B&R SYSTEM 2000

PG2000 SOFTWARE
USER'S MANUAL

Version: **1.0 (April 1995)**
Model Number: **MASYS2PG-E**

MS-DOS® is a registered trademark of the Microsoft Corporation
IBM® is a registered trademark of the International Business Machines Corporation
Profibus® is a registered trademark of the Profibus User Organisation

This is a preliminary version of the B&R SYSTEM 2000 PG2000 Software User's Manual intended for client information. Some sections will be extended or changed. The information contained within this manual is believed to be correct at the date of publication and the software will perform substantially in accordance with the documentation.

We reserve the right to change the contents of this manual at anytime without any warning. Bernecker und Rainer Industrie-Elektronik Ges.m.b.H. specifically disclaims, including but not limited to, all implied or inferred warranties of merchantability or fitness for a particular purpose, with regards to the products or documentation contained within this manual. Under no circumstances will Bernecker und Rainer Industrie-Elektronik Ges.m.b.H. be liable for special, incidental, consequential, indirect or any other damages or claims arising from or in connection with the use of these products, including, but not limited to, loss of profits or any other commercial damages, even if Bernecker und Rainer Industrie-Elektronik has been advised of the possibility of such damages.

WHAT'S INSIDE THIS MANUAL

This manual describes and explains the installation and usage the PG2000 Programming System, and other programs that are included in the PG2000 software package as well as the operation of the PLC operating system. What each chapter contains:

Title	Contents
Contents	Table of contents for the entire manual.
Chapter 1 Overview	Quick overview of the software components of the B&R SYSTEM 2000 PLC Family.
Chapter 2 Operating System PLCSW	Contains a detailed description of the Operating System PLCSW.
Chapter 3 Software Installation	Instructions for installing the PG2000 software package.
Chapter 4 Programming System PG2000	Detailed description of the PG2000 Programming System.
Appendix A PLCSW Error Numbers	List of all PLCSW error numbers with their corresponding error description.
B&R Sales and Service	Addressed, Telephone and Fax numbers of all B&R Subsidiaries and Partners.
Index	Index with page number cross reference to help you find information on a topic quickly.

OTHER REFERENCES

There are some other B&R SYSTEM 2000 manuals and documentation that you may find useful, especially as you get to the more complex areas:

B&R SYSTEM 2000 Programming Languages Manual

This manual contains a detailed description of the three programming languages (Ladder Diagram, Statement List and PL2000) of the B&R SYSTEM 2000 family with many application examples.

B&R SYSTEM 2000 Library Reference Manual

Describes all the functions of the libraries provided by B&R in detail. These functions can save you a lot of time and hassle as they are tested code sections created to perform a particular function.

B&R SYSTEM 2000 Hardware User's Manual

Describes all the modules of the B&R SYSTEM 2000 including all technical data for putting systems together.

PG2000 On-line Help

The extensive on-line help system provides you with instant information on the features of the programming system as well as a reference guide to the languages. Sometimes this saves you having to dig into manuals like this one when you only need to refresh your memory.

CONTENTS

CONTENTS

Chapter 1 - Overview Contents-5

Chapter 2 - PLCSW Operating System Contents-5

Chapter 3 - Software Installation Contents-7

Chapter 4 - PG2000 Programming System Contents-7

Appendix A - PLCSW Error Numbers Contents-10

B&R Sales and Service Contents-10

Index Contents-10

CHAPTER 1 - OVERVIEW

1 Introduction 1-5
2 Operating System PLC Software 1-6
2.1 Make Up of the Operating System PLC Software 1-6
2.2 Application Programs 1-7
2.3 Task Classes 1-7
2.4 Monitoring Functions of the Operating System 1-9
2.5 Error Handling 1-9
3 Programming System PG2000 1-10
3.1 PC Requirements 1-11
3.3 Programming in Statement List (STL) 1-12
3.2 Programming in PL2000 1-12
3.4 Programming in Ladder Diagram (LAD) 1-13
3.5 Functions 1-13
3.6 Programming in C 1-14
3.7 Variable Declaration 1-15
3.8 Graphic Design Method (GDM) 1-16
3.9 Source Level Debugger for PL2000 and STL 1-17
3.10 Ladder Diagram Debugger 1-18
3.11 Process Variable Monitor 1-19
4 PLC Configurator 1-20
5 Profiler 1-21

