8 PROGRAMMABLE CONTROLLERS BERNECKER&RAINER, BASIC PROGRAMMING.

Study duration: 90 min.

8.1 Operating system – deterministic multitasking and its description, task classes, task, cyclic program

Multitasking

On multitasking systems, the **processor performance** is **divided over several tasks**. These tasks are worked on simultaneously. [21, 26]

Division of the tasks provides the following advantages:

- Structure the application.
- Task is created in the best suitable programming language.
- Program individual functions and test them (modularity).
- Simple maintenance of the individual tasks.

A multitasking system meets the following requirements for PCC applications:

- Parallel processing of multiple control tasks.
- Deterministic Multitasking (with accurate time predictions).
- Different and adjustable cycle times with monitoring.
- Identical (consistent) I/O image for every task class.

B&R System make use of the deterministic multitasking, that means each task class is provided in periodical time intervals for example task class Cyclic#1 (figure 8.1) This task class is called in periodical 10 milisecond intervals. The task class Cyclic#1 contains two tasks – l_logic1 and l_logic2. The project doesn´t have to contain only one task class but it can contain several tasks and task classes. Below is the describtion of individual task classes.

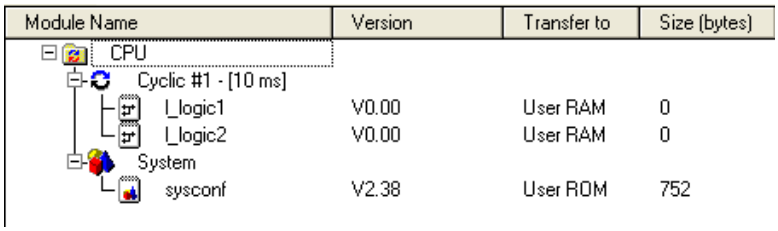

Fig. 8.1: Task class Cyclic#1 and its tasks.

When the task classes start, first is executed the process image. All task of the task class Cyclic#1 have to execute during 10 miliseconds. First the image inputs and outputs for task l_logic1 and l_logic2 are read. After the first task – l_logic1 is executed. When the task l_logic1 finishes, the task l_logic2 is executed. When the task 2 - l_logic2 finishes, are the image outputs written.

For example the task l_logic1 was executed during 3 miliseconds and task l_logic2 was executed during 5 miliseconds so there are still left 2 miliseconds (10 -5-3=2miliseconds). During 2 miliseconds can be executed the task with the lowest priority for example the task which executes the communication between remote I/O. The whole cycle show in figure 8.2.

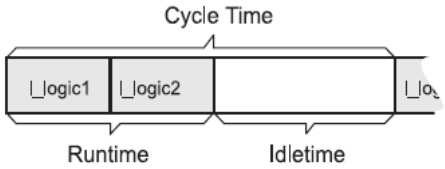

Fig. 8.2: Cycle Time.

I/O Image

For example these are two task classes TC#1 and TC#2 as figure 8.3. shows. The image inputs/outputs are read separatelly for each task classes TC#1 and TC#2. First are the inputs/outputs read for the task class TC#1, after the task class TC#1 is executed. When the task class TC#1 is finished, the task scheduler writes the process image of all outputs. In the next 10 miliseconds interval are the inputs/outputs for task classes TC#1 and TC#2 read again. After the task class TC#1 is finished , the task scheduler write outputs . Because the cycle time doesn´t expire the task class TC#2 this will be executed also. After the task class TC#2 is finished the task scheduler writes outputs. The task class TC#2 is only executed in 50 miliseconds interval as figure 8.3. shows.

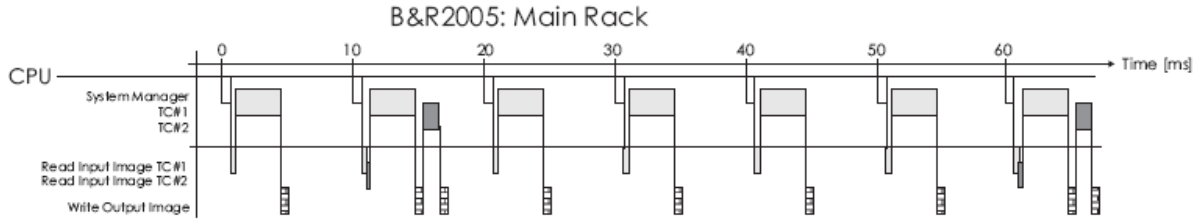

Fig. 8.3: I/O Image.

