identifier	="S"
	="5"
Header length	=16d
Ident.OP code	=01
OP Code length	=03
OP Code	**=05**
ORG block	=03
ORG block length	=08
ORG identifier	
DBNR	
Start address	H
	L
Length	H
	L
Dummy block	=FFh
Dummy block length	=02

Acknowledgment message

System identifier	="S"
	="5"
Header length	=16d
Ident.OP code	=01
OP Code length	=03
OP Code	**=06**
Ack. block	=0Fh
Ack. Block length	=03
Error No.	=No.
Dummy block	=FFh
Dummy block length	=07
not used	
64K data only if error no.=0	

SEND / RECEIVE of the type TRADA

TRADA stands for **Tra**nsparent **Da**ta exchange. A transparent data exchange may transfer application data with varying block lengths. A 16Byte header that defines the length of the application data precedes the application data.

With TRADA you may enter a wildcard length into the PLC application.

If you enter "-1" as length into the RECEIVE-FB (parameter: ZLAE) you are defining a variable length (wildcard length) for the application data. With wildcard lengths the actual length of the data is retrieved from the respective TRADA header.

With the TRADA functionality the following header will precede a SEND job and it is analyzed by the RECEIVE function.

SEND of type TRADA
OP-code = 07

System identifier	="S"	
	="5"	
Header length	=16d	
Ident.OP code	=01	
OP code length	=03	
OP code	**=07**	
ORG block	=03	
ORG block length	=08	
ORG identifier		
DBNR (**irrelevant**)		
Start address	H	
	L	
Length	**H**	→ Length of the user data
	L	
Dummy block	=FFh	
Dummy block length	=02	
64K data if error no.	=0	

Length

The length file contains the number of bytes in a data block.

If you are synchronizing with a block size of 6 (512Byte) the length is entered in words.

Test program for TCP/IP connections

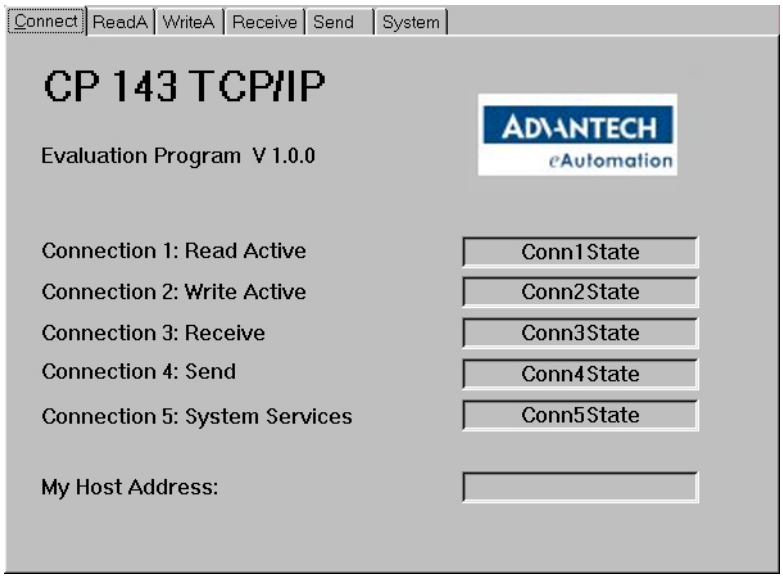
The test program TCPTest is included with WinNCS. You may use this test program to create simple TCP/IP connections and analyze them.

After the installation of WinNCS TCPTest is to find in the directory NCS.

The following section provides a short introduction to the test program.

For this purpose please start TCPTEST.EXE. The test program is executed and displays the following window:

Initial display



Tab sheets

The menu has the appearance of tab sheets. The respective dialog window may be displayed by left clicking with the mouse.

Tab sheets

Connect	window containing the status of the connections and the local IP address.
ReadA	configuration window for READ ACTIVE connections (FETCH).
WriteA	configuration window for WRITE AKTIVE connections.
Receive	configuration window for RECEIVE orders.
Send	configuration window for SEND orders.
System	control windows for status requests and toggling between RUN/STOP of the CP.

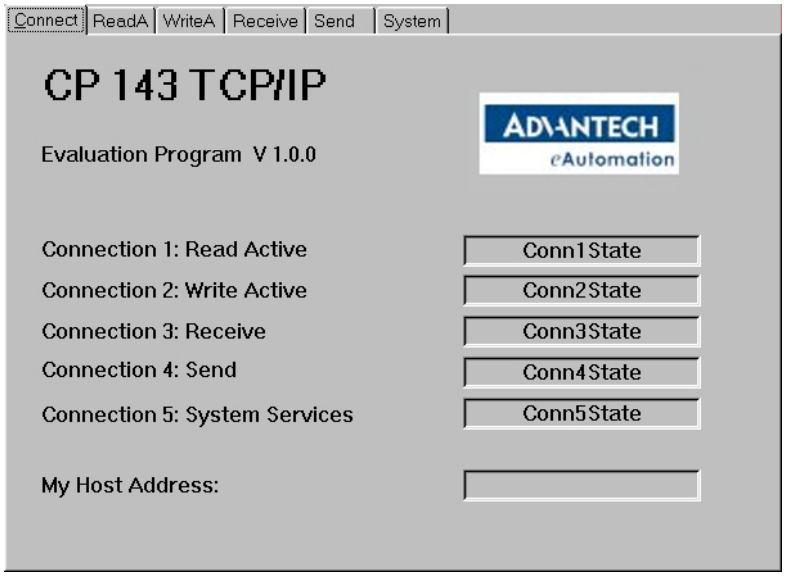
Context menu (right mouse key)

You may activate a context menu in each tab sheet. This is activated by means of the right mouse key.

You may always access the context menu by clicking the right mouse key. This menu offers the following selection:

Save All	save all parameters.
Save Conn1 to *Save Conn5*	saves the respective connection.
Save Win Pos	saves the current window position.
Show Hints	When you place the cursor on an input field or on a button, a hint is displayed if you selected "Show Hints".

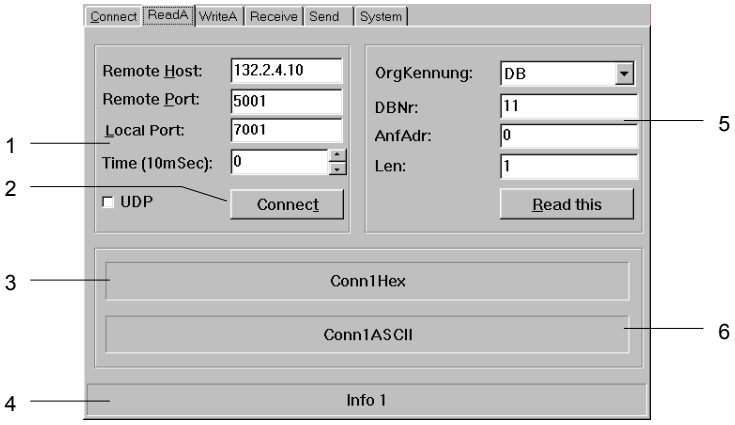
Connect tab (Status)



This window displays the status of all the connections that can be configured in this program. Here you may recognize in one screen, which connections are stable and which are unstable. When a status changes in a register, the change is displayed in this window.

For reference, your own IP address is also displayed in the window.

ReadA tab



[1] port data
[2] establish a connection
[3] hexadecimal number
[4] information window for the status of the connection
[5] source data
[6] ASCII formatted display of the data received

Here you may configure an active read connection.

In addition to the data required to establish the connection, you also have to specify the source from where the data should be read.

Input fields

Remote Host	IP address of the station from which you wish to read data.
Remote Port	Port address of the remote station.
Local Port	Port address of your own (local) station. To simplify matters you can specify the same port address for remote and local.
Time (10mSec)	Definable interval for cyclic read operations.
OrgKennung	Type of the source block.
DBNr	Number of the source block.
AnfAdr	Start address of the source block.
Len	Word length of the source block.

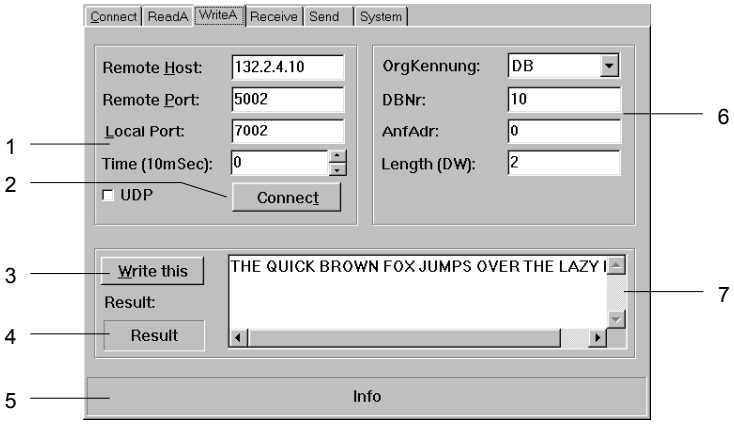
Tick-box

UDP	This tick mark selects unsecured communications. No virtual connections are used by unsecured communication links. In this manner you may only display UDP messages.

Buttons

Connect	The connection is established and prepared for the read operation.
Read this	The data requested is read via this connection.

WriteA tab



[1] port data
[2] establish a connection
[3] transfer the data via the connection
[4] result-code of the write job
[5] information window for the status of the connection
[6] source data
[7] ASCII-text that has to be transferred to the CP

This is where you activate an active write connection.

In the same way as for the READ active command, you declare the destination block from where the data has to be transferred in addition to the data required for establishing the connection.

Input fields

Remote Host	IP address of the station to which you wish to write the data.
Remote Port	Port address of the remote station.
Local Port	Port address of your own (local) station. To simplify matters you can specify the same port address for remote and local.
Time (10mSec)	Definable interval for cyclic write operations. The minimum timer value for cyclic writes is 5.
OrgKennung	Type of destination block.
DBNr	Number of destination block.
AnfAdr	Start address of destination block.
Len	Word length of destination block.

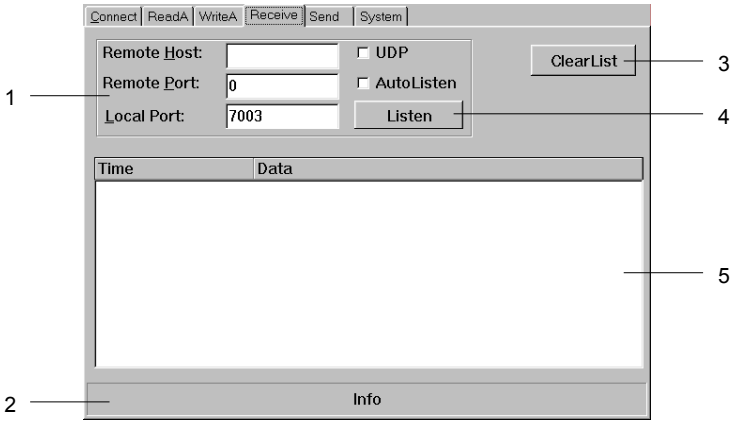
Tick-box

UDP	This tick mark selects unsecured communications. No virtual connections are used by unsecured communication links. In this manner you may only display UDP messages.

Buttons

Connect	The connection is established and prepared for the write operation.
Write this	Data entered into the ASCII field is transferred to the CP via the connection that was established by means of *Connect*.

Receive tab



[1] Connection data
[2] connection status information bar
[3] clear received list
[4] list messages
[5] list of received messages

In this dialog window you configure the reception of messages from a specific host.

Input fields

Remote Host IP address of the station where the data will be saved.

Remote Port Port address of the remote station.

Local Port Port address of own (local) station. To simplify matters you may specify the same port address for remote and local.

Tick-box

UDP This tick mark selects unsecured communications. No virtual connections are used by unsecured communication links. In this manner you may only display UDP messages.

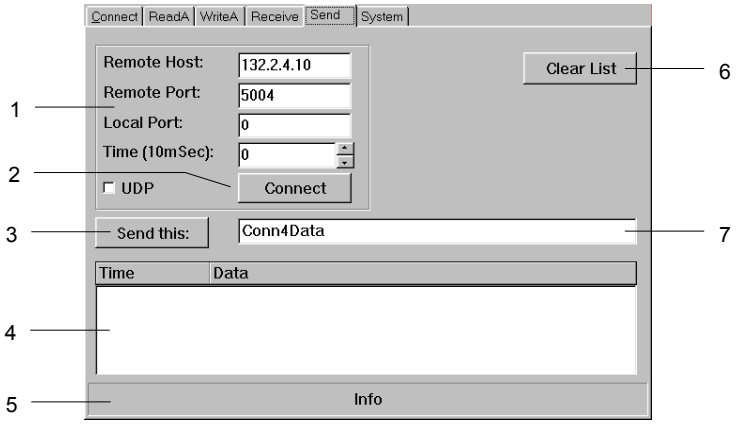
AutoListen If you select "AutoListen", the program switches to receive mode. Every message received from the remote CP is displayed in the list. Interruptions of the connection are detected and displayed, however, the program remains ready to receive data. As soon as the connection is re-established messages will again be listed.

Buttons

Listen Any received messages are entered into the list. The listing is stopped when you click the "STOP" button or the connection is interrupted. You may also stop the listing by entering a new set of connection parameters.

ClearList Clears the received list, new entries will appear at the top of the list.

Send tab



[1] port data
[2] establish a connection
[3] transfer data via the connection
[4] list of transmitted messages
[5] information bar for the connection status
[6] clear the list of messages
[7] ASCII-text that must be transferred to the CP

You may use this dialog window to send a message to a specific host.

Input fields

Remote Host IP address of the station where the data must be saved.

Remote Port Port address of the remote station.

Local Port Port address of own (local) station. To simplify matters you can specify the same port address for remote and local.

Time (10mSec) Definable interval for cyclic write operations. The minimum timer value for cyclic writes is 5.

Tick-box

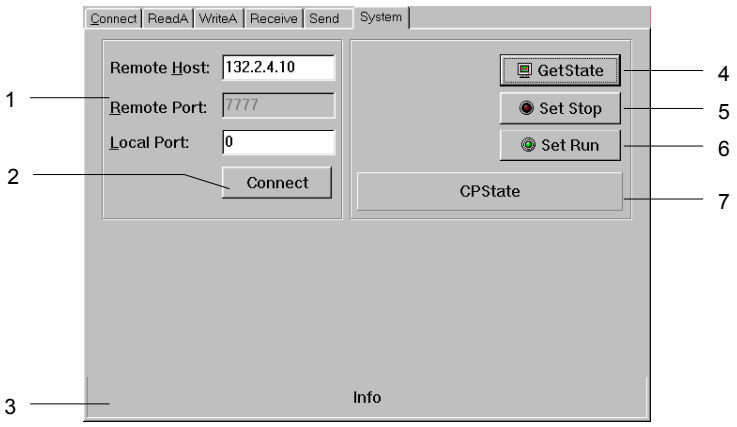
UDP This tick mark selects unsecured communications. No virtual connections are used by unsecured communication links. In this manner you may only display UDP messages.

Buttons

Connect The connection is established and prepared for the write operation.

Send this Data entered into the ASCII field is transferred to the CP via the connection that was established by means of *Connect*.

System tab



[1] port data
[2] establish a connection
[3] information bar for the connection status
[4] CP status query
[5] CP in STOP
[6] CP in RUN
[7] Status monitor, requested by GetState

This dialog window gives you information about your specified Host-CP.

Input fields

Remote Host IP address of the station where the data will be saved.

Remote Port Port address of the remote station.

Local Port Port address of own (local) station. To simplify matters you may specify the same port address for remote and local.

Buttons

Connect The connection is established and prepared for the write operation.

GetState Via the connection established by *Connect*, the status of the CP is transferred and monitored in the status window. This may be seen:

- Hardware-Stop (RUN/STOP lever at the CP is in STOP position)
  The CP is not remoteable with the test program.
- Hardware-Run (RUN/STOP lever at the CP is in RUN position)
  The CP is remoteable with the test program.
- Software-Stop (RUN/STOP lever at the CP is in RUN position)
  The CP has been switched to STOP with *SetStop*.
- Software-Run (RUN/STOP lever at the CP is in RUN position)
  The CP has been switched to RUN with *SetRun*.

SetStop The CP is switched to STOP. This function is only available, if the RUN/STOP lever at the CP is in RUN position.

SetRun The CP is switched to RUN. This function is only available, if the RUN/STOP lever at the CP is in RUN position.

Chapter 5 Deployment of the CPU 821xDPM

Outline

Content of this chapter is the deployment of the CPU 821xDPM under Profibus. After a short introduction into the Profibus system, the project engineering and the usage under MPI is shown.

This chapter ends with information to the operating modes of the DP master and to the commissioning.

The following text describes:

- Principles of Profibus-DP
- Proyect engineering of a CPU 821xDPM
- Deployment of the MPI-interface and MMC
- Operating modes of the DP master
- Commissioning

Content

Principles

General

Profibus is an open fieldbus standard for building, manufacturing and process automation. Profibus defines the technical and functional properties of a serial fieldbus system that is used to create a network of distributed digital field automation equipment on the lower (sensor-/actuator level) to middle performance level (process level).

Profibus comprises various compatible versions. The specifications contained in this description refer to Profibus-DP.

Profibus-DP

Profibus-DP is particularly suitable for applications in production automation. DP is very fast, offers plug&play and is a cost-effective alternative to parallel cabling between PLC and the decentral/distributed periphery. Profibus-DP is conceived for high-speed data exchange on the sensor-actuator level.

The data exchange is processed cyclically. During a one bus cycle the master reads the input values from the various slaves and writes new output information into the slaves.

Master and slaves

Profibus distinguishes between active stations (masters) and passive stations (slaves).

Master equipment

Master equipment controls the data traffic on the bus. There may be also several masters at one Profibus. This is referred to as multi master operation. The bus protocol establishes a logical token ring between the intelligent devices connected to the bus.

A master like in the CPU 821xDPM may send unsolicited messages if it has the bus access permission (Token). In the Profibus protocol masters are also referred to as active stations.

Slave equipment

Typical slave equipment holds data of peripheral equipment, sensors, actuators or transducers. The ADAM Profibus are modular slave equipment, transferring data between the System 82xx periphery and the leading master.

These devices do not have bus access permission in accordance with the Profibus standard. They may only acknowledge messages or transfer messages to a master if requested by this. Slaves occupy a very limited part of the bus protocol. Slaves are also referred to as passive stations.

Communication

The bus communication protocol provides two procedures for accessing the bus:

Master to Master

Communications with the master is also referred to as token passing procedure. Token passing guarantees that the station receives access permission to the bus. This access right to the bus is passed between the stations in form of a "token". A token is a specific message that is transferred via the bus.

When a master possesses the token, it has the access right to the bus and is allowed to communicate with all other active and passive stations. The token retention time is defined when the system is being configured. When the token retention time has expired, the token is passed along to the next master that acquires the bus access rights with the token so that this may now communicate with all other stations.

Master slave procedure

Data is exchanged in a fixed repetitive sequence between the master and the slaves assigned to this respective master. When you configure the system, you define which slaves are assigned to a certain master. You may also specify which DP slave is included in the cyclic exchange of application data and which ones are excluded.

The master slave data transfer is divided into parameterization, configuration and data transfer phases. Before a DP slave is included into the data transfer phase, the master verifies during the parameterization and configuration phase whether the specified configuration agrees with the effective configuration. This verification process checks the device type, format and length as well as the number of inputs and outputs. This provides you with effective protection against configuration errors.

The master handles application data transfers independently. In addition you may also send new configuration data to a bus coupler.

If in the status DE „Data Exchange“, the master is sending new basic data to the slave and the receipt of the slave transfers the recent input data to the master.

Data consistency

Data is referred to as being consistent, if it has the same logical contents. Data that belongs together is: the high- and low-byte of an analog value (word consistency) and the control and the status byte with the respective parameter word, required to access the registers.

The data consistency during the interaction between the peripherals and the controller is only guaranteed for 1Byte. That is, the bits of one byte are acquired together and they are transmitted together. Byte-wise consistency is sufficient for the processing of digital signals.

Transfer medium

As transfer medium Profibus uses an isolated drilled 2 core line based upon the RS485 interface or a duplex optical waveguide (OWG). The transfer rate is for both methods max. 12Mbaud.

More information about this topic is available at the installation guideline.

Electrical system over RS485

The RS482 interface is working with voltage differences. Though it is less irritable from failures than a voltage or a current interface. You are able to configure the network as well linear as in a tree structure. Your ADAM Profibus coupler includes a 9pin slot where you link up the Profibus coupler into the Profibus network as a slave.

The bus structure under RS485 allows an easy connection resp. disconnection of stations as well as starting the system step by step. Later expansions don't have any influence on stations that are already integrated. The system realizes automatically if one partner had a fail down or if it is new in the network.

Optical system via optical waveguide (OWG)

The optical waveguide system uses monochromatical light impulses. The optical waveguide is totally independent from disturbing voltage from other machines. An optical waveguide system is built up linear. Every module has to be connected with two links: one input link and one back. You don't need to terminate the last module.

For the structure is a linear one, connecting and disconnecting stations is not free of consequences.

Addressing

Every partner of the Profibus network has to identify itself with a certain address. This address may be existing only one time in the bus system and has a value between 0 and 125.

At the CPU 821xDPM you choose the address via your software tool.

Electronic Data Sheet (GSD-file)

To configure the slave connections in your own configuration tool, you've got all the information about your ADAM modules in form of an electronic data sheet file (German: ***G****erätes****t****amm****d****atei* = *GSD-file*).

Structure and content of this file are dictated by the Profibus User Organization (PNO) and may be seen there.

Install the GSD-file in your configuration tool. Look for more information in the online help of the according tool.

Project engineering CPU with integrated Profibus-DP master

Outline

For the project engineering of the Profibus-DP master you have to use the hardware manager from Siemens. Your Profibus projects are transferred via MPI into the CPU 821xDPM by means of the **PLC** functions. The CPU passes the data on to the Profibus-DP master.

Compatibility to STEP®7 manager from Siemens

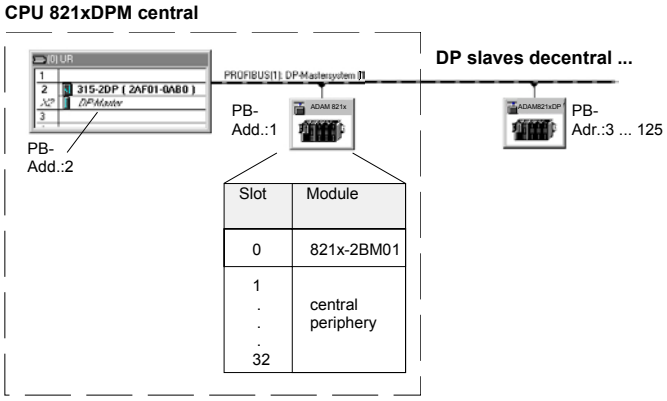
The address allocation and the parameterization of the directly plugged-in modules takes place via the STEP®7 manager from Siemens in form of a virtual Profibus system.

For the Profibus interface is standardized in software, you'll have the complete functionality of our modules at your disposal by including the GSD-file into the STEP®7 manager.

Steps of engineering

To be compatible with the STEP®7 projecting tool from Siemens, you have to execute the following steps for the System 82xx:

- Project the CPU 315-2DP with the DP master system (address 2).
- Add the Profibus slave with address 1.
- Insert the CPU type 8**21xDPM** at plug-in location 0 of the slave system.
- Include directly plugged-in peripheral modules also via this slave at the plug-in locations 1...32.



Note!

For the project engineering of the CPU and the Profibus-DP masters a thorough knowledge of the STEP®7 manager and the hardware configurator from Siemens is required!

Preconditions

For the project engineering of the Profibus-DP master in the CPU 821xDPM the following preconditions have to be fulfilled:

- STEP®7 manager from Siemens is installed.
- When using Profibus-DP slaves of the Systems ADAM 8xxx from Advantech: GSD-files are included into the hardware configurator
- There is a transfer possibility between configuration tool and CPU 821xDPM.

Install Siemens hardware configurator

The hardware configurator is a part of the STEP®7 projecting tool from Siemens. It serves the project engineering. The modules that may be configured here are listed in the hardware catalog.

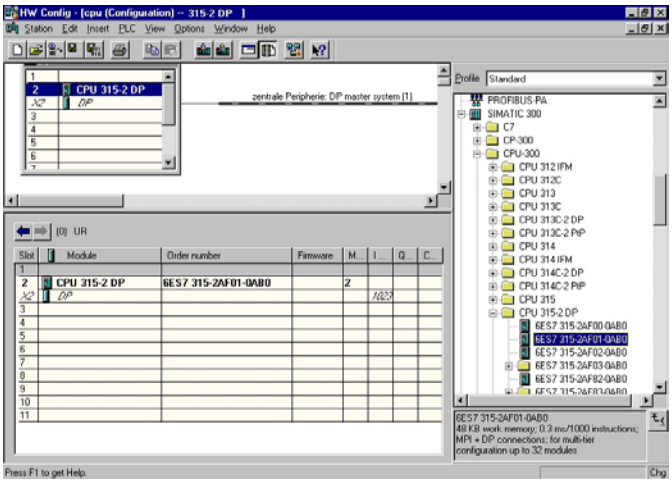
For the deployment of the Profibus-DP slaves of the Systems ADAM 8xxx from Advantech you have to include the modules into the hardware catalog by means of the GSD-file from Advantech.

Project engineering CPU 821xDPM

The following section describes the single steps of the project engineering.

Create a virtual Profibus system

- Create a new project System 300V.
- Add a profile rail from the hardware catalog.
- In the hardware catalog the CPU with Profibus master is listed as: Simatic300 > *CPU-300* > *CPU315-2DP* > **6ES7 315-2AF01-0AB0**
- Insert the CPU 315-2DP (**6ES7 315-2AF01-0AB0**).
- Type the Profibus address of your master.
- Click on "DP", choose the operating mode "DP master" under *object properties* and confirm your entry with OK.
- Click on "DP" with the right mouse button and choose "add master system". Create a new Profibus subnet.



Configure CPU section

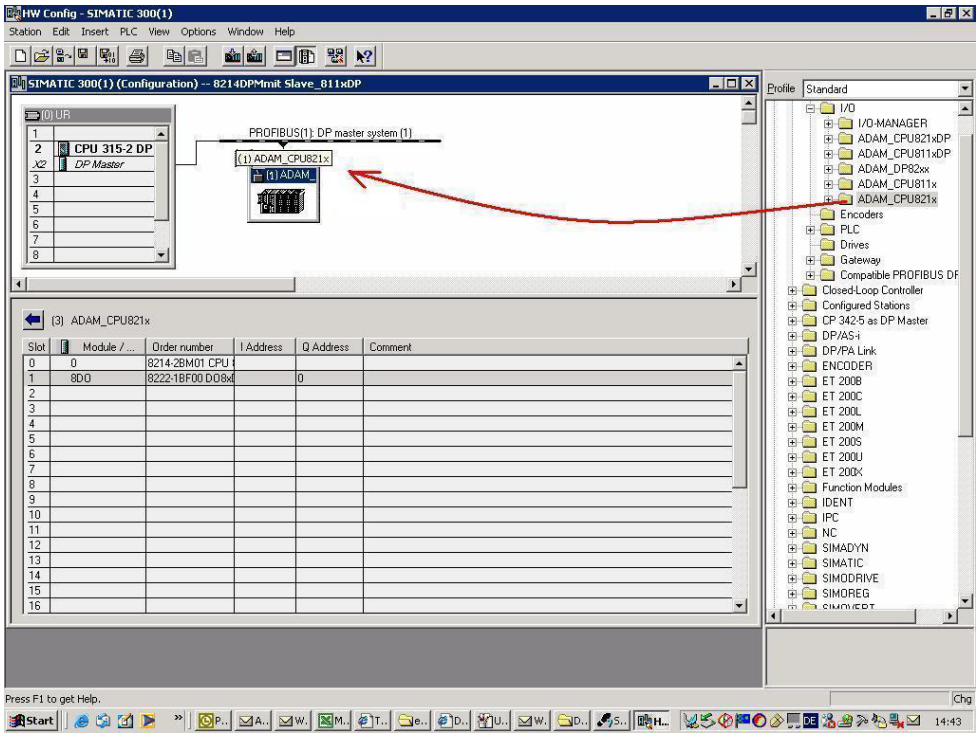
To be, like mentioned above, compatible to the STEP®7 projecting tool from Siemens, you have to include the CPU section explicitly.

- Add the System "DP8000" to the Profibus subnet. You find this in the hardware catalog under *PROFIBUS DP > Additional field devices > IO > DP8000*.
- Assign the Profibus address 1 to this slave.
- Place the CPU 821x-2BM01 from Advantech at the plug-in location 0 in your configurator. **The plug-in location 0 is mandatory!**

Now the project engineering of your Profibus-DP master and your CPU is finished. The following shows how to include the directly plugged-in System 8xxx modules.