CHAPTER 2 - PLCSW OPERATING SYSTEM

1 Performance Characteristics 2-5
2 Memory Areas of the PLC 2-6
2.1 System RAM 2-6
2.2 Dual Ported RAM (DPR) 2-6
2.3 System ROM 2-7
2.4 User ROM 2-7
2.5 User RAM 2-7
3 How the Operating System Fits Together 2-8
3.1 Operating System Kernel 2-8
3.2 PLC Specific Software Packages 2-8
3.3 Application Programs 2-10
4 Application Programs 2-11
4.1 Cyclic Tasks 2-11
4.2 Non-cyclic Tasks 2-11

5 Task Classes 2-12
5.1 Normal Task Classes 2-12
5.2 High Speed Task Classes (HS Task Classes) 2-12
5.3 Task Class Priority 2-13
5.4 Cyclic Run-time Behaviour of the Task Classes 2-14
5.5 Execution Order of Tasks within a Task Class 2-15
5.6 System Loading 2-16
6 Exception Task Class 2-17
7 Interrupt Task Class 2-19
8 Boot Modes 2-20
8.1 TOTALINIT 2-22
8.2 INIT 2-23
8.3 RESET 2-24
8.4 RECONFIG 2-24
8.5 ERROR 2-25
8.6 DIAGNOSE 2-25
8.7 SERVICE 2-26
9 Power On Behaviour 2-27
10 Boot Behaviour 2-28
10.1 RESET with PG2000 2-28
10.2 RESET with the operation elements on the B&R 2010 CPU Module 2-29
10.3 RESET with the Operation Elements of the B&R 2005 CPU Module 2-31
11 Start Behaviour of Tasks after a TOTALINIT/INIT 2-32
12 I/O Data Transfer 2-34
12.1 Conventional I/O Data Transfer 2-34
12.2 I/O Data Transfer with B&R SYSTEM 2010 2-34
12.3 I/O Data Transfer on B&R SYSTEM 2005 2-38
12.4 Data Exchange between the CPU and I/O Processor 2-39
12.5 Multitasking of the I/O Processor 2-39
13 Monitoring Functions of the Operating System 2-40
13.1 Checksum of B&R Modules 2-40
13.2 System Structures 2-40
13.3 Stack Overflow 2-40
13.4 I/O Data Transfer 2-40
13.5 Cycle Time Monitoring 2-41
13.6 Hardware Watchdog 2-41
14 Error Handling 2-42
14.1 Fatal Errors 2-42
14.2 Exceptions 2-42

CHAPTER 3 - SOFTWARE INSTALLATION

1 Contents of Delivery 3-5
2 Hardware Requirements 3-5
3 New Installation 3-6
3.1 New Installation of the PG2000 Programming Systems 3-6
3.2 New Installation of the Utility Box 3-10
3.3 New Installation of the PLC-OS OPTIONS 3-12
3.4 New Installation of the Function Libraries 3-12
3.5 Directory Structure 3-13
4 Replacing an Old Software Version 3-14
5 System Configuration and Memory Optimization 3-15
5.1 BIOS Setup 3-15
5.2 Settings in the CONFIG.SYS file 3-15
5.3 Settings in the Batch File AUTOEXEC.BAT 3-17

CHAPTER 4 - PG2000 PROGRAMMING SYSTEM

1 Highlights 4-7
1.1 Operation 4-7
1.2 Programming 4-7
2 Start Preparations 4-8
2.1 System Configuration 4-8
2.2 Use of a Pointing Device 4-12
2.3 Printing Requirements for PG2000 4-12
2.4 Starting the Programming System 4-12
3 General Operation information 4-13
3.1 Creating a Test Project 4-14
3.2 Screen Components 4-15
3.5 Conventions 4-16
3.3 Using the Mouse 4-16
3.4 Using the Keyboard 4-16
3.6 Main Menu Bar and Pull Down Menus 4-17
3.7 Working with Windows 4-19
3.8 Foot Line Menu 4-25
3.9 Context Sensitive Pop Up Menus 4-26
3.10 Dialog Boxes 4-27
3.11 File Selection Box 4-30
3.12 General Hot Keys 4-31
3.13 Help System 4-32
4 Introductory Example 4-35
4.1 Example Description 4-35
4.2 Hardware Requirements 4-35
4.3 Creating the Project 4-36

