9 PROGRAMMABLE LOGIC CONTROLLERS BERNECKER-RAINER – EXTENDED PROGRAMMING FUNCTIONS, ANALOG I/O, COMMUNICATIONS

Study duration: 90 min.

9.1 Library Manager

The Library Manager is the interface used to completely manage the libraries used in a project. This includes managing standard libraries and libraries from third-party suppliers as well as offering support when creating user libraries.

The Library Manager is fully integrated in Automation Studio and can be accessed via the menu option Open: Library Manager. The Library Manager is divided into two main sections:

The libraries integrated in the project are displayed in the left window with the corresponding functions and function blocks.

Information and properties of the element selected in the left window are displayed on different tabs in the right window. [27, 29]

This includes:

- Data types and constants used by the library.
- Additional dependencies from other libraries.
- Parameter declaration of the functions and function blocks.
- Management of the source code files for ANSI C libraries.

B&R provides a comprehensive package of standard libraries with Automation Runtime.

The function range of these standard libraries begins with simple functions and function blocks, which are not contained in the standard code of the respective programming language or which can be deleted with simple loops.

Examples of this include the following:

- Timer (delays), counter, edge detection.
- String processing.
- Arithmetic.
- Logic operations.

Highly complex and powerful functions and function blocks are also contained, which considerably minimize the development effort required for applications and save a great deal of program code.

Here a few examples of this:

- Control algorithms.
- Data objects and file management.
- Webserver data exchange.

- Network functions.
- Axis control.
- Graphics functions.

Tab. 9.1: Overview of Library.

Library	**Description**
Acp10_mc	Motion function block for ACOPOS drives, specified in PLCopen - TechnicalCommittee 2 –Task Force "Function blocks for motion control V1.0"
AsArCfg	Reading and writing Automation Runtime configuration settings
AsArProf	Operation of the AR profiler from an application
AsCont	Support of hardware modules
AsHost	Conversion of IP addresses to domain names and vice-versa
AsHW	Reading information from the respective target system
Astma	Activation of the INA2000 manager
AsIO	Determining the states and values of a data point, force handling
AsIOAcc	Reading and writing access to non-cyclic I/O data points
AsIODiag	Creation and evaluation of IO module information
AsIOMMan	Activation and deactivation of IO configuration modules or mapping modules
AsKey	Functions for querying the dongle
AsL2DP	Operation of the L2DP slave module 3IF661.9 and 3IF761.9
AsMath	Mathematic functions not covered by the "OPERATOR" library
AsMem	Managing memory blocks in large memory areas (memory partitions)
AsPlkSup	Access to different configuration words for the 2003 screw-in modules
AsString	Memory manipulation and string handling
AsTime	Support of date and time functions on the controller
AsUPS	Communication with a UPS
AsWeigh	Function blocks for using strain gauge modules (e.g. AI261).
AsWStr	Processing of 16-bit wide character (Unicode)
C220man	Operation of the panel controllers
CAN_Lib	Operation of the CAN controller
CANIO	Operation of B&R2003 CAN nodes
Commserv	System expansion for INAclnt library
CONVERT	Conversion functions according to IEC61131-3
DataObj	Handling of data objects
DM_Lib	Storage of data objects in nonvolatile memory
DRV_3964	3964R protocol
DRV_mbus	Modbus protocol
DRV_mn	MiniNet protocol

Library	Description
DVFrame	Frame driver library for serial interface operation
Ethernet	Data exchange via UDP or TCP
EthSock	Integration of socket functions
FDD_Lib	Operation of the external floppy disk station MFDD70S
FileIO	File and directory management
IF361	Operation of the IF361 interface module (Profibus DP Slave)
IF661	Operation of IF661 interface module (Profibus DP Slave)
INAclnt	INA2000 client communication
IOConfig	Execution of shovel tasks on 2003
IOCtrl	Operation of 2003 IOs
IO_lib	Operation of the I/O modules
Logging	Operation of the profiler from an application
LoopConR	Implementation of tasks for control technology (calculation with REAL values)
LoopCont	Implementation of tasks for control technology (calculation with Integer values)
NET2000	NET2000 protocol
OPERATOR	Operators according to IEC61131-3
PB_lib	Profibus protocol (FMS)
PowerLnk	Handling of the Powerlink interface board IF686.
PPdpr	Exchange of data between CPU and PP
RIO_lib	Operation of remote I/O
runtime	Internal support functions and function blocks
Spooler	Allows the spooling of data on IPs
SRAM200x	Functions for handling the SRAM200x
Default	IEC61131-3 standard functions and function blocks
SYS_lib	Various system functions
TCPIPMGR	Data exchange via UDP or TCP
VCScrsht	Saving a current image of the target system as a bitmap (*.bmp)
visapi	Programming interface for controlling the visualization while the system is running
VNCServ	Visualizations that run on SG4 targets and support VGA, can be viewed on the PC

9.2 Timers

Most control applications need time functions for controlling and monitoring system and machine processes.

This is subdivided into the following groups:

- **Timer Functions** (on/off delays, pulse generating).
- **Time Determination** (read out and set the real time clock).
- **Time Measurement** (read out 10 msec and µsec timer).

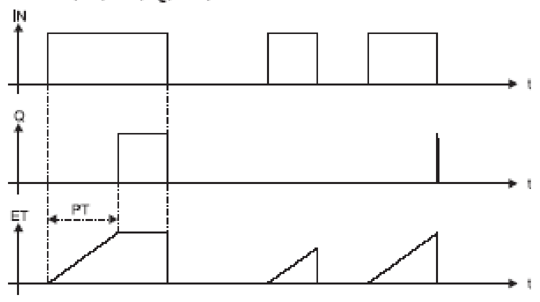
Timer Functions

Time function elements make it easier for you to control time procedures.
All three time function element function blocks have the same parameters:

Parameters:

IN .... Input information.
PT .... Preset time in 10 msec intervals.
Q .... Output information.
ET .... Elapsed time in 10 msec intervals.

Timing of TON(IN, PT, Q, ET).



Timing of TOF(IN, PT, Q, ET).

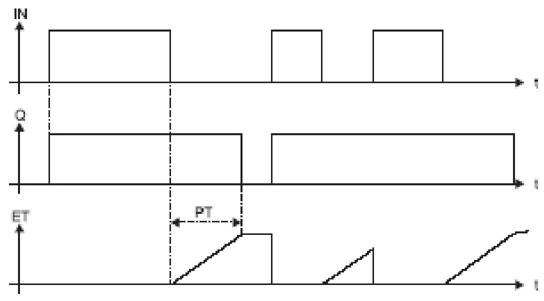


Fig.9.1: TON and TOF Timer.