Configure central periphery

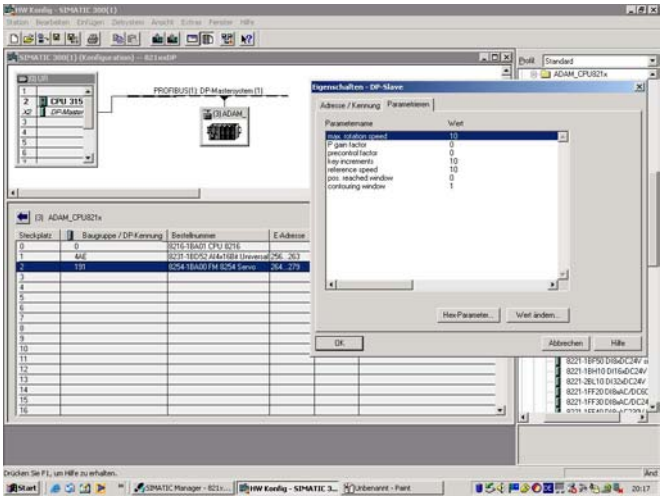
To include the modules plugged-in at the bus, you drag the according System 8xxx modules from the hardware catalog under *ADAM821x* and drop it on the locations below the CPU. Start with plug-in location 1.



Parameterize modules

System 8xxx modules may get up to 16Byte parameter data from the CPU. Using the STEP®7 manager from Siemens, you may assign parameters for parameterizable System 8xxx modules at any time.

For this purpose double-click in the plug-in location overview on the concerning module. The following picture shows the parameterization of the positioning module FM 8254:

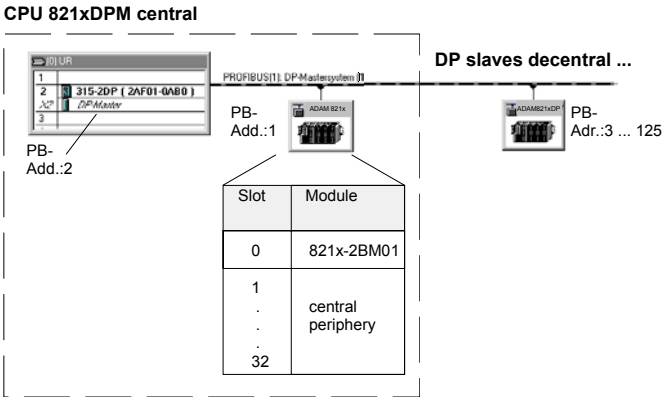


Configure DP-Slaves

For the project engineering of Profibus-DP slaves coupled at the DP master of the CPU 821xDPM, you approach analog to the ADAMCPU821x system.

Search the concerning Profibus-DP slave ADAM CPU 821x_DP in the hardware catalog and drag&drop it in the subnet of your master.

Assign a valid Profibus address to the DP slave (> 3).



Project transfer

Outline

There are 2 possibilities for the transfer of your project into the CPU:

- Transfer via MPI
- Transfer via MMC at deployment of a MMC reading device

Transfer via MPI

The structure of a MPI net is in principal same to the one of a 1.5MBaud Profibus net. This means, the same rules are valid and you use the same components for assembly.

Per default, the MPI net is running with 187kBaud.

Every bus participant identifies itself at the bus with an unique MPI address.

You connect the single participant by means of bus interface plugs and the Profibus bus cable.

Terminating resistor

A conductor has to be terminated with its ripple resistor. Herefore you activate the terminating resistor at the first and the last participant of a net or a segment.

Please regard, that the participants with the activated terminating resistors are always supplied with voltage during start-up and operation.

Approach

- Connect your PU resp. your PC via MPI with the CPU.
  If your programming device has no MPI-interface, you may use the ADAM "Green Cable" to establish a serial point-to-point-connection from your PC to MPI.
  The "Green Cable" has the order no. ADAM-8950-0KB00 and may only be used with the ADAM CPUs of the Systems 8xxx.
  Please regard the notes about the "Green Cable" in chapter 1!
- Configure the MPI-interface of your PC.
- With **PLC** > *Load to module* in your projecting tool, you transfer your project into the CPU.
- For an additional backup of your project on the MMC, you plug in a MMC and transfer the user application to the MMC by means of **PLC** > *Copy RAM to ROM*.
  During the writing process the "MC"-LED at the CPU is blinking. Due to the system, a successful writing operation is signaled to soon. The writing operation is only finished, when the LED extinguishes.

Note!

Further information to MPI is to find under "Commissioning".

Configure MPI

Hints for the configuration of a MPI-interface is to find in the documentation of your programming software.

Here, only the usage of the ADAM "Green Cable" together with the programming tool from Siemens shall be shown.

The "Green Cable" establishes a serial point-to-point connection between the COM-interface of the PC and the MP²I-interface of the CPU.

Attention!

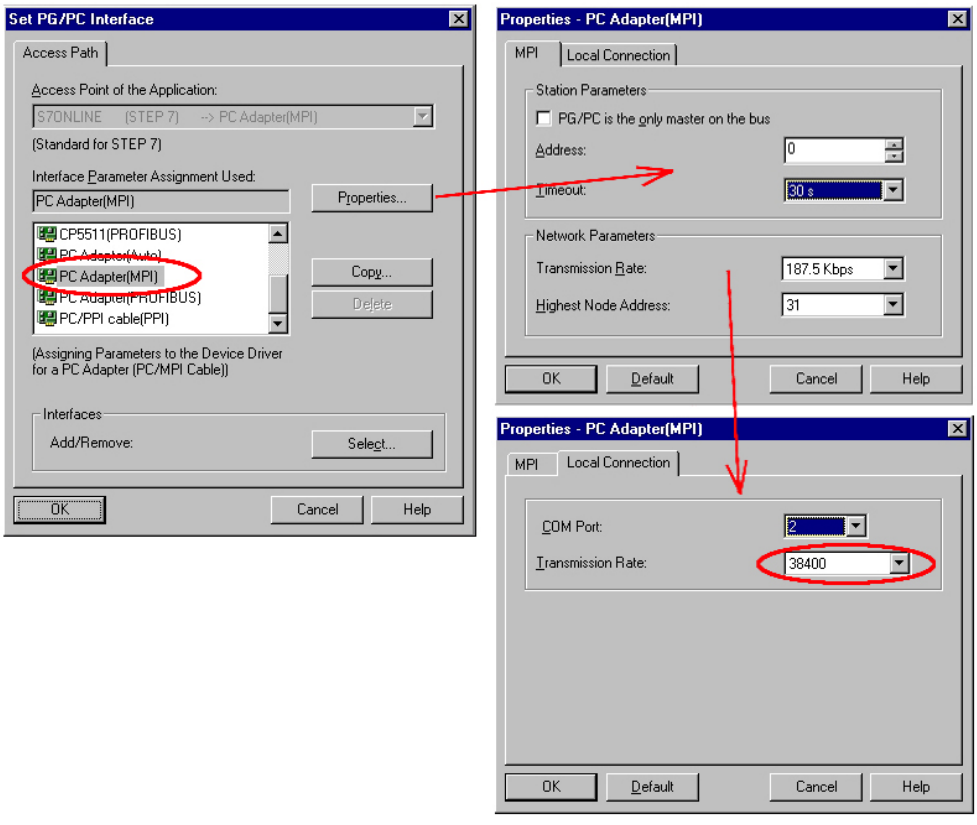
Please regard, that you may use the Green Cable exclusively at the MP²I-interfaces of the Systems ADAM 8xxx from Advantech!

Please regard the notes about the "Green Cable" in chapter 1!

Approach

- Start the STEP®7 manager from Siemens.
- Choose **Options** > *Set PG/PC interface*

  → The following dialog window appears, where you may configure the wanted MPI-interface:



- Choose "PC Adapter (MPI)" from the list; if necessary you have to add it first.
- Click on < Properties >

→ In the following 2 sub dialogs you may configure your PC adapter, like shown in the picture.

Note!

Please make sure to adjust the transmission rate to 38400Baud when using the "Green Cable".

Usage of the MMC

As external storage medium the Multi Media Card (MMC) is used. The MMC is available from Advantech under the order number ADAM-8953-0KX00.

The reading of the MMC takes always place after an OVERALL_RESET.

The writing on the MMC starts via a WRITE command from the hardware configurator from Siemens or via a MMC reading device from Advantech (Order No.: ADAM-8950-0AD00). Thus it is possible to create applications at the PC, copy them to the MMC and transfer them to the ADAM CPU by plugging-in the MMC.

The MMC modules are delivered from Advantech preformatted with the file system FAT16.

Required Files

There may be several projects and subfolders on one MMC storage module. Therefore you have to make sure, that your recent project is located in the root directory and has the following file name: **S7PROG.WLD.**

Transfer CPU → MMC

If there is a plugged MMC in the CPU, the content of the battery buffered RAM is transferred to the MMC by means of a WRITE command.

The write command is started from the hardware configurator from Siemens via **PLC** > *Copy RAM to ROM.*

During the writing process the yellow "MC"-LED of the CPU is blinking.

Transfer MMC → CPU

The transfer of the user application from the MMC into the CPU always takes place after an OVERALL_RESET. The blinking of the yellow LED "MC" on the CPU marks the transfer process.

If there is no valid user application on the plugged MMC or the transfer fails, an OVERALL_RESET of the CPU takes place and the STOP-LED blinks for three times.

Now the master is linked up to the network with the following default parameters:

Default-Bus-Parameter: Address: 1, Transfer rate: 1.5MBaud

Note!

If the user application is larger than the user memory of the CPU, the content of the MMC is not transferred into the CPU.

If you initialize the writing process without plugged MMC, this leads to an error message about inadequate memory.

Before transferring the user application to the MMC it is convenient to initialize a compression.

DP master operating modes

STOP → RUN (automatically)

After POWER_ON and with valid configuration data in the CPU, the master switches automatically into RUN. There is no operating mode lever for the master.

Now the communication to the DP slaves is established. During this time only the RUN-LED is on. At successful communication and valid bus parameters, the DP master switches to Data Exchange (DE). The LEDs RUN and DE are blinking.

At invalid parameters the DP master switches to RUN and monitors a parameterization error via the IF-LED.

Now the DP master is linked up at the bus with the following default bus parameters:

Default-Bus-Parameter: Address:1, Transfer rate:1.5MBaud.

RUN

In the RUN mode, the RUN- and the DE-LEDs are blinking. Now data may be exchanged. If an error occurs, like e.g. DP slave failure, this is shown at the DP master via the ERR-LED and an alarm is initialized to the CPU.

Note!

If the CPU goes into STOP during operation, the DP master stays in RUN.

By means of the BASP signal all outputs of the peripheral modules, linked up via the DP slaves, are set to zero.

Commissioning and Start-up behavior

Check list for commissioning

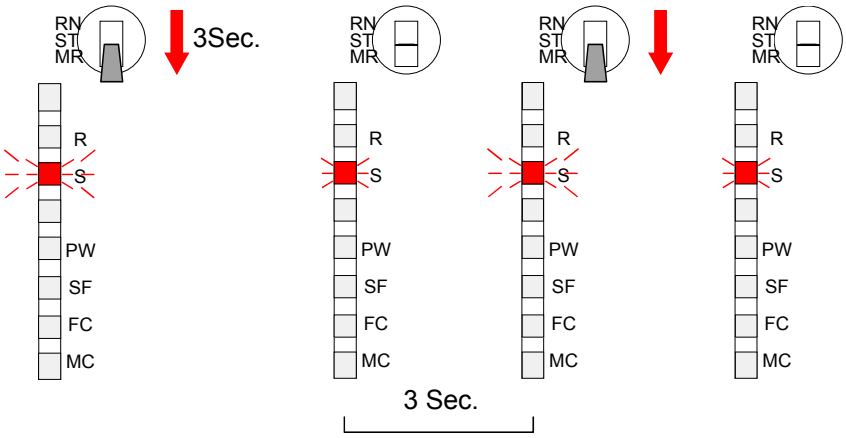
- Turn off your power supply
- Build up your system
- Cable your system
- Plug in your MMC with CPU program and Profibus project
- Turn on your power supply
- For transferring your project from the MMC into the CPU, request an OVERALL_RESET

Turn on power supply

Turn on the power supply. The course of events, described under "start-up behavior", is following.

OVERALL_RESET

The following picture shows the approach once more:



Default boot procedure, as delivered

When the CPU is delivered it has been reset.

After a STOP→RUN transition the CPU switches to RUN without program.

After NETZ_EIN (power on), the DP master tries to get parameters from the CPU.

For the master doesn't get valid parameters from the CPU, it starts with default parameters (Addr.:1, 1.5MBit) from its ROM and shows this via the IF-LED.

Boot procedure with valid data in the CPU

The CPU switches to RUN with the program stored in the battery buffered RAM.

The DP master receives valid parameters and starts with them.

Boot procedure with valid Memory Card

The reading of a MMC only takes place after an OVERALL_RESET.

After the OVERALL_RESET, the DP master proofs the validity of the parameters in the CPU.

If these are valid, they are taken over.

If they are invalid, the CPU switches to STOP and the DP master starts with the default values.

Start-up at empty battery

The battery is loaded directly via the integrated power supply by means of a special load electronic and provides a buffer of max. 30 days. Is this time exceeded, there may be a total discharge of the battery and the battery buffered RAM is erased.

Now the CPU proceeds an overall_reset. If a MMC is plugged-in, the application on it is transferred into the RAM and the DP master is supported with parameters.

If these are valid, the DP master links-up to the bus with those parameters.

If they are not valid, the master starts with default values of its ROM (Add.:1; 1.5MBit) and monitors this via the IF-LED.

Depending on the operating mode selected at the module, the CPU switches to RUN resp. stays in STOP.

This procedure is fixed in the diagnostic buffer by means of the following entry: "Automatic start OVERALL_RESET (unbuffered NETZ_EIN/ Power_on)".

Chapter 6 Deployment of the CPU 821xDP

Outline

This chapter describes applications of the CPU 821xDP under Profibus. You'll get all information for the deployment of an intelligent Profibus-DP slave. An example for the CP 821xDP and the CPU 821xDPM concludes the chapter.

This chapter contains a description of:

- the principles of Profibus-DP
- configuration and parameterization of a CPU 821xDP
- diagnostic and status messages
- assembly and commissioning
- communication example

Contents

Principles

General

Profibus is an international open fieldbus standard for building, manufacturing and process automation. Profibus defines the technical and functional properties of a serial fieldbus system that can be used to create a network of distributed digital field automation equipment on the lower (sensor-/actuator level) to middle performance level (process level).

Profibus comprises various compatible versions. The specifications contained in this description refer to Profibus-DP.

Profibus-DP

Profibus-DP is particularly suitable for applications in production automation. DP is very fast, offers plug&play and is a cost-effective alternative to parallel cabling between PLC and the distributed/decentralized periphery. Profibus-DP is conceived for high-speed data exchange on the sensor-actuator level.

The data exchange happens cyclically. During one bus cycle the master reads the input values from the various slaves and writes new output information into the slaves.

Master and slaves

Profibus distinguishes between active stations (masters) and passive stations (slaves).

Master equipment

Master equipment controls the data traffic on the bus. There may be also several masters at one Profibus. This is referred to as multi master operation. The bus protocol establishes a logical token ring between the intelligent devices connected to the bus.

A master like in the CPU 821xDPM is allowed to send unsolicited messages if it has the bus access permission (Token). In the Profibus protocol these masters are also referred to as active stations.

Slave equipment

Typical slave equipment holds data of peripheral equipment, sensors, actuators, transducers. The ADAM Profibus couplers are modular slave devices that transfer data between the System 8xxx periphery and the leading master.

These devices do not have bus access permission in accordance with the Profibus standard. They may only acknowledge messages or transfer messages to a master if requested by the respective master. Slaves are also referred to as passive stations.

Communication

The bus communication protocol provides two procedures for accessing the bus:

Master to Master

Communication with the master is also referred to as token passing procedure. Token passing guarantees that the station receives access permission to the bus. This access right to the bus is passed between the stations in form of a "token". A token is a specific message that is transferred via the bus.

When a master possesses the token it also has the access right to the bus and is allowed to communicate with all other active and passive stations. The token retention time is defined at system configuration. When the token retention time has expired, the token is passed along to the next master that acquires the bus access rights with the token so that it may communicate with all other stations.

Master slave procedure

Data is exchanged in a fixed repetitive sequence between the master and the slaves assigned to the respective master. When you configure the system you define which slaves are assigned to a certain master. You may also specify which DP slave is included in the cyclic exchange of application data and which ones are excluded.

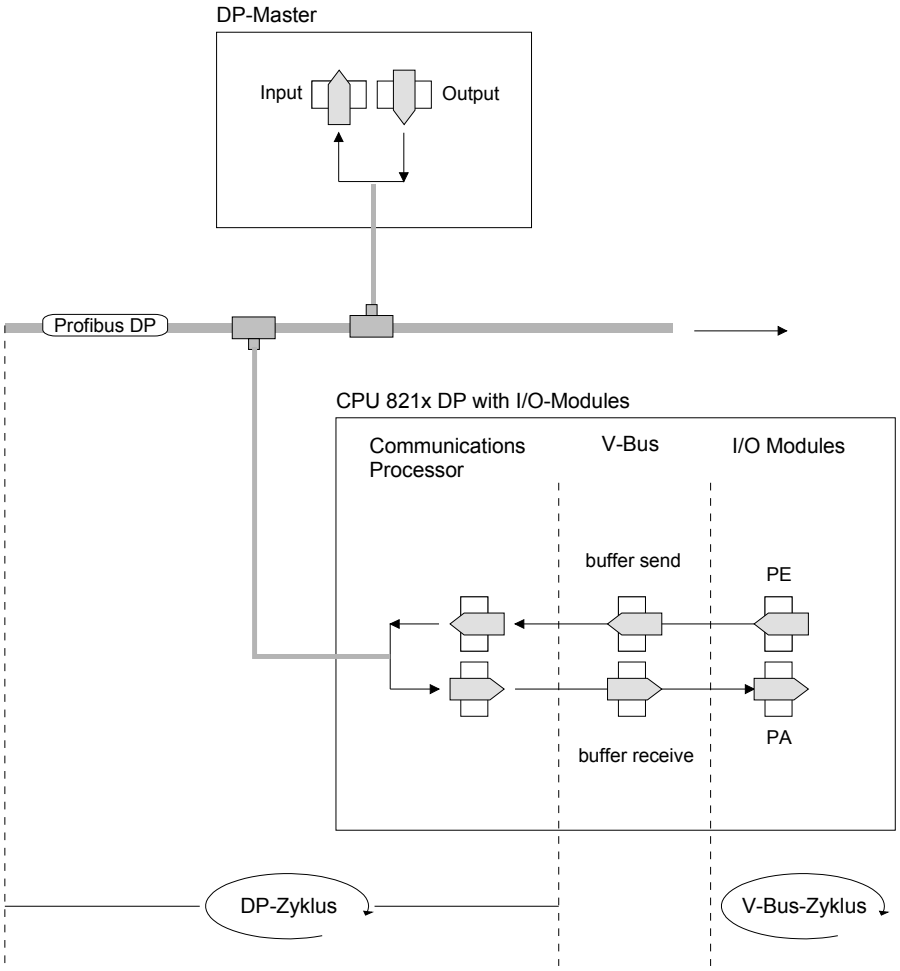
The master slave data transfer is divided into parameterization, configuration and data transfer phases. Before a DP slave is included in the data transfer phase the master verifies during the parameterization and configuration phase, whether the specified configuration agrees with the effective configuration. This verification process checks the device type, format and length as well as the number of inputs and outputs. This provides you with effective protection against configuration errors.

The master handles application data transfers independently. In addition you may also send new configuration data to a bus coupler.

If in the status DE „Data Exchange“, the master is sending new basic data to the slave and the responding telegram of the slave transfers the recent input data to the master.

The principle of data transfer operations

The data exchange between the DP master and the DP slave is performed in a cycle using send and receive buffers.



PE: Process picture of the inputs

PA: Process picture of the outputs

V-bus cycle

In one V-bus cycle (i.e. backplane bus) all input data of the single modules are collected in the PE and all output data from the PA are transferred to the output modules. After the data exchange is completed, the PE is transferred to the sending buffer (buffer send) and the content of the input buffer (buffer receive) is transferred to PA.

DP cycle

In one Profibus cycle the master contacts all its slaves with a data exchange. There the memory areas assigned to the Profibus are written resp. read.

Afterwards the DP master transmits data of the input area to the receive buffer of the communication processor and the data of the send buffer is transferred into the Profibus output area.

The DP master to DP slave data exchange on the bus is repeated cyclically and does not depend on the V-bus cycle.

V-bus cycle vs. DP cycle

To guarantee a simultaneous data transfer the V-bus cycle time should always be same or lower than the DP cycle time.

In the delivered GSD you'll find the parameter **min_slave_interval = 3ms**.

Thus guarantees that the Profibus data on the V-bus is updated latest every 3 ms. Though you are allowed to execute one data exchange with the slave every 3 ms.

Data consistency

Data is referred to as being consistent, if it has the same logical contents. Data that belongs together is: the high- and low-byte of an analog value (word consistency) and the control and the status byte with the respective parameter word required to access the registers.

The data consistency during the interaction between the peripherals and the controller is only guaranteed for 1 byte. That is, the bits of one byte are acquired together and they are transmitted together. Byte-wise consistency is sufficient for the processing of digital signals.

Where the length of the data exceeds a single byte, e.g. analog values, the data consistency must be expanded. Profibus guarantees consistency for the required length of data. Please ensure that you use the correct method to read consistent data from the Profibus master into your PLC.

For additional information please refer to the manual on your Profibus master as well as the one for the interface module.

Restrictions

If a high-level master fails, this is not recognized automatically by the CPU. You should always pass along a control byte to indicate the presence of the master thereby identifying valid master data.

The example at the end of this chapter also explains the use of the control byte.

Diagnosis

There is a wide range of diagnosis functions under Profibus-DP to allow a fast error localization. The diagnosis data are broadcasted by the bus system and summarized at the master.

Transfer medium

As transfer medium Profibus uses an isolated twisted-pair cable based upon the RS485 interface or a duplex photo cable. The transfer rate is for both methods max. 12Mbaud.

More information about this topic is available at „installation guideline“.

Electrical system over RS485

The RS482-interface is working with voltage differences. Though it is less irritable from failures than a voltage or a current interface. You are able to configure the network as well linear as in a tree structure. Your ADAM Profibus coupler includes a 9pin slot where you connect the Profibus coupler into the Profibus network as a slave.

The bus structure under RS485 allows an easy connection resp. disconnection of stations as well as starting the system step by step. Later expansions don't have any influence on stations that are already integrated. The system realizes automatically if one partner had a fail down or is new in the network.

Optical system via optical waveguide (OWG)

The optical waveguide system uses monochromatically light impulses. The optical waveguide is totally independent from disturbing voltage from other machines. An optical waveguide system is built up linear. Every module has to be connected with two links: one input link and one back. You don't need to terminate the last module.

For the structure is a linear one, connecting and disconnecting stations is not free of consequences.

Addressing

Every partner of the Profibus network has to identify itself with a certain address. This address may be existing only one time in the bus system and has a value between 0 and 125.

At the CPU 821xDPM you choose the address via your software tool.

GSD-file

To configure the slave connections in your own configuration tool, you've got all the information about your ADAM modules in form of an electronic data sheet file.

Structure and content of this file are dictated by the Profibus User Organization (PNO) and can be seen there.

Install this file in your configuration tool. Look for more information in the online help of the according tool.

CPU 821xDP configuration

Outline

In contrast to the ADAM Profibus slave IM 8253DP, the Profibus coupler in the CPU 821xDP is an "intelligent coupler".

The "intelligent coupler" processes data that is available from an input or an output area of the CPU. This area and an area for status and diagnostic data are fixed via the CPU 821xDP properties. Separate memory areas are used for input and for output data. Those areas are to handle with your PLC program.

Due to the system, the address areas occupied by the coupler are not displayed in the hardware configurator from Siemens. For the directly plugged-in System 82xx modules are also included in the periphery address range, there may occur address overlappings during the automatic address allocation.

Note!

For configuring the CPU and the Profibus-DP master a thorough knowledge of the STEP®7 manager and the hardware configurator from Siemens is required!

Possibility of configuration in the STEP®7 manager from Siemens

The address allocation and the parameterization of the directly plugged-in System 82xx modules takes place via the STEP®7 manager from Siemens in form of a virtual Profibus system.

For the Profibus interface is software standardized, ADAM is able to guarantee the availability of the complete functions of the System 82xx modules by including the GSD-file in the manager.

Steps of the CPU 821xDP configuration

To be compatible with the STEP®7 projecting tool from Siemens, you have to execute the following steps:

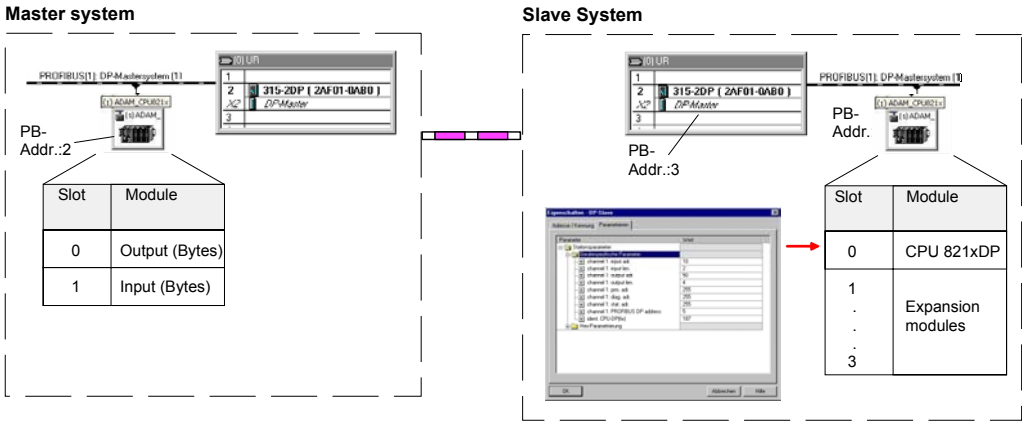
- Project the CPU 315-2DP with DP master system (address 2).
- Add Profibus slave "DP8000" with address 1.
- Include the CPU type **821xDP** at plug-in location 0 of the slave system.
- Adjust the Profibus parameters of the CPU 821xDP.
- Include directly plugged-in periphery modules at the plug-in locations 1...32.
- Transfer project via MPI into the CPU 821xDP

Steps of the master configuration

- Project CPU with DP master system (address 2).
- Add Profibus slave 821xDP (GSD-file required).
- Fix the Profibus input and output areas starting with plug-in location 0.

Reference between master and slave

The following picture illustrates the project engineering at the master and the slave:



Project engineering of the CPU 821xDP

The CPU 821xDP is configured in the STEP®7 Manager from Siemens. Following requirements have to be fulfilled before configuration:

Preconditions

- STEP®7 Manager from Siemens is installed.
- GSD-file has been integrated in the hardware configurator of Siemens.
- There is a possibility for data transfer between PU and CPU.

Install hardware configurator from Siemens

The hardware configurator is part of the STEP®7 projecting tool from Siemens for project engineering. The modules parameterizable via this tool are in the hardware catalog of this tool.

For the deployment of the Profibus-DP slaves of the Systems ADAM 8xxx from Advantech you have to include the modules via an GSD-file from Advantech in the hardware catalog.

Include GSD-file

Start the hardware configurator from Siemens. For including a new GSD-file, no project may be opened.

Open the GSD-file window under **Options** > *Install new GSD-file*. Insert the delivered data device and select the wanted GSD-file. Via <Open> you install the GSD-file.

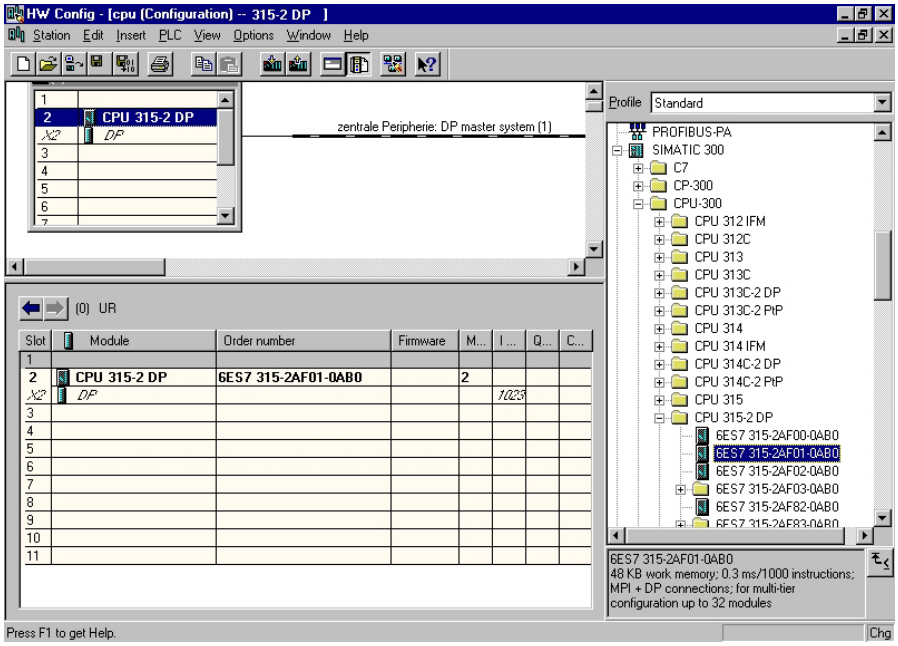
Normally you'll find the ADAM modules included via GSD-file in the hardware catalog under *Profibus-DP* > *Additional Field devices* > *I/O* > *ADAM*.