4.4 Calling the LAD Editor via the GDM 4-37
4.5 Creating the Ladder Diagram 4-39
4.6 Saving the Ladder Diagram 4-40
4.7 Variable Declaration 4-41
4.8 Connecting to the PLC 4-44
4.9 Compiling and Downloading to the PLC 4-45
4.10 Testing the Example Program with the LAD Debugger 4-45
5 Project Management 4-48
5.1 Creating a Project 4-49
5.1.1 Project Databank 4-51
5.1.2 Program Files 4-52
5.2 Opening a Project 4-54
5.3 Copying a Project 4-54
5.4 Including Projects in Project Management 4-55
5.5 Purging a Project 4-56
5.6 Deleting a Project 4-56
6 GDM - Graphical Design Method 4-57
6.1 General Information 4-57
6.2 Calling the GDM 4-58
6.3 Calling the Task Layer 4-59
6.4 Placing an Object in the Task Layer 4-60
6.5 Object Parameters 4-62
6.5.1 Parameters for Normal, High Speed, IRQ and EXC Tasks 4-62
6.5.2 Parameters for Data Modules and B&R Modules 4-65
6.5.3 PLC Configuration Module Parameters 4-66
6.6 Editing an Object 4-69
6.7 Ladder Diagram - LAD 4-70
6.7.1 General Information 4-70
6.7.2 Ladder Diagram Syntax 4-71
6.7.3 LAD Editor 4-74
6.8 Statement List - STL 4-84
6.8.1 General Information 4-84
6.8.2 Syntax of the Programming Language - STL 4-84
6.8.3 STL Editor 4-88
6.8.4 Configuring the STL Editor 4-94
6.9 PL2000 4-96
6.9.1 General Information 4-96
6.9.2 Syntax of the Programming Language - PL2000 4-96
6.9.3 PL2000-Editor 4-99
6.9.4 Configuring the PL2000 Editor 4-105
6.10 Variable Declaration 4-107
6.10.1 Symbolic Names 4-107
6.10.2 Scope 4-108
6.10.3 Data Type 4-109
6.10.4 Length 4-110
6.10.5 I/O Type 4-111
6.10.6 Init Value 4-119

6.10.7 Long Name 4-119
6.10.8 Configuring the Variable Declaration Editor 4-120
6.11 Initialization Sub-Programs 4-123
6.11.1 Creating an INIT-SP 4-123
6.11.2 Opening or Deleting an INIT-SP 4-123
6.12 Compiling Objects 4-124
6.12.1 Compiler Components 4-124
6.12.2 Compiler Options 4-125
6.13 Download object in User RAM 4-128
6.13.1 Download Function 4-128
6.13.2 Overload Function 4-129
6.14 Download Object in the User ROM or FIX-RAM 4-130
6.15 Delete Objects 4-131
6.16 Process Variable Monitor (PV Monitor) 4-132
6.17 Debugger 4-139
6.17.1 Ladder Diagram Debugger (LAD Debugger) 4-140
6.17.2 STL and PL2000 Debugger 4-144
6.18 Putting in a Second PLC 4-149
6.19 Foot Line Menu of the GDM 4-150
6.20 PLC Layer Pop Up Menu 4-154
6.21 Task Layer Pop Up Menu 4-158
6.22 GDM Keyboard Commands 4-162
6.23 Synchronizing the PLC and GDM 4-163

7 Functions 4-164
7.1 Standard Functions 4-164
7.1.1 General Information 4-164
7.1.2 Importing Function Libraries 4-164
7.1.3 Help on the Function Libraries 4-165
7.1.4 Deleting a Function Library 4-166
7.2 Application Specific Function Blocks 4-167
7.2.1 General Information 4-167
7.2.2 Creating An Application Specific Function Block 4-167

8 PLC Utilities 4-172
8.1 Task Management Utilities 4-172
8.2 Task Class Utilities 4-174
8.3 Module - Display and Manipulation 4-175
8.4 Error Analysis 4-178
8.5 Other Functions 4-180
8.6 Modem 4-184
8.7 PV Monitor 4-185

APPENDIX A - PLCSW ERROR NUMBERS

B&R SALES AND SERVICE

INDEX

CHAPTER 1
OVERVIEW

CONTENTS

1 Introduction 1-5

2 Operating System PLC Software 1-6
2.1 Make Up of the Operating System PLC Software 1-6
2.2 Application Programs 1-7
2.3 Task Classes 1-7
2.4 Monitoring Functions of the Operating System 1-9
2.5 Error Handling 1-9