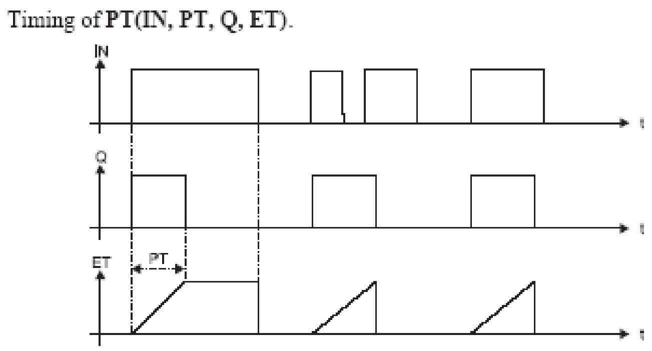


Fig.9.2: TP Timer.

9.3 Functions/Function Blocks

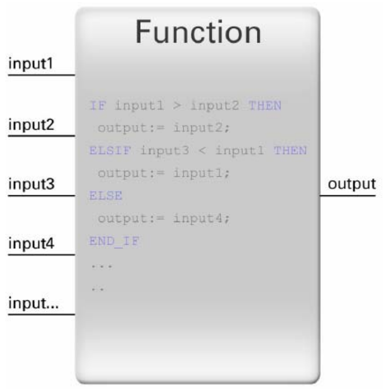
The program functionalities grouped in libraries are divided into two types:

- Functions.
- Function blocks.

These two types differ in behavior and how they are used.

9.3.1 Function

A function is a program organizational unit which returns exactly one value. Therefore, it has just one output, but can have any number of inputs. Unlike function blocks, functions do not have any static memory. With only a few exceptions (e.g. time and IO – read functions), it means that it always returns the same output value when called repeatedly with the same input parameters. [26]

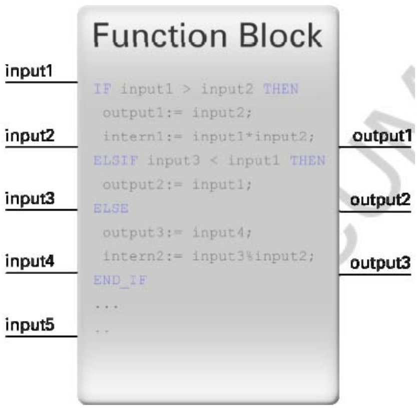


```
IF input1 > input2 THEN
 output:= input2;
ELSIF input3 < input1 THEN
 output:= input1;
ELSE
 output:= input4;
END_IF
...
..
```

Fig. 9.3: Function.

9.3.2 Function blocks

A function block is a program organizational unit which can return one or more values. Therefore, it can have one or more inputs and outputs. [26]



```
IF input1 > input2 THEN
 output1:= input2;
 intern1:= input1*input2;
ELSIF input3 < input1 THEN
 output2:= input1;
ELSE
 output3:= input4;
 intern2:= input3%input2;
END_IF
...
..
```

Fig. 9.4: Function Block.

An instance of a function block must be created before it can be used. This is essentially a data structure, which contains all of the parameters that the function block uses (i.e. inputs, outputs, and internal variables). By using a data structure, function blocks have a static memory. When called repeatedly with the same input parameters, the output values can also change. In some cases, function blocks, which require a great deal of system resources or access hardware, might have to be called repeatedly using multiple cycles. This makes it possible to wait for a response from the hardware and can reduce the load that the function block puts on the system.

C Functions

C functions cannot be created with the Library Manager; instead, they are created within a C task. Whether or not variables are preserved during a function call depends on the definition of the variables used (STATIC, defined outside the function, etc., see Comparison). Calling functions in a C function is allowed, but calling function blocks in C functions is not permitted.

Tab. 9.2: Comparison functions and function blocks.

	Functions	**Function Blocks**	**C Functions**
Creation is possible in the programming languages...	all	all	ANSI C
Usage is possible in the programming languages...	all	all	ANSI C
VAR_INPUT	Variables are transferred from the program to the function and are not changed in the function.	Variables are transferred from the program to the function block and are not changed in the function block.	-----
VAR_OUTPUT	Return value (one value) of the function. Variable is defined with the return value of the function.	Return value of the function block. Variables are defined by the function block.	-----

	Functions	**Function Blocks**	**C Functions**
VAR	Only exist within the function. Values are valid only within the function. These variables are located on the stack.	Only exist within the function block. Variables are valid outside of function block calls. There is a unique variable memory - function block instance per call	-----
VAR_DYNAMIC	Dynamic variable in the function which is assigned to a pointer. Can only be used in Automation Basic or ANSI C.	Dynamic variable in the function block which is assigned to a pointer. Can only be used in Automation Basic or ANSI C.	-----
VAR_INPUT_DYNAMIC	Dynamic input/output variable supplied by a pointer when a function is called. Can only be used in Automation Basic or ANSI C.	Dynamic input/output variable supplied by a pointer when a function block is called. Can only be used in Automation Basic or ANSI C.	-----
Local C variables	Can only be used with functions written in ANSI C (Library Manager). Values are not valid outside the call unless the variable is defined as STATIC. These variables are located on the stack.	Can only be used with function blocks written in ANSI C (Library Manager). Values are not valid outside the call unless the variable is defined as STATIC. These variables are located on the stack.	-----
Global variables	Can only be used with functions written in ANSI C (Library Manager). Values are valid outside of the call. (defining global C variables takes place outside the function)	Can only be used with function blocks written in ANSI C (Library Manager). Values are valid outside of the call. (defining global C variables takes place outside the function)	Values are valid outside of the call. (defining global C variables takes place outside the function)
Re-entrant	Yes	yes (each function block possesses its own instance)	Due to the integration of C functions in B&R systems, a C function

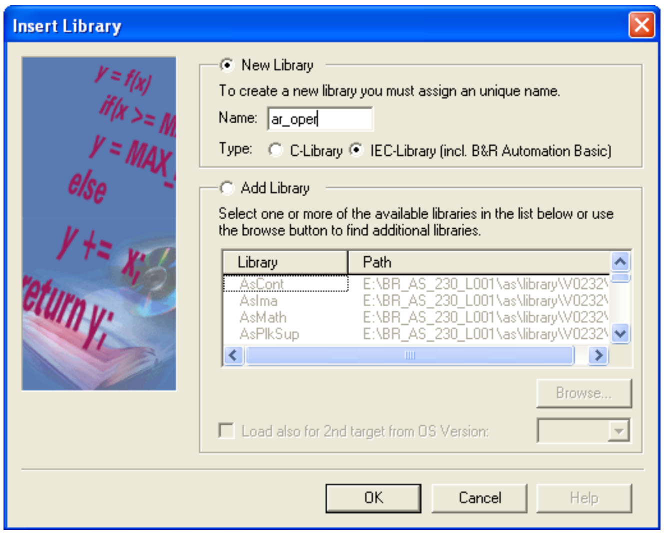
	Functions	**Function Blocks**	**C Functions**
			re-entrant call cannot occur.
Recursive call	Not possible	Not possible	possible
Calling/Using functions	possible	possible	possible
Calling/Using function blocks	Not possible	possible	Not possible
Calling/Using C functions	Possible (only in ANSI C)	Possible (only in ANSI C)	possible

<u>Example create a new function block – new library</u>

The same wizard that was used to integrate existing libraries is also used to create a new library. You must first begin by pressing the "Insert Library" button.