Create a virtual Profibus system

- Create a new System 300 project.
- Add a profile rail from the hardware catalog.
- Add the CPU 315-2DP (**6ES7 315-3AF01-0AB0**).
- You find the CPU with Profibus master in the hardware catalog under:

Simatic > *CPU-300* > *CPU 315-2DP* > **6ES7 315-3AF01-0AB0**.

- Assign the Profibus address 2 to your master.
- Click at DP and select the operating mode "DP master" under *Object properties*. Confirm your entry with OK.
- By clicking on "DP" with the right mouse button, the context menu opens. Choose "Add master system". Create a new Profibus subnet via NEW. The following picture shows the created master system:



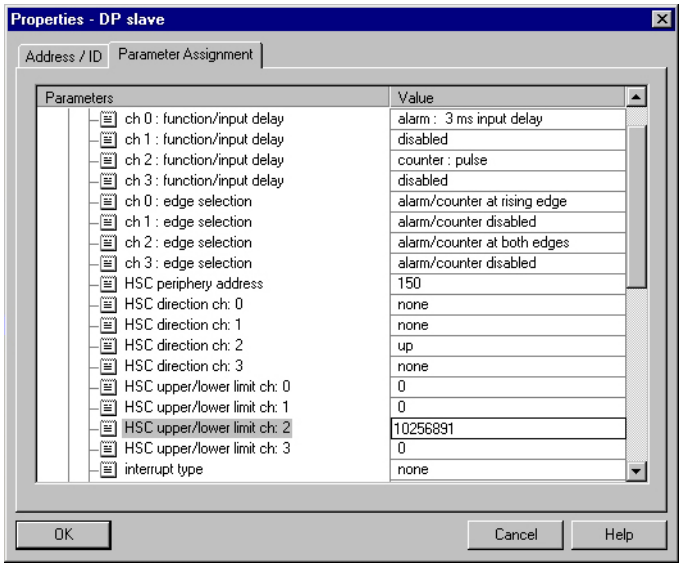
Project CPU 821xDP and modules

To be compatible with the STEP®7 projecting tool from Siemens you have to include the CPU 821xDP explicitly like mentioned above.

- Add the System "ADAM_CPU821x" to your subnet. The module is to find under *PROFIBUS-DP* > *Additional Field devices* > *I/O* > *DP8000*. Assign the Profibus address 1 to the DP slave.
- Place the CPU 821x-2BP01 from Advantech at the plug-in location 0 of the hardware configurator. **Plug-in location 0 is mandatory!**
- You may fix the data areas of the Profibus section in the CPU parameter window. More details are under "Include Profibus section".
- Include your System 82xx modules in the assembled order starting with plug-in location 1.
- If there is a DP master at the backplane bus, you have to include this at the concerning plug-in location, too. Here the configuration takes place via WinNCS from Advantech.
- Save your project.
- Transfer your project via MPI to the CPU 821xDP.
  Please regard the hints in chapter 1!

Including the Profibus section

The Profibus section creates an image of its data area in the addressing space of the CPU. The assignment of the areas happens via the properties of the CPU 821xDP. Via a double-click at the CPU 821xDP you reach the dialog window for parameterization of the data areas for the Profibus slave. Details are to find in the chapter "DP slave parameters".



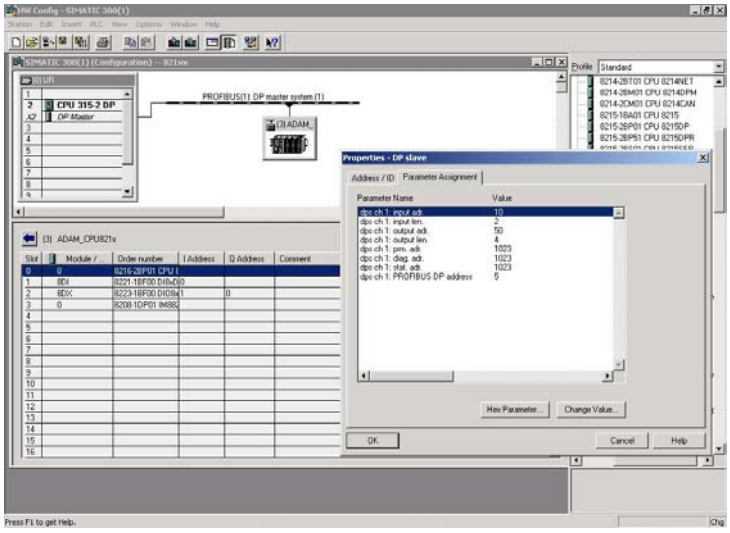
Attention!

Please regard, that the lengths of the data areas are identical at the master and the slave project engineering.

Due to the restrictions imposed by the system, the memory areas of the CPU that are used for the Profibus section may only be displayed in the CPU configuration window.

View in the hardware configurator from Siemens

In the following all relevant dialog windows of the slave project engineering are shown. Here you may also see, how to include your System ADAM 8xxx:

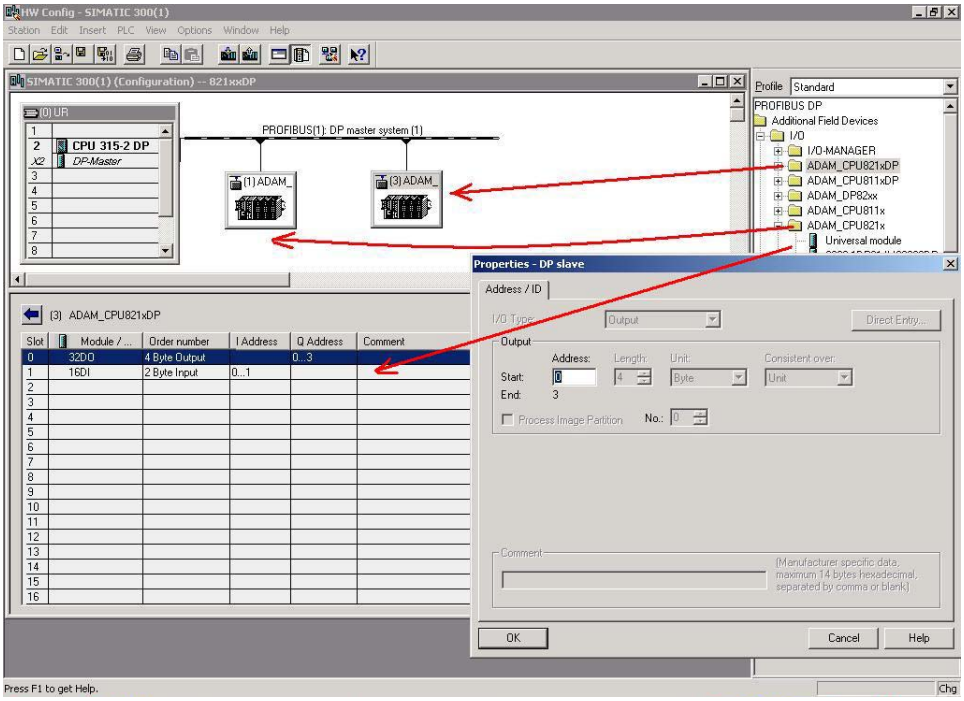


Configuration in the leading DP master

To include the CPU 821xDP into a leading master system, the GSD-file included in the package is required.

- Start your configuration program and configure the Profibus-DP master that will be the leader of your CPU 821xDP.
- Add a DP slave of the station type „CPU821xDP“. This is to find in the hardware catalog under *Profibus-DP > Additional field devices > I/O > ADAM_CPU821xDP*.
- Assign a valid Profibus address.
- Assign memory space from the address area of the CPU to the Profibus section in form of „modules“ for in- and output.
- Save your project and transfer it into the CPU of your master system.

In the following the relevant dialog windows for the master configuration are shown:



Note!

If your DP master system is a System 82xx module from Advantech, you may parameterize the directly plugged-in modules by including a "DP8000" slave system.

To enable the ADAM CPU to recognize the project as central system, you have to assign the Profibus address 1 to your slave system!

DP slave parameters

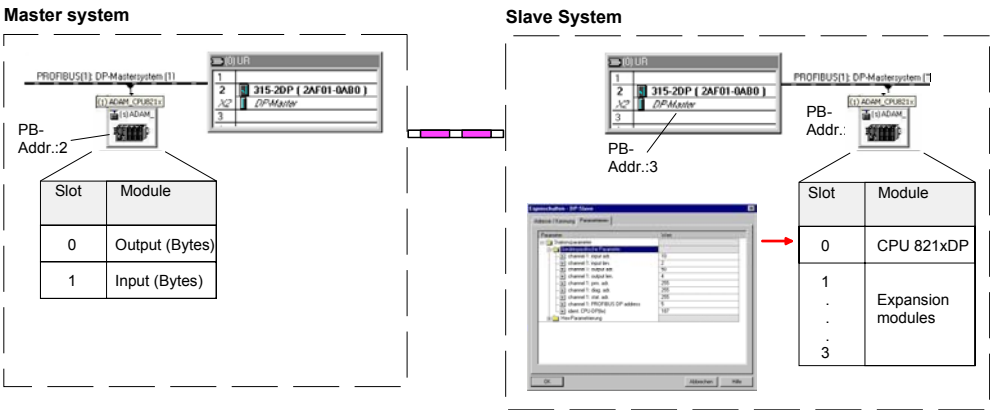
Outline

With an "intelligent slave", the Profibus section writes its data areas into the memory area of the CPU 821xDP. The assignment of the areas happens via the properties of the CPU 821xDP.

The input resp. output areas have to be supported with an according PLC program.

Attention!

The length values for in- and output area have to be identical to the byte values of the master configuration. Otherwise there is no Profibus communication possible and the master notes a slave failure!



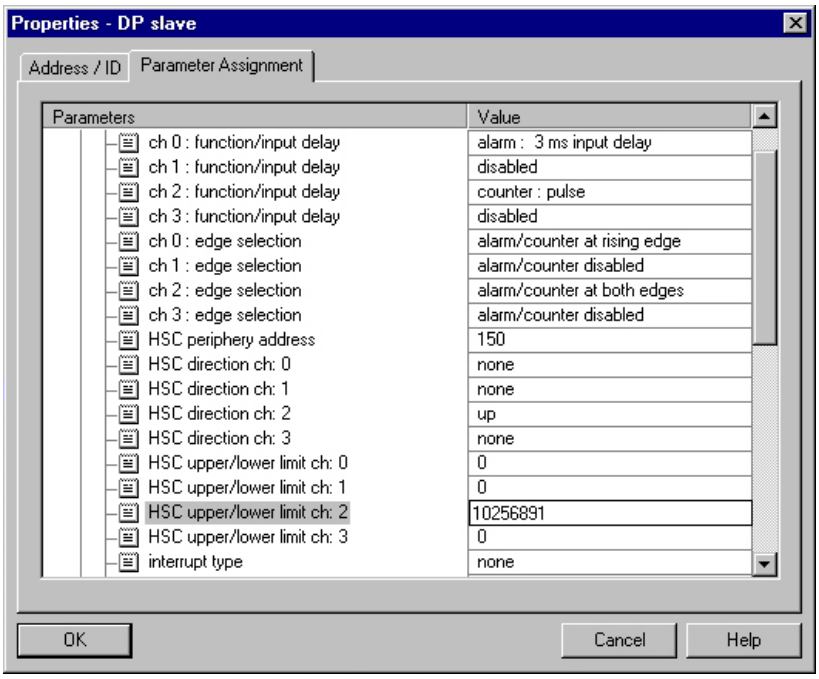
Release CPU areas

As soon as you set a length value to 0, the concerning data does not occupy any memory space in the CPU.

By assigning 255 (memory limit) at the parameters PRN, DIAG and STAT you may also release memory space in the CPU.

Description of the parameter data

Via a double-click at the CPU 821xDP in the hardware configurator from Siemens, the following dialog window for configuring the Profibus slave data areas appears:



input adr, len

Address, from where on the data coming from Profibus shall be stored in the CPU, together with the according length.

The length value of 0 occupies no CPU memory space for the input area.

output adr, len

Address, from where on the data are stored that shall be send via Profibus. here also you predefine the data length via *len*.

The length value of 0 occupies no CPU memory space for the input area.

prm. adr.

The parameter data are an extract of the parameter telegram. The parameter telegram is created during the master configuration and is send to the slave when:

- the CPU 821xDP is in start-up
- the connection between CPU 821xDP and master has been interrupted, like e.g. short-time release of the bus connector.

A parameter telegram exists of Profibus-specific data (bus parameters) and user specific data, where at the CPU 821xDP the in- and output bytes are defined.

The user specific data (Byte 7...31) is mapped into the memory area of the CPU with a fixed length of 24Byte, starting from the address fixed under *prm*.

Thus you may proof the parameters your slave is getting from the master.

diag. adr.

The various diagnostic functions of Profibus-DP allow a fast error detection and localization. The diagnostic messages are transferred via the bus and collected at the master.

The CPU 821xDP is sending diagnostic data at request from the master or in case of an error. The diagnostic data consists of:

- Norm diagnostic data (Byte0...5),
- Device-related diagnostic data (Byte6...10)
- **User-specific diagnostic data (Byte11...15)**

Via *diag* you define the start address, from where on the 6Byte user-specific diagnostic data shall be stored in the CPU.

You may initiate and influence diagnostics by controlled access to this area.

Note!

More detailed information about the structure and the control possibilities on diagnostic messages are to find under "Diagnostic functions".

stat. adr.

The recent state of the Profibus communication is readable from a 2Byte status area in the periphery address area of the CPU, starting from the status address.

Note!

More detailed information about the structure of a status message is to find under "Status message internal to CPU".

Profibus DP address

Via this parameter you assign a Profibus address to your Profibus system.

Release CPU areas

As soon as you assign the length 0, the according data don't occupy any memory space in the CPU.

You may also release CPU memory space by assigning the address range limit (255 resp. 1023 for CPU firmware versions > 2.2.0) to the parameters *prn*, *diag* and *stat*.

Diagnostic functions

Outline

Profibus-DP is provided with an extensive set of diagnostic functions that may be used to locate problems quickly and effectively. Diagnostic messages are transferred via the bus and collected by the master.

The CPU 821xDP transmits diagnostic data when requested by the master or when an error occurs. Since a portion of the diagnostic data (Byte11...15) is located in the peripheral address area of the CPU, you may start the diagnostics and modify the diagnostic data. Diagnostic data consists of:

- standard diagnostic data (Byte 0.. 5),
- equipment related diagnostic data (Byte 6...15).

Structure

The structure of the diagnostic data is as follows:

Standard diagnostic data

Byte 0	Station status 1
Byte 1	Station status 2
Byte 2	Station status 3
Byte 3	Master address
Byte 4	Ident number (low)
Byte 5	Ident number (high)

Device related diagnostic data

Byte 6	Length and code of device related diagnostics
Byte 7	Equipment related diagnostic messages
Byte 8 ... Byte 10	reserved
Byte 11 ... Byte 15	User-specific diagnostic data is mapped into the peripheral addressing range of the CPU and may be modified and send to the master.

Standard diagnostic data

Details on the structure of the standard diagnostic data are available from the literature on the Profibus standards. This documentation is available from the Profibus User Organization.

The slave standard diagnostic data have the following structure:

Byte	Bit 7 ... Bit 0
0	Bit 0: fixed at 0 Bit 1: slave not ready for data transfer Bit 2: configuration data is not identical Bit 3: slave got extern diagnostic data Bit 4: slave does not provide this function Bit 5: fixed at 0 Bit 6: wrong parameterization Bit 7: fixed at 0
1	Bit 0: slave needs new parameterization Bit 1: statistical diagnosis Bit 2: fixed at 1 Bit 3: response control active Bit 4: freeze command received Bit 5: sync command received Bit 6: reserved Bit 7: fixed at 0
2	Bit 0...Bit 6: reserved Bit 7: Diagnostic data overflow
3	Master address after parameterization FFh: Slave is without parameterization
4	Ident number High Byte
5	Ident number Low Byte

Device related diagnostic data

The device related diagnostic data provide detailed information on the slave and the peripheral modules. The length of the device related diagnostic data is fixed at 10Byte.

Byte	Bit 7...Bit 0
6	Bit 0...5: length equipment related diagnostic data 001010: length 10Byte (fixed) Bit 6...7: Code for equipment related diagnostics 00: Code 00 (fixed)
7	Bit 0...Bit 7: equipment related diagnostic message 12h: Error: data length parameters 13h: Error: data length configuration data 14h: Error: configuration entry 15h: Error: VPC3 buffer calculation 16h: Error: missing configuration data 17h: Error: Difference DP parameterization and configuration 40h: User defined diagnostic is valid
8...10	reserved
11...15	**User specific diagnostic data that are stored behind the diagnostic status byte in the process picture of the CPU. This data may be overwritten and forwarded to the master.**

Starting diagnostics

In case of a diagnostic action the contents of Byte 11...15 of the equipment related diagnostic data will be transferred to the process image of the CPU and get a status byte in front.

Where this diagnostic block with a length of 6Byte is located in the process image is definable via the CPU parameters.

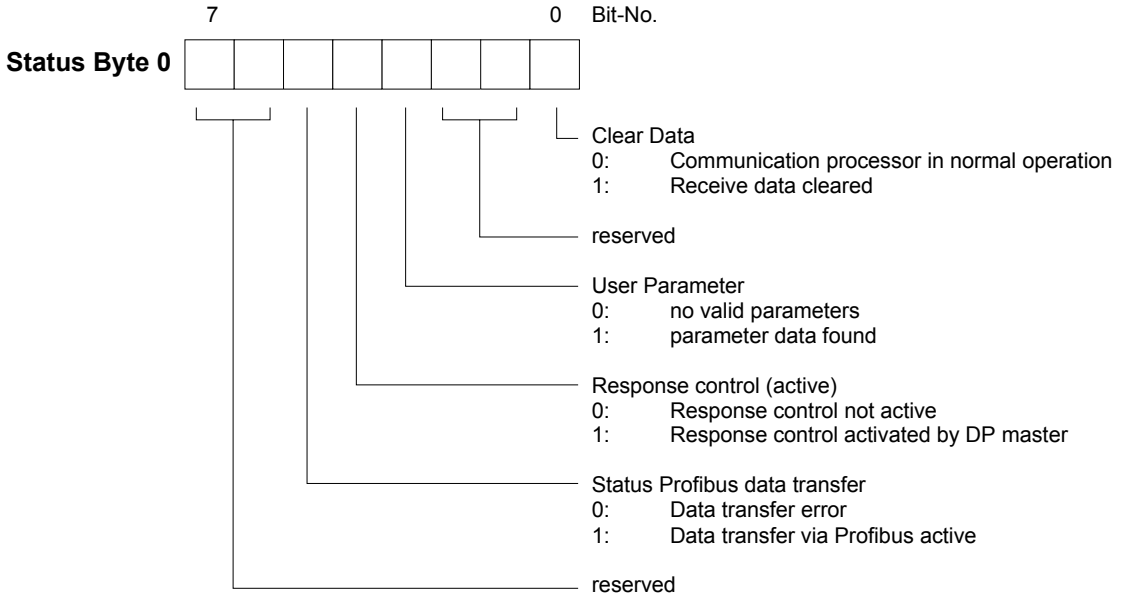
You start diagnostics by means of a status change from 0 $\rightarrow$ 1 in the diagnostic status byte. This transmits the respective diagnostic message to the master. **A status of 0000 0011 is ignored!**

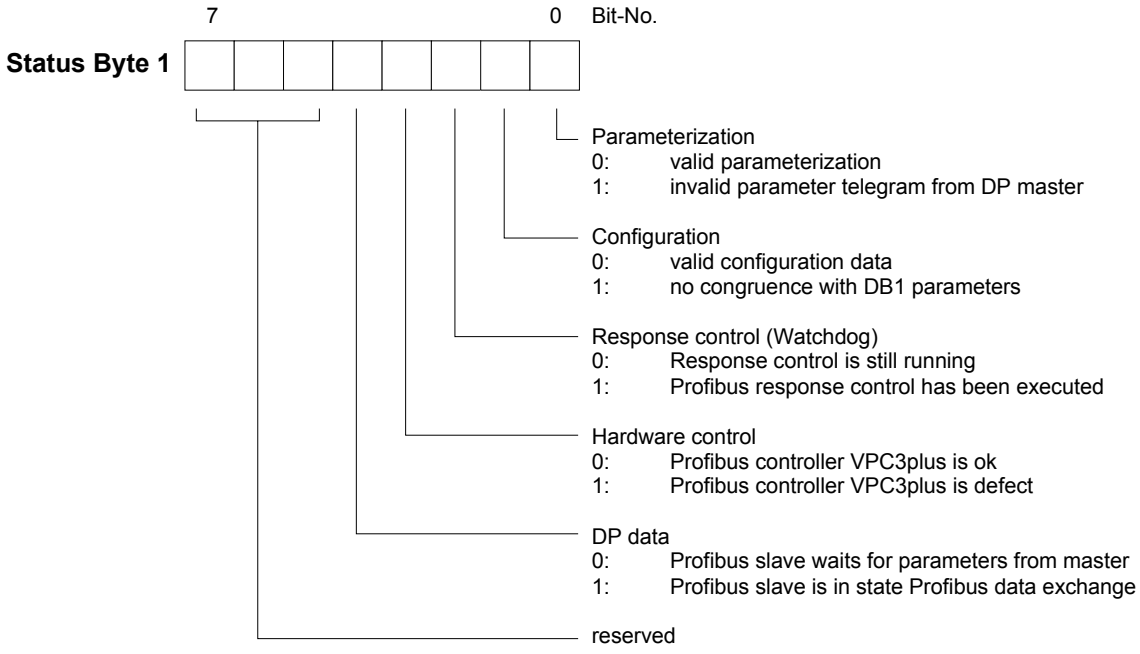
The diagnostic block of the CPU has the following structure:

Byte	Bit 7...Bit 0
0	diagnostic status byte: Bit 0: user specific diagnostic data 0: invalid diagnostic data 1: valid diagnostic data (starting a diagnosis) Bit 1: delete diagnostic 0: diagnostic deletion invalid 1: diagnostic deletion valid Bit 2...Bit 7: reserved
1...5	Bit 0...Bit 7: user specific diagnostic data equal to Byte 11...15 of equipment related diagnostic

Internal status messages to CPU

The current status of the Profibus communication procedure is obtainable from the status messages that are mapped into the peripheral addressing range of the CPU. Status messages consist of 2Byte with the following structure:





Parameter

Clear Data	The transmit and receive buffers are cleared when an error occurs.
reserved	These bits are reserved for future use.
User Parameter	Indicates validity of the parameter data. Parameter data is entered into the master configuration tool.
Response monitoring (active)	Indicates the status of the activation of the response monitor in the leading Profibus master. The slave will terminate communications when the response monitoring time is exceeded.
Profibus data exchange status	Status indicator for master communications. A bad configuration or bad parameters will terminate communications and the respective error is indicated by means of this bit.
Parameter configuration	Displays the status of the configuration data. The length of the configuration data and the number of parameter bytes is analyzed. The configuration is only accepted as being correct if these are equal and if no more than 31Byte of parameter data is transferred.
Configuration	Status indicator of the configuration data that are received from the Profibus master. You define the configuration by means of the master configuration tool.
Response monitoring (Watchdog)	Indicates the status of the response monitor in the Profibus master. This location contains an error when the response monitor has been activated and the required response time has been exceeded in the slave.
Hardware monitoring	This bit is set if the Profibus controller in the CPU 821xDP is defective. In this case you should contact the Advantech.
DP data	Any transfer error on the Profibus will set this error.

Profibus Installation guidelines

Profibus in general

- The ADAM Profibus-DP network must have a linear structure.
- Profibus-DP consists of at least one segment with a minimum of one master and one slave.
- A master must always be used in conjunction with a CPU.
- Profibus supports a max. of 125 stations.
- A max. of 32 stations are permitted per segment.
- The maximum length of a segment depends on the data transfer rate:

9.6...187.5kBaud	→	1000m
500kBaud	→	400m
1.5MBaud	→	200m
3...12MBaud	→	100m

- A maximum of 10 segments may be established. Segments are connected by means of repeaters. Every repeater is regarded as a station.
- All the stations communicate at the same baudrate. Slaves adapt automatically to the baudrate.
- The bus has to be terminated at both ends.
- Masters and slaves may be installed in any sequence.

Note!

When using optical participants, you should place a cover over the socket for the next station at the end of the bus, to avoid eye damage and to eliminate the chance of disturbance by external radiation. Use the rubber inserts for this purpose by inserting them into the two remaining openings of the FO-connector.

Assembly and installation into Profibus

- Assemble your Profibus system complete with the respective modules.
- Set the address of your bus coupler to an unused address.
- Transfer the GSD-file supplied with the modules into your configuration system and configure the system.
- Transfer the configuration into the master.
- Connect the Profibus cable to the coupler and turn the power supply on.

Note!

The Profibus cable has to be terminated with a terminating resistor of the characteristic impedance of the cable. Please ensure to install a terminating resistor at the last station on the bus.

Transfer medium

Profibus employs a screened twisted two-core cable as communication medium, which is based on the RS485 interface.

The RS485 interface employs differential voltages. For this reason it is less sensitive to electrical interference than a voltage or a current based interface. You may configure networks with linear as well as with tree geometry.

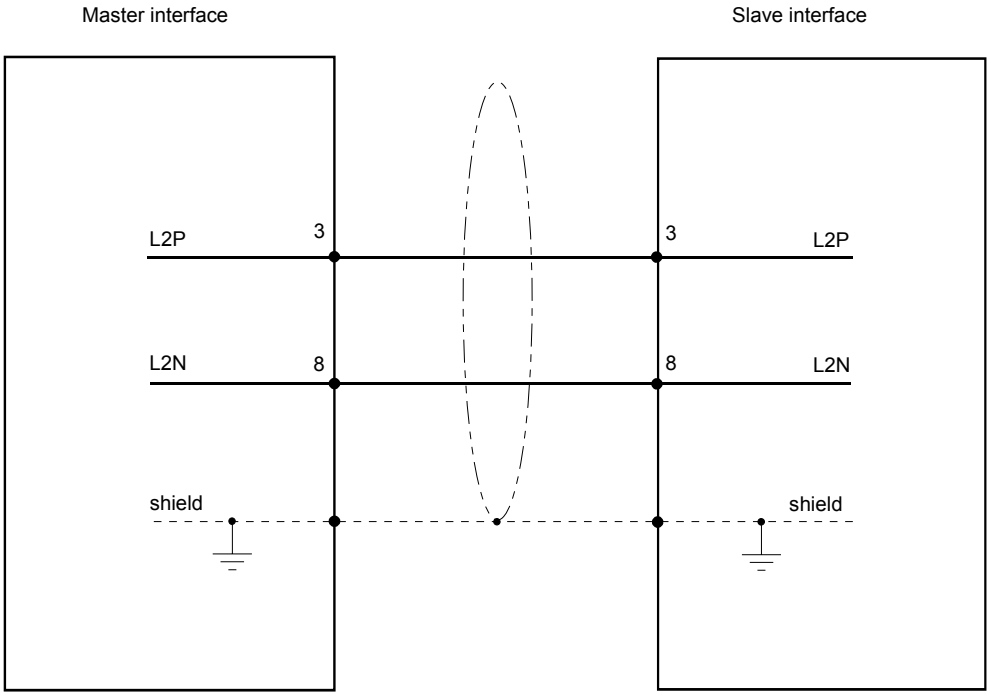
Your ADAM CPU 821xDP carries a 9pin socket. You connect the Profibus coupler directly to your Profibus network as a slave by means of this connector.

Every segment supports a maximum of 32 stations. Different segments are connected by means of repeaters. That maximum length of a segment depends on the data communication rate.

The rate of data transfer of a Profibus-DP link is set to a value between 9.6kBaud and 12MBaud. Slaves are configured automatically. All the stations on the network communicate at the same baudrate.

The structure of the bus is such that stations may be inserted or removed without repercussions or to commission the system in different stages. Extensions to the network do not influence those stations that have already been commissioned. New stations or stations that have failed are detected automatically.

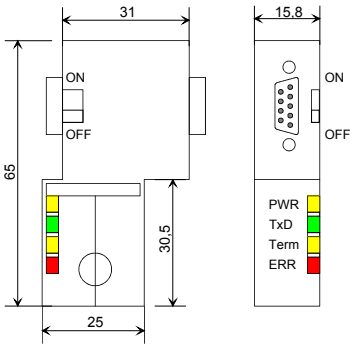
The picture shows a Profibus connection under RS485 with indicated terminating resistors:



Bus connector

In systems with more than two stations all partners are wired in parallel. For that purpose the bus cable must be connected in a continuous uninterrupted loop.

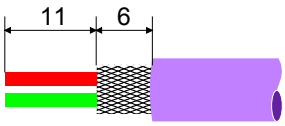
Via the order number ADAM-8972-0DP10 you may order the bus connector "EasyConn". This is a bus connector with switchable terminating resistor and integrated bus diagnosis.