If the task class isn´t executed in its cycle interval, this event is written into the Logbook and the task class gets another time interval. If the time interval is stepped over the error is written into the Logbook and PLC is in stop mode.

Task classes

Tasks can be assigned to various task classes to define priority between the individual cyclic tasks. Task classes can be configured with their own cycle times.
All tasks in the task class are executed once within this cycle time. Important and time critical tasks are put into a task class with a short cycle time and are executed more often than tasks in a task class with a longer cycle time.

The user has two different kinds of task classes available, normal task classes and high speed task classes (not supported on some target systems). Normal task classes are activated by the system manager (operating system); high speed task classes are activated by interrupts.

- **Cyclic tasks** are the usual PLC applications and are processed exactly once within a defined time (= cycle time). The Automation Runtime makes sure that the cycle time is observed. Cycle time transgressions trigger exceptions. By default, a system emergency stop is then carried out, i.e. the controller boots in ERROR mode and then enters SERVICE mode. However, it is also possible to handle exceptions in an application specific manner using exception To assign priorities, tasks can be put into different task classes (different cycle times). Important and time critical tasks are put into a task class with a short cycle time and are executed more often than tasks in a task class with a longer cycle time.

Tab.8.1: Task classes and its cycle time.

Resource	Cycle Time	Tolerance
Timer #1	3ms	-----
Cyclic #1	10 ms	20 ms
Cyclic #2	50 ms	50 ms
Cyclic #3	100 ms	100 ms
Cyclic #4	10 ms	30 000 ms

- **EXC- Exception task classes** Exceptions are fatal errors which occur while the Automation Runtime is running and cannot be corrected by the operating system alone. The exception task class has the highest priority in the system. When an exception occurs, the accompanying exception task interrupts both normal and high speed tasks, as well as interrupt tasks.

For example:

- The exception occurs while Task1 is running. Task1 is interrupted because the EXC task class has the highest priority.
- The exception task assigned to the exception is executed.
- Processing of the cyclic tasks is resumed from the point at which it was interrupted.

The EXC task class shares its task class global data area with Cyclic#1. This results in faster access times and the possibility (without PLC-global variables) to exchange information between the Cyclic#1 and the EXC task class.

Tab. 8.2: Exceptions.

Exception number	Exception description
2	Bus error
3	Address error
4	Illegal instruction
5	Division by zero
6	Range overflow

Exception number	Exception description
7	Null pointer
8	Privilege violation
10	Unimplemented instruction
24	Spurious interrupt
128	I/O Exception
144	TC cycle time violation
145	TC maximum cycle time violation
146	TC Input - cycle time violation
147	TC Output - cycle time violation
160	HSTC maximum cycle time violation
161	HSTC I/O - cycle time violation
162	System cycle time violation
170	CAN I/O exception
176	RIO HSTC - cycle time violation
177	RIO - Exception when Ready/Error

Example creating of exception task class:

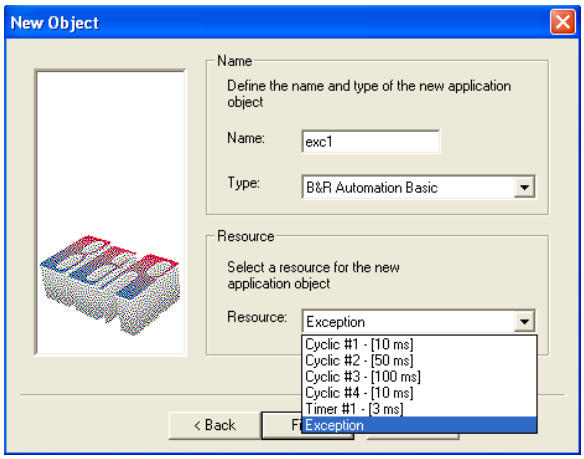

Fig. 8.6: Select of task class.