The library must be given a name with a maximum of 8 characters (e.g. *ar_oper*). The type determines the programming languages that will be possible for the implementation of the functions and function blocks. If *C-Library* is selected, then functions and function blocks must be programmed in ANSIC.

However, if an *IEC-Library* is created, then the programming languages Instruction List, Structured Text and Automation Basic are available.


Insert Library

New Library

To create a new library you must assign an unique name.

Name: ar_oper

Type: C-Library [✓ IEC-Library (incl. B&R Automation Basic)]

Add Library

Select one or more of the available libraries in the list below or use the browse button to find additional libraries.

Library	Path
AsCont	E:\BR_AS_230_L001\as\library\V0232\
AsIma	E:\BR_AS_230_L001\as\library\V0232\
AsMath	E:\BR_AS_230_L001\as\library\V0232\
AsPlkSup	E:\BR_AS_230_L001\as\library\V0232\

Browse...

Load also for 2nd target from OS Version:

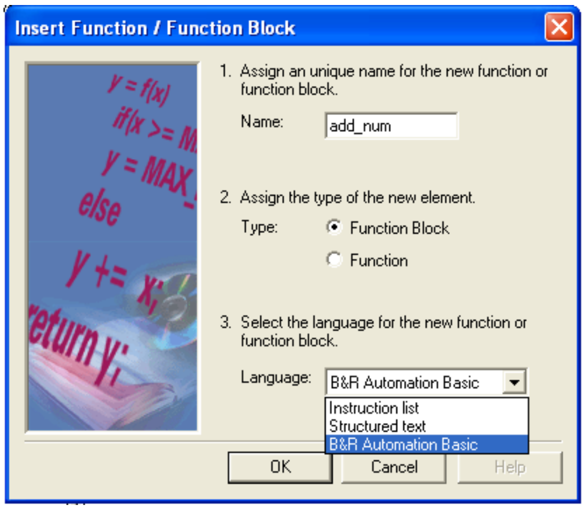
OK Cancel Help

Fig. 9.5: Creating a new library.

A function or a function block can be create using the *Insert Function / Function Block* button or by right-clicking on the library.

In the window *Insert Function/Function Block* you have to assign name for a new function/function blocks e.g.add_num. Select if you create a new function or function block (e.g. Function Blocks), and in popup menu select programming language for respective e.g. function block.

After confirming the selection with *OK*, the function block is created and you can go on to declare the interface and create the source code. At this point, the function block should e.g. add the two inputs and write the result to the output.



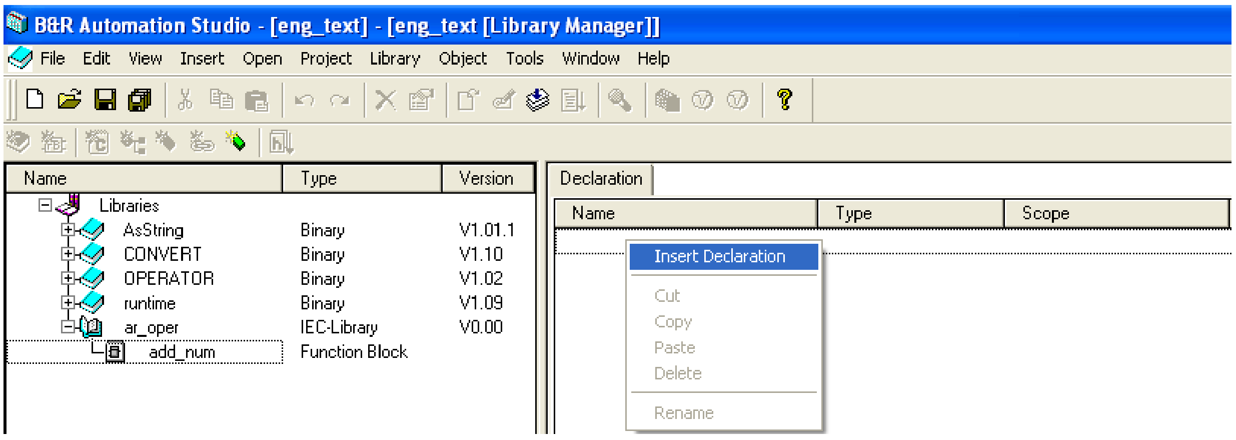
1. Assign an unique name for the new function or function block.
   Name: add_num

2. Assign the type of the new element.
   Type: [✓ Function Block] Function

3. Select the language for the new function or function block.
   Language: B&R Automation Basic
   - Instruction list
   - Structured text
   - B&R Automation Basic

Fig. 9.6: Wizard inserting a new function/function block.

The function block has e.g. two inputs variables and one output variable. With the right mouse click and the selection in the dialog window *Insert Declaration* (fig.9.7.) you insert a new variable.



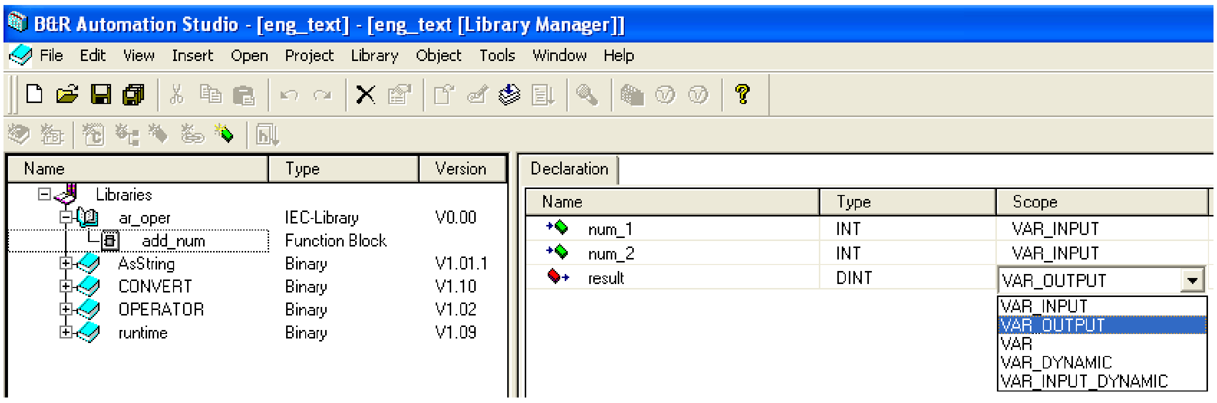
Name	Type	Version
Libraries		
AsString	Binary	V1.01.1
CONVERT	Binary	V1.10
OPERATOR	Binary	V1.02
runtime	Binary	V1.09
ar_oper	IEC-Library	V0.00
add_num	Function Block	

Name	Type	Scope

Fig. 9.7: Insert a new variables.