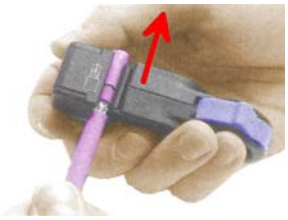


all in mm

To connect this connector please use the standard Profibus cable type B according to EN50170.

Under the order no. ADAM-8905-6AA00 Advantech offers the "EasyStrip" deisolating tool, that makes the connection of the EasyConn much easier.





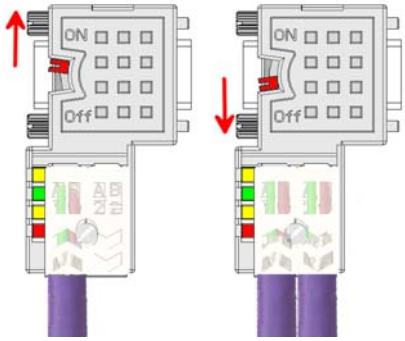
all in mm

Attention!

The bus cable has always to be terminated with the ripple resistor to avoid reflections and therefore communication problems!

Termination

The bus connector is provided with a switch that may be used to activate a terminating resistor.



Attention!

The terminating resistor is only effective, if the connector is installed at a slave and the slave is connected to a power supply.

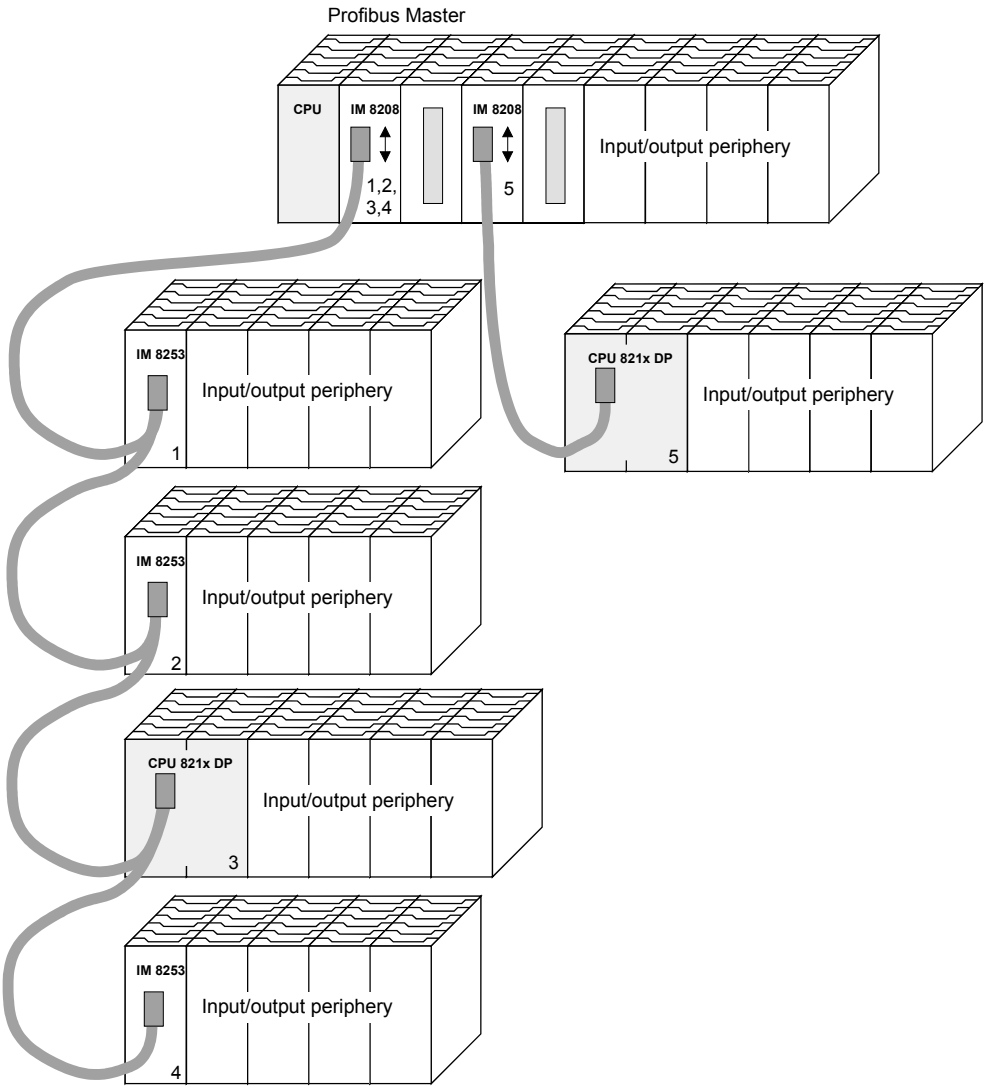
Note!

A complete description of installation and deployment of the terminating resistors is delivered with the connector.

Examples for Profibus networks

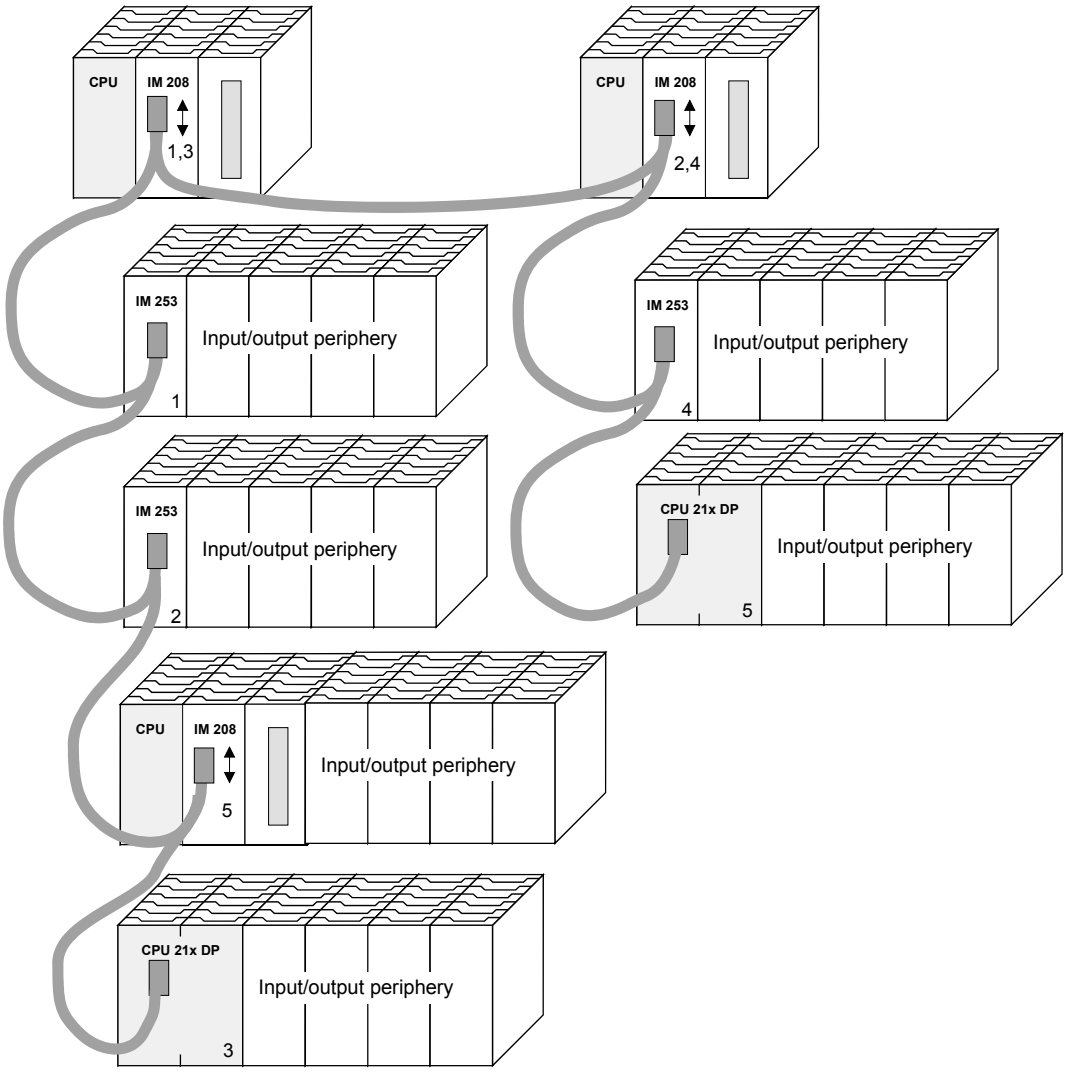
One CPU and several master interfaces

The CPU should have a short cycle time to ensure that the data of slave No.5 (at the right) is always up to date. This scheme is only viable if the slower line (at the left) is connected to slaves that do not require up to date data. This portion of the line should also not be connected to modules that issue alarms.



Multi master system

More than one master and multiple slaves connected to one bus:



Commissioning

Outline

- Install the CPU 821xDP.
- Configure the CPU 821xDP in your master system.
- Configure the I/O periphery that is connected to the backplane bus.
- Connect the CPU 821xDP to your Profibus.
- Turn on the power supply.
- Transfer your project to the CPUs.

Installation

Assemble the CPU 821xDP with the required peripheral modules. Do not exceed the maximum current capacity of your power supply.

Attention!

The bus cable has always to be terminated with the ripple resistor to avoid reflections and therefore communication problems!

Configuration in the master system

Configure your CPU 821xDP in your master system. You may use the Advantech WinNCS package for this purpose. To configure the System 82xx Profibus slave modules from Advantech, you need to include the according GSD-file into the configuration tool from Siemens.

Configuration CPU 821xDP and I/O periphery

The System 82xx peripheral modules that are connected with the CPU 821xDP directly via the backplane bus are automatically mapped into the CPU address area. You may alter the address allocation in the hardware configurator from Siemens at any time.

Voltage supply

The CPU 821xDP has an integrated power supply that has to be provided with DC 24V.

Via the voltage supply not only the CPU is provided with voltage but also the bus coupler and, via the backplane bus, the peripheral modules. Please regard, that the integrated power supply may provide the backplane bus with max. 3A.

Profibus and backplane bus are galvanically separated from each other.

Transfer project

The transfer of the hardware configuration into the CPU takes place via MPI.

- Connect your PU res. the PC via MPI with the CPU.
  If your programming device has no MPI slot, you may use the Advantech Green Cable to establish a serial point-to-point connection.

  The Green Cable has the order no. ADAM-8950-0KB00 and only be used with the ADAM CPUs of the Systems 8xxx.
- Configure the MP interface of your PC.
- With **PLC** > *Load to module* in your projecting tool, you transfer your project into the CPU.
- For the additional security copy of your project on MMC, you plug-in a MMC and transfer the user application to the MMC via **PLC** > *Copy RAM to ROM*.
  During write operation the MC-LED on the CPU blinks. Due to the system, the successful writing is signalized too soon. The write command has only been completed, when the LED extinguishes.

Attention!

Please regard the hints for deploying the Green Cable and the MP^2I jack in chapter 1.

Initialization phase

The Profibus coupler executes a self-test routine after it is powered on. On this occasion it checks the internal operations, the backplane bus communication and the Profibus communication.

When the test is completed successfully, the parameters of the CPU are read and the Profibus slave parameters are checked.

After the successful test, the bus coupler status changes to "READY".

If the bus coupler detects communication errors on the backplane bus, its status first changes to STOP and after a delay of 2 seconds it is initialized again. As soon as the test is completed successfully, the RD-LED blinks.

The DE-LED is turned on when communication is started.

Example

Objective

This example is intended to show the communication between a master CPU 8214DPM and a slave CPU 8214DP.

The counters are to communicate via the Profibus and to be displayed at the output module of the respective partner.

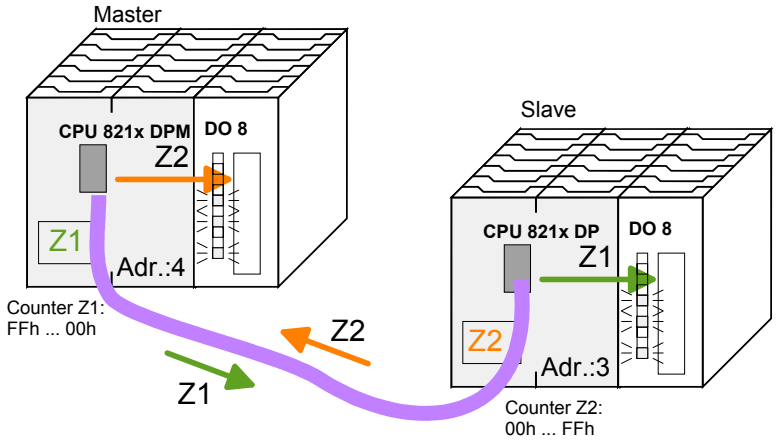
Detailed description of the objective

The CPU 8214DPM shall count from FFh...00h and transfer the count cyclically into the output area of the Profibus master. The master has then to transfer this value to the slave of the CPU 821xDP.

The value received shall be saved in the peripheral input area of the CPU and monitored at the output module (at address 0) via the backplane bus.

On the other hand, the CPU 8214DP should count from 00h to FFh. This count shall also be saved in the output area of the CPU slave and be transferred to the master via Profibus.

This value shall be monitored at the output module (address 0) of the CPU 8214DPM.



Configuration data

CPU 821xDPM

Counter: MB 0 (FFh...00h)

Profibus address:	4	
Input area:	Address 10	length: 2 Byte
Output area:	Address 20	length: 2 Byte

CPU 821xDP

Counter: MB 0 (00h...FFh)

Profibus address:	3	
Input area:	Address 30	length: 2 Byte
Output area:	Address 40	length: 2 Byte
Parameter data:	Address 50	length: 24 Byte (fixed)
Diagnostic data:	Address 60	length: 6 Byte (fixed)
Status data:	Address 100	length: 2 Byte (fixed)

Configuration of CPU 821xDPM (Master)

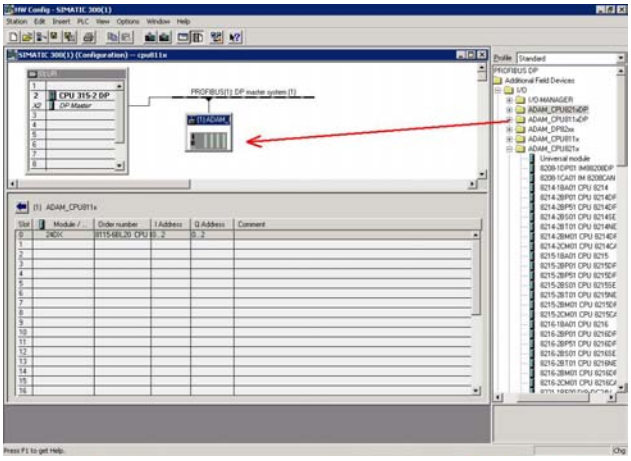
To be compatible to the STEP®7 projecting tool from Siemens, you have to execute the following steps for the System 82xx:

- Start the hardware configurator from Siemens.
- Configure a CPU 315-2DP with DP master system (address 2).
- Add a Profibus slave **ADAM 821x** at address 1.
- Include the CPU **8214-2BM01** at the plug-in location 0 of the slave system.
- Include the output module 8222-1BF00 at plug-in location 1.

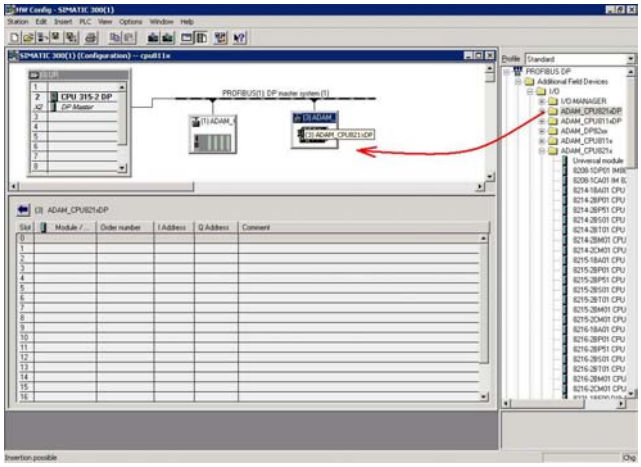
For linking-up the CPU 8214DP you have to execute the following steps after including the GSD-file:

- Add the Profibus slave **"CPU82xxDP"** (address 3). The DP slave is in the hardware catalog under:
  Profibus-DP > Additional field devices > I/O > ADAM > ADAM_CPU82xxDP.
- Assign memory areas of the CPU to the in- and output of the Profibus-DP master section in form of Byte blocks. For this, you have to include the "2Byte output" element on plug-in location 0 and select the output address 20. Include the "2Byte input" element on plug-in location 1 and select the input address 10.
- Save your project.

Including directly connected modules



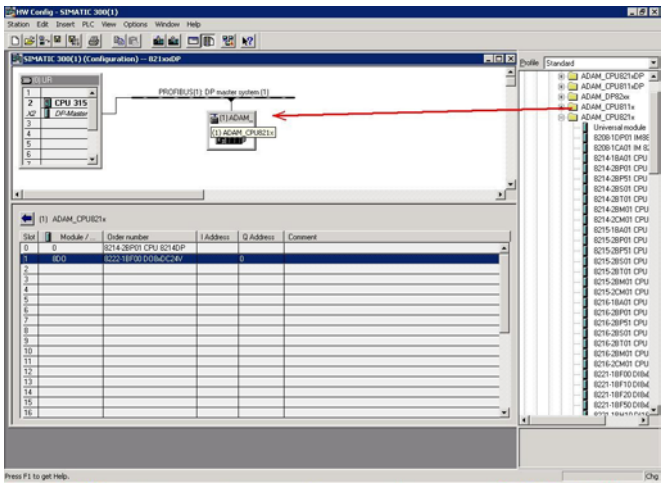
Including CPU 821xDP



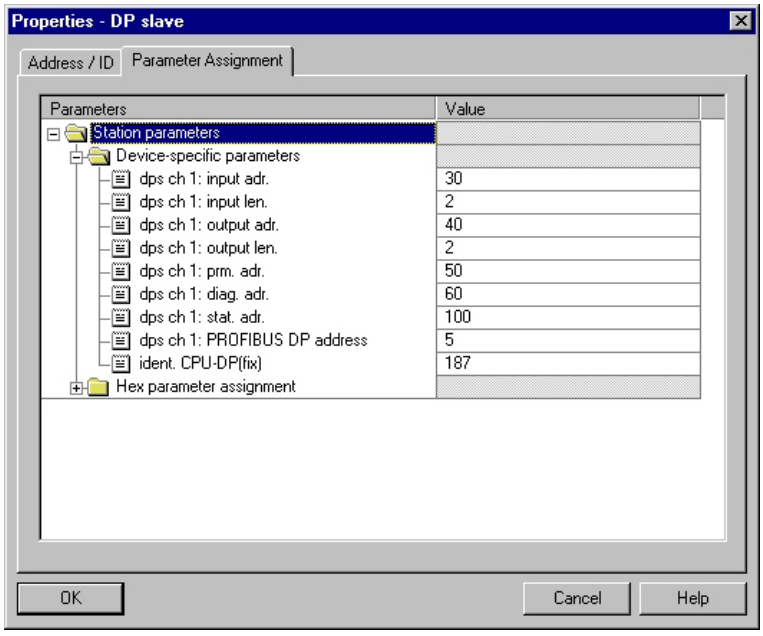
Configuration CPU 821xDP (Slave)

To be compatible to the STEP®7 projecting tool from Siemens, you have to execute the following steps for the System 82xx:

- Start the hardware configurator from Siemens.
- Configure a CPU 315-2DP with DP master system (address 2).
- Add a Profibus slave "DP8000" at address 1.
- Include the CPU **8214-2BP01** at the plug-in location 0 of the slave system.
- Include the output module 8222-1BF00 at plug-in location 1.



- Choose the following parameters in the parameter window of the CPU 8214-2BP01:



- Save your project.

Application program in the CPU 8214DPM

The application program in the CPU 8214DPM has two tasks that are handled by two OBs:

- to test communications by means of the control byte.
  Load the input byte from Profibus and transfer the value to the output module.

  OB 1 (cyclic call)

```
L    B#16#FF
T    AB  20             Control byte for Slave-CPU

L    B#16#FE            Load control value 0xFE
L    EB  10             Was the control byte transferred
<>I                     correctly from the slave CPU?
BEB                     No -> End

                        ------------------------------
                        Data exchange via Profibus

L    EB  11             Load input byte 11 (output data
                        of the CPU214DP) and
T    AB   0             transfer to output byte 0

BE
```

- Read counter value from MB0, decrement, save in MB0 and put it out to CPU 214DP via Profibus.

  OB 35 (timer-OB)

```
L    MB   0             Counter from 0xFF to 0x00
L    1
-I
T    MB   0

T    AB  21             Transfer into output byte 21
                        (input data of CPU214DP)
BE
```

At this point the programming of the CPU 8214DPM is completed. The Profibus communication has also been defined for both sides.

Transfer your project into the CPU 8214DPM via MPI by means of the **PLC**-functions.

Application program in the CPU 8214DP

As shown above, the application program in the CPU has two tasks that are handled by two OBs:

- Load the input byte from the Profibus slave and transfer the value to the output module.

OB 1 (cyclic call)

```
L    PEW 100          Load status data and save in
T    MW 100           flag word

UN   M  100.5         Commissioning by the DP master
BEB                   occurred? No -> End

U    M  101.4         Valid receive data?
BEB                   No -> End
L    B#16#FF          Load control value and compare
L    PEB  30          to control byte
<>I                   (1st input byte)
BEB                   Received data does not contain
                      valid values

L    B#16#FE          Control byte for master-CPU
T    PAB  40
                      --------------------------------
                      Data exchange via Profibus

L    PEB  31          Load peripheral byte 31 (input
                      data from Profibus slave) and
T    AB   0           transfer into output byte 0

BE
```

- Read counter value from MB0, increment, save into MB0 and put it out to CPU 821x via Profibus.

OB 35 (timer-OB)

```
L    MB   0           Counter from 0x00 to 0xFF
L    1
+I
T    MB   0

T    PAB  41          Transfer counter value into
                      peripheral byte 41 (output data
                      of the Profibus slave)
BE
```

Chapter 7 Integrated OBs, SFBs and SFCs

Outline

Here you find the description of the integrated OBs, SFBs and SFCs of the ADAM PLC-CPUs from Advantech for STEP®7 from Siemens.

The information about the listed blocks is valid for the CPUs 811x, 821x and 851x.

Another part of the chapter are the ADAM specific SFCs that are exclusively used with ADAM CPUs.

Note!

Please regard that some ADAM specific SFCs are only integrated in certain CPUs. For example, the SFCs for high-speed counter and pulse duration modulation are only integrated in the CPUs of the System 81xx.

The assignment of the according SFCs to the CPUs is to find in the sub-chapter "ADAM specific SFCs".

Here you'll also find hints for the description of the ADAM specific SFCs.

The following text describes:

- Overview over the integrated OBs, SFBs, SFCs
- ADAM specific SFCs and their inclusion
- SFCs for access to the MMC
- SFCs for the System 8xxx
- SFC for access to TD 200
- SFCs for direct access to page frame and page frame communication

Content

Integrated OBs and SFBs

General

The system program of the CPU 811x offers you some additional functions, that you may use by calling FBs, FCs or OBs. Those additional functions are part of the system program and don't use any work memory. Although the additional functions may be requested, they can not be read or altered.

The calling of an additional function via FB, FC or OB is registered as block change and influences the nesting depth for blocks.

Integrated OBs

The following organization blocks (OBs) are available:

OB	Description
OB 1	Free cycle
OB 10	Clock alarm
OB 20	Delay alarm
OB 35	Prompter alarm
OB 40	Process alarm
OB 80	Cycle time exceeded or clock alarm run out
OB 82	Diagnostic alarm
OB 85	OB not available
OB 86	Slave failure res. restart
OB 100	Reboot
OB 121	Synchronous errors
OB 122	Periphery error at n^{th} access

Integrated SFBs

The following system function blocks (SFBs) are available:

SFB	Label	Description
SFB 0	CTU	Count up
SFB 1	CTD	Count down
SFB 2	CTUD	Count up and down
SFB 3	TP	Create pulse
SFB 4	TON	Create turn-on delay
SFB 5	TOF	Create turn-off delay
SFB 32	DRUM	realize a step-by-step switch with maximum 16 steps

Integrated standard SFCs

Standard SFCs

The following standard system functions (SFCs) are available:

SFC	Label	Description
SFC 0	SET_CLK	Set clock
SFC 1	READ_CLK	Read clock
SFC 2	SET_RTM	Set operating time counter
SFC 3	CTRL_RTM	Start/stop operating time counter
SFC 4	READ_RTM	Read operating time counter
SFC 6	RD_SINFO	Read start information of the recent OB
SFC 12	D_ACT_DP	Activation or deactivation of DP slaves
SFC 13	DPNRM_DG	Read slave diagnostic data
SFC 14	DPRD_DAT	Read consistent user data (also from DP slaves → DP master FW ≥ V3.00)
SFC 15	DPWR_DAT	Write consistent user data (also to DP slaves → DP master FW ≥ V3.00)
SFC 20	BLKMOV	Copy variable inside work memory
SFC 21	FILL	Predefine field inside work memory
SFC 22	CREAT_DB	Create data block
SFC 23	DEL_DB	Delete data block
SFC 24	TEST_DB	Test data block
SFC 28	SET_TINT	Set clock alarm
SFC 29	CAN_TINT	Cancel clock alarm
SFC 30	ACT_TINT	Activate clock alarm
SFC 31	QRY_TINT	Request clock alarm
SFC 32	SRT_DINT	Start delay alarm
SFC 33	CAN_DINT	Cancel delay alarm
SFC 34	QRY_DINT	Request delay alarm
SFC 36	MASK_FLT	Mask synchronous error events
SFC 37	DMASK_FLT	Demask synchronous error events
SFC 38	READ_ERR	Read event status register
SFC 41	DIS_AIRT	Delay alarm events
SFC 42	EN_AIRT	Cancel alarm event delay
SFC 43	RE_TRIGR	Re-trigger cycle time surveillance
SFC 44	REPL_VAL	Transfer replacement value into ACCU1
SFC 46	STP	Switch CPU in STOP
SFC 47	WAIT	Delay program processing additionally to delay time
SFC 49	LGC_GADR	Search the plug-in location concerning to a logical address
SFC 50	RD_LGADR	Search all logical addresses of a block

continued ...

... continue standard SFCs

SFC 51	RDSYSST	Read information from system state list
SFC 52	WR_USMSG	Write user entry into diagnostic buffer (sending via MPI in preparation)
SFC 54	RD_DPARM	Read predefined parameters
SFC 55	WR_PARM	Write dynamic parameters (only for analog, digital blocks, FM350, CP340 / not possible via Profibus)
SFC 56	WR_DPARM	Write predefined parameters (only for analog, digital blocks, FM350, CP340 / not possible via Profibus)
SFC 57	PARM_MOD	Parameterize block (only for analog, digital blocks, FM350, CP340 / not possible via Profibus)
SFC 58	WR_REC	Write record set (only for analog, digital blocks, FM350, CP340 / not possible via Profibus)
SFC 59	RD_REC	Read record set (only for analog, digital blocks, FM350, CP340 / not possible via Profibus)
SFC 64	TIME_TICK	Read millisecond timer
SFC 65	X_SEND	Send data to external partner
SFC 66	X_RCV	Receive data from external partner
SFC 67	X_GET	Read data from external partner
SFC 68	X_PUT	Write data to external partner
SFC 69	X_ABORT	Interrupt connection to external partner

ADAM specific SFCs

General

The integrated SFCs are programmed in Assembler and for this they have a fast processing time. They do not occupy space in the internal program memory.

The integrated blocks are called in the user application.

The following pages show the ADAM specific blocks that may be called for special functions in the control application.

Assignment table SFC ←→ CPU

For not every SFC is integrated in every CPU family, this table shows the assignment between SFC and CPU.

The first number specifies the CPU family, e.g. 811x means: CPU 811x .

SFC	Label	Description	811x	821x	851x
SFC 220	MMC_CR_F	Create file on MMC	✓	✓	✓
SFC 221	MMC_RD_F	Read file on MMC	✓	✓	✓
SFC 222	MMC_WR_F	Write to file on MMC	✓	✓	✓
SFC 223	PWM	Parameterize pulse duration modulation	✓		
SFC 224	HSC	Parameterize high-speed counter	✓		
SFC 225	HF_PWM	Parameterize HF pulse duration modulation (up to 50kHz)	✓		
SFC 227	TD_PRM	Parameterization for TD200 communication	✓	✓	✓
SFC 228	RW_Kachel	Read/write page frame		✓	✓
SFC 230	Send	Send via page frame (page frame comm.)		✓	✓
SFC 231	Receive	Receive via page frame (page frame comm.)		✓	✓
SFC 232	Fetch	Fetch via page frame (page frame comm.)		✓	✓
SFC 233	Control	Control for page frame communication		✓	✓
SFC 234	Reset	Reset for page frame communication		✓	✓
SFC 235	Synchron	Synchron for page frame communication		✓	✓
SFC 236	Send_All	Send_All via page frame (page frame comm.)		✓	✓
SFC 237	Recv_All	Receive_All via page frame (page frame com.)		✓	✓
SFC 238	Control1	Control for page frame communication with Type ANZW: pointer and parameter IND.		✓	✓

For a better overview, in the following every SFC repeats an extract of this assignment table.