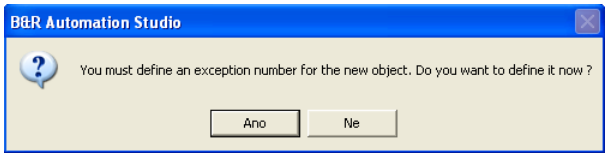

Fig.8.7: Types of exception error

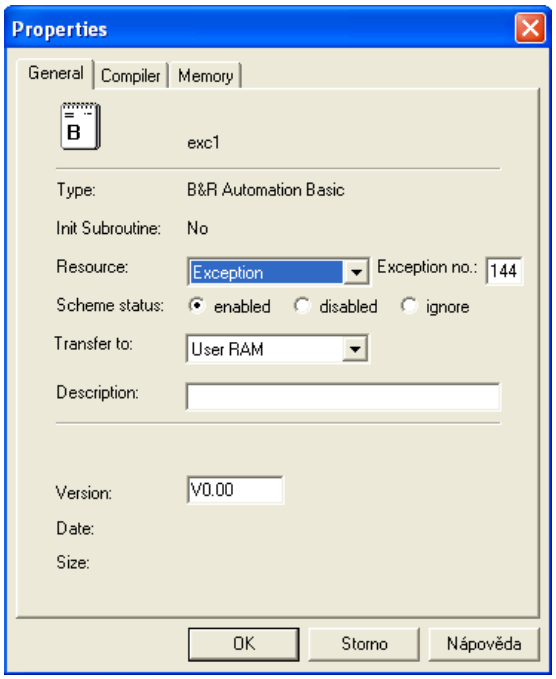

Fig. 8.8: Setting number of exception.

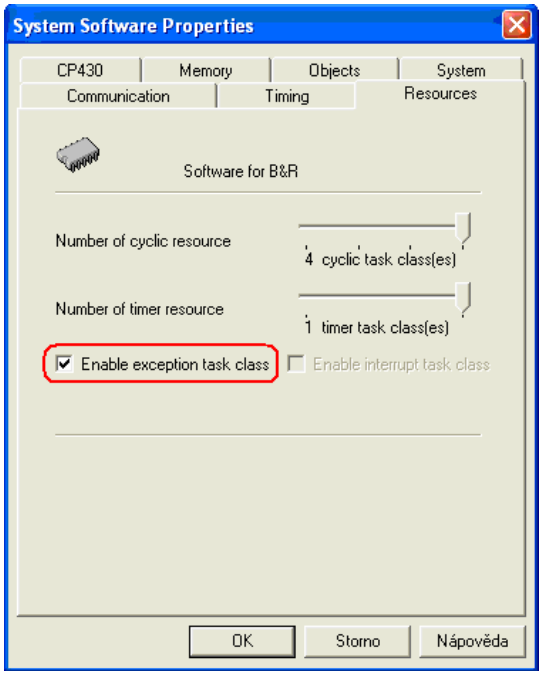

Fig.8.9: Enable execution exception task class.

- IRQ - INTERRUPT TASK CLASSES Interrupts are asynchronous events (triggered by hardware) that interrupt cyclic program execution. The interrupt task class has a higher priority than all normal task classes; however, it has a lower priority than the exception task class. This means that the interrupt task class can interrupt all normal tasks, but can also be interrupted by the exception task. Each interrupt capable module can be assigned an **interrupt task** (**IRQ task** for short) which is processed each time the corresponding interrupt is triggered. However, this additional processing of interrupts is only possible with processors from B&R 2010 systems (interrupt handling is not possible with CPUs and multi/parallel processors from B&R 2005 and B&R 2003 systems).
- HIGH SPEED TASK CLASSES High-speed task classes are called using separate interrupts, not by the operating system. This results in the following differences between normal task classes:
    - Timer HSCTC By default, HS task classes are called using the hardware timer (interrupts). These HS task classes are called 'timer HS task classes' (Timer HSTC for short). Cycle times for Timer HSTCs can be configured using B&R Automation Studio. All tasks assigned to the Timer HSTC are executed exactly once within this cycle time. The cycle time has nothing to do with the actual runtime of the tasks on the PLC.
    - RIO HSTC In addition to Timer HSTCs, separate remote I/O HS task classes (RIO HSTC for short) can be configured for remote I/O systems. RIO HSTCs are triggered by an interrupt generated by the Remote Master at the end of each RIO cycle. They are especially suited for fast access to current remote I/O data. The cycle time for the RIO HSTC can be configured using B&R Automation Studio. The set cycle time is the minimum cycle time. In other words, if the RIO cycle lasts for 1 ms, and 3 ms was specified for the cycle time, then the Remote Master activates the RIO HSTC only every 3 ms even though the RIO cycle is already completed after 1 ms. This prevents a very fast RIO cycle from

overloading the system. However, if the RIO cycle lasts longer than the set cycle time, then the RIO HSTC accordingly starts later.