The names of the variables are written in the column *Name* e.g.num_1, num_2. The type of the variable is written in the column *Type* (see item data types). If the variable is as input or output it is written in the column *Scope*.

The solution could look like as shows figure 9.8


B&R Automation Studio - [eng_text] - [eng_text [Library Manager]]
File Edit View Insert Open Project Library Object Tools Window Help

Name	Type	Version
Libraries		
ar_oper	IEC-Library	V0.00
add_num	Function Block	
AsString	Binary	V1.01.1
CONVERT	Binary	V1.10
OPERATOR	Binary	V1.02
runtime	Binary	V1.09

Declaration

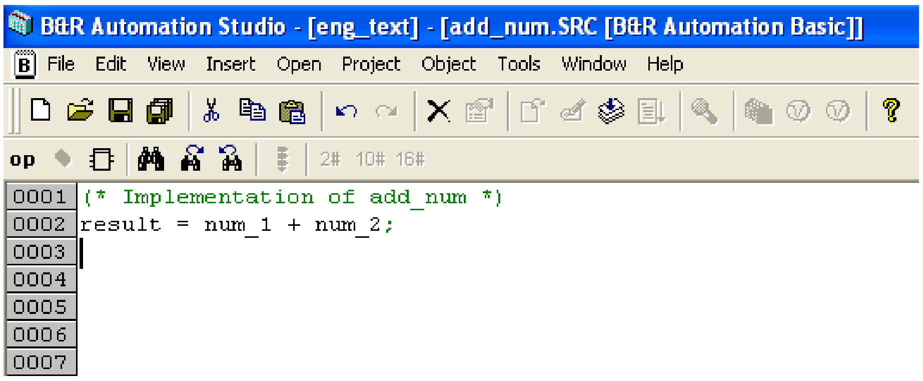
Name	Type	Scope
num_1	INT	VAR_INPUT
num_2	INT	VAR_INPUT
result	DINT	VAR_OUTPUT

VAR_INPUT
VAR_OUTPUT
VAR
VAR_DYNAMIC
VAR_INPUT_DYNAMIC

Fig. 9.8: Declaring of a variable.

The function block will add two values in inputs num_1, num_2. The source code editor is opened by double-clicking on the function block icon *add_num* (on the left side of the window Automation Studio) when the Automation Basic opens you have to write the arithmetic operation: *result = num_1 + num_2* (fig.9.9).

The source code for this function looks like this:


B&R Automation Studio - [eng_text] - [add_num.SRC [B&R Automation Basic]]
File Edit View Insert Open Project Object Tools Window Help
op
2# 10# 16#

```
(* Implementation of add_num *)
result = num_1 + num_2;
```

Fig. 9.9: Arithmetic operation.

Confirm the saving of this function block: File→Save.

The function block add_num can be used in the user task. When this function block insert into the task you have to assign the name of this function block. Thereby you create an instance function block add_num.

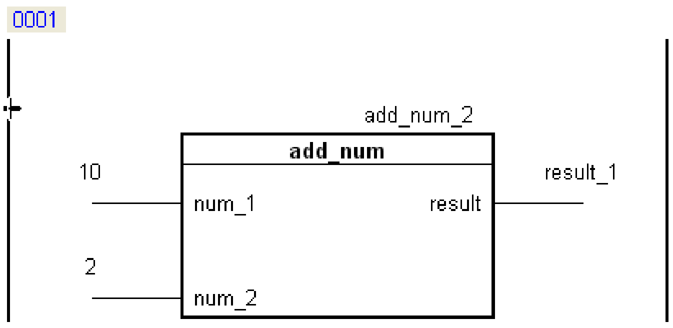


Fig. 9.10: The function block in a task.

9.4 Module CM 211

The module CM 211 contains digital and analog inputs and outputs. Digital inputs can be used with special function e.g. counter inputs, measure frequency. The states of the digital inputs are shown with status LEDs.

Fig. 9.11: Expansion module CM211.

Tab. 9.3: Digital and analog inputs and outputs.

DI/DO	8/8
AI/AO	2/2 ±10V,0-20mA 12bit
Special function	3 event counters, 3x period measurement, 3x gate measurement, 2x incremental encoder , 1x comparator

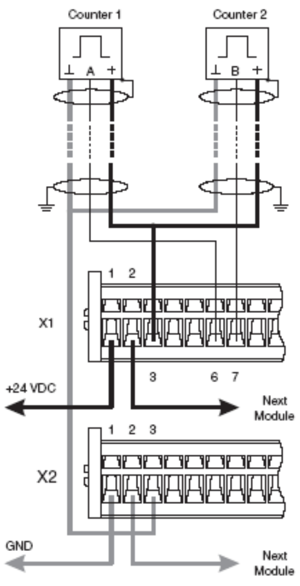


Fig. 9.12: Connection Example Event Counter Operation

Analog inputs and outputs

Analog inputs are used for connection analog value to PLC as signals from sensor with continuous outputs. The main groups are sensors as temperature, flow- rate, pressure, etc. The analog inputs often have from voltage or current range. The current ranges are immunity for disturbance, it is useful for industry applications or where it is the most possibility disturbance.

The most widely used range are 0-20mA and ±10V of B&R module

Tab. 9.4: Range of analog inputs.

Analog inputs/outputs	Range		Data type
Analog inputs 1/2	Voltage +10V	$7FFF	INT 16
	0V	$0000	
	Voltage -10V	$8001	
	Current 20mA	$7FFF	
	0mA	$0000	
Analog outputs 1/2	Voltage +10V	$7FFF	INT 16
	0V	$0000	
	Voltage -10V	$8001	

9.5 Communication

Fieldbus and IT networks are standards in automation today. B&R systems support the most common fieldbus systems and networks.

Today, extensive communication possibilities are a basic requirement for any automation solution. Ethernet is easily experiencing the strongest growth in this sector. CAN bus systems are also experiencing significant gains.

Flexible communication and networking are a fundamental part of B&R products. Most CPUs are already equipped with an integrated 10/100 Mbps Ethernet interface.

Communication must adapt to meet the requirements of the application. That is why there is a wide product range of interface modules. They can be used with all x86-based CPUs from the 2003 and 2005 systems as well as with the Power Panel 200 series. The modules are based on the B&R aPCI standard. Plug-in cards are available in the PCI format for B&R Industrial PCs.

9.5.1 Networks for Automation

For the most part, higher demands are placed on communication in the field than those in office communication. Real-time capability and deterministic behavior are of utmost importance in this respect. Jitter behavior in the micro second range and extremely high resistance to disturbance are important factors.