The next pages describe the ADAM specific SFCs.

Include ADAM library

Outline

The ADAM specific SFCs are included in consignment in form of libraries. The libraries are self-extracting exe-files.

When you want to use the ADAM specific SFCs, you have to import them into your project.

Please follow this steps:

- Execute F200000z_Vxxx.exe to extract the library
- "De-archivate" the library
- Open the library and transfer SFCs into the project

Extract F200000z_Vxxx.exe

The self-extraction of F200000z_Vxxx.exe is started via a double-click on the file. Choose a destination directory and select <Extract>.

The following structure is created in the destination directory:

De-archivate library

To de-archivate the SFC library you start the STEP®7 manager from Siemens. Via **File** > *De-archivate*, you open a dialog window to select an archive. The SFC library is to find in the directory structure above under ADAM_S7. The file name is ADAM.ZIP.

Choose ADAM.ZIP and click on <open>.

Choose a destination directory where to store the blocks and start the extraction via <OK>.

Open library and transfer SFCs into project

After the extraction, open the library.

Open your project and copy the required SFCs from the library into the directory "Blocks" of your project.

Now your user application allows the access to the ADAM specific blocks.

SFC 220 MMC_CR_F

11	21	51
x	x	x

Description

Deploying this block, you may create a new res. access an existing file when a MMC is plugged-in.

As long as you do not open another file, you may access this file via read/write commands.

Restrictions

For deploying the SFCs 220, 221 and 222, you have to regard the following restrictions:

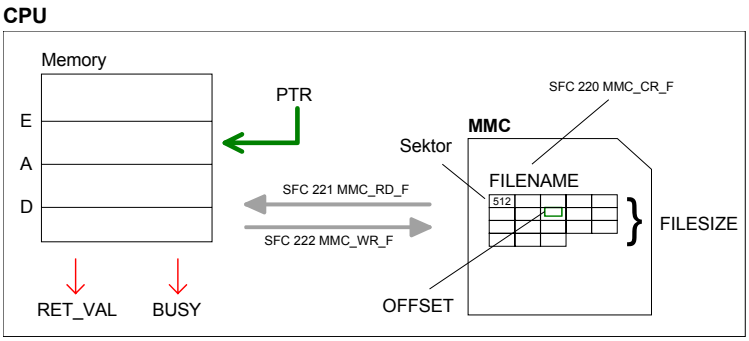
- A read res. write access to the MMC is only possible after creation res. opening of the file via SFC 220.
- The data on MMC must be unfragmentated, for only complete data blocks may be read res. written.

When transferring data to the MMC from an external reading device, they may be fragmentated, i.e. the data is divided into blocks. This may be avoided by formatting the MMC before the write access.

At a write access from the CPU to the MMC, the data is always stored unfragmentated.

- When opening an already existing file, you have to use the same file name and file size that you used at creation of this file.
- A MMC is structured into sectors. Every sector has a size of 512Byte. Sector overlapping writing or reading is not possible. Access to sector overlapping data is only possible by using a write res. read command for every sector. By giving the offset, you define the according sector.

The following picture shows the usage of the single SFCs and their variables:



Note!

For read and write accesses to the MMC, you firstly have to open the file with SFC 220!

Parameters

Calling this SFC, you specify the name and the size of the file that has to be created res. to open.

When calling this SFC 220, you have to transfer the following parameters:

Adresse	Deklaration	Name	Typ	Anfa	Kommentar
0.0	in	FILENAME	STRING[25		name of the file
256.0	in	FILESIZE	DWORD		size of the file
260.0	out	RET_VAL	WORD		return value

File name

Type the file name used to store the data on the MMC. The file name may not exceed a maximum length of 8+3 signs (FAT).

File size

The file size defines the size of the user data in Byte.

Note!

When accessing an already existing file, it is mandatory to give not only the file name but also the file size. The entry of a "Joker" length is not supported at this time.

RET_VAL

Return Value

Word that returns a diagnostic/error message. 0 means OK.

See the table below for the concerning messages:

Value	Description
Diagnostic messages	
0000h	no errors
0001h	File already exists, is not fragmentated and the length value is identical or higher. This message is always sent when opening an existing file.
8001h	No or wrong MMC is plugged-in.
Error messages	
8002h	No FAT on MMC found.
A001h	File name missing.
A002h	File name wrong.
A003h	File exists but file size too low.
A004h	File exists but is fragmentated and cannot be opened.
A005h	Not enough space on MMC.
A006h	No free entry in root directory. Depending on the used MMC there may be min. 16 up to max. 512 entries in the root directory.
B000h	An internal error occurred.

SFC 221 MMC_RD_F

11	21	51
x	x	x

Description

Via the SFC 221 you may read data from a MMC. For read and write accesses to the MMC, you firstly have to open the file with SFC 220 and it has to be unfragmentated.

For more detailed information to this and to the restrictions see the description of the SFC 220.

Parameters

Adresse	Deklaration	Name	Typ	Anfa	Kommentar
0.0	in	PTR	ANY		variable to read
10.0	in	OFFSET	DWORD		offset in the file
14.0	out	BUSY	BOOL		busy flag
16.0	out	RET_VAL	WORD		return value

PTR

This variable of the type pointer points to a data area in the CPU where the content of the MMC has to be written to.

OFFSET

Here you define the start address inside the file on the MMC from where on the data has to be transferred to the CPU.

BUSY

During data transfer this Bit remains set. The Bit is reset as soon as the data transfer is complete.

RET_VAL

Return Value

Word, where a diagnostic/error message is returned to. 0 means OK. The following messages may be set:

Value	Description
0000h	no errors
8001h	No or wrong MMC
8002h	No FAT found on MMC
9001h	Pointer value is wrong
9002h	Bit reading has been tried (variable of Boolean type). Bit reading is not possible.
9003h	File length exceeded
9004h	Sector limit of 512 has been tried to overrun. Sector overrun reading is not possible.
B000h	An internal error occurred.

SFC 222 MMC_WR_F

11	21	51
x	x	x

Description

Via the SFC 222, you may write to the MMC. For read and write accesses to the MMC, you firstly have to open the file with SFC 220 and it has to be unfragmentated.

For more detailed information to this and to the restrictions see the description of the SFC 220.

Parameters

Adresse	Deklaration	Name	Typ	Anfa	Kommentar
0.0	in	PTR	ANY		variable to write from
10.0	in	OFFSET	DWORD		offset in the file
14.0	out	BUSY	BOOL		busy flag
16.0	out	RET_VAL	WORD		return value

PTR

This variable of the type pointer points to a data area from where on the data starts that will be written to the MMC.

OFFSET

This defines the beginning of the data inside the file on the MMC where the data is written to.

BUSY

During data transfer this Bit remains set. The Bit is reset as soon as the data transfer is complete.

RET_VAL

Return Value

Word, where a diagnostic/error message is returned to. 0 means OK. The following messages may be set:

Value	Description
0000h	no errors
8001h	No or wrong MMC
8002h	No FAT found on MMC
9001h	Pointer value is wrong
9002h	Bit reading has been tried (variable of Boolean type). Bit reading is not possible.
9003h	File length exceeded
9004h	Sector limit of 512 has been tried to overrun. Sector overrun reading is not possible.
B000h	An internal error occurred.

SFC 223 PWM

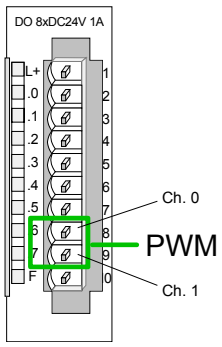
11	21	51
x	x	x

Description

This block serves the parameterization of the pulse duration modulation for the last two output channels of X5.

Parameters

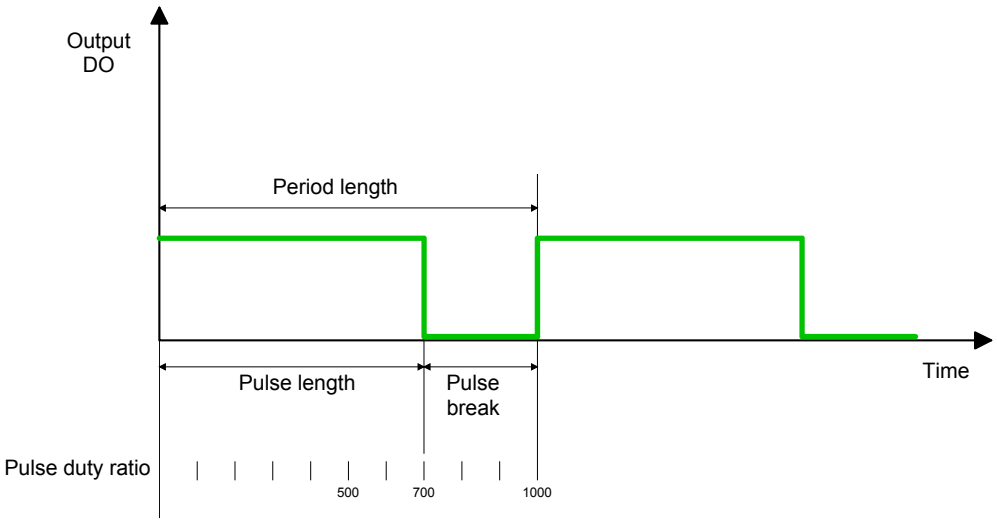
Adresse	Deklaration	Name	Typ	Anfa	Kommentar
0.0	in	Channel	INT		PWM Channel 0..1
2.0	in	Enable	BOOL		True = Start PWM, False = Stop PWM
4.0	in	Timebase	INT		Timebase for period/duty (0 = 0,1ms / 1 = 1,0ms)
6.0	in	Period	DINT		Period of PWM (0..60000)
10.0	in	Duty	DINT		Outputvalue per mille (0..1000)
14.0	in	MinLen	DINT		Minimum pulse (0..60000)
18.0	out	RET_VAL	WORD		Errorcode (0 = no Error)



You define a timebase, a period, the pulse duty ratio and min. pulse length. The CPU determines an pulse series with an according pulse/break relation and issues this via the according output channel.

The SFC returns a certain error code. You can see the concerning error messages in the table at the following page.

The PWM parameters have the following relationship:



Period length = timebase x period

Pulse length = (period length / 1000) x pulse duty ratio

Pulse break = period length - pulse length

The parameters have the following meaning:

Channel Define the output channel that you want to address.
Value range: 0 ... 1

Enable Via this parameter you may activate the PWM function (true) res. deactivate it (false).
Value range: true, false

Timebase Timebase defines the resolution and the value range of the pulse, period and minimum pulse length per channel.
You may choose the values 0 for 0.1ms and 1 for 1ms.
Value range: 0 ... 1

Period Through multiplication of the value defined at period with the timebase you get the period length.
Value range: 0 ... 60000

Duty This parameter shows the pulse duty ratio in promille. Here you define the relationship between pulse length and pulse break, concerned on one period.
1 Promille = 1 Timebase
If the calculated pulse duration is no multiplication of the timebase, it is rounded down to the next smaller time base limit.
Value range: 0 ... 1000

MinLen Via MinLen you define the minimal pulse length. Switches are only made, if the pulse exceeds the here fixed minimum length.
Value range: 0 ... 60000

Ret_Val Via the parameter Ret_Val you get an error number in return. See the table below for the concerning error messages:

Value	Description
0000h	no error
8005h	Parameter “MinLen” outside the permissible range
8006h	Parameter “Duty” outside the permissible range
8007h	Parameter “Period” outside the permissible range
8008h	Parameter “Timebase” outside the permissible range
8009h	Parameter “Channel” outside the permissible range
9001h	Internal error: There was no valid address for a parameter
9002h	Internal hardware error: Please call the Advantech-Service.

SFC 224 HSC

11	21	51
x	x	x

Description This SFC serves for parameterization of the counter functions (**h**igh **s**peed **c**ounter) for the first 4 inputs.

Parameters

Adresse	Deklaration	Name	Typ	Anfangswert	Kommentar
0.0	in	Channel	INT		HSC Channel 0..3
2.0	in	Enable	BOOL		True = Start HSC, False = Stop HSC
4.0	in	Direction	INT		0=no counting, 1=up, 2=down
6.0	in	PresetValue	DINT		Preset Value for HSC (0..0xFFFFFFFF)
10.0	in	Limit	DINT		Endvalue for Counting up / StartValue for Counting down (0..0xFFFFFFFF)
14.0	out	RET_VAL	WORD		Errorcode (0 = no Error)
16.0	in_out	SetCounter	BOOL		True = Copy PresetValue to HSC (Bit will be reset from SFC)

Channel Type the input channel that you want to activate as counter.
Value range: 0 ... 3

Enable Via this parameter you may activate the counter (true) res. deactivate it (false).
Value range: true, false

Direction Fix the counting direction.
Hereby is: 0: Counter is deactivated, means *Enable*=false
1: count up
2. count down

PresetValue Here you may preset a counter content, that is transferred to the according counter via *SetCounter*=true.
Value range: 0 ... FFFFFFFFh

Limit Via Limit you fix an upper res. lower limit for the counting direction (up res. down). When the limit has been reached, the according counter is set zero and started new. If necessary an alarm occurs.
Value range: 0 ... FFFFFFFFh

Ret_Val

Via the parameter Ret_Val you get an error number in return. See the table below for the concerning error messages:

Value	Description
0000h	No error
8002h	The chosen channel is not configured as counter (Error in the hardware configuration)
8008h	Parameter “Direction” outside the permissible range
8009h	Parameter “Channel” outside the permissible range
9001h	Internal error: There was no valid address for a parameter
9002h	Internal hardware error: Please call the Advantech-Service.

SetCounter

Per SetCounter=true the value given by PresetValue is transferred into the according counter.

The Bit is set back from the SFC.

Value range: true, false

SFC 225 HF_PWM

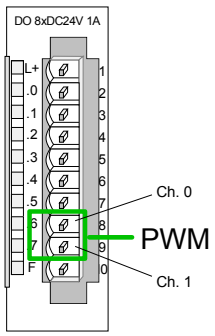
11	21	51
x	x	x

Description

This block serves the parameterization of the pulse duration modulation for the last two output channels. This block is function identical to SFC 223. Instead of timebase and period, the SFC 225 works with a predefined frequency (up to 50kHz).

Parameters

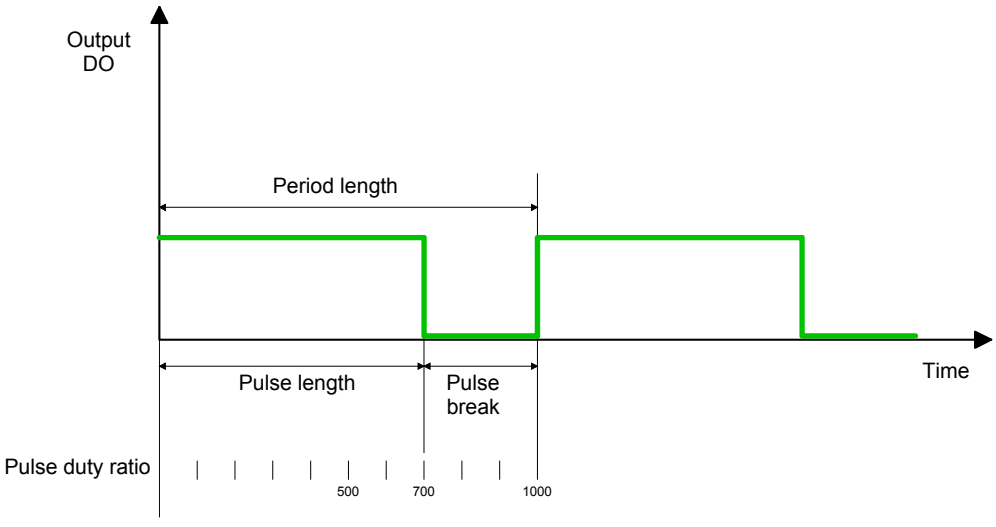
Adresse	Deklaration	Name	Typ	Anfa	Kommentar
0.0	in	Channel	INT		PWM Channel 0..1
2.0	in	Enable	BOOL		True = Start PWM, False = Stop PWM
4.0	in	Frequence	WORD		Frequence of HF PWM in Hz (1250 .. 50000)
6.0	in	Duty	DINT		Outputvalue per mille (0..1000)
10.0	in	MinLen	DINT		Minimum pulse (0..60000)
14.0	out	RET_VAL	WORD		Errorcode (0 = no Error)



You define a frequency, the pulse duty ratio and min. pulse length. The CPU determines an pulse series with an according pulse/break relation and issues this via the according output channel.

The SFC returns a certain error code. You can see the concerning error messages in the table at the following page.

The PWM parameters have the following relationship:



Period length = 1 / frequency

Pulse length = (period length / 1000) x pulse duty ratio

Pulse break = period length - pulse length

Channel Define the output channel that you want to address.
Value range: 0 ... 1

Enable Via this parameter you may activate the PWM function (true) res. deactivate it (false).
Value range: true, false

Frequency Type the frequency in Hz.
Value range: 1250 ... 50000

Duty This parameter shows the pulse duty ratio in Promille. Here you define the relationship between pulse length and pulse break, concerned on one period.
1 Promille = 1 Timebase
If the calculated pulse duration is no multiplication of the timebase, it is rounded down to the next smaller time base limit.
Value range: 0 ... 1000

MinLen Via MinLen you define the minimal pulse length. Switches are only made, if the pulse exceeds the here fixed minimum length.
Value range: 0 ... 60000

Ret_Val Via the parameter Ret_Val you get an error number in return. See the table below for the concerning error messages:

Value	Description
0000h	no error
8005h	Parameter “MinLen” outside the permissible range
8006h	Parameter “Duty” outside the permissible range
8007h	Parameter “Period” outside the permissible range
8008h	Parameter “Timebase” outside the permissible range
8009h	Parameter “Channel” outside the permissible range
9001h	Internal error: There was no valid address for a parameter
9002h	Internal hardware error: Please call the Advantech-Service.

SFC 227 - TD_PRM

11	21	51
x	x	x

Description

The SFC 227 supports the connection of the TD200 terminal from Siemens to the ADAM CPUs.

Please regard that you have to include a data block with the TD200 configuration data before calling the SFC 227. This data block may be created with the TDWizard from ADAM. This data block contains the general settings like language and display mode and the messages that are comfortably createable with the TDWizard from ADAM. TDWizard is a part of WinPLC7 and is available from Advantech.

Parameters

The call of the SFC 227 specifies the terminal to communicate with. To call the SFC you have to transfer the following parameters:

Adresse	Deklaration	Name	Typ	Anfa	Kommentar
0.0	in	MPI_ADDR	BYTE		MPI address of TD
2.0	in	TD_STRUCT_PTR	ANY		Pointer to begining of TD's data (must be in datablock)
12.0	in	FUNC_KEY_PTR	ANY		Pointer to area for keys
22.0	in	OFFSET_INPUT	BYTE		Forced values offset in input area
23.0	in	OFFSET_OUTPUT	BYTE		Forced values offset in output area
24.0	out	RET_VAL	BYTE		Errorcode (0 = no Error)

MPI_ADR

MPI address

Enter the MPI address of the connected TD200 terminal.

Parameter type: Byte

TD_STRUCT_PTR

Terminal Structure Pointer

Points to the start of the data block containing the parameterization and the text blocks of the terminal.

You may create the data block with TDWizard. This tool is a part of WinPLC7, the programming, test, diagnostic and simulation software from Advantech.

Parameter type: Pointer

Convenient range: DB

FUNC_KEY_PTR

Function Key Pointer

Points to the area where the status byte for hitten keys is stored.

Parameter type: Pointer

Convenient range: DB, M

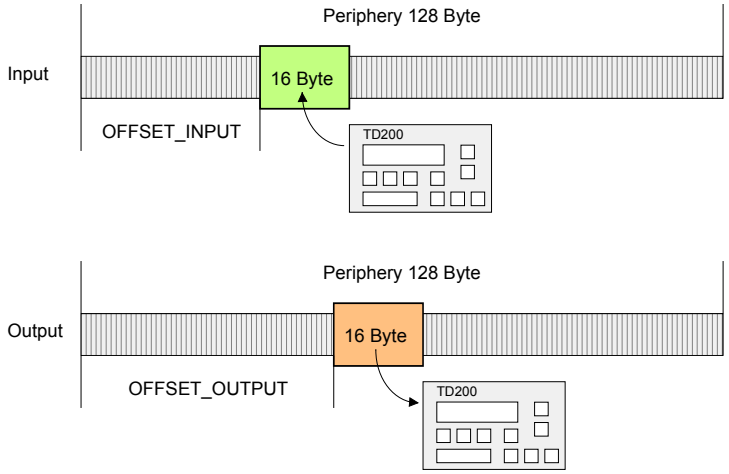
OFFSET_INPUT
OFFSET_OUTPUT

Offset Input, Offset Output

The terminal allows you to set in- and output byte. The TD200 from Siemens supports only a size of 16Byte for in- and outputs.

The ADAM CPUs from Advantech enables the access to the complete process image (each 128Byte for in- and outputs).

By setting an offset, you may overlay the periphery range of 128Byte with a window of 16Byte for in- and output. The following picture illustrates this:



RET_VAL

Return Value

Type the bit memory byte (marker byte) where the resulting message should be stored.

For specification of the (error) messages see the table below.

Messages

Value	Description
00h	no error
10h	Error at MPI_ADR
11h	MPI_ADR contains MPI address of the CPU
12h	Value in MPI_ADR exceeds max. MPI address
20h	Error in OFFSET_INPUT
21h	Value in OFFSET_INPUT is too high
30h	Error in OFFSET_OUTPUT
31h	Value in OFFSET_OUTPUT is too high
40h	Error in TD_STRUCT_PTR
41h	TD_STRUCT_PTR points not to a DB
50h	Error in FUNC_KEY_PTR
51h	Error in FUNC_KEY_PTR
52h	FUNC_KEY_PTR points not to an I, Q or M area
60h	An internal error occurred

SFC 228 - RW_KACHEL

11	21	51
x	x	x

Description

This SFC allows you the direct access to the page frame area of the CPU with a size of 4KByte. The page frame area is divided into four page frames, each with a size of 1KByte.

By fixing the page frame (i.e. "Kachel") no., offset and data width, the SFC 228 enables read and write access to an eligible page frame area.

Note!

This SFC has been developed for test purposes and for building-up proprietary communication systems and is completely at the user's disposal. Please regard that a write access to the page frame area influences a communication directly!

Parameter

Adresse	Deklarat	Name	Typ	An	Kommentar
0.0	in	K_Nr	INT		Interface number
2.0	in	Offset	INT		Offset in Byte
4.0	in	R_W	INT		0=Read, 1=Write
6.0	in	Size	INT		1, 2 or 4 Bytes
8.0	out	RET_VAL	BYTE		Errorcode, 0=OK
10.0	in_out	Value	ANY		Value, which is read from interface or should be written to interface

K_NR

Page frame (i.e. "Kachel") no.

Type the page frame no. that you want to access.

Value range: 0 ... 3

OFFSET

Page frame offset

Fix here an offset within the specified page frame.

Value range: 0 ... 1023

R_W

Read/Write

This parameter specifies a read res. write access.

0 = read access, 1 = write access

SIZE

Size

The size defines the width of the data area fixed via K_NR and OFFSET. You may choose between the values 1, 2 and 4Byte.

RET_VAL

Return Value

Byte where an error message is returned to.

IN_OUT

In-/output area

This parameter fixes the in- res. output area for the data transfer.

At a read access, this area up to 4Byte width contains the data read from the page frame area.

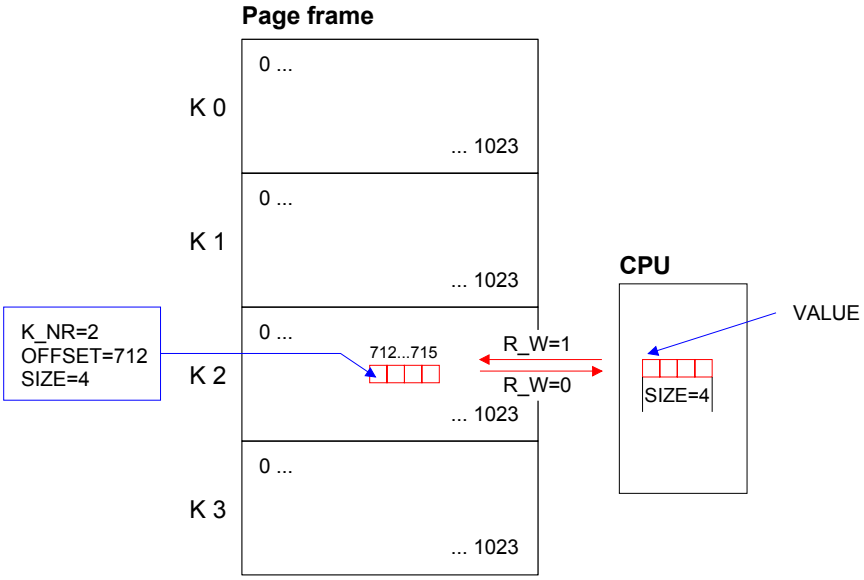
At a write access, the data up to 4Byte width is transferred to the page frame area.

Parameter type: Pointer

Example

The following example shows the read access to 4Byte starting with Byte 712 in page frame 2. The read 4Byte are stored in DB10 starting with Byte 2. Herefor the following call is required:

```
CALL SFC 228
K_NR      :=2
OFFSET    :=712
R_W       :=0
SIZE      :=4
RET_VAL   :=MB10
VALUE     :=P#DB10.DBX 2.0 Byte 4
```



Error messages

Value	Description
00h	no error
01h ... 05h	Internal error: No valid address found for a parameter
06h	defined page frame does not exist
07h	parameter SIZE ≠ 1, 2 or 4 at read access
08h	parameter SIZE ≠ 1, 2 or 4 at write access
09h	parameter R_W ≠ 0 or 1

Page frame communication - Parameter

11	21	51
x	x	x

General

The delivered handling blocks allow the deployment of communication processors in the ADAM CPUs from Advantech.

This increases the efficiency markable.

The handling blocks control the complete data transfer between CPU and the CPs.

Advantages of the handling blocks:

- you loose only few user application memory space
- short runtimes of the blocks

The handling blocks don't need:

- bit memory area
- time areas
- counter areas

Parameter description

All handling blocks described in the following use an identical interface to the user application with this parameters:

SSNR:	Interface number
ANR:	Order number
ANZW:	Indicator word (double word)
IND:	Indirect fixing of the relative start address of the data source res. destination
QANF/ZANF:	Relative start address within the type
PAFE:	Parameterization error
BLGR:	Block size

A description of these parameters follows on the next pages.

SSNR Interface number

Number of the logical interface (page frame address) to which the according order refers to.

Parameter type : Integer

Convenient range : 0 ... 255

ANR Job number

The called job number for the logical interface.

Parameter type : Integer

Convenient range : 1 ... 223

ANZW Indicator word (double word)

Address of the indicator double word in the user memory where the processing of the order specified under ANR is shown.

Parameter type : Double word

Convenient range : DW or MW; use either DW and DW+1 or MW and MW+2

The value DW refers to the data block opened before the incoming call or to the directly specified DB.

IND Kind of parameterization (direct, indirect)

This parameter defines the kind of data on which the pointer QUANF points.

0: QANF points directly to the initial data of the source res. destination data.

1: the pointer QANF/ZANF points to a memory cell, from where on the source res. destination data are defined (indirect).

2: the pointer QANF/ZANF points to a memory area where the source res. destination information lies (indirect).

5: the pointer QANF/ZANF points to a memory cell, from where on the source res. destination data and parameters of the indicator word are defined (indirect).

6: the pointer QANF/ZANF points to a memory area where the source res. destination data and parameters of the indicator word are laying (indirect).

Parameter type : Integer

Convenient entries : 0, 1, 2, 5, 6

Note!

Please regard, that at IND=5 res. IND=6, the parameter ANZW is ignored!

QANF/ZANF Relative start address of the data source res. destination and at IND=5 res. IND=6 of the indicator word.

This parameter of the type „pointer“ (Any-Pointer) allows you fix the relative starting address and the type of the data source (at SEND) res. the data destination (at RECEIVE).

At IND=5 res. IND=6 the parameters of the indicator word are also in the data source.

Parameter type : Pointer

Convenient range : DB, M, A, E

Example:
```
P#DB10.DBX0.0 BYTE 16
P#M0.0 BYTE 10
P#E 0.0 BYTE 8
P#A 0.0 BYTE 10
```

BLGR Block size

During the boot process the stations agree about the block size (size of the data blocks) by means of SYNCHRON.

A high block size = high data throughput but longer run-times and higher cycle load.

A small block size = lower data throughput but shorter run-times of the blocks.

These block sizes are available:

Value	*Block size*	*Value*	*Block size*
0	Default (64Byte)	4	128Byte
1	16Byte	5	256Byte
2	32Byte	6	512Byte
3	64Byte	255	512Byte

Parameter type : Integer

Convenient range : 0 ... 255

PAFE Error indication at parameterization defects

This "BYTE" (output, marker) is set if the block detects a parameterization error, e.g. interface (plug-in) not detected or a non valid parameterization of QUANF/ZANF.