- MP HSTC In addition to Timer HSTCs, multi/parallel processors can configure separate multiprocessor HS task classes (MP HSTC for short). The MP HSTC is triggered by the MP_trigger() function in the main CPU application instead of by a hardware timer. This means that the cycle time depends completely on the cycle time of the task which called the MP_trigger() function. This function is not supported by B&R Automation Studio at the moment.

Tab 8.3: Available task classes for B&R target.

Processor type		**Task class available?**			
		Task class Cyclic#1	**Task class Cyclic#2**	**Task class Cyclic#3**	**Task class Cyclic#4**
B&R 2010	CPU	YES	YES	YES	YES
	MP/PP	Not supported yet			
B&R 2005	CPU	YES	Only RIO HSTC	NO	NO
	MP/PP	Not supported yet			
B&R 2003	CPU	YES	NO	NO	NO
	MP/PP	Not supported yet			

Task

A task is an independent part of the program representing many processes (digital and analog connectins, control, positioning, evaluation of measurements, etc.). Each task class can be written in alternative programming language.

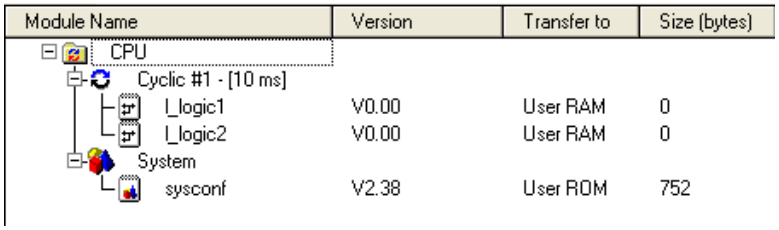

Fig 8.10: Example task l_logic1 and l_logic2.

8.2 Programming languages, data types

Programming is carried out using B&R Automation Studio™. Several programming languages are available:

- Automation Basic (previously PL2000).
- ANSI C.
- IEC 1131 Ladder Diagram (LAD).

- IEC 1131 Sequential Function Chart (SFC).
- IEC 1131 Structured Text (ST).
- IEC 1131 Instruction List (IL).

B&R Automation Basic create Bernecker Rainer for their programmable logic controllers.

Data types

Nowadays, programming is done using symbolic elements with names instead of using fixed memory addresses. These elements are called variables. Simple data types determine the size of the memory, which the variables occupy (value range) and how the contained value is interpreted (with/without sign or decimal point, ASCII text, date/time). An array is a collection of several variables of the same data type, which are addressed using a fixed name and an index. Structures are variables with user-specific data types that the user can create.

In programming, there are simple data types also known as basic data types. All available basic data types in accordance with IEC 61131-3 are displayed and categorized in the following list according to their possible areas of use.

Tab. 8.4a: Data types.

Type	Size	Range	Použití
BOOL	1	0..1	e.g.digital I/O
DINT	32	-2 147 483 648.. 2 147 483 647	
INT	16	-32 768 .. 32767	e.g.analog I/O
SINT	8	-128..127	
UDINT	32	0.. 4 294 967 295	
UINT	16	0..65 535	
USINT	8	0..255	
REAL	32	-3,4e38.. 3,4e38	

Tab. 8.4b: Data types.

Type	Size	Range	Použití
STRING	1-xx byte	2 znaků – 32767 znaků	“Text”
TIME	4 bytes	0..4 294 967 295 ms	Time differences
DATE_AND_TIME	4 bytes	Second since 1970	Date calculations

8.3 B&R Automation Studio

B&R Automation Software™ combines all the software packages necessary for configuring, programming, diagnosing, and operating all B&R Control Systems™, B&R Motion Systems™ und B&R Panel Systems™ components.

B&R Automation Studio™ is the integrated software development environment which includes tools for all parts of an automation project, making it the foundation for applications of any size and scope. Regardless of which stage a project is in – planning, implementation, testing, production, commissioning, or service – this same environment always makes up the interface to the machine.

Advantages of Automation Studio™:

- Cost-saving since it is suitable for all types of applications.
- Seamless integration and easier communication since everybody uses the same tool.
- Increased productivity since the same environment is used for all parts of a project.
- Reduced maintenance since only one software environment is installed.
- A single provider for the entire hardware and software platform.
- A completely integrated environment with total compatibility between the various tools.
- More capacity to concentrate on your primary core competence since no extra effort is needed for integrating tools.
- Extremely easy for new employees to use.
- Improved motivation with an integrated environment instead of the irritation caused by an uncontrollable environment.

AS (Automation Studio) is hardware oriented programming system. The objects are used to organize a project (system / machine). All program objects on a system / machine are assigned to a project using structured organizational support.