9.5.2 True Real-time for Standard Ethernet

ETHERNET Powerlink provides a standard protocol for Fast Ethernet, which has proven its tough real-time characteristics in thousands of applications. The ETHERNET Powerlink Standardization Group (EPSG) ensures openness and continuous advancement. Powerlink as

Standard Ethernet based system represents the second generation of fieldbus. This makes it possible to apply the full power of IT technologies to the automation field for the first time. ETHERNET Powerlink is comparably suited for drives, I/Os, visualization and data exchange between PLC systems.

9.5.3 <u>CAN Bus in Automation</u>

CAN bus has had much success, particularly in machine manufacturing, and is steadily gaining importance. High resistance to disturbance, high-speed data transfer, ease of use and deterministic real-time behavior are among the reasons for this success. CAN is the ideal fieldbus for applications with a manageable number of remote I/O nodes and few axes.

As a fieldbus, CAN bus reaches its limits when dealing with larger and more complex machines. For these applications however, ETHERNET Powerlink is the ideal expansion in the higher performance range.

9.5.4 <u>Decentralized Backplane</u>

Decentralization is the dominating trend in automation technology. This has come about as a result of economic considerations and the total cost reduction of machines, by those who see the clear advantages of a decentralized structure for multiple applications. These demands brought about the idea of running the conventional backplane for the I/O modules in a PLC system or a bus controller in one cable. The result is the extremely high-speed I/O connection, X2X Link.

9.5.5 <u>Serial Communication</u>

Now, just as ever, interfaces such as RS232, RS422 and RS485 play an important role in the automation world. Robust, simple and nevertheless efficient, these interfaces still find wide usage. The classic RS232 interface is fully capable of meeting the demands for system programming and maintenance.

9.5.6 <u>Real-Time and Standard Ethernet</u>

Regular Ethernet is not capable of handling data transfer in real-time. Additional measures like fully switched Ethernet and frame prioritization are not suitable either. Firstly, it does not fit the flexibility needs of automation network topologies. Secondly, deterministic data transfer and timing precision still cannot be guaranteed. And thirdly, it is quite complex to configure network utilization by selecting appropriate node and frame priorities.

Multiple industry groups have therefore introduced various new mechanisms, like using non-CSMA/CD access mechanisms on Ethernet physics, introducing special time-based switching mechanisms, bit-stream decoding with ASICs or shortening Ethernet frames to lower transfer times. All these approaches have major drawbacks, either violating established worldwide standards, and/or forcing implementers to depend on proprietary ASICs.

Ethernet as an open standard demands openness when it comes to real-time data communication. Therefore, from the very beginning ETHERNET Powerlink always took the standard conform approach, extending IEEE 802.3 Ethernet with a mixed polling and time slicing mechanism to:

- Guarantee transfer of time-critical data within very short and precise isochronous cycles with configurable timing.
- Synchronize networked nodes with high precision in the microsecond range.
- Transmit less time-critical data in a reserved asynchronous channel.

Current implementations have reached 200µs cycle time with a timing deviation (jitter) below 1µs. Until now, this was only possible with dedicated motion bus systems.

Due to its standard compliancy it is possible to leverage and continue using any standard Ethernet silicon, infrastructure component or test and measurement equipment with ETHERNET Powerlink. All IP-based protocols on higher layers, like TCP, UDP and above, can be further used without modifications. In particular, ETHERNET Powerlink conforms to the following international standards:

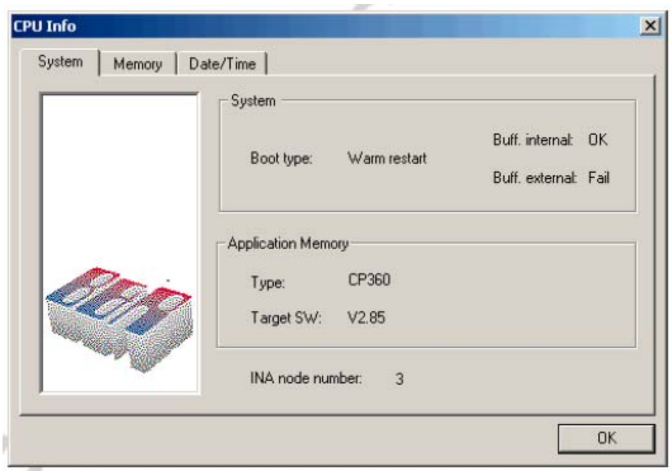
- IEEE 802.3 Fast Ethernet (incl. frame formats) .
- IP-based protocols (TCP, UDP, etc). Standard device profiles according to CANopen EN50325-4.
- Any Ethernet chip and hardware for implementation and operation - no ASICs necessary.
- IEEE 1588 for distributed real-time domain synchronization in one of the next extension.

ETHERNET Powerlink distinguishes between real-time domains and non real-time domains. This separation matches typical machine and plant concepts. It also satisfies the increasing security demands to prevent hacker attacks at the machine level or harm through erroneous data communication on higher network hierarchies. True real-time requirements are met within the real-time domain. Less time critical data is routed transparently between the real-time domain and non-real-time domain using standard IP frames. A clear boundary between a machine and factory network prevents potential security flaws from the very beginning while keeping full data transparency.

9.6 Program debugging and monitoring

The Online Info is an overview window that displays information about the target system. This includes information about available memory, battery status, node numbers on the CAN bus or Ethernet interface, and clock settings on the target system.

Online Info is opened by selecting *Online info...* from the CPU's shortcut menu.