Parameter type : Byte

Convenient range : AB 0 ... AB127, MB 0...MB 255

Page frame communication - Parameter transfer

11 x	21 x	51 x

Direct/indirect parameterization

A handling block may be parameterized directly or indirectly. Only the "PAFE" parameter must always been set directly.

When using the direct parameterization, the handling block works off the parameters given immediately with the block call.

When using the indirect parameterization, the handling block gets only pointers per block parameters. These are pointing to other parameter fields (data blocks or data words).

The parameters SSNR, ANR, IND and BLGR are of the type "integer", so you may parameterize them indirectly.

Example for direct and indirect parameter transfer

Direct parameter transfer

```
CALL  SFC  230
       SSNR:=0
       ANR :=3
       IND :=0
       QANF:=P#A 0.0 BYTE 16
       PAFE:=MB79
       ANZW:=MD44
```

Indirect parameter transfer

Please note that you have to load the bit memory words with the corresponding values before.

```
CALL  SFC  230
       SSNR:=MW10
       ANR :=MW12
       IND :=MW14
       QANF:=P#DB10.DBX0.0 BYTE 16
       PAFE:=MB80
       ANZW:=MD48
```

The direct res. indirect transfer of the source and destination parameters are described in the following section.

Page frame communication - Source res. destination definition

11	21	51
x	x	x

Outline

You have the possibility to set the entries for source, destination and ANZW directly or store it indirectly in a block to which the QANF/ZANF res. ANZW pointer points.

The parameter IND is the switch criterion between direct and indirect parameterization.

Direct parameterization of source and destination details (IND=0)

With IND=0 you fix that the pointer QANF/ZANF shows directly to the source res. destination data.

The following table shows the possible QANF/ZANF parameters at the direct parameterization:

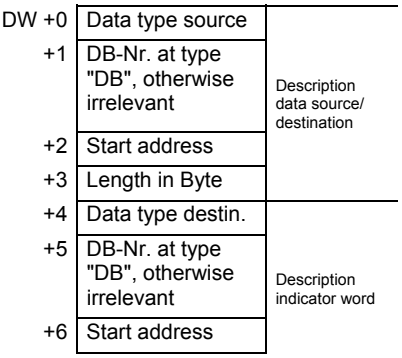
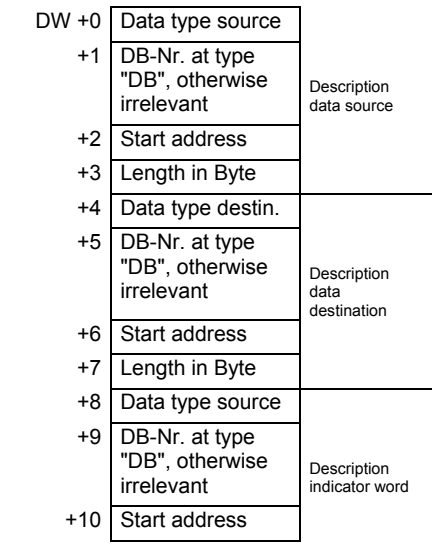
QTYP/ZTYP Description	Data in DB	Data in MB	Data in AB Process image of the outputs	Data in EB Process image of the inputs
Pointer: Example:	`P#DB`a`.DBX` b`.0 BYTE` C `P#DB10.DBX 0.0 BYTE 8`	`P#M` b`.0 BYTE` c `P#M 5.0 BYTE 10`	`P#A` b`.0 BYTE` c `P#A 0.0 BYTE 2`	`P#E` b`.0 BYTE` c `P#E 20.0 BYTE 1`
DB, MB, AB, EB Definition	`P#DB`a "a" means the DB-No., from where the source data is fetched or where to the destination data is transferred.	`P#M` The data is stored in a MB.	`P#A` The data is stored in the output byte.	`P#E` The data is stored in the input byte.
Valid range for "a"	0...32767	irrelevant	irrelevant	irrelevant
Data / Marker Byte, AB, EB Definition	DB-No., where data fetch or write starts.	Bit memory byte no., where data fetch or write starts.	Output byte no., where data fetch or write starts.	Input byte no., where data fetch or write starts.
Valid range for "b"	0.0...2047.0	0...255	0...127	0...127
`BYTE` c Definition	Length of the Source/ Destination data blocks in Words.	Length of the Source/ Destination data blocks in Bytes.	Length of the Source/ Destination data blocks in Bytes.	Length of the Source/ Destination data blocks in Bytes.
Valid range for "c"	1...2048	1...255	1...128	1...128

Indirect parameterization of source and destination details (IND=1 or IND=2)

Indirect addressing means that QANF/ZANF points to a memory area where the addresses of the source res. destination areas and the indicator word are stored.

In this context you may either define one area for data source, destination and indicator word (IND=5) or each, data source, data destination and the indicator word, get an area of their own (IND=6).

The following table shows the possible QANF/ZANF parameters for indirect parameterization:

QTYP/ZTYP Description	IND=5	IND=6
Definition	Indirect addressing for source **or** destination parameters and indicator word. The source or destination parameters and ANZW are stored in a DB. 	Indirect addressing for source **and** destination parameters and ANZW. The source and destination parameters and ANZW are stored in a DB in a sequential order. 
valid DB-No.	0...32767	0...32767
Data word Definition	DW-No., where the stored data starts	DW-No., where the stored data starts
Valid range	0.0...2047.0	0.0...2047.0
Length Definition	Length of the DBs in Byte	Length of the DBs in Byte
Valid range	14 fix	22 fix

Page frame communication - Indicator word ANZW

11 x	21 x	51 x

Status and error reports

Status and error reports are created by the handling blocks:

- by the indicator word ANZW (information at order commissioning),
- by the parameter error byte PAFE (indication of a wrong order parameterization).

Content and structure of the indicator word ANZW

The "Indicator word" shows the status of a certain order on a CP.

In your PLC program you should keep one indicator word for each defined order at hand.

The indicator word has the following structure:

Byte	Bit 7 ... Bit 0
0	*State management CPU* Bit 0: Handshake convenient (data exists) =0: RECEIVE blocked =1: RECEIVE released Bit 1: order commissioning is running =0: SEND/FETCH released =1: SEND/FETCH blocked Bit 2: Order ready without errors Bit 3: Order ready with errors *Data management handling block* Bit 4: Data receive/send is running Bit 5: Data transmission active Bit 6: Data fetch active Bit 7: Disable/Enable data block =0: released =1: blocked
1	Bit 0 ... Bit 3: Error management CPU =0: no error =1...5: CPU-Error =6...15: CP-Error Bit 4 ... Bit 7: reserved
2 ... 3	Length word handling block

In the „length word“ the handling blocks (SEND, RECEIVE) store the data that has already been transferred, i.e. received data in case of a RECEIVE order, send data when there is a SEND order.

The announcement in the length word is always in byte and absolute.

Status management Byte 0, Bit 0 to Bit 3

Here you may see if an order has already been started, if an error occurred or if this order is blocked, e.g. a virtual connection doesn't exist any longer.

Bit 0 **Handshake convenient**

Set: Per plug-in according to the „delete"-announcement in the order status bit: Handshake convenient (=1) is used at the RECEIVE block (telegram exists at PRIO 1 or RECEIVE impulse is possible at PRIO 2/3)

Analyze: Per RECEIVE block: The RECEIVE initializes the handshake with the CP only if this bit is set.

Per application: for RECEIVE request (request a telegram at PRIO 1).

Bit 1 **Order is running**

Set: Per plug-in: when the CP received the order.

Delete: Per plug-in: when an order has been commissioned (e.g. receipt received).

Analyze: Per handling blocks: A new order is only send, when the order before is completely commissioned.

Per user: when you want to know, if triggering a new order is convenient.

Bit 2 **Order ready without errors**

Set: Per plug-in: when the according order has been commissioned without errors.

Delete: Per plug-in: when the according order is triggered for a second time.

Analyze: Per user: to proof that the order has been commissioned without errors.

Bit 3 **Order ready with errors**

Set: Per plug-in: when the according order has been commissioned with errors. Error causes are to find encrypted in the high-part of the indicator word.

Delete: Per plug-in: when the according order is triggered for a second time.

Analyze: Per user: to proof that the order has been commissioned with errors. If set, the error causes are to find in the high-byte of the indicator word.

Data management Byte 0, Bit 4 to Bit 7

Here you may check if the data transfer is still running or if the data fetch res. transmission is already finished. By means of the bit "Enable/Disable" you may block the data transfer for this order (Disable=1; Enable=0).

Bit 4 **Data fetch / Data transmission is active**

Set: Per handling block SEND or RECEIVE, if the fetch/transmission has been started, e.g. when data is transferred with the ALL-function (DMA-replacement), but the impulse came per SEND-DIRECT.

Delete: Per handling blocks SEND or RECEIVE, if the data transfer of an order is finished (last data block has been transferred).

Analyze: Per user: During the data transfer CP << >>AG the user must not change the record set of an order. This is uncritical with PRIO 0/1 orders, because here the data transfer is realizable in one block cycle. Larger data amounts however are transferred in blocks during more AG cycles. To ensure data consistency you should proof that the data block isn't in transfer any more before you change the content!

BIT 5 **Data transmission is active**

Set: Per handling block SEND, when the data transition for an order is ready.

Delete: Per handling block SEND, when the data transfer for a new order has been started (new trigger).

Per user: When analysis is ready (flank creation).

Analyze: Per user: Here you may ascertain, if the record set of an order has already been transferred to the CP res. at which time a new record set concerning a running order (e.g. cyclic transition) may be started.

Bit 6 **Data fetch active**

Set: Per RECEIVE, when data fetch for a new order has been finished.

Delete: Per RECEIVE, when data transfer to AG for a new order (new trigger) has been started. Per user, when analyzing (edge creation).

Analyze: Per user: Here you may ascertain, if the record set of an order has already been transferred to the CP res. at what time a new record set for the current order has been transferred to the AG.

Bit 7 **Disable/Enable data block**

Set: Per user: to avoid overwriting an area by the RECEIVE block res. data transition of an area by the SEND block (only for the first data block).

Delete: Per user: to release the according data area.

Analyze: Per handling blocks SEND and RECEIVE: if Bit 7 is set, there is no data transfer anymore, but the blocks announce an error to the CP.

Error management Byte 1, Bit 0 to Bit 3

Those bits announce the error messages of the order. The error messages are only valid if the bit „Order ready with error“ in the status bit is set simultaneously.

The following error messages may occur:

0 **no error**

If the bit "Order ready with error" is set, the CP had to reinitialize the connection, e.g. after a reboot or RESET.

1 **wrong Q/ZTYP at HTB**

The order has been parameterized with the wrong type label.

2 **AG area not found**

The order impulse had a wrong parameterized DB-No.

3 **AG area too small**

Q/ZANF and Q/ZLAE overwrite the range boundaries. Handling with data blocks the range boundary is defined by the block size. With flags, timers, counters etc. the range size depends on the AG.

4 **QVZ-Error in the AG**

This error message means, that you chose a source res. destination parameter of the AG area, where there is either no block plugged in or the memory has a defect. The QVZ error message can only occur with the type Q/ZTYP AS, PB, QB or memory defects.

5 **Error at indicator word**

The parameterized indicator word can not be handled. This error occurs, if ANZW declared a data word res. double word, that is not (any more) in the specified data block, i.e. DB is too small or doesn’t exist.

6 **no valid ORG-Format**

The data destination res. source isn’t declared, neither at the handling block (Q/TYP="NN") nor at the coupler block.

7 **Reserved**

8 **no available transfer connections**

The capacity for transfer connections is at limit. Delete unnecessary connections.

9 **Remote error**

There was an error at the communication partner during a READ/WRITE-order.

A **Connection error**

The connection is not (yet) established. The message disappears as soon as the connection is stable. If all connections are interrupted, please check the block itself and the bus cable. Another possibility for the occurrence of this error is a wrong parameterization, like e.g. inconsistent addressing.

B **Handshake error**

This could be a system error or the size of the data blocks has been defined out of range.

C **Initial error**

The wrong handling block tried to initialize the order or the size of the given data block was too large.

D **Cancel after RESET**

This is a normal system message. With PRIO 1 and 2 the connection is interrupted but will be established again, as soon as the communication partner is online. PRIO 3 connections are deleted, but can be initialized again.

E **Order with basic load function**

This is a normal system message. This order is a READ/WRITE-PASSIV and can not be started from the AG.

F **Order not found**

The called order is not parameterized on the CP. This error may occur when the SSNR/A-No. combination in the handling block is wrong or no connection block is entered.

The bits 4 to 7 of Byte 2 are reserved for extensions.

Length word Byte 2 and Byte 3

In the length word the handling blocks (SEND, RECEIVE) store the already transferred data of the current order, i.e. the received data amount for receiving orders, the sent data amount for sending orders.

Describe: Per SEND, RECEIVE during the data transfer. The length word is calculated from: **current transfer amount + amount of already transferred data**

Delete: Per overwrite res. with every new SEND, RECEIVE, FETCH.

If the bit "order ready without error" res. "Data fetch/data transition ready " is set, the "Length word" contains the current source res. destination length.

If the bit "order ready with error" is set, the length word contains the data amount transferred before the failure occurred.

Status and error reports

Important status and error reports of the CPU

The following section lists important status and error messages that can appear in the "Indicator word". The representation is in HEX patterns. The literal X means "not declared" res. "irrelevant"; No. is the error number.

Possible indicator words

Indicator word: X F X A

The error index "F" shows, that the according order is not defined on the CP. The state index "A" causes a block of this order (for SEND/FETCH and RECEIVE).

Indicator word: X A X A

The error index "A" shows that the connection of the communication order is not (yet) established. Together with the state index "A" SEND, RECEIVE and FETCH are blocked.

Indicator word: X 0 X 8

The connection has been established again (e.g. after a CP reboot), the SEND order is released (SEND-communication order).

Indicator word: X 0 X 9

The connection has been established again, the RECEIVE order is released (RECEIVE-communication order).

Indicator word: X 0 2 4

SEND has been worked off without errors, the data was transferred.

Indicator word: X 0 4 5

RECEIVE was successful, the data arrived at the AG.

Indicator word: X 0 X 2

The SEND-, RECEIVE-, READ- res. WRITE order is still running. At SEND the partner is not yet ready for RECEIVE or vice versa.

Important indicator word states

The following table shows the most important indicator word states:

Messages at SEND

State under H1	Prio 0/1	Prio 2	Prio 3/4
State under TCP/IP	Prio 1	Prio 2	Prio 3
after reboot	0 A 0 A	0 A 0 A	0 0 0 8
after connection start	X 0 X 8	X 0 X 8	.....
after initial impulse	X 0 X 2	X 0 X 2	X 0 X 2
ready without error	X 0 2 4	X 0 2 4	X 0 2 4
ready with error	X No X 8	X No X 8	X No X 8
after RESET	X D X A	X D X A	X D X 8

Messages at RECEIVE

State under H1	Prio 0/1	Prio 2	Prio 3/4
State under TCP/IP	Prio 1	Prio 2	Prio 3
after reboot	0 A 0 A	0 A 0 A	0 0 0 1
after connection start	X 0 X 4	X 0 0 9	.....
after initial impulse	X 0 X 2	X 0 X 2	X 0 X 2
Telegram received	X 0 X 1	.....	.....
ready without error	X 0 4 1	X 0 4 5	X 0 4 5
ready with error	X No X 8	X No X 9	X No X 9
after RESET	X D X A	X D X A	X D X 9

Messages at READ/WRITE-ACTIVE

State under H1	Prio 0/1	Prio 2	Prio 3/4
State under TCP/IP	Prio 1	Prio 2	Prio 3
after reboot		0 A 0 A	
after connection start		X 0 0 8	
after initial impulse		X 0 X 2	
READ ready		X 0 4 4	
WRITE ready		X 0 2 4	
ready with error		X No X 8	
after RESET		X D X A	

Page frame communication - Parameterization error PAFE

11	21	51
x	x	x

The parameterization error byte PAFE is set (output or bit memory), when the block detects a parameterization error., e.g. there is no interface or there is an invalid parameterization of QANF/ZANF.

PAFE has the following structure:

Byte	Bit 7 ... Bit 0
0	Bit 0: error =0: no error =1: error, error-No. in Bit 4 to Bit 7 Bit 1 ... Bit 3: reserved Bit 4 ... Bit 7: error-No. =0: no error =1: wrong ORG-Format =2: area not found (DB not found) =3: area too small =4: QVZ- error =5: wrong indicator word =6: no Source-/Destinationparameters at SEND/RECEIVE ALL =7: interface not found =8: interface not specified =9: interface overflow =A: reserved =B: invalid order-No. =C: interface of CP doesn't quit or is negative =D: Parameter BLGR not allowed =E: reserved =F: reserved

SFC 230 - SEND

11	21	51
x	x	x

Description

The SEND block initializes a send order to a CP.

Normally SEND is called in the cyclic part of the user application program. Although the insertion of this block into the interrupt or the time-alarm program part is possible, the indicator word (ANZW), however, may not be updated cyclically. This should be taken over by a CONTROL block.

The connection initialization with the CP for data transmission and for activating a SEND impulse is only started, if:

- the FB VKE received "1"
- the CP released the order. (Bit "order active" in ANZW =0).

During block stand-by, only the indicator word is updated.

Parameter

Adresse	Deklarat	Name	Typ	Anfa	Kommentar
0.0	in	SSNR	INT		Interface number, number of logical interface
2.0	in	ANR	INT		The job that must be initiated at the interface, i.e. start transmission
4.0	in	IND	INT		Mode of Adressing (direct / indirect)
6.0	in	ZANF	ANY		Pointer to data destination
16.0	out	PAFE	BYTE		Error indicator for configuration errors
18.0	in_out	ANZW	DWORD		Indicatorword (progress of started jobs is displayed)

SEND_ALL for data transmission

If the CP is able to take over the data directly, the SEND block transfers the requested data in one session. If the CP requests only the order parameters or the amount of the depending data is too large, the CP only gets the sending parameters res. the parameter with the first data block. The according data res. the assigned serials blocks for this order are requested from the CP by SEND_ALL to the CPU. Herefor it is necessary that the block SEND_ALL is called minimum one time per cycle.

The user interface is for all initialization types equal, only the transfer time of the data is postponed for minimum one CPU cycle.

SFC 231 - RECEIVE

11	21	51
x	x	x

Description

The RECEIVE block receives data from a CP.

Normally the RECEIVE block is called in the cyclic part of the user application program. Although the insertion of this block into the interrupt or the waking program part is possible, the indicator word can not be updated cyclically. This should be taken over by a CONTROL block.

The handshake with the CP (order initialization) and for activating a RECEIVE block is only started, if

- the FB VKE received "1"
- the CP released the order (Bit "Handshake convenient" = 1).

Parameter

Adresse	Deklarat	Name	Typ	Anfa	Kommentar
0.0	in	SSNR	INT		Interface number, number of logical interface
2.0	in	ANR	INT		The job that must be initiated at the interface, i.e. start transmission
4.0	in	IND	INT		Mode of Adressing (direct / indirect)
6.0	in	ZANF	ANY		Pointer to data destination
16.0	out	PAFE	BYTE		Error indicator for configuration errors
18.0	in_out	ANZW	DWORD		Indicatorword (progress of started jobs is displayed)

If the block runs in stand-by only the indicator word is updated.

The RECEIVE block reacts different depending from the kind of supply and the CP reaction:

- If the CP transmits a set of parameters although the RECEIVE block itself got destination parameters, the parameter set of the block has the priority above those of the CP.
- Large amounts of data can only be transmitted in blocks. Therefore you have to transmit the assigned serial blocks by means of RECEIVE_ALL to the CPU. It is necessary that the block RECEIVE_ALL is called minimum one time per application cycle and CP interface, if you want to transmit larger data amounts. You also have to integrate the RECEIVE_ALL cyclically, if the CP only uses the RECEIVE for releasing a receipt telegram and the data is transmitted via the background communication of the CPU.

SFC 232 - FETCH

11	21	51
x	x	x

Description

The FETCH block initializes a FETCH order in the partner station.

The FETCH order defines data source and destination and the data source is transmitted to the partner station.

The CPU ADAM 821x from Advantech realizes the definition of source and destination via a pointer parameter.

The partner station provides the *Source* data and transmits them via SEND_ALL back to the requesting station. Via RECEIVE_ALL the data is received and is stored in *Destination*.

The update of the indicator word takes place via FETCH res. CONTROL.

The handshake for initializing FETCH is only started, if

- the FB VKE receives "1"
- the function has been released in the according CP indicator word (order active = 0).

Parameter

Adresse	Deklarat	Name	Typ	Anfa	Kommentar
0.0	in	SSNR	INT		Interface number, number of logical interface
2.0	in	ANR	INT		The job that must be initiated at the interface, i.e. start transmission
4.0	out	PAFE	BYTE		Error indicator for configuration errors
6.0	in_out	ANZW	DWORD		Indicatorword (progress of started jobs is displayed)

Note!

More information for indirect parameterization you will find in this chapter in "Page frame communication - Source res. ".

SFC 233 - CONTROL

11	21	51
x	x	x

Description

The purpose of the CONTROL block is the following:

- Update of the indicator word
- Query if a certain order of the CP is currently active, e.g. request for a receipt telegram
- Query the CP which order is recently in commission.

The CONTROL block is not responsible for the handshake with the CP, it just transfers the announcements in the order status to the parameterized indicator word. The block is independent from the VKE and should be called from the cyclic part of the application.

Parameter

Adresse	Deklara	Name	Typ	Anfa	Kommentar
0.0	in	SSNR	INT		Interface number, number of logical interface
2.0	in	ANR	INT		The job that must be initiated at the interface, i.e. start transmission
4.0	out	PAFE	BYTE		Error indicator for configuration errors

ANR

If ANR ≠0, the indicator word is built up and handled equal to all other handling blocks.

If the parameter ANR gets 0, the CONTROL command transmits the content of the order state cell 0 to the LOW part of the indicator words

The order state cell 0 contains the number of the order that is in commission, e.g. the order number of a telegram (set by the CP).

SFC 234 - RESET

11	21	51
x	x	x

Description

The RESET_ALL function is called via the order number 0. This resets all orders of this logical interface, e.g. deletes all order data and interrupts all active orders.

With a direct function (ANR≠0) only the specified order will be reset on the logical interface.

The block depends on the VKE and may be called from cyclic, time or alarm controlled program parts.

Parameter

Adresse	Deklara	Name	Typ	Anfa	Kommentar
0.0	in	SSNR	INT		Interface number, number of logical interface
2.0	in	BLGR	INT		Blocksize
4.0	out	PAFE	BYTE		Error indicator for configuration errors

Operating modes

The block has two different operating modes:

- RESET ALL
- RESET DIRECT

SFC 235 - SYNCHRON

11	21	51
x	x	x

Description

The SYNCHRON block initializes the synchronization between CPU and CP during the boot process. For this it has to be called from the starting OBs. Simultaneously the transition area of the interface is deleted and predefined and the CP and the CPU 821x agree about the block size.

Parameter

Adresse	Deklaration	Name	Typ	Anfa	Kommentar
0.0	in	SSNR	INT		Interface number, number of logical interface
2.0	out	PAFE	BYTE		Error indicator for configuration errors
4.0	in_out	ANZW	DWORD		Indicatorword (progress of started jobs is displayed)

Block size

To avoid long cycle run-times it is convenient to split large data amounts into smaller blocks for transmitting them between CP and CPU. You declare the size of this blocks by means of „block size“.

A large block size = high data throughput, but also longer run-times and therefore a high cycle time strain.

A small block size = smaller data throughput, but also shorter run-times of the blocks.

Following block sizes are available:

Value	*Block size*	*Value*	*Block size*
0	Default (64Byte)	4	128Byte
1	16Byte	5	256Byte
2	32Byte	6	512Byte
3	64Byte	255	512Byte

Parameter type : Integer

Valid range : 0 ... 255

SFC 236 - SEND_ALL

11	21	51
x	x	x

Description

Via the SEND_ALL block, the data is transmitted from the CPU to the CP by using the declared block size.

Location and size of the data area that is to transmit with SEND_ALL, must be declared before by calling SEND res. FETCH.

In the indicator word that is assigned to the concerned order, the bit "ENABLE/DISABLE" is set, "Data transmission starts" and "Data transmission running" is calculated or altered.

Parameter

Adresse	Deklaration	Name	Typ	Anf	Kommentar
0.0	in	SSNR	INT		Interface number, number of logical interface
2.0	out	PAFE	BYTE		Error indicator for configuration errors
4.0	in_out	ANZW	DWORD		Indicatorword (progress of started jobs is displayed)

ANZW

In the indicator word of the block, the indicator word, that is parameterized in the SEND_ALL block, the current order number is stored (0 means stand-by).

The amount of the transmitted data for one order is shown in the data word of SEND_ALL which follows the indicator word.

Note!

In the following cases, the SEND_ALL command has to be called for minimum one time per cycle of the block OB1:

- if the CP is able to request data from the CPU independently
- if a CP order is initialized via SEND, but the CP still has to request the background communication data of the CPU for this order.
- if the amount of data, that should be transmitted by this SEND to the CP, is higher than the declared block size.

SFC 237 - RECEIVE_ALL

11	21	51
x	x	x

Description

Via the RECEIVE_ALL block, the data received from the CP is transmitted from the CP to the CPU by using the declared block size.

Location and size of the data area that is to transmit with RECEIVE_ALL, must be declared before by calling RECEIVE.

In the indicator word that is assigned to the concerned order, the bit "ENABLE/DISABLE" is set, "Data transition starts" and "Data transition/fetch running" is analyzed or altered. The receiving amount is shown in the following word.

Parameter

Adresse	Deklaration	Name	Typ	Anfa	Kommentar
0.0	in	SSNR	INT		Interface number, number of logical interface
2.0	in	ANR	INT		The job that must be initiated at the interface, i.e. start transmission
4.0	in	IND	INT		Reserved (at the moment)
6.0	out	PAFE	BYTE		Error indicator for configuration errors
8.0	in_out	ANZW	ANY		Indicatorword (progress of started jobs is displayed)

ANZW

In the indicator word of the block, the indicator word, , that is parameterized in the RECEIVE_ALL block, the current order number is stored. In the stand-by running mode of RECEIVE_ALL the block indicator word is deleted.

Note!

In the following cases, the RECEIVE_ALL command has to be called for minimum one time per cycle of the block OB1:

- if the CP should send data to the CPU independently
- if a CP order is initialized via RECEIVE, but the CP still has to request the background communication data of the CPU for this order.
- if the amount of data, that should be transmitted to the CPU by this RECEIVE, is higher than the declared block size.

SFC 238 - CTRL1

11	21	51
x	x	x

Description

This block is identical to the CONTROL block SFC 233 except that the indicator word is of the type Pointer and that it additionally includes the parameter IND, reserved for further extensions.

The purpose of the CONTROL block is the following:

- Update of the indicator word
- Query if a certain order of the CP is currently active, e.g. request for a receipt telegram
- Query the CP which order is recently in commission.

The CONTROL block is not responsible for the handshake with the CP, it just transfers the announcements in the order status to the parameterized indicator word. The block is independent from the VKE and should be called from the cyclic part of the application.

Parameter

Adresse	Deklaration	Name	Typ	Anfa	Kommentar
0.0	in	SSNR	INT		Interface number, number of logical interface
2.0	in	ANR	INT		The job that must be initiated at the interface, i.e. start transmission
4.0	in	IND	INT		Reserved (at the moment)
6.0	out	PAFE	BYTE		Error indicator for configuration errors
8.0	in_out	ANZW	ANY		Indicatorword (progress of started jobs is displayed)

ANR

If ANR ≠0, the indicator word is built up and handled equal to all other handling blocks.

If the parameter ANR gets 0, the CONTROL command transmits the content of the order state cell 0 to the LOW part of the indicator words

The order state cell 0 contains the number of the order that is in commission, e.g. the order number of a telegram (set by the CP).

IND

The parameter IND has no functionality at this time and is reserved for further extensions.

ANZW

The indicator word ANZW is of the type Pointer. This allows you to store the indicator word in a data block.

Chapter 8 Instruction list

Outline

The following chapter lists the available commands of the CPU 821x from Advantech. The instruction list intends to give you an overview over the commands and their syntax. The commands are sorted by topics in alphabetical order.

Via the content the different topics are available.

The alphabetical instruction list gives you direct access to the instructions.

For the parameters are integrated in the instruction list, there is no extra parameter list.