3 Programming System PG2000 1-10
3.1 PC Requirements 1-11
3.3 Programming in Statement List (STL) 1-12
3.2 Programming in PL2000 1-12
3.4 Programming in Ladder Diagram (LAD) 1-13
3.5 Functions 1-13
3.6 Programming in C 1-14
3.7 Variable Declaration 1-15
3.8 Graphic Design Method (GDM) 1-16
3.9 Source Level Debugger for PL2000 and STL 1-17
3.10 Ladder Diagram Debugger 1-18
3.11 Process Variable Monitor 1-19

4 PLC Configurator 1-20

5 Profiler 1-21

1 INTRODUCTION

The software in the B&R 2000 PLC family is based on a real-time multitasking operating system. B&R has this basic functionality adapted especially for use in control technology without losing the advantages of multitasking. You can read about how the operating system works and where the advantages for your application are in the "Operating System PLCSW" chapter.

The PG2000 programming system is a powerful software package for programming the B&R 2000 PCC-PLC family. An installation guide for the PG2000 software package can be found in the "Software Installation" chapter. The programming system PG2000 allows you to program all CPU modules of the B&R 2010 and B&R 2005 control systems. The individual programming steps are integrated in a simple and clearly arranged development desktop. There is a uniform user interface for all the editors and debuggers. This means that after a short introductory and learning phase, nothing stands between you and your application.
The possibilities that PG2000 offers you are described in detail in the "PG2000 Programming System" chapter.

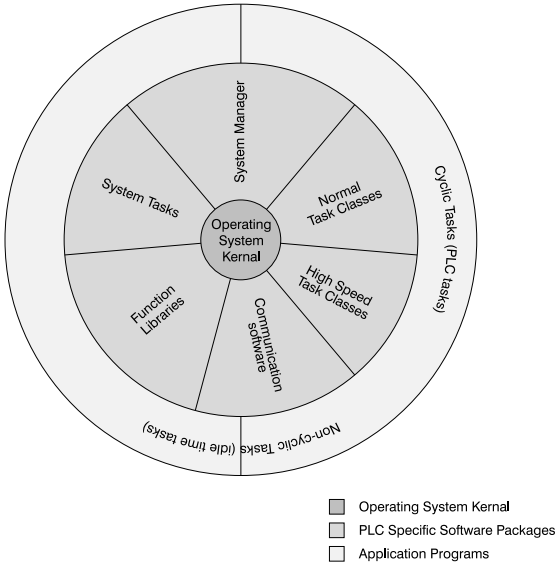
When you want to optimize the configuration of your controller for a particular application, there is the PG2000 "PLC Configurator". This utility allows you to modify the run times of the individual time intervals, the size of the memory areas, etc. to suit your application.
By way of multitasking, the CPU power is divided among the individual tasks. Basically, the processing time is not defined by the run time of the application,but by the operating system. The "Profiler" is a PG2000 utility that displays the system information (e.g. CPU loading) in a meaningful form for the user. In this way, the optimization of the time usage of the application is not a difficult and tedious trial method, instead it is the simple setting of a few program parameters.

2 OPERATING SYSTEM PLC SOFTWARE

The operating system **PLC Software (PLC SW)** is a configurable deterministic real time multitasking system. This means the system functions and application programs run in parallel in a definable, time consistent and controllable time frame.

2.1 MAKE UP OF THE OPERATING SYSTEM PLC SOFTWARE

The operating system PLC software is modular and consists of an operating system kernel and PLC specific software packages.



Operating System Kernel

The operating system kernel is a multitasking capable standard operating system. Its primary jobs are resource management for applications and multitasking.

PLC Specific Software Packages

The multitasking capable operating system is adapted especially for controller technology with the PLC specific software packages.

Application Programs

The application programs are managed by the operating system. Several application programs can be run in one CPU simultaneously because the operating system PLC software is multitasking capable.