CPU Info

System | Memory | Date/Time

System

Boot type:	Warm restart	Buff. internal:	OK
		Buff. external:	Fail

Application Memory

Type:	CP360
Target SW:	V2.85

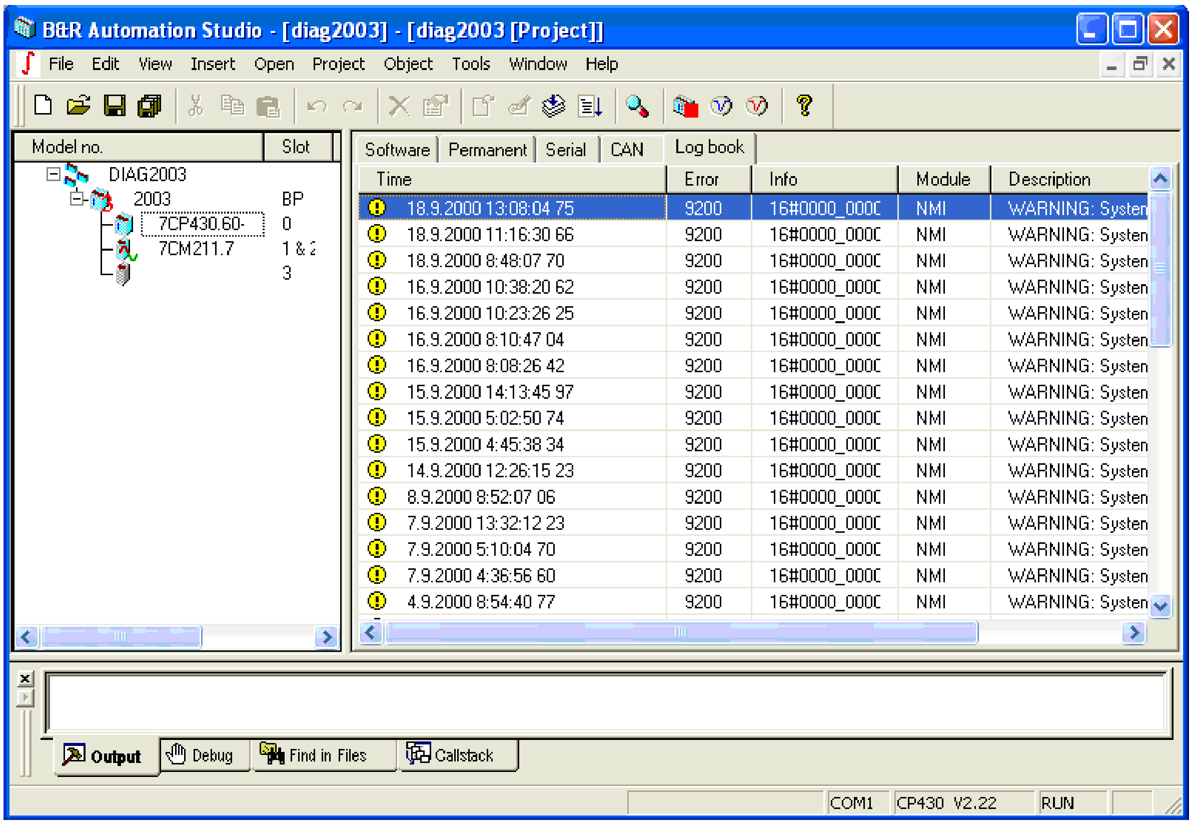
INA node number: 3

OK

Fig. 9.13: CPU information.

9.6.1 Log book

All serious errors that occur in the context of an application (software errors in the target system such as cycle time violations, installation errors, etc.) are logged by the operating system. This log is stored in its own error log book and can be viewed in the project window. To do this, choose a module in the hardware configuration (e.g. CPU) that provides a log book and click on the Log Book tab list in the right-hand section of the project window:


B&R Automation Studio - [diag2003] - [diag2003 [Project]]

Model no.	Slot
DIAG2003	
2003	BP
7CP430.60-	0
7CM211.7	1 & 2
	3

Software | Permanent | Serial | CAN | Log book

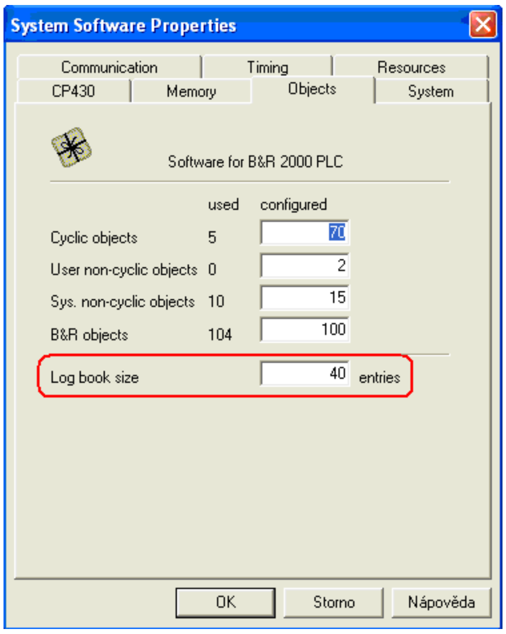
Time	Error	Info	Module	Description
18.9.2000 13:08:04 75	9200	16#0000_000C	NMI	WARNING: Systen
18.9.2000 11:16:30 66	9200	16#0000_000C	NMI	WARNING: Systen
18.9.2000 8:48:07 70	9200	16#0000_000C	NMI	WARNING: Systen
16.9.2000 10:38:20 62	9200	16#0000_000C	NMI	WARNING: Systen
16.9.2000 10:23:26 25	9200	16#0000_000C	NMI	WARNING: Systen
16.9.2000 8:10:47 04	9200	16#0000_000C	NMI	WARNING: Systen
16.9.2000 8:08:26 42	9200	16#0000_000C	NMI	WARNING: Systen
15.9.2000 14:13:45 97	9200	16#0000_000C	NMI	WARNING: Systen
15.9.2000 5:02:50 74	9200	16#0000_000C	NMI	WARNING: Systen
15.9.2000 4:45:38 34	9200	16#0000_000C	NMI	WARNING: Systen
14.9.2000 12:26:15 23	9200	16#0000_000C	NMI	WARNING: Systen
8.9.2000 8:52:07 06	9200	16#0000_000C	NMI	WARNING: Systen
7.9.2000 13:32:12 23	9200	16#0000_000C	NMI	WARNING: Systen
7.9.2000 5:10:04 70	9200	16#0000_000C	NMI	WARNING: Systen
7.9.2000 4:36:56 60	9200	16#0000_000C	NMI	WARNING: Systen
4.9.2000 8:54:40 77	9200	16#0000_000C	NMI	WARNING: Systen

COM1 | CP430 V2.22 | RUN

Fig. 9.14: Log book.

If the full text is not shown, the entire text can be displayed in a small pop-up window. To do this, move the mouse cursor over the required text and then hold it still. The complete text will appear after a short time:

The number of log book entries can be set in the *System Software* properties (*Software Objects* tab list). In other words, if a value of 20 is set, the last 20 errors are logged in the error log book. Older entries are overwritten.



System Software Properties

Communication | Timing | Resources
CP430 | Memory | Objects | System

Software for B&R 2000 PLC

	used	configured	
Cyclic objects	5	70	
User non-cyclic objects	0	2	
Sys. non-cyclic objects	10	15	
B&R objects	104	100	
Log book size		40	entries

Fig. 9.15: Set the size of the LogBook.

9.6.2 Logger

All fatal errors, warnings, and information which occur within the framework of the application are recorded by the operating system. This log is stored in a separate system logbook in the controller's memory. It can be displayed using the logger and saved as a file as long as there is an online connection. The logger can be opened via the main menu *Open→Logger*.

9.6.3 Monitors

The monitors can be used to get information about the target system. The system monitor shows the version and status of the task, the I/O monitor provides information about the status of the data points and the monitor for programming languages offers extensive possibilities for checking the program flow. All monitors are started with the icon .

In graphical programming languages, variable values are displayed right next to their symbols. There are also optical aids that show the program flow in a very simple manner.

In the image above, the signal flow is shown by coloring both the lines and the symbols. In addition to the values in the program, this provides another way to carry out diagnostics.