The following chapter describes:

- Instruction and abbreviation list
- Structure of the registers and addressing examples
- Instruction list
- ADAM specific diagnostic entries (Event-IDs)

Content

Alphabetical instruction list

Abbreviations

Abbreviation	Description
/FC	First check bit
2#	Binary constant
a	Byte address
ACCU	Register for processing bytes, words and double words
AR	Address registers, contain the area-internal or area-crossing addresses for the instructions addressed register-indirect
b	Bit address
B	area-crossing, register-indirect addressed Byte
B (b1,b2)	Constant, 2Byte
B (b1,b2,b3,b4)	Constant, 4Byte
B#16#	Byte hexadecimal
BR	Binary result
c	Operand range
C	Counter
C#	Counter constant (BCD-coded)
CC0	Condition code
CC1	Condition code
D	area-crossing, register-indirect addressed double word
D#	IEC date constant
DB	Data block
DBB	Data byte in the data block
DBD	Data double word in the data block
DBW	Data word in the data block
DBX	Data bit in the data block
DI	Instance data block
DIB	Data byte in the instance DB
DID	Data double word in the instance DB
DIW	Data word in the instance DB
DIX	Data bit in the instance DB
DW#16#	Double word hexadecimal
f	Timer/Counter No.
FB	Function block
FC	Functions
g	Operand range
h	Operand range
I	Input (in the PII)
i	Operand range
i8	Integer (8Bit)
i16	Integer (16Bit)
i32	Integer (32 Bit)
IB	Input byte (in the PII)
ID	Input double word (in the PII)
IW	Input word (in the PII)
k8	Constant (8 Bit)
k16	Constant (16 Bit)
k32	Constant (32 Bit)
Abbreviation	Description

L	Local data
L#	Integer constant (32Bit)
LABEL	Symbolic jump address (max. 4 characters)
LB	Local data byte
LD	Local data double word
LW	Local data word
m	Pointer constant P#x.y (pointer)
M	Bit memory bit
MB	Bit memory byte
MD	Bit memory double word
MW	Bit memory word
n	Binary constant
OB	Organization block
OR	Or
OS	Stored overflow
OV	Overflow
p	Hexadecimal constant
P#	Pointer constant
PIQ	Process image of the outputs
PII	Process image of the inputs
PIB	Periphery input byte (direct periphery access)
PID	Periphery input double word (direct periphery access)
PIW	Periphery input word (direct periphery access)
PQB	Periphery output byte (direct periphery access)
PQD	Periphery output double word (direct periphery access)
PQW	Periphery output word (direct periphery access)
Q	Output (in the PIQ)
q	Real number (32bit floating-point number)
QB	Output byte (in the PIQ)
QD	Output double word (in the PIQ)
QW	Output word (in the PIQ)
r	Block no.
RLO	Result of (previous) logic instruction
S5T#	S5 time constant (16Bit), loads the S5-Timer
SFB	System function block
SFC	System function
STA	Status
T	Timer (times)
T#	Time constant (16/32Bit)
TOD#	IEC time constant
W	area-crossing, register-indirect addressed word
W#16#	Word hexadecimal

Registers

ACCU1 and ACCU2 (32Bit)

The ACCUs are registers for the processing of Byte, words or double words. Therefore the operands are loaded in the ACCUs and combined. The result of the instruction is always in ACCU1.

ACCU	Bit
ACCUx (x=1 to 4)	Bit 0 to Bit 31
ACCUx-L	Bit 0 to Bit 15
ACCUx-H	Bit 16 to Bit 31
ACCUx-LL	Bit 0 to Bit 7
ACCUx-LH	Bit 8 to Bit 15
ACCUx-HL	Bit 16 to Bit 23
ACCUx-HH	Bit 24 to Bit 31

Address register AR1 and AR2 (32Bit)

The address registers contain the area-internal or area-crossing addresses for the register-indirect addressed instructions. The address registers are 32Bit wide.

The area-internal or area-crossing addresses have the following structure:

area-internal address:

00000000 00000bbb bbbbbbbb bbbbbxxx

area-crossing address:

10000yyy 00000bbb bbbbbbbb bbbbbxxx

Legend:

b	Byte address
x	Bit number
Y	Range ID (see chapter "Addressing examples")

Status word (16Bit)

The values are analyzed or set by the instructions.
The status word is 16Bit wide.

Bit	Assignment	Description
0	/FC	First check bit*
1	RLO	Result of (previous) logic instruction
2	STA	Status*
3	OR	Or*
4	OS	Stored overflow
5	OV	Overflow
6	CC0	Condition code
7	CC1	Condition code
8	BR	Binary result
9 to 15	not used	-

* Bit may not be analyzed with the instruction L STW in the user application, for the Bit isn't updated during application run-time.

Addressing examples

Addressing example	Description
Immediate addressing	
L +27	Load 16Bit integer constant "27" in ACCU1.
L L#-1	Load 32Bit integer constant "-1" in ACCU1.
L 2#1010101010101010	Load binary constant in ACCU1.
L DW#16#A0F0_BCFD	Load hexadecimal constant in ACCU1.
L 'End'	Load ASCII code in ACCU1.
L T#500ms	Load time value in ACCU1.
L C#100	Load counter value in ACCU1.
L B#(100,12)	Load constant as 2Byte.
L B#(100,12,50,8)	Load constant as 4Byte.
L P#10.0	Load area-internal pointer in ACCU1.
L P#E20.6	Load area-crossing pointer in ACCU1.
L -2.5	Load real number in ACCU1.
L D#1995-01-20	Load date.
L TOD#13:20:33.125	Load time-of-day.
Direct addressing	
A I 0.0	AND operation of input bit 0.0
L IB 1	Load input byte 1 in ACCU1.
L IW 0	Load input word 0 in ACCU1.
L ID 0	Load input double word 0 in ACCU1.

Indirect addressing timer/counter

SP T [LW 8]	Start timer; timer no. is in local data word 8.
CU C [LW 10]	Start counter; counter no. is in local data word 10.
Memory-indirect, area-internal addressing	
A I [LD 12] e.g.: LP#22.2 T LD 12 A I [LD 12]	AND instruction; input address is in local data double word 12 as pointer.
A I [DBD 1]	AND instruction; input address is in data double word 1 of the DB as pointer.
A Q [DID 12]	AND instruction; output address is in data double word 12 of the instance DB as pointer.
A Q [MD 12]	AND instruction; output address is in bit memory double word 12 as pointer.
Register-indirect, area-internal addressing	
A I [AR1,P#12.2]	AND instruction; input address is calculated "pointer value in address register 1 + pointer P#12.2".
Register-indirect, area-crossing addressing	

For the area-crossing, register indirect addressing the address needs an additional range-ID in the Bits 24-26. The address is in the address register.

Range-ID	**Binary code**	**hex.**	**Area**
P	1000 0**000**	80	Periphery area
I	1000 0**001**	81	Input area
Q	1000 0**010**	82	Output area
M	1000 0**011**	83	Bit memory area
DB	1000 0**100**	84	Data area
DI	1000 0**101**	85	Instance data area
L	1000 0**110**	86	Local data area
VL	1000 0**111**	87	Preceding local data area (access to the local data of the calling block)

L B [AR1,P#8.0]	Load byte in ACCU1; the address is calculated "pointer value in address register 1 + pointer P#8.0".
A [AR1,P#32.3]	AND instruction; operand address is calculated "pointer value in address register 1 + pointer P#32.3".
Addressing via parameters	
A parameter	The operand is addressed via the parameter.

Example for pointer calculation

Example when sum of bit addresses ≤ 7:

```
LAR1 P#8.2
A    I [AR1,P#10.2]
```

Result: The input 18.4 is addressed (by adding the byte and bit addresses)

Example when sum of bit addresses > 7:

```
L MD 0          at will calculated pointer, e.g. P#10.5
LAR1
A    I [AR1,P#10.7]
```

Result: Addressed is input 21.4 (by adding the byte and bit addresses with carry)

Command	Operand	Parameter	Status word									Function	Length in words
			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC		
												: Instruction depends on	
												: Instruction influences	

Math instructions

Fixed-point arithmetic (16Bit)			*Status word*									Math instructions of two 16Bit numbers. The result is in ACCU1 res. ACCU1-L.	
+I	-		BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Add up two integers (16Bit).	1
			-	-	-	-	-	-	-	-	-	(ACCU1-L)=(ACCU1-L)+(ACCU2-L)	
-I	-		-	Y	Y	Y	Y	-	-	-	-	Subtract two integers (16Bit).	1
												(ACCU1-L)=(ACCU2-L)-(ACCU1-L)	
*I	-											Multiply two integers (16Bit).	1
												(ACCU1-L)=(ACCU2-L)*(ACCU1-L)	
/I	-											Divide two integers (16Bit).	1
												(ACCU1-L)=(ACCU2-L):(ACCU1-L)	
												The remainder is in ACCU1-H.	
Fixed-point arithmetic (32Bit)			*Status word*									Math instructions of two 32Bit numbers. The result is in ACCU1.	
+D	-		BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Add up two integers (32Bit).	1
			-	-	-	-	-	-	-	-	-	(ACCU1)=(ACCU2)+(ACCU1)	
-D	-		-	Y	Y	Y	Y	-	-	-	-	Subtract two integers (32Bit).	1
												(ACCU1)=(ACCU2)-(ACCU1)	
*D	-											Multiply two integers (32Bit).	1
												(ACCU1)=(ACCU2)*(ACCU1)	
/D	-											Divide two integers (32Bit).	1
												(ACCU1)=(ACCU2):(ACCU1)	
MOD	-											Divide two integers (32Bit) and load the rest of the	1
												division in ACCU1.	
												(ACCU1)=remainder of [(ACCU2):(ACCU1)]	
Floating-point arithmetic (32Bit)			*Status word*									The result of the math instructions is in ACCU1. The execution time of the instruction depends on the value to calculate.	
+R	-		BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Add up two real numbers (32Bit)	1
			-	-	-	-	-	-	-	-	-	(ACCU1)=(ACCU2)+(ACCU1)	
-R	-		-	Y	Y	Y	Y	-	-	-	-	Subtract two real numbers (32Bit).	1
												(ACCU1)=(ACCU2)-(ACCU1)	
*R	-											Multiply two real numbers (32Bit).	1
												(ACCU1)=(ACCU2)*(ACCU1)	
/R	-											Divide two real numbers (32Bit).	1
												(ACCU1)=(ACCU2):(ACCU1)	
NEGR	-		BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Negate the real number in ACCU1.	1
			-	-	-	-	-	-	-	-	-		
ABS	-		-	-	-	-	-	-	-	-	-	Form the absolute value of the real number in ACCU1.	1

Command	Operand	Parameter	Status word BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Function	Length in words
												: Instruction depends on : Instruction influences	

			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC		
Square root an square instructions (32Bit)			*Status word*									The result of the instructions is in ACCU1. The instructions may be interrupted by alarms.	
SQRT	-		-	-	-	-	-	-	-	-	-	Calculate the Square root of a real number in ACCU1.	1
SQR	-		-	Y	Y	Y	Y	-	-	-	-	Form the square of a real number in ACCU1.	1
Logarithmic function (32Bit)			*Status word*									The result of the logarithm function is in ACCU1. The instructions may be interrupted by alarms.	
LN	-		-	-	-	-	-	-	-	-	-	Calculate the natural logarithm of a real number in ACCU1.	1
EXP	-		-	Y	Y	Y	Y	-	-	-	-	Calculate the exponential value of a real number in ACCU1 on basis e (=2.71828).	1
Trigonometrical functions (32Bit)			*Status word*									The result of the trigonometrical function is in ACCU1. The instructions may be interrupted by alarms.	
SIN [1]	-		-	-	-	-	-	-	-	-	-	Calculate the sine of the real number.	1
ASIN [2]	-		-	Y	Y	Y	Y	-	-	-	-	Calculate the arcsine of the real number.	1
COS [1]	-											Calculate the cosine of the real number.	1
ACOS [2]	-											Calculate the arccosine of the real number.	1
TAN [1]	-											Calculate the tangent of the real number.	1
ATAN [2]	-											Calculate the arctangent of the real number.	1
Addition of constants												Addition of integer constants to ACCU1. The condition code bits are not affected	
+	i8											Add an 8Bit integer constant.	1
+	i16											Add a 16Bit integer constant.	2
+	i32											Add a 32Bit integer constant.	3
Addition via address register												Adding a 16Bit integer to contents of address register. The value is in the instruction or in ACCU1-L. Condition code bits are not affected	
+AR1	-											Add the contents of ACCU1-L to AR1.	1
+AR1	m											Add a pointer constant to the contents of AR1.	2
+AR2	-											Add the contents of ACCU1-L to those of AR2.	1
+AR2	m											Add pointer constant to the contents of AR2.	2

1 Specify the angle in radians; the angle must be given as a floating point value in ACCU 1.

2 The result is an angle in radians.

Command	Operand	Parameter	Status word BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Function	Length in words
												: Instruction depends on : Instruction influences	

Block instructions

Block call instructions			*Status word* BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC		
CALL	FB r, DB r		-	-	-	-	-	-	-	-	-	Unconditional call of a FB, with parameter transfer	1
CALL	SFB r, DB r		-	-	-	-	0	0	1	-	0	Unconditional call of a SFB, with parameter transfer	2
CALL	FC r											Unconditional call of a function, with parameter transfer	1
CALL	SFC r											Unconditional call of a SFC, with parameter transfer	2
UC	FB r FC r Parameter											Unconditional call of blocks, without parameter transfer FB/FC call via parameters	1
CC	FB r FC r Parameter		BR - -	CC1 - -	CC0 - -	OV - -	OS - 0	OR - 0	STA - 1	RLO Y -	/FC - 0	Conditional call of blocks, without parameter transfer FB/FC call via parameters	1
OPN	DB r DI r Parameter		BR - -	CC1 - -	CC0 - -	OV - -	OS - -	OR - -	STA - -	RLO - -	/FC - -	Open a data block Open a instance data block Open a data block via parameter	1/2 2 2
Block end instructions			*Status word* BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC		
BE			-	-	-	-	-	-	-	-	-	End block.	1
BEU			-	-	-	-	0	0	1	-	0	End block unconditionally.	1
BEC			BR - -	CC1 - -	CC0 - -	OV - -	OS - Y	OR - 0	STA - 1	RLO Y 1	/FC - 0	End block if RLO="1".	
Exchanging shared data block an instance data block												Exchanging the two current data blocks. The current shared data block becomes the current instance data block and vice versa. The condition code bits are not affected	
CDB												Exchange shared data block and instant data block.	1

Command	Operand	Parameter	Status word									Function	Length in words
			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC		
												: Instruction depends on	
												: Instruction influences	

Program display and null instruction instructions

Program display and null operation instructions				The status word is not affected.	
BLD	0 ... 255			Program display instruction: is treated by the CPU like a null operation instruction.	1
NOP	0 1			Null operation instruction	1

Edge-triggered instructions

Edge-triggered instructions			*Status word*									Detection of an edge change. The current signal state of the RLO is compared with the signal state of the instruction or edge bit memory. FP detects a change in the RLO from "0" to "1". FN detects a change in the RLO from "1" to "0".	
			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC		
FP	I/Q a.b	0.0 to 127.7										Detecting the positive edge in the RLO. The bit addressed	2
	M a.b	0.0 to 1023.7	-	-	-	-	-	-	-	Y	-	in the instruction is the auxiliary edge bit memory.	2
	L a.b	0.0 to 1043.7	-	-	-	-	-	0	Y	Y	1		2
	DBX a.b	0.0 to 8191.7											2
	DIX a.b	0.0 to 8191.7											2
	c [AR1,m]												2
	c [AR2,m]												2
	[AR1,m]												2
	[AR2,m]												2
	Parameter												2
FN	I/Q a.b	0.0 to 127.7										Detecting the negative edge in the RLO. The bit addressed	2
	M a.b	0.0 to 1023.7	-	-	-	-	-	-	-	Y	-	in the instruction is the auxiliary edge bit memory.	2
	L a.b	0.0 to 1043.7	-	-	-	-	-	0	Y	Y	1		2
	DBX a.b	0.0 to 8191.7											2
	DIX a.b	0.0 to 8191.7											2
	c [AR1,m]												2
	c [AR2,m]												2
	[AR1,m]												2
	[AR2,m]												2
	Parameter												2

Command	Operand	Parameter	Status word								Function	Length in words
			BR	CC1	CC0	OV	OS	OR	STA	RLO /FC		
											: Instruction depends on	
											: Instruction influences	

Load instructions

Load instructions				Loading address identifiers into ACCU1. The contents of ACCU1 and ACCU2 are saved first. The status word is not affected.	
L				Load ...	
	IB a	0 to 127		input byte	1/2
	QB a	0 to 127		output byte	1/2
	PIB a	0 to 1023		periphery input byte	2
	MB a	0 to 1023		bit memory byte	1/2
	LB a	0 to 1043		local data byte	2
	DBB a	0 to 8191		data byte	2
	DIB a	0 to 8191		instance data byte	2
				... in ACCU1	2
	g [AR1,m]			register-indirect, area-internal (AR1)	2
	g [AR2,m]			register-indirect, area-internal (AR2)	2
	B [AR1,m]			area-crossing (AR1)	2
	B [AR2,m]			area-crossing (AR2)	2
	Parameter			via parameters	2
L				Load ...	
	IW a	0 to 126		input word	1/2
	QW a	0 to 126		output word	1/2
	PIW a	0 to 1022		periphery input word	
	MW a	0 to 1022		bit memory word	1/2
	LW a	0 to 1042		local data word	2
	DBW a	0 to 8190		data word	1/2
	DIW a	0 to 8190		instance data word	1/2
				... in ACCU1-L	
	h [AR1,m]			register-indirect, area-internal (AR1)	2
	h [AR2,m]			register-indirect, area-internal (AR2)	2
	W [AR1,m]			area-crossing (AR1)	2
	W [AR2,m]			area-crossing (AR2)	2
	Parameter			via parameters	2

Command	Operand	Parameter	Status word BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Function	Length in words
												: Instruction depends on : Instruction influences	
L												Load ...	
	ID a	0 to 124										input double word.	1/2
	QD a	0 to 124										output double word.	1/2
	PID a	0 to 1020										periphery input double word.	2
	MD a	0 to 1020										bit memory double word.	1/2
	LD a	0 to 1040										local data double word.	2
	DBD a	0 to 8188										data double word.	2
	DID a	0 to 8188										instance data double word.	2
												... in ACCU1-L.	
	i [AR1,m]											register-indirect, area-internal (AR1)	2
	i [AR2,m]											register-indirect, area-internal (AR2)	2
	D [AR1,m]											area-crossing (AR1)	2
	D [AR2,m]											area-crossing (AR2)	2
	Parameter											via parameters	2
L												Load ...	
	k8											8Bit constant in ACCU1-LL.	1
	k16											16Bit constant in ACCU1-L.	2
	k32											32Bit constant in ACCU1.	3
	Parameter											Load constant in ACCU1 (addressed via parameters)	2
L	2#n											Load 16Bit binary constant in ACCU1-L.	2
												Load 32Bit binary constant in ACCU1.	3
L	B#8#p											Load 8Bit hexadecimal constant in ACCU1-LL.	1
	W#16#p											Load 16Bit hexadecimal constant in ACCU1-L.	2
	DW#16#p											Load 32Bit hexadecimal constant in ACCU1.	3
L	x											Load one character.	
L	xx											Load two characters.	2
L	xxx											Load three characters.	
L	xxxx											Load four characters.	3
L	D# Date											Load IEC-date (BCD-coded).	3
L	S5T# time value											Load time constant (16Bit).	2
L	TOD# time value											Load 32Bit time constant. (IEC-time-of-day)	3
L	T# time											Load 16Bit time constant.	2
	value											Load 32Bit time constant.	3
L	C# counter value											Load 16Bit counter constant.	2
L	P# bit pointer											Load bit pointer.	3
L	L# Integer											Load 32Bit integer constant.	3
L	Real											Load real number.	3

Command	Operand	Parameter	Status word									Function	Length in words
			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC		
												: Instruction depends on	
												: Instruction influences	

Load instructions for timer and counter				Load a time or counter value in ACCU1, before the recent content of ACCU1 is saved in ACCU2. The status word is not affected.	
L	T f	0 to 127		Load time value.	1/2
	Timer parameter			Load time value (addressed via parameters).	2
L	C f	0 to 255		Load counter value.	1/2
	Counter parameter			Load counter value (addressed via parameters).	2
LD	T f	0 to 127		Load time value BCD-coded.	1/2
	Timer parameter			Load time value BCD-coded (addressed via parameters).	2
LD	C f	0 to 255		Load counter value BCD-coded.	1/2
	Counter parameter			Load counter value BCD-coded (addressed via parameters).	2

Command	Operand	Parameter	Status word									Function	Length in words
			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC		
												: Instruction depends on	
												: Instruction influences	

Shift instructions

Shift instructions			*Status word*									Shifting the contents of ACCU1 and ACCU1-L to the left or right by the specified number of places. If no address identifier is specified, shift the number of places into ACCU2-LL. Any positions that become free are padded with zeros or the sign. The last shifted bit is in condition code bit CC1.	
SLW	-		BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Shift the contents of ACCU1-L to the left.	1
SLW	0 ... 15		-	-	-	-	-	-	-	-	-	Positions that become free are provided with zeros	
SLD	-		-	Y	Y	Y	-	-	-	-	-	Shift the contents of ACCU1 to the left.	1
SLD	0 ... 32											Positions that become free are provided with zeros.	
SRW	-											Shift the contents of ACCU1-L to the right.	1
SRW	0 ... 15											Positions that become free are provided with zeros.	
SRD	-											Shift the contents of ACCU1 to the right.	1
SRD	0 ... 32											Positions that become free are provided with zeros.	
SSI	-											Shift the contents of ACCU1-L to the right with sign.	1
SSI	0 ... 15											Positions that become free are provided with the sign (Bit 15) .	
SSD	-											Shift the contents of ACCU1 to the right with sign	1
SSD	0 ... 32												
Rotation instructions			*Status word*									Rotate the contents of ACCU1 to the left or right by the specified number of places. If no address identifier is specified, rotate the number of places into ACCU2-LL.	
RLD	-		BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Rotate the contents of ACCU1 to the left.	1
RLD	0 ... 32		-	-	-	-	-	-	-	-	-		
RRD	-		-	Y	Y	Y	-	-	-	-	-	Rotate the contents of ACCU1 to the right.	1
RRD	0 ... 32												
RLDA	-		BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Rotate the contents of ACCU1 one bit position to the left,	
			-	-	-	-	-	-	-	-	-	via CC1 bit.	
RRDA	-		-	Y	0	0	-	-	-	-	-	Rotate the contents of ACCU1 one bit position to the right,	
												via CC1 bit.	

Command	Operand	Parameter	Status word									Function	Length in words
			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC		
												: Instruction depends on	
												: Instruction influences	

Setting/resetting bit addresses

Set/Reset bit addresses			*Status word*									Assign the value "1" or "0" or the RLO to the addressed instructions.	
S			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Set ...	
	I/Q a.b	0.0 to 127.7	-	-	-	-	-	-	-	Y	-	input/output to "1".	1/2
	M a.b	0.0 to 1023.7	-	-	-	-	-	0	Y	-	0	set bit memory to "1".	1/2
	L a.b	0.0 to 1043.7										local data bit to "1".	2
	DBX a.b	0.0 to 8191.7										data bit to "1".	2
	DIX a.b	0.0 to 8191.7										instance data bit to "1".	2
	c [AR1,m]											register-indirect, area-internal (AR1)	2
	c [AR2,m]											register-indirect, area-internal (AR2)	2
	[AR1,m]											area-crossing (AR1)	2
	[AR2,m]											area-crossing (AR2)	2
	Parameter											via parameters	2
R			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Reset ...	
	I/Q a.b	0.0 to 127.7	-	-	-	-	-	-	-	Y	-	input/output to "0".	1/2
	M a.b	0.0 to 1023.7	-	-	-	-	-	0	Y	-	0	set bit memory to "0".	1/2
	L a.b	0.0 to 1043.7										local data bit to "0".	2
	DBX a.b	0.0 to 8191.7										data bit to "0".	2
	DIX a.b	0.0 to 8191.7										instance data bit to "0".	2
	c [AR1,m]											register-indirect, area-internal (AR1)	2
	c [AR2,m]											register-indirect, area-internal (AR2)	2
	W [AR1,m]											area-crossing (AR1)	2
	W [AR2,m]											area-crossing (AR2)	2
	Parameter											via parameters	2
=			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Assign ...	
	I/Q a.b	0.0 to 127.7	-	-	-	-	-	-	-	Y	-	RLO to input/output	1/2
	M a.b	0.0 to 1023.7	-	-	-	-	-	0	Y	-	0	RLO to bit memory	1/2
	L a.b	0.0 to 1043.7										RLO to local data bit	2
	DBX a.b	0.0 to 8191.7										RLO to data bit	2
	DIX a.b	0.0 to 8191.7										RLO to instance data bit	2
	c [AR1,m]											register-indirect, area-internal (AR1)	2
	c [AR2,m]											register-indirect, area-internal (AR2)	2
	[AR1,m]											area-crossing (AR1)	2
	[AR2,m]											area-crossing (AR2)	2
	Parameter											via parameters	2

Command	Operand	Parameter	Status word									Function	Length in words
			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC		
												: Instruction depends on	
												: Instruction influences	

Instructions directly affecting the RLO			*Status word*									The following instructions have a directly effect on the RLO.	
CLR			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Set RLO to "0".	1
			-	-	-	-	-	-	-	-	-		
			-	-	-	-	-	0	0	0	0		
SET			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Set RLO to "1".	1
			-	-	-	-	-	-	-	-	-		
			-	-	-	-	-	0	1	1	0		
NOT			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Negate RLO.	1
			-	-	-	-	-	Y	-	Y	-		
			-	-	-	-	-	-	1	Y	-		
SAVE			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Saving RLO into BR-Bit.	1
			-	-	-	-	-	-	-	Y	-		
			Y	-	-	-	-	-	-	-	-		

Command	Operand	Parameter	Status word									Function	Length in words
			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC		
												: Instruction depends on	
												: Instruction influences	

Jump instructions

Jump instructions			*Status word*									Jump, depending on conditions. Please note that the jump destination always forms the **beginning** of a Boolean logic string in the case of jump instructions. The jump destination must not be included in the logic string.	
JU	LABEL		BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Jump unconditionally.	½
			-	-	-	-	-	-	-	-	-		
			-	-	-	-	-	-	-	-	-		
JC	LABEL		BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Jump if RLO="1".	½
JCN	LABEL		-	-	-	-	-	-	-	Y	-	Jump if RLO="0".	2
			-	-	-	-	-	0	1	1	0		
JCB	LABEL		BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Jump if RLO="1".	2
			-	-	-	-	-	-	-	Y	-	Save the RLO in the BR-Bit	
JNB	LABEL		Y	-	-	-	-	0	1	1	0	Jump if RLO="0".	2
												Save the RLO in the BR-Bit	
JBI	LABEL		BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Jump if BR="1".	2
JNBI	LABEL		Y	-	-	-	-	-	-	-	-	Jump if BR="0".	2
			-	-	-	-	-	0	1	-	0		
JO	LABEL		BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Jump on stored overflow (OV="1").	½
			-	-	-	Y	-	-	-	-	-		
			-	-	-	-	-	-	-	-	-		
JOS	LABEL		BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Jump on stored overflow (OS="1").	2
			-	-	-	-	Y	-	-	-	-		
			-	-	-	-	0	-	-	-	-		
JUO	LABEL		BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Jump if "unordered instruction" (CC1=1 and CC0=1).	2
JZ	LABEL		-	Y	Y	-	-	-	-	-	-	Jump if result=0 (CC1=0 and CC0=0).	½
JP	LABEL		-	-	-	-	-	-	-	-	-	Jump if result>0 (CC1=1 and CC0=0).	½
JM	LABEL											Jump if result<0 (CC1=0 and CC0=1).	½
JN	LABEL											Jump if result≠0. (CC1=1 and CC0=0) or (CC1=0) and (CC0=1)	½
JMZ	LABEL											Jump if result≤0. (CC1=0 and CC0=1) or (CC1=0 and CC0=0)	2
JPZ	LABEL											Jump if result≥0. (CC1=1 and CC0=0) or (CC1=0 and CC0=0)	2

Command	Operand	Parameter	Status word BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Function	Length in words
												: Instruction depends on : Instruction influences	

			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC		
JL	LABEL		-	-	-	-	-	-	-	-	-	Jump distributor This instruction is followed by a list of jump instructions. The operand is a jump label to subsequent instructions in this list. ACCU1-L contains the number of the jump instruction to be executed.	2
LOOP	LABEL											Decrement ACCU1-L and jump if ACCU1-L _ 0 (loop programming)	2

Transfer instructions

Transfer instructions				Transfer the contents of ACCU1 into the addressed operand. The status word is not affected.	
T				Transfer the contents of ACCU1-LL to...	
	IB a	0 to 127		input byte.	1/2
	QB a	0 to 127		output byte.	1/2
	PQB a	0 to 1023		periphery output byte.	1/2
	MB a	0 to 1023		bit memory byte.	1/2
	LB a	0 to 1043		local data byte.	2
	DBB a	0 to 8191		data byte.	2
	DIB a	0 to 8191		instance data byte.	2
	g [AR1,m]			register-indirect, area-internal (AR1)	2
	g [AR2,m]			register-indirect, area-internal (AR2)	2
	B [AR1,m]			area-crossing (AR1)	2
	B [AR2,m]			area-crossing (AR2)	2
	Parameter			via parameters	2
T				Transfer the contents of ACCU1-L to ...	
	IW	0 to 126		input word.	1/2
	QW	0 to 126		output word.	1/2
	PQW	0 to 1022		periphery output word.	1/2
	MW	0 to 1022		bit memory word.	1/2
	LW	0 to 1042		local data word.	2
	DBW	0 to 8190		data word.	2
	DIW	0 to 8190		instance data word.	2
	h [AR1,m]			register-indirect, area-internal (AR1)	2
	h [AR2,m]			register-indirect, area-internal (AR2)	2
	W [AR1,m]			area-crossing (AR1)	2
	W [AR2,m]			area-crossing (AR2)	2
	Parameter			via parameters	2