2.2 APPLICATION PROGRAMS

Application programs are basically split into two categories **cyclic tasks** (also called PLC tasks) and **non-cyclic tasks** (also called idle time tasks):

- **Cyclic Tasks** (PLC tasks) are processed exactly once within a definable time (= cycle time). The operating system PLCSW monitors if this cycle time is kept.
  In order to assign priorities, you can put the cyclic tasks in various task class (different cycle times).
- **Non-cyclic Tasks** (idle time tasks) are only executed when neither the operating system nor the PLC tasks require CPU time. Idle time tasks can only be created with the C Toolbox.

2.3 TASK CLASSES

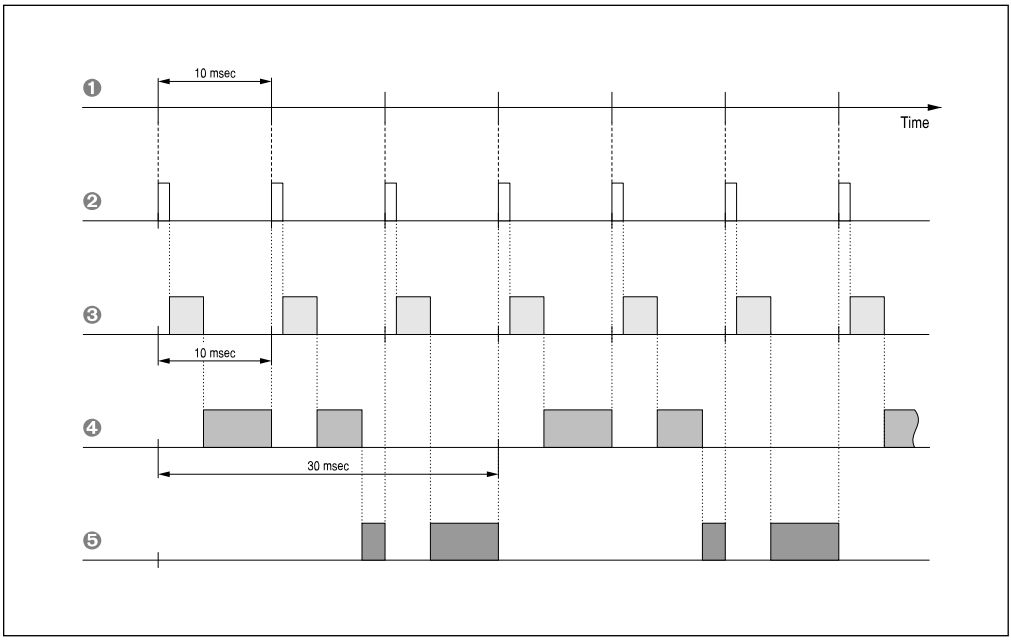
In order to set PLC tasks priority relative to one another, you can put the tasks in different task classes. You can set an individual cycle time for each task class. All tasks within the task class are processed exactly once within the cycle time. Tasks which are important and time critical are put into a task class with a short cycle time. This means that they are executed more often than those in a task class with a longer cycle time. The cycle time has nothing to do with the actual run-time of the tasks on the PLC.
You have two types of task classes available: **normal task classes** and **high speed task classes**:

- The switching between tasks within **normal task classes** is done by the operating system (system manager). The CPU modules of the B&R 2000 family offer up to four normal task classes. The cycle times of the task classes can be set with the PLC Configurator or the PG2000 Programming System (V2.10 and later).
- Tasks in the **high speed task classes (HS** task classes) are not called by the operating system, but by individual hardware timers (interrupts). The cycle times of the task classes can be set with the PLC Configurator or the PG2000 Programming System (V2.10 and later). The CPU modules of the B&R 2010 offer up to four HS task classes and for the B&R 2005, only one HS task class is possible.

To correctly share the processor power, task classes are assigned a priority (higher priority task classes can interrupt lower priority task classes). The priority of the task classes does **not** depend on the cycle time ! The following table shows the relative priorities of the task classes and the default settings for the cycle times:

Task Class	Priority	Cycle Time (default)	Cycle Time Violation Allowed (default)	Where Available
HS-Task Class 1	high	3000 µsec	No cycle time violation allowed	B&R 2010 and B&R 2005
HS-Task Class 2		5000 µsec	No cycle time violation allowed	B&R 2010
HS-Task Class 3		7000 µsec	No cycle time violation allowed	
HS-Task Class 4		9000 µsec	No cycle time violation allowed	
Task Class 1		10 msec	20 msec	B&R 2010 and B&R 2005
Task Class 2		50 msec	50 msec	
Task Class 3		100 msec	100 msec	
Task Class 4	low	10 msec	30000 msec	