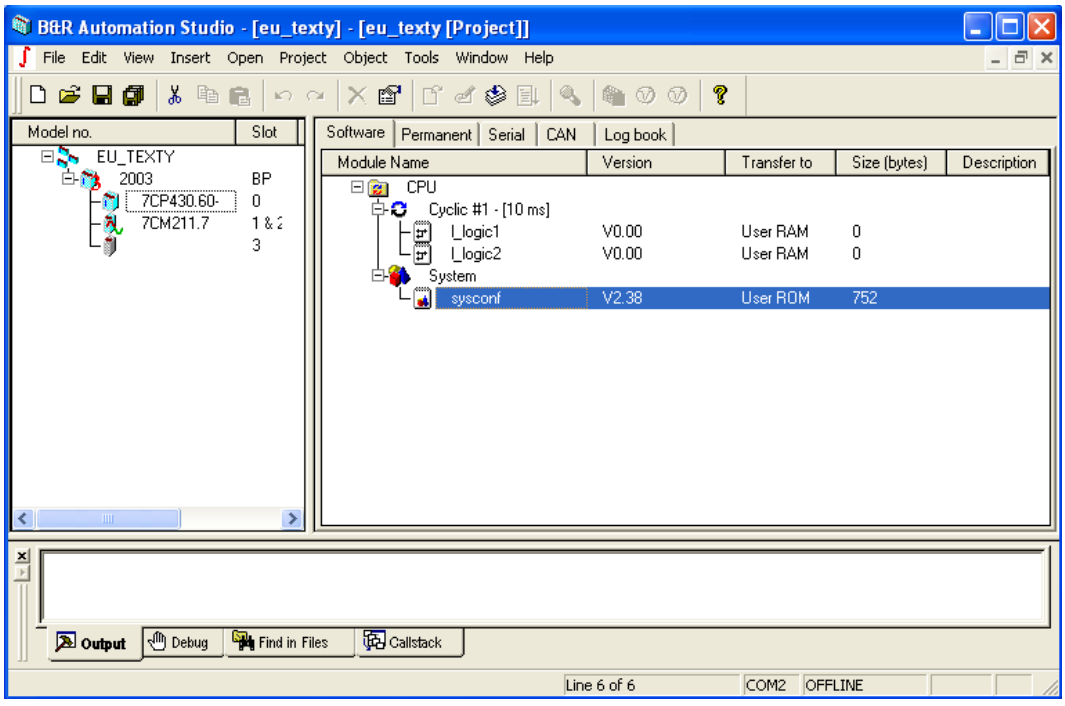

Fig. 8.11: B&R Automation Studio.

An integral component of Automation Studio™ is Automation Runtime, the software kernel which allows applications to run on a target system. This runtime environment offers numerous important advantages:

- Guaranteed highest possible performance for the hardware being used.
- Runs on all B&R target systems.
- Makes the application hardware-independent.
- Applications can be easily ported between B&R target systems.
- Cyclic system guarantees deterministic behavior.
- Configurable jitter tolerance in all task classes.
- Supports all relevant programming language such as IEC 61131-3 and C.
- Extensive function library conforming to IEC 61131-3 as well as the expanded B&R Automation library.
- Integrated into Automation NET. Access to all networks and bus systems via function calls or from the Automation Studio™ configuration.

Automation NET/PVI

Transparent Communication with Automation NET

Automation Studio™ provides nearly unlimited possibilities for communicating with automation components. Connections via RS232, the CAN bus, or Ethernet always use the same services from Automation NET.

One Interface for Many Networks

Automation NET uses the same interface to connect to any network, making it easy to migrate existing applications to new bus systems or networks.

Transparent Networking

Because of the ability to route network connections over any network nodes, it is easy to reach all controllers in an existing system using the programming and analysis tool. Automation NET allows the user to access any controller, even if it is not directly connected to the PC used for programming. For the application, the connection is completely transparent.

With Automation NET, Automation Studio™ integrates technology for the transparent networking of control systems.

Example of create a new project

Before we begin the programming, the connection should be checked. The online interface COM1 or COM2 must be correctly set so that the hardware configuration can be loaded from the controller.