Command	Operand	Parameter	Status word								Function	Length in words
			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	
												: Instruction depends on
												: Instruction influences
T											Transfer the contents of ACCU1 to...	
	ID	0 to 124									input double word.	1/2
	QD	0 to 124									output double word.	1/2
	PQD	0 to 1020									periphery output double word.	1/2
	MD	0 to 1020									bit memory double word.	1/2
	LD	0 to 1040									local data double word.	2
	DBD	0 to 8188									data double word.	2
	DID	0 to 8188									instance data double word.	2
	i [AR1,m]										register-indirect, area-internal (AR1)	2
	i [AR2,m]										register-indirect, area-internal (AR2)	2
	D [AR1,m]										area-crossing (AR1)	2
	D [AR2,m]										area-crossing (AR2)	2
	Parameter										via parameters	2
Load and transfer instructions for address register											Load a double word from a memory area or a register into AR1 or AR2.	
LAR1											Load the contents from...	
	-										ACCU1.	1
	AR2										address register 2.	1
	DBD a	0 to 8188									data double word.	2
	DID a	0 to 8188									instance data double word.	2
	m										32Bit constant as pointer.	3
	LD a	0 to 1040									local data double word.	2
	MD a	0 to 1020									bit memory double word.	2
											... into AR1.	
LAR2											Load the contents from ...	
	-										ACCU1.	1
	DBD a	0 to 8188									data double word.	2
	DID a	0 to 8188									instance data double word.	2
	m										32Bit constant as pointer.	3
	LD a	0 to 1040									local data double word.	2
	MD a	0 to 1020									bit memory double word.	2
											... into AR2.	
TAR1											Transfer the contents from AR1 to...	
	-										ACCU1.	1
	AR2										address register 2.	1
	DBD a	0 to 8188									data double word.	2
	DID a	0 to 8188									instance data double word.	2
	LD a	0 to 1040									local data double word.	2
	MD a	0 to 1020									bit memory double word.	2

Command	Operand	Parameter	Status word BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Function	Length in words
												: Instruction depends on : Instruction influences	
TAR2												Transfer the contents from AR2 to...	
	-											ACCU1.	1
	DBD a	0 to 8188										data double word.	2
	DID a	0 to 8188										instance data double word.	2
	LD a	0 to 1040										local data double word.	2
	MD a	0 to 1020										bit memory double word.	2
TAR												Exchange the contents of AR1 and AR2	1
Load and transfer instructions for the status word			*Status word*									Load status word from a memory or a register of AR1 or AR2.	
L	STW		BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Load status word in ACCU1.	
	-		Y	Y	Y	Y	Y	0	0	Y	0		
			-	-	-	-	-	-	-	-	-		
T	STW		BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Transfer ACCU1 (Bits 0 to 8) into status word.	
	-		-	-	-	-	-	-	-	-	-		
			Y	Y	Y	Y	Y	-	-	Y	-		
Load instructions for DB number and DB length												Load the number/length of a data block to ACCU1. The old contents of ACCU1 are saved into ACCU2. The condition code bits are not affected	
L	DBNO											Load number of data block.	1
L	DINO											Load number of instance data block.	1
L	DBLG											Load length of data block into byte.	1
L	DILG											Load length of instance data block into byte.	1
ACCU transfer instructions, increment, decrement												The status word is not affected.	
CAW	-											Reverse the order of the bytes in ACCU1-L. LL, LH becomes LH, LL.	1
CAD	-											Reverse the order of the bytes in ACCU1. LL, LH, HL, HH becomes HH, HL, LH, LL.	1
TAK	-											Swap the contents of ACCU1 and ACCU2	1
PUSH	-											The contents of ACCU1 are transferred to ACCU2.	1
POP	-											The contents of ACCU2 are transferred to ACCU1.	1
INC	0 ... 255											Increment ACCU1-LL.	1
DEC	0 ... 255											Decrement ACCU1-LL.	1

Command	Operand	Parameter	Status word									Function	Length in words
			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC		
												: Instruction depends on	
												: Instruction influences	

Data type conversion instructions

Data type conversion instructions			*Status word*									The results of the conversion are in ACCU1. When converting real numbers, the execution time depends on the value.	
BTI	-		BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Convert contents of ACCU1 from BCD to integer (16Bit)	1
			-	-	-	-	-	-	-	-	-	(**B**CD **T**o **I**nt.).	
BTD	-		-	-	-	-	-	-	-	-	-	Convert contents of ACCU1 from BCD to integer (32Bit).	1
												(**B**CD **T**o **D**oubleint.).	
DTR	-											Convert cont. of ACCU1 from integer (32Bit) to Real number	1
												(32Bit) (**D**oubleint. **T**o **R**eal).	
ITD	-											Convert contents of ACCU1 from integer (16Bit) to integer	1
												(32Bit) (**I**nt. **T**o **D**oubleint.).	
ITB	-		BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Convert contents of ACCU1 from integer (16Bit) to BCD	1
			-	-	-	-	-	-	-	-	-	0 to +/-999 (**I**nt. **T**o **B**CD).	
DTB	-		-	-	-	Y	Y	-	-	-	-	Convert contents of ACCU1 from integer (32Bit) to BCD	1
												0 to +/-9 999 999 (**D**oubleint. **T**o **B**CD).	
RND	-		BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Convert a real number to 32Bit integer.	1
RND-	-		-	-	-	-	-	-	-	-	-	Convert a real number to 32Bit integer.	1
			-	-	-	Y	Y	-	-	-	-	The number is rounded next hole number..	
RND+	-											Convert real number to 32Bit integer.	1
												It is rounded up to the next integer.	
TRUNC	-											Convert real number to 32Bit integer.	1
												The places after the decimal point are truncated.	
Complement creation			*Status word*										
INVI	-		BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Forms the ones complement of ACCU1-L (integer).	1
INVD	-		-	-	-	-	-	-	-	-	-	Forms the ones complement of ACCU1.	1
			-	-	-	-	-	-	-	-	-		
NEGI	-		BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Forms the twos complement of ACCU1-L.	1
NEGD	-		-	-	-	-	-	-	-	-	-	Forms the twos complement of ACCU1 (double integer).	1
			-	Y	Y	Y	Y	-	-	-	-		

Command	Operand	Parameter	Status word									Function	Length in words
			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC		
												: Instruction depends on	
												: Instruction influences	

Comparison instructions

Comparison instructions with integer (16Bit)			*Status word*									Comparing the integer (16Bit) in ACCU1-L and ACCU2-L. RLO=1, if condition is satisfied.	
==I	-		BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	ACCU2-L=ACCU1-L	1
<>I	-		-	-	-	-	-	-	-	-	-	ACCU2-L≠ACCU1-L	1
<I	-		-	Y	Y	0	-	0	Y	Y	1	ACCU2-L<ACCU1-L	1
<=I	-											ACCU2-L<=ACCU1-L	1
>I	-											ACCU2-L>ACCU1-L	1
>=I	-											ACCU2-L>=ACCU1-L	1
Comparison instructions with integer (32Bit)			*Status word*									Comparing the integer (32Bit) in ACCU1 and ACCU2. RLO=1, if condition is satisfied.	
==D	-		BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	ACCU2=ACCU1	1
<>D	-		-	-	-	-	-	-	-	-	-	ACCU2≠ACCU1	1
<D	-		-	Y	Y	0	-	0	Y	Y	1	ACCU2<ACCU1	1
<=D	-											ACCU2<=ACCU1	1
>D	-											ACCU2>ACCU1	1
>=D	-											ACCU2>=ACCU1	1
Comparison instructions with 32Bit real number			*Status word*									Comparing the 32Bit real numbers in ACCU1 and ACCU2. RLO=1, is condition is satisfied. The execution time of the instruction depends on the value to be compared.	
==R	-		BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	ACCU2=ACCU1	1
<>R	-		-	-	-	-	-	-	-	-	-	ACCU2≠ACCU1	1
<R	-		-	Y	Y	Y	Y	0	Y	Y	1	ACCU2<ACCU1	1
<=R	-											ACCU2<=ACCU1	1
>R	-											ACCU2>ACCU1	1
>=R	-											ACCU2>=ACCU1	1

Command	Operand	Parameter	Status word									Function	Length in words
			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC		
												: Instruction depends on	
												: Instruction influences	

Bit logic instructions (Bit)

Bit logic instructions with bit operands			*Status word*									Examining the signal state of the addressed instruction and gating the result with the RLO according to the appropriate logic function.	
A			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	AND operation at signal state "1"	
	I/Q a.b	0.0 ... 127.7	-	-	-	-	-	Y	-	Y	Y	Input/output	1/2
	M a.b	0.0 ... 1023.7	-	-	-	-	-	Y	Y	Y	1	Bit memory	1/2
	L a.b	0.0 ... 1043.7										Local data bit	2
	DBX a.b	0.0 ... 8191.7										Data bit	2
	DIX a.b	0.0 ... 8191.7										Instance data bit	2
	c [AR1,m]											register-indirect, area-internal (AR1)	2
	c [AR2,m]											register-indirect, area-internal (AR2)	2
	[AR1,m]											area-crossing (AR1)	2
	[AR2,m]											area-crossing (AR2)	2
	Parameter											via parameters	2
AN			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	AND operation of signal state "0"	
	I/Q a.b	0.0 ... 127.7	-	-	-	-	-	Y	-	Y	Y	Input/output	1/2
	M a.b	0.0 ... 1023.7	-	-	-	-	-	Y	Y	Y	1	Bit memory	1/2
	L a.b	0.0 ... 1043.7										Local data bit	2
	DBX a.b	0.0 ... 8191.7										Data bit	2
	DIX a.b	0.0 ... 8191.7										Instance data bit	2
	c [AR1,m]											register-indirect, area-internal (AR1)	2
	c [AR2,m]											register-indirect, area-internal (AR2)	2
	[AR1,m]											area-crossing (AR1)	2
	[AR2,m]											area-crossing (AR2)	2
	Parameter											via parameters	2
O			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	OR operation at signal state "1"	
	I/Q a.b	0.0 ... 127.7	-	-	-	-	-	-	-	Y	Y	Input/output	1/2
	M a.b	0.0 ... 1023.7	-	-	-	-	-	0	Y	Y	1	Bit memory	1/2
	L a.b	0.0 ... 1043.7										Local data bit	2
	DBX a.b	0.0 ... 8191.7										Data bit	2
	DIX a.b	0.0 ... 8191.7										Instance data bit	2
	c [AR1,m]											register-indirect, area-internal (AR1)	2
	c [AR2,m]											register-indirect, area-internal (AR2)	2
	[AR1,m]											area-crossing (AR1)	2
	[AR2,m]											area-crossing (AR2)	2
	Parameter											via parameters	2

Command	Operand	Parameter	Status word BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Function	Length in words
												: Instruction depends on : Instruction influences	

			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC		
ON												OR operation at signal state "0"	
	I/Q a.b	0.0 ... 127.7	-	-	-	-	-	-	-	Y	Y	Input/output	1/2
	M a.b	0.0 ... 1023.7	-	-	-	-	-	0	Y	Y	1	Bit memory	1/2
	L a.b	0.0 ... 1043.7										Local data bit	2
	DBX a.b	0.0 ... 8191.7										Data bit	2
	DIX a.b	0.0 ... 8191.7										Instance data bit	2
	c [AR1,m]											register-indirect, area-internal (AR1)	2
	c [AR2,m]											register-indirect, area-internal (AR2)	2
	[AR1,m]											area-crossing (AR1)	2
	[AR2,m]											area-crossing (AR2)	2
	Parameter											via parameters	2
X												EXCLUSIVE-OR operation at signal state "1"	
	I/Q a.b	0.0 ... 127.7	-	-	-	-	-	-	-	Y	Y	Input/output	2
	M a.b	0.0 ... 1023.7	-	-	-	-	-	0	Y	Y	1	Bit memory	2
	L a.b	0.0 ... 1043.7										Local data bit	2
	DBX a.b	0.0 ... 8191.7										data bit	2
	DIX a.b	0.0 ... 8191.7										Instance data bit	2
	c [AR1,m]											register-indirect, area-internal (AR1)	2
	c [AR2,m]											register-indirect, area-internal (AR2)	2
	[AR1,m]											area-crossing (AR1)	2
	[AR2,m]											area-crossing (AR2)	2
	Parameter											via parameters	2
XN												EXCLUSIVE-OR operation at signal state "0"	
	I/Q a.b	0.0 ... 127.7	-	-	-	-	-	-	-	Y	Y	Input/output	2
	M a.b	0.0 ... 1023.7	-	-	-	-	-	0	Y	Y	1	Bit memory	2
	L a.b	0.0 ... 1043.7										Local data bit	2
	DBX a.b	0.0 ... 8191.7										Data bit	2
	DIX a.b	0.0 ... 8191.7										Instance data bit	2
	c [AR1,m]											register-indirect, area-internal (AR1)	2
	c [AR2,m]											register-indirect, area-internal (AR2)	2
	[AR1,m]											area-crossing (AR1)	2
	[AR2,m]											area-crossing (AR2)	2
	Parameter											via parameters	2

Command	Operand	Parameter	Status word									Function	Length in words
			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC		
												: Instruction depends on	
												: Instruction influences	

Bit logic instructions with parenthetical expressions			*Status word*									Saving the bits BR, RLO, OR and a function ID (A, AN, ...) at the nesting stack. For each block 7 nesting levels are possible.	
A(			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	AND left parenthesis	1
AN(			Y	-	-	-	-	Y	-	Y	Y	AND-NOT left parenthesis	1
O(			-	-	-	-	-	0	1	-	0	OR left parenthesis	1
ON(												OR-NOT left parenthesis	1
X(												EXCLUSIVE-OR left parenthesis	1
XN(												EXCLUSIVE-OR-NOT left parenthesis	1
)			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Right parenthesis, popping an entry off the nesting stack,	1
			-	-	-	-	-	-	-	Y	-	gating RLO with the current RLO in the processor.	
			Y	-	-	-	-	Y	1	Y	1		
ORing of AND operations			*Status word*									The ORing of AND operations is implemented according the rule: AND before OR.	
O			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	OR operations of AND functions according the rule:	1
			-	-	-	-	-	Y	-	Y	Y	AND before OR	
			-	-	-	-	-	Y	1	-	Y		

Command	Operand	Parameter	Status word BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Function	Length in words
												: Instruction depends on	
												: Instruction influences	

Logic instructions with timer and counters			*Status word* BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Examining the signal state of the addressed timer/counter an gating the result with the RLO according to the appropriate logic function.	
A												AND operation at signal state	
	T f	0 to 127	-	-	-	-	-	Y	-	Y	Y	Timer	1/2
	C f	0 to 255	-	-	-	-	-	Y	Y	Y	1	Counter	1/2
	Timer para. Counter p.											Timer addressed via parameters Counter addressed via parameters	2
AN												AND operation at signal state	
	T f	0 to 127	-	-	-	-	-	Y	-	Y	Y	Timer	1/2
	C f	0 to 255	-	-	-	-	-	Y	Y	Y	1	Counter	1/2
	Timer para. Counter p.											Timer addressed via parameters Counter addressed via parameters	2
O												OR operation at signal state	
	T f	0 to 127	-	-	-	-	-	-	-	Y	Y	Timer	1/2
	C f	0 to 255	-	-	-	-	-	0	Y	Y	1	Counter	1/2
	Timer para. Counter p.											Timer addressed via parameters Counter addressed via parameters	2
ON												OR operation at signal state	
	T f	0 to 127	-	-	-	-	-	-	-	Y	Y	Timer	1/2
	C f	0 to 255	-	-	-	-	-	0	Y	Y	1	Counter	1/2
	Timer para. Counter p.											Timer addressed via parameters Counter addressed via parameters	2
X												EXCLUSIVE-OR operation at signal state	
	T f	0 to 127	-	-	-	-	-	-	-	Y	Y	Timer	2
	C f	0 to 255	-	-	-	-	-	0	Y	Y	1	Counter	2
	Timer para. Counter p.											Timer addressed via parameters Counter addressed via parameters	2
XN												EXCLUSIVE-OR operation at signal state	
	T f	0 to 127	-	-	-	-	-	-	-	Y	Y	Timer	2
	C f	0 to 255	-	-	-	-	-	0	Y	Y	1	Counter	2
	Timer para. Counter p.											Timer addressed via parameters Counter addressed via parameters	2

Command	Operand	Parameter	Status word									Function	Length in words
			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC		
												: Instruction depends on	
												: Instruction influences	

Evaluating Conditions Using AND, OR and EXCLUSIVE OR			*Status word*									Examining the specified conditions for their signal status, and gating the result with the RLO according to the appropriate function.	
A			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	AND operation at signal state "1"	
	==0		Y	Y	Y	Y	Y	Y	-	Y	Y	Result=0 (CC1=0) and (CC0=0)	1
	>0		-	-	-	-	-	Y	Y	Y	1	Result>0 (CC1=1) and (CC0=0)	1
	<0											Result<0 (CC1=0) and (CC0=1)	1
	<>0											Result≠0 ((CC1=0) and (CC0=1)) or ((CC1=1) and (CC0=0))	1
	<=0											Result<=0 ((CC1=0) and (CC0=1)) or ((CC1=0) and (CC0=0))	1
	>=0											Result>=0 ((CC1=1) and (CC0=0)) or ((CC1=1) and (CC0=0))	1
	UO											unordered math instruction (CC1=1) and (CC0=1)	1
	OS											OS=1	1
	BR											BR=1	1
	OV											OV=1	1
AN			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	AND operation at signal state "0"	
	==0		Y	Y	Y	Y	Y	Y	-	Y	Y	Result=0 (CC1=0) and (CC0=0)	1
	>0		-	-	-	-	-	Y	Y	Y	1	Result>0 (CC1=1) and (CC0=0)	1
	<0											Result<0 (CC1=0) and (CC0=1)	1
	<>0											Result≠0 ((CC1=0) and (CC0=1)) or ((CC1=1) and (CC0=0))	1
	<=0											Result<=0 ((CC1=0) and (CC0=1)) or ((CC1=0) and (CC0=0))	1
	>=0											Result>=0 ((CC1=1) and (CC0=0)) or ((CC1=1) and (CC0=0))	1
	UO											unordered math instruction (CC1=1) and (CC0=1)	1
	OS											OS=0	1
	BR											BR=0	1
	OV											OV=0	1
O			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	OR operation at signal state "1"	
	==0		Y	Y	Y	Y	Y	-	-	Y	Y	Result=0 (CC1=0) and (CC0=0)	1
	>0		-	-	-	-	-	0	Y	Y	1	Result>0 (CC1=1) and (CC0=0)	1
	<0											Result<0 (CC1=0) and (CC0=1)	1
	<>0											Result≠0 ((CC1=0) and (CC0=1)) or ((CC1=1) and (CC0=0))	1
	<=0											Result<=0 ((CC1=0) and (CC0=1)) or ((CC1=0) and (CC0=0))	1
	>=0											Result>=0 ((CC1=1) and (CC0=0)) or ((CC1=1) and (CC0=0))	1
	UO											unordered math instruction (CC1=1) and (CC0=1)	1
	OS											OS=1	1
	BR											BR=1	1
	OV											OV=1	1

Command	Operand	Parameter	Status word									Function	Length in words
			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC		
												: Instruction depends on	
												: Instruction influences	

ON			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	OR operation at signal state "0"	
	==0		Y	Y	Y	Y	Y	-	-	Y	Y	Result=0 (CC1=0) and (CC0=0)	1
	>0		-	-	-	-	-	0	Y	Y	1	Result>0 (CC1=1) and (CC0=0)	1
	<0											Result<0 (CC1=0) and (CC0=1)	1
	<>0											Result≠0 ((CC1=0) and (CC0=1)) or ((CC1=1) and (CC0=0))	1
	<=0											Result<=0 ((CC1=0) and (CC0=1)) or ((CC1=0) and (CC0=0))	1
	>=0											Result>=0 ((CC1=1) and (CC0=0)) or ((CC1=1) and (CC0=0))	1
	UO											unordered math instruction (CC1=1) and (CC0=1)	1
	OS											OS=0	1
	BR											BR=0	1
	OV											OV=0	1
X			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	EXCLUSIVE-OR operation at signal state "1"	
	==0		Y	Y	Y	Y	Y	-	-	Y	Y	Result=0 (CC1=0) and (CC0=0)	1
	>0		-	-	-	-	-	0	Y	Y	1	Result>0 (CC1=1) and (CC0=0)	1
	<0											Result<0 (CC1=0) and (CC0=1)	1
	<>0											Result≠0 ((CC1=0) and (CC0=1)) or ((CC1=1) and (CC0=0))	1
	<=0											Result<=0 ((CC1=0) and (CC0=1)) or ((CC1=0) and (CC0=0))	1
	>=0											Result>=0 ((CC1=1) and (CC0=0)) or ((CC1=1) and (CC0=0))	1
	UO											unordered math instruction (CC1=1) and (CC0=1)	1
	OS											OS=1	1
	BR											BR=1	1
	OV											OV=1	1
XN			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	EXCLUSIVE-OR operation at signal state "0"	
	==0		Y	Y	Y	Y	Y	-	-	Y	Y	Result=0 (CC1=0) and (CC0=0)	1
	>0		-	-	-	-	-	0	Y	Y	1	Result>0 (CC1=1) and (CC0=0)	1
	<0											Result<0 (CC1=0) and (CC0=1)	1
	<>0											Result≠0 ((CC1=0) and (CC0=1)) or ((CC1=1) and (CC0=0))	1
	<=0											Result<=0 ((CC1=0) and (CC0=1)) or ((CC1=0) and (CC0=0))	1
	>=0											Result>=0 ((CC1=1) and (CC0=0)) or ((CC1=1) and (CC0=0))	1
	UO											unordered math instruction (CC1=1) and (CC0=1)	1
	OS											OS=0	1
	BR											BR=0	1
	OV											OV=0	1

Command	Operand	Parameter	Status word									Function	Length in words
			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC		
												: Instruction depends on	
												: Instruction influences	

Word logic instructions with the contents of ACCU1

Word Logic Instructions with the Contents of ACCU1			*Status word*									Gating the contents of ACCU1 and/or ACCU1-L with a word or double word according to the appropriate function. The word or double word is either a constant in the instruction or in ACCU2. The result is in ACCU1 and/or ACCU1-L.	
			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC		
AW			-	-	-	-	-	-	-	-	-	AND ACCU2-L	1
AW	k16											AND 16Bit constant	2
OW			-	Y	0	0	-	-	-	-	-	OR ACCU2-L	1
OW	k16											OR 16Bit constant	2
XOW												EXCLUSIVE OR ACCU2-L	1
XOW	k16											EXCLUSIVE OR 16Bit constant	2
AD												AND ACCU2	1
AD	k32											AND 32Bit constant	3
OD												OR ACCU2	1
OD	k32											OR 32Bit constant	3
XOD												EXCLUSIVE OR ACCU2	1
XOD	k32											EXCLUSIVE OR 32Bit constant	3

Timer instructions

Time instructions			*Status word*									Starting or resetting a timer (addressed directly or via parameters). The time value must be in ACCU1-L.	
			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC		
SP	T f	0 to 127										Start time as pulse on edge change from "0" to "1".	1/2
	Timer para.		-	-	-	-	-	-	-	Y	-		2
SE	T f	0 to 127	-	-	-	-	-	0	-	-	0	Start timer as extended pulse on edge change from	1/2
	Timer para.											"0" to "1".	2
SD	T f	0 to 127										Start timer as ON delay on edge change	1/2
	Timer para.											from "0" to "1".	2
SS	T f	0 to 127										Start timer as saving start delay on edge change	1/2
	Timer para.											from "0" to "1".	2
SA	T f	0 to 127										Start timer as OFF delay on edge change from	1/2
	Timer para.											"1" to "0".	2
FR	T f	0 to 127										Enable timer for restarting on edge change from "0" to "1"	1/2
	Timer para.											(reset edge bit memory for starting timer).	2
R	T f	0 to 127										Reset timer.	1/2

Command	Operand	Parameter	Status word									Function	Length in words
			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC		
												: Instruction depends on : Instruction influences	
	Timer para.												2

Command	Operand	Parameter	Status word									Function	Length in words
			BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC		
												: Instruction depends on	
												: Instruction influences	

Counter instructions

Counter instructions			*Status word*									The counter value is in ACCU1-L res. in the address transferred as parameter.	
S	C f	0 to 255	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC	Presetting of counter on edge change from "0" to "1".	1/2
	Counter p.		-	-	-	-	-	-	-	Y	-		2
R	C f	0 to 255	-	-	-	-	-	0	-	-	0	Reset counter to "0".	1/2
	Counter p.												2
CU	C f	0 to 255										Increment counter by 1 on edge change from "0" to "1".	1/2
	Counter p.												2
CD	C f	0 to 255										Decrement counter by 1 on edge change from "0" to "1".	1/2
	Counter p.												2
FR	C f	0 to 255										Enable counter on edge change from "0" to "1".	1/2
	Counter p.											(reset the edge bit memory for up and down counting)	2

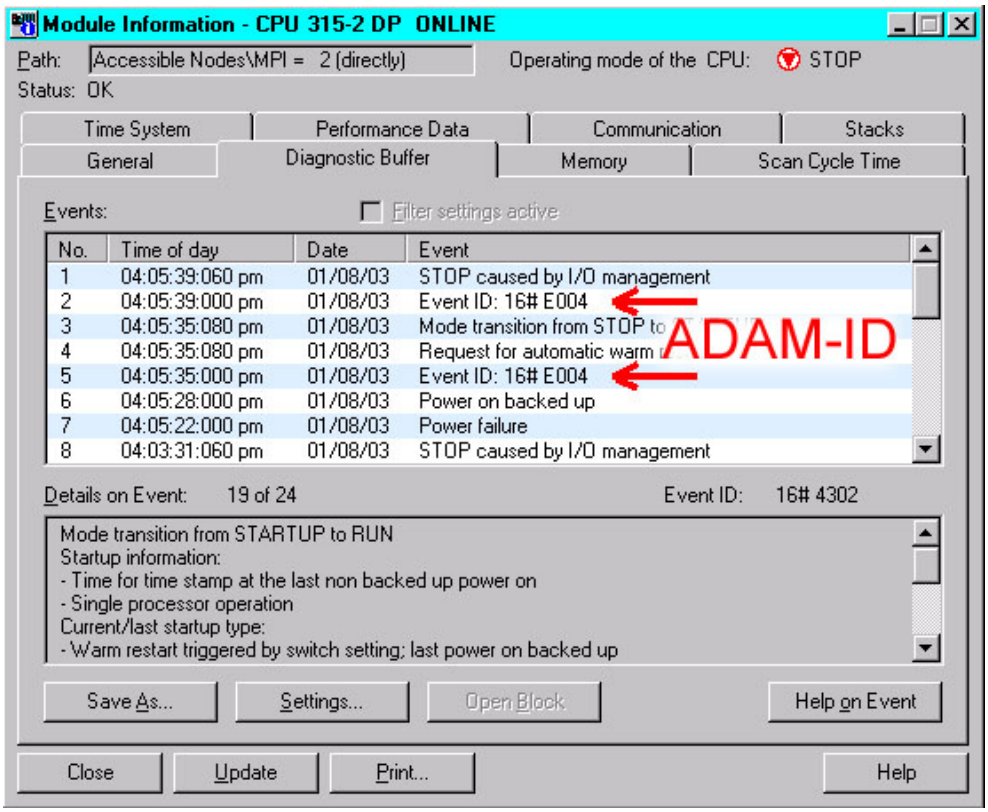
ADAM specific diagnostic entries

Entries in the diagnostic buffer

You may read the diagnostic buffer of the CPU via the STEP®7 Manager from Siemens. Besides of the standard entries in the diagnostic buffer, the ADAM CPUs support some additional specific entries in form of event-IDs.

Monitoring the diagnostic entries

To monitor the diagnostic entries you choose the option **PLC** > *Module Information* in the STEP®7 Manager. Via the register "Diagnostic Buffer" you reach the diagnostic window:



The diagnosis is independent from the operating mode of the CPU. You may store a max. of 100 diagnostic entries in the CPU.

The following page shows an overview of the ADAM specific Event-IDs.

Overview of the Event-IDs

Event-ID	Description
0xE000	reserved
0xE001	reserved
0xE002	reserved
0xE003	Error at parameterization/ configuration ADAM-BUS
0xE004	Multi-parameterization of a peripheral address
0xE005	Error at address information parsing
0xE006	Error at DP slave information parsing
0xE007	Configured in-/output bytes do not fit in the peripheral area (DP-Slave / High-speed counter)
0xE008	Error at High-speed counter information parsing
0xE009	reserved
0xE010	Not defined module detected at backplane bus
0xE0CC	Communication error MPI / Serial
0xE100	MMC access error
0xE101	MMC file system error
0xE102	MMC FAT error
0xE200	MMC writing ready (Copy Ram2Rom)
0xE300	Internal Flash writing ready (Copy Ram2Rom)