The execution order of the task classes is a direct result of the task class priorities (low priority task classes are interrupted by higher priority task classes). The following diagram demonstrates the principle of task class run-time behaviour:



❶ Time Frame (unchangable): 10 msec Operating System Clock
❷ Operating System (System Manager)
❸ Task Class 1 (Cycle timet: 10 msec); Task run-time: approx. 3 msec
❹ Task Class 2 (Cycle time: 30 msec); Task run-time: approx. 10 msec
❺ Idle time tasks use the remaining time CPU time

2.4 MONITORING FUNCTIONS OF THE OPERATING SYSTEM

The operating system PLC software has various monitoring and safety functions that are cyclically executed to guarantee the availability of the system:

The Operating System monitors ...
- ... the B&R module checksums
- ... the system structures
- ... stack overflow
- ... the I/O data transfer
- ... the cycle time of the individual task classes
- ... the operating system through the hardware watchdog.

2.5 ERROR HANDLING

All errors that occur within the scope of an application (e.g. cycle time violations, bus errors, error from incorrect use,...) are reported by the operating system. This report is put in a separate error module and can be read with the PG2000 Programming System or by the application itself. This means you can find out the cause of the error and how to avoid it in future.

3 PROGRAMMING SYSTEM PG2000

PG2000 has all the features of modern software packages: mouse operation, dialog boxes, pop-up and pull-down menus. There is also instant online help for all functions.

The PG2000 Programming System consists of the following parts:

Project Management ..... to create, open, delete copy, etc. a project
Programming Environment ... to program a project (create, compile code, download to PLC, etc.)

Before the PG2000 programming system can be utilized, a project has to be selected. After starting PG2000 for the very first time, a reduced main menu (Project Management) is shown, containing only those functions necessary for selecting a project. The functions in this menu can be used to search through and select a project on your hard disk, create new projects, duplicate existing projects or delete projects that are no longer needed.

After selecting a project, the project menu is replaced with the main menu (**Programming Environment**).



PG2000 Programming System Project Management

Anytime you want to change projects you can *Close* the current project and you are then returned to the project menu.



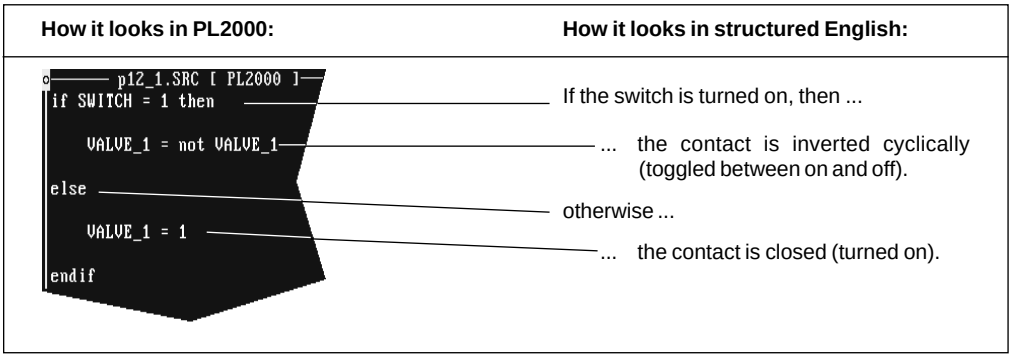
PG2000 Programming System Programming Environment

3.1 PC REQUIREMENTS

The PG2000 Programming System runs on PCs with MSDOS operating system version 3.30 or later. The connection to the PLC is made by the Programming System via either COM1 or COM2 (selectable). If a PROFIBUS network module is used, you can also program the PLC over the network.

3.2 PROGRAMMING IN PL2000

The PL2000 programming language is a high level language especially developed for structured programming of automation tasks. This high level language is easy to learn as well as being ideally suited to the control technology area.