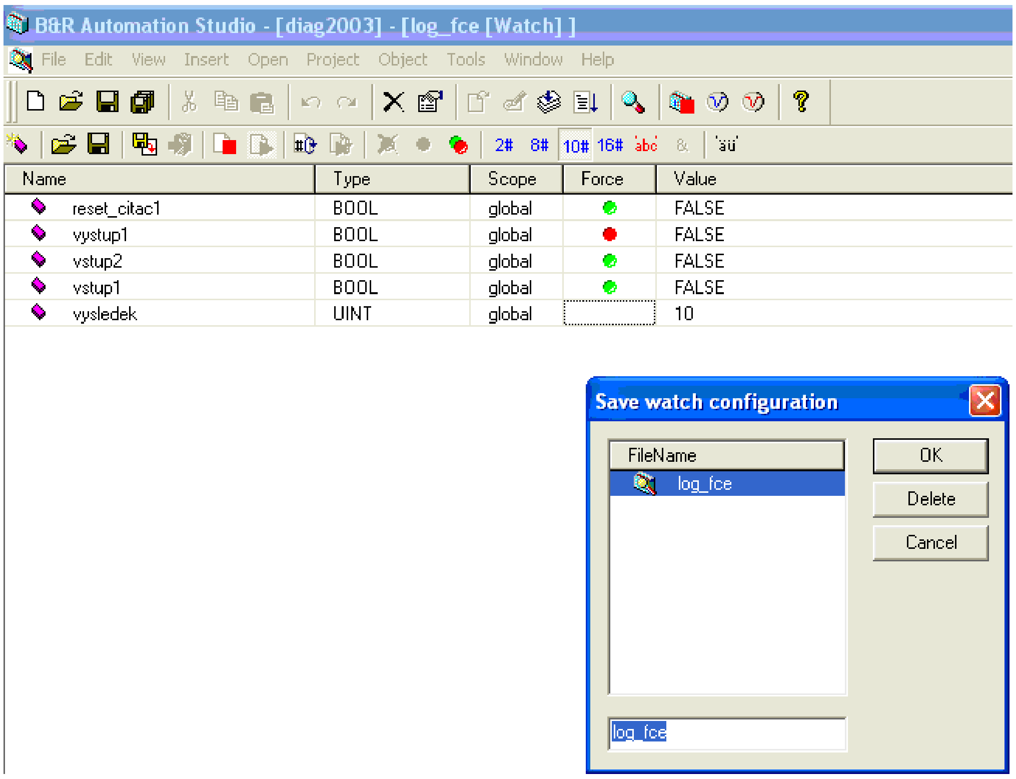
The signal flow display can be turned on with the icon.

9.6.4 Watch

All functions in the Watch window relate to the task that was selected in the software tree when it is opened. In other words, only variables from this task can be selected. If the Watch window is opened when the CPU is highlighted in the software tree, then all global variables can be displayed. This window is opened by clicking on the Watch item in the shortcut menu.

The primary purpose of the Watch window is to view and modify controller variables. In addition to the values of variable, other important information can be displayed about them (data type, scope, I/O data point, etc.). Variables can be managed in the form of a list, and different configurations (groups of variables) can be stored.

The Watch window displays I/O variables with red (outputs) and green (inputs) LEDs. When attempting to change the value of such an I/O variable, the force function can be enabled by confirming the message that appears. When forcing is turned on, the value entered into the Watch window is written to the software or hardware regardless of the state of the I/O data point.

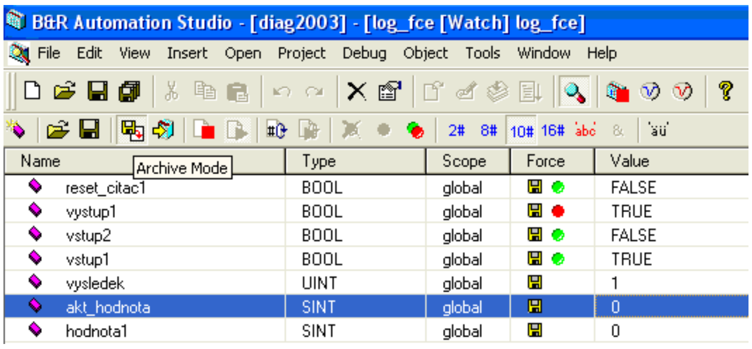

B&R Automation Studio - [diag2003] - [log_fce [Watch]]
File Edit View Insert Open Project Object Tools Window Help
2# 8# 10# 16# 'abc' & 'äü'
Save watch configuration
FileName
log_fce
OK
Delete
Cancel
log_fce

Name	Type	Scope	Force	Value
reset_citac1	BOOL	global	🟢	FALSE
vystup1	BOOL	global	🔴	FALSE
vstup2	BOOL	global	🟢	FALSE
vstup1	BOOL	global	🟢	FALSE
vysledek	UINT	global		10

Fig. 9.16: Watch and save watch configuration.

It is no problem changing ONE variable in the Watch window. The new value is transferred to the target system as soon as the entry has been confirmed.

However, archive mode can also be used to transfer more than one variable at the same time. The variable values are no longer refreshed when archive mode is activated. This allows you to easily change the values of multiple variables and to transfer them to the target system with a single command (write values). This also allows you to put together small recipes (collections of values that are related to one another) that can be loaded and saved at any time. When a Watch configuration is saved, the values are saved as well. This data is called archive data.


B&R Automation Studio - [diag2003] - [log_fce [Watch] log_fce]

File Edit View Insert Open Project Debug Object Tools Window Help

Archive Mode

Name	Type	Scope	Force	Value
reset_citac1	BOOL	global	💾 🟢	FALSE
vystup1	BOOL	global	💾 🔴	TRUE
vstup2	BOOL	global	💾 🟢	FALSE
vstup1	BOOL	global	💾 🟢	TRUE
vysledek	UINT	global	💾	1
akt_hodnota	SINT	global	💾	0
hodnota1	SINT	global	💾	0

Fig. 9.17: Archive data.

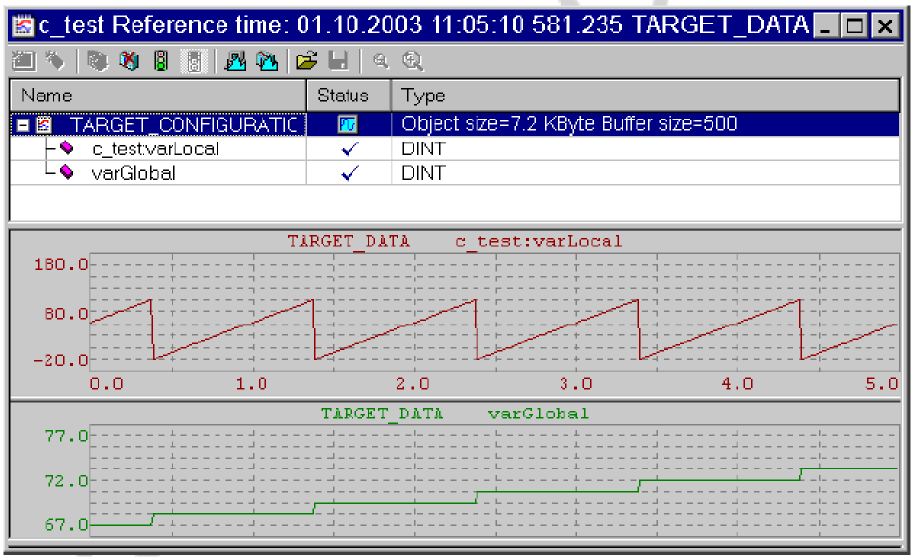
9.6.5 Trace

Until now, the different monitors, line coverage, and the Watch Windows were sufficient for viewing both static and ever-changing variables. However, there are some values that change rapidly and cannot be followed in the Watch window. The Tracer is able to record how values change over time and display them in a diagram. The diagrams which result can be stored to be viewed or processed later. The Trace function records the values directly on the controller. An upload function takes the values from the controller and displays them as a diagram. This function makes it possible to record values from different task classes in a precisely defined timeframe.