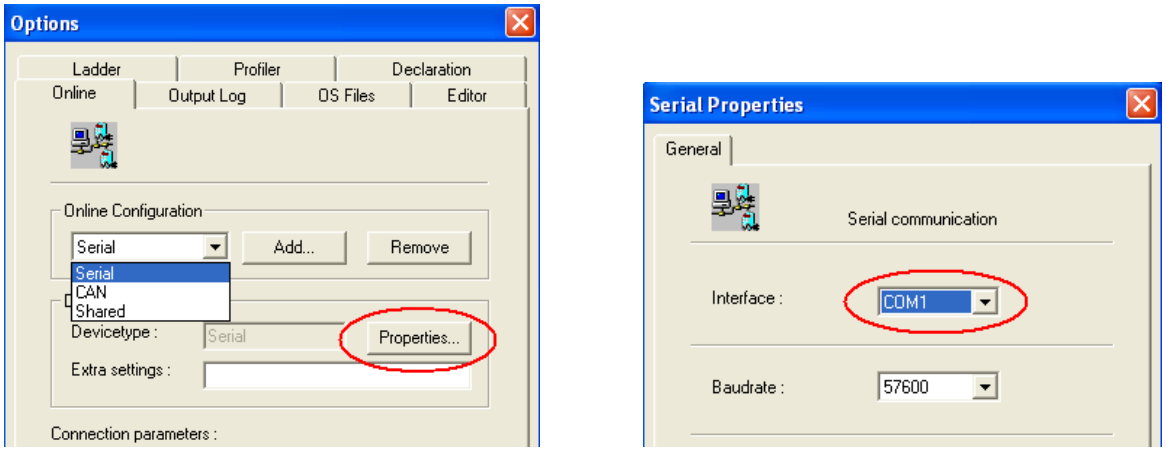

Fig. 8.12: Setting communication between PLC and PC.

Select the command in the menu *File* → *New project* for creating a new project.

When creating a new project, the project data must be entered as shown in the following picture 8.13. Activate the option ***Upload hardware from target***!!

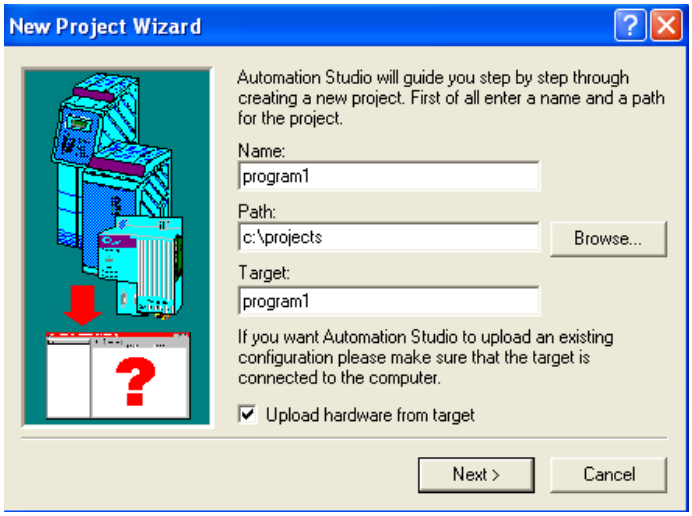

Fig. 8.13: Upload hardware.

On the left side (hardware configuration), individual modules can be selected and inserted. The right side adjusts itself automatically.

Insert task classes and task

Each program is divided into tasks and task classes. Right click on the icon *CPU* and after the selection of *Insert Object*, you start the wizard inserting a new object as follows the picture.

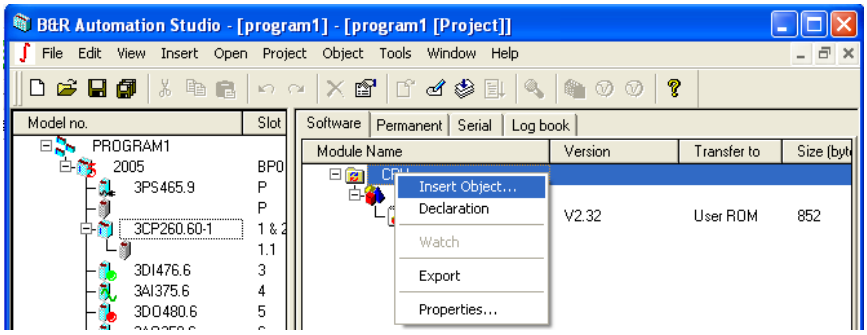

Fig. 8.14: Creation of new task.

For example when you create a new cyclic object, select the cyclic object from the window *Insert Object* (fig. 8.15). After you click on the button Next, in the window *New Objects* you assign the name of the new task class, than you select a language for the task class and the priority for the new task class (fig. 8.16). Confirm with the clicking on the button *Finish*. The new task will be created.