Programming in PL2000

Advantages:

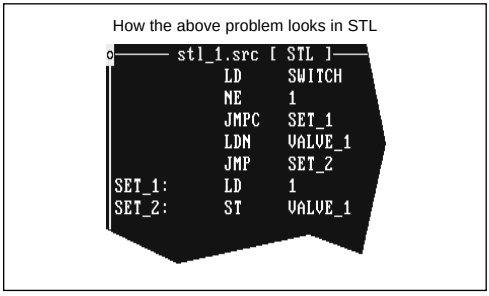
- Writing programs for larger and more complex automation problems is simplified through the availability of **Language Constructs** (IF, ELSE, ...).
- The variables used are given **symbolic names**. The relationship between the symbolic names and the hardware (switches, sensors, valves, ...) can be set later using the **variable declaration**.
- The **syntax sensitive editor** recognizes syntax errors in the input line and informs you of the problem.
- This readability of PL2000 programs is increased by comments that can be inserted at any position.

3.3 PROGRAMMING IN STATEMENT LIST (STL)

STL is a standardized programming language in alphanumerical form. It is a tried and tested assembler type language for writing programs in the automation area. The variables are given symbolic names. The STL in PG2000 corresponds to the IEC 1131 standard.

Advantage:

- The standardized language STL allows for software ported between PLC systems of different manufacturers.

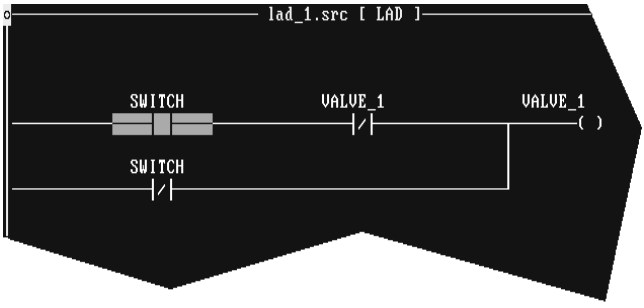


Programming in STL

3.4 PROGRAMMING IN LADDER DIAGRAM (LAD)

Ladder diagram is a circuit diagram representation of a control task. You can enter the individual logic paths in the Ladder Diagram Editor.

How the previous problem looks in LAD



Programming in LAD

Advantage:

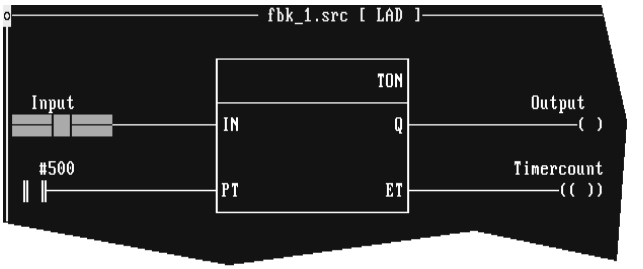
- ❍ Simple control logic is displayed in a fashion similar to a circuit diagram.

3.5 FUNCTIONS

Functions are frequently used program sections that have already been written. These program sections (e.g. edge recognition, delayed switching, etc.) can be called in any application program (PL2000, STL, LAD) as often as you want.

PG2000 includes a large number of libraries (function libraries). The number and types of functions are always being extended.

Function TON (Delayed switching on)



Using a function in LAD

Advantages:

- ❍ Saves program development time and program code by connection functions.
- ❍ Creation and usage of program libraries.

Creating User Specific Function Blocks

If a program section is used several times, a user specific function block (FBK) can be created with PG2000 that contains this section of code.

Procedure to create a function block:

- Write code for the function block in one of the three programming languages (PL2000, STL or LAD).
- Define the inputs and outputs of the function block in an FBK editor.
- Compile the function block.

3.6 PROGRAMMING IN C

With the "C-Toolbox" available from B&R, you can...

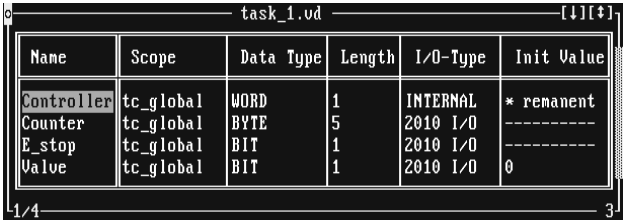
- Create and write functions in "C".
- Write **idle time tasks** in "C" and link them into a project on the PLC.



The C-Toolbox is optionally available from B&R with the corresponding C compiler.

3.7 VARIABLE DECLARATION

The connection between the variables in your program and the external I/O points (switches, encoders, valves,...) is made using the variable declaration. Each variable is assigned a defined scope, i.e. variables are available for either only one task, one task class or for all task classes. The variable declaration can be called at any time and is a table editor.



Variable Declaration