The smallest unit of time is the task class cycle time in the task for which the trace is opened. The values can be sampled at the start or end of a task. A maximum of 8 values can be recorded at one time. One trace can be opened for each task.

The trace can be configured using the shortcut menu from TARGET_CONFIGURATION: Properties. The maximum trace duration depends on the buffer size and the time base (task class cycle time and scaling). Furthermore, you can also define whether the trace should record continuously or should be started by a trigger condition. Once the variables have been selected and the Trace has been configured, it must be installed on the controller using the icon .

The Tracer can be started with the icon and stopped with the icon . When the Tracer is stopped, the icon can be used to load the data from the controller.


c_test Reference time: 01.10.2003 11:05:10 581.235 TARGET_DATA

Name	Status	Type
TARGET_CONFIGURATIC		Object size=7.2 KByte Buffer size=500
c_test:varLocal	✓	DINT
varGlobal	✓	DINT

Fig. 9.18: Trace.

9.6.6 Operating system

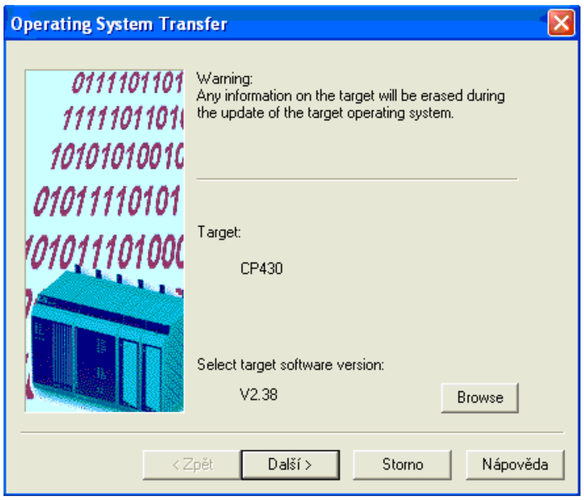
Every computer requires an operating system before it can be used. This provides the user with functionalities that make it easier to use the hardware. The more functions that the operating system handles, the faster and easier it is for the user to create software. Functions that are handled by the operating system do not have to be programmed by the user. This shortens the amount of time needed to create a program and reduces the number of errors.

An operating system essentially has two main tasks. On the one hand, it manages a system's hardware and software resources. On the other hand, it provides the user and his application with a stable and uniform way of accessing hardware without having to know the details.

There are different types of operating systems, which are differentiated according to their characteristics and how they work. A real-time multitasking operating system runs on B&R Automation Targets. This operating system is also called Automation Runtime (AR).

All program modules, also called tasks, are assigned to different task classes. These task classes determine the time frame in which the tasks will be processed. The time frame is always the same and is separate from the execution time of the individual tasks. This is the meaning of "real-time". The operating system is thus responsible for timing stability on the controller system.

The Record or the update operating systems to target is shown below on the picture 9.19-9.21


Operating System Transfer

Warning:
Any information on the target will be erased during the update of the target operating system.

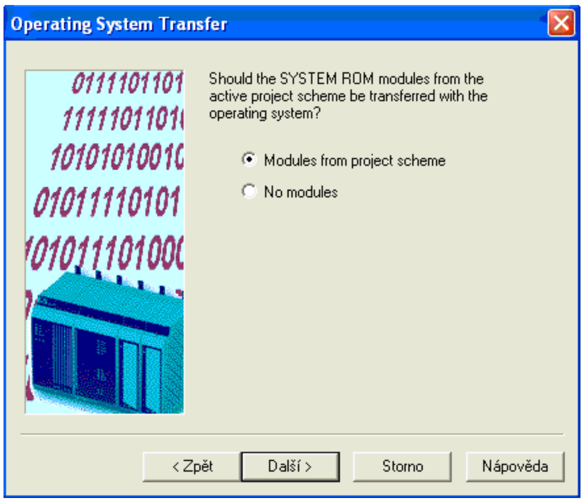
Target:

CP430

Select target software version:

V2.38

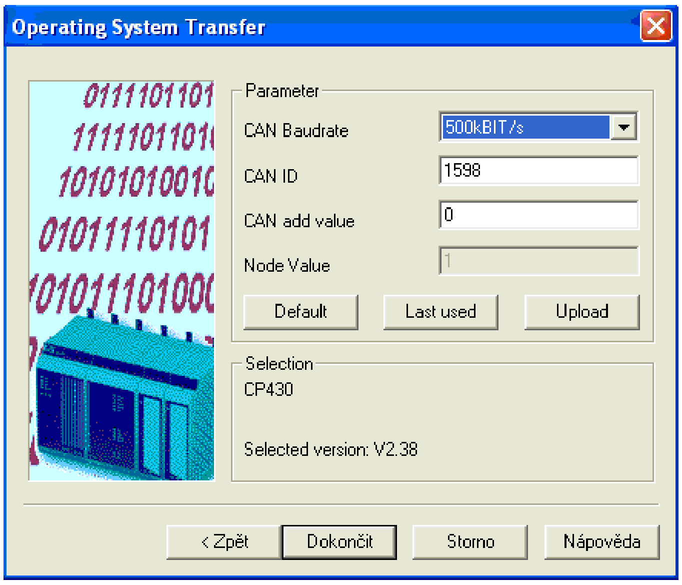
Fig. 9.19: Select target software version.


Operating System Transfer

Should the SYSTEM ROM modules from the active project scheme be transferred with the operating system?

[✓ Modules from project scheme]
No modules

Fig. 9.20: Types of PLC.


Operating System Transfer

Parameter

CAN Baudrate	500kBIT/s
CAN ID	1598
CAN add value	0
Node Value	1

Selection
CP430

Selected version: V2.38

Fig. 9.21: Select Baudrate.

Questions:

1. *What is the difference between the function and the function block?*
2. *What is the LogBook?*
3. *What is the Monitor?*
4. *When do you use monitor in a task can you change value of variable in program or no?*
5. *Can we re-load operation system in the PLC?*