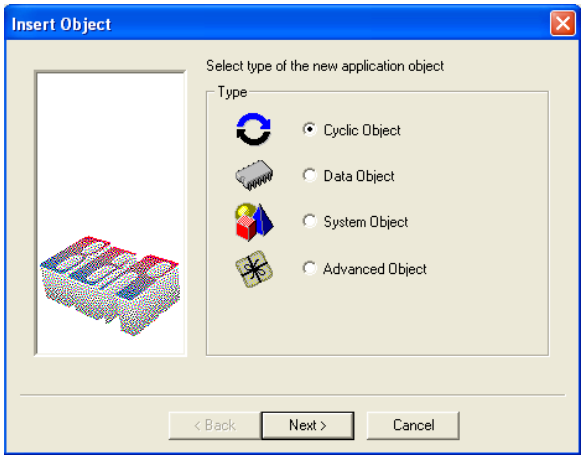

Fig. 8.15: Window Insert Object.

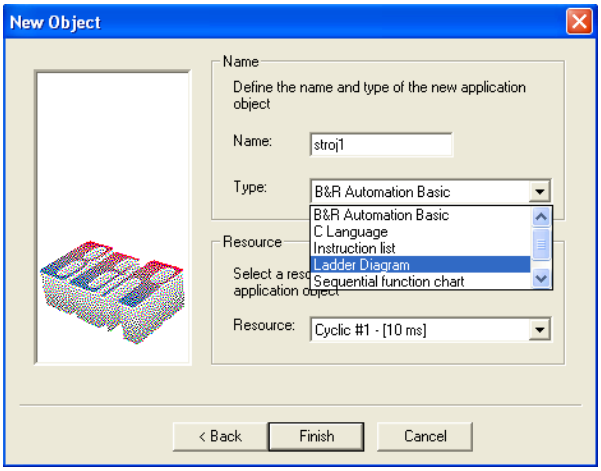

Fig. 8.16: Setting properties task class.

Each tasks consists two parts, initialization part and cyclic part. Select the command in the menu *View* → *Init Subroutine*, is displayed initialization part a task.

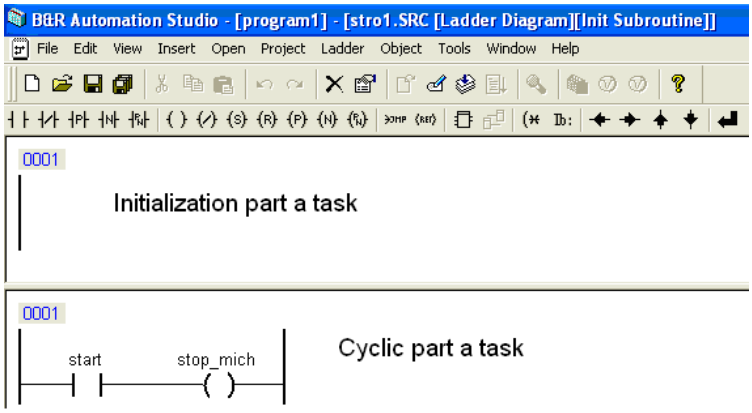

Fig. 8.17: Parts of task.

Download to PLC

Select the command in the menu *Project* → *Build All* (fig. 8.18.) compiling your project or select the command in the menu *Project* → *Transfer To Target* (fig. 8.19.) compiles all tasks and downloads then to the controller.

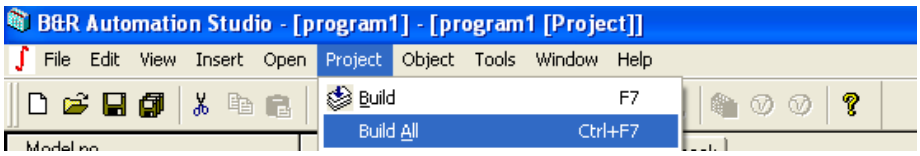

Fig. 8.18: Compile all tasks.

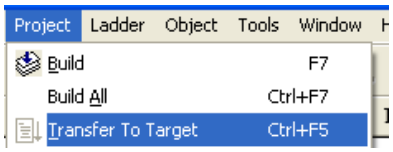

Fig. 8.19: Download to PCC.

8.4 Emulator in Automation Studio

AR000 is an Automation Runtime system based on Windows-32 which is not real-time capable, but essentially corresponds to the functionality of all controllers. Since the AR000 is conceived for testing purposes, no hardware is used.

Therefore, functions and function blocks that do not directly access the hardware or are not designed for specific hardware configurations can be easily tested on the AR000.

The AR000 is started using the menu *Tools*→*AR000*. Once the online connection has been set on the AR000, it can be used like a real SG4 controller and can be used to test the application.

Although the AR000 runtime emulator can't handle real-time functions, it allows us to test different program sections in cyclic operation without needing a real CPU.

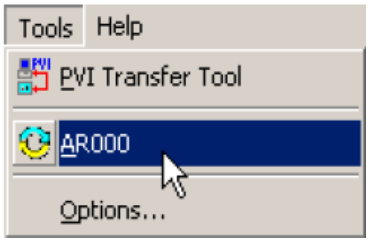

Fig. 8.20: Emulator.

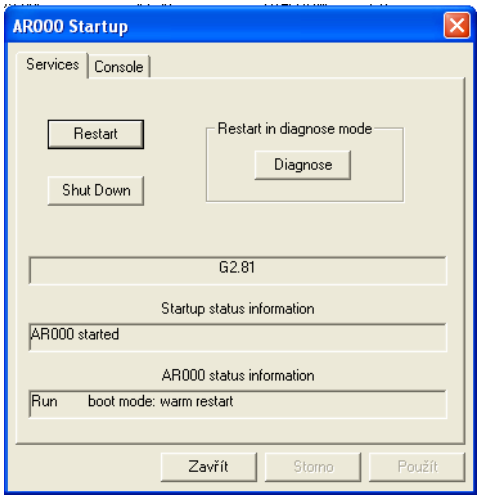

Fig. 8.21: The information that the emulator started.

Creating a connection

In order to be able to work with a controller, it's necessary to establish a connection to it. This is because we need to transfer the project to the target system so that we can test it. The following describes how to specify the type of connection.

Open up the *Tools* → *Options...* menu item.

This opens the following window:

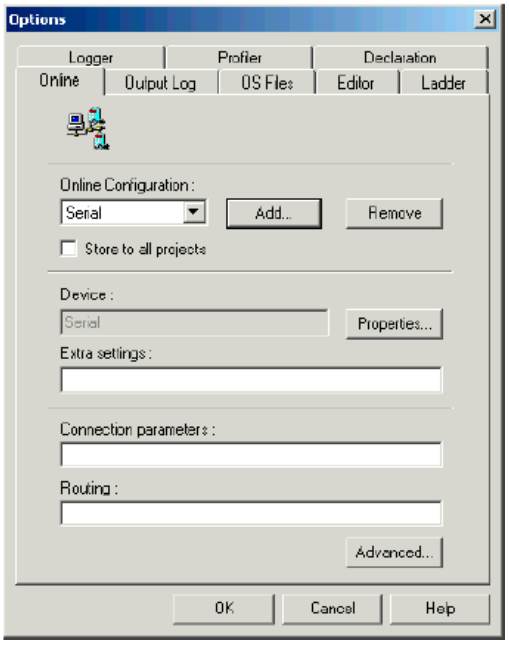

Fig. 8.22: Connection settings.

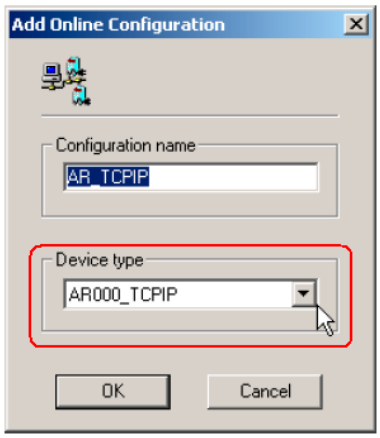

Fig. 8.23: On-line Configuration.

Add a new connection by clicking on the Add... button. Give the new connection a configuration name and change the device type to AR000_TCPIP so that your dialog box appears as follows:

Save these settings by clicking on OK.

The connection settings for emulation have now been completely set up. This information does not have to be entered each time since this connection is saved under the specified configuration name. Accept the settings by clicking on OK.

Questions:

1. *Which types of programmable computer controller are produced by the company Bernecker Rainer?*
2. *How is the memory divided?*
3. *Can you describe deterministic multitasking?*
4. *What is the difference between the task classes and the task?*
5. *What is the difference between the timer task class and a cyclic task class?*
6. *Can you specify task classe?.*
7. *Can you specify the software packages B&R Automation Software?*
8. *Can you specify some data types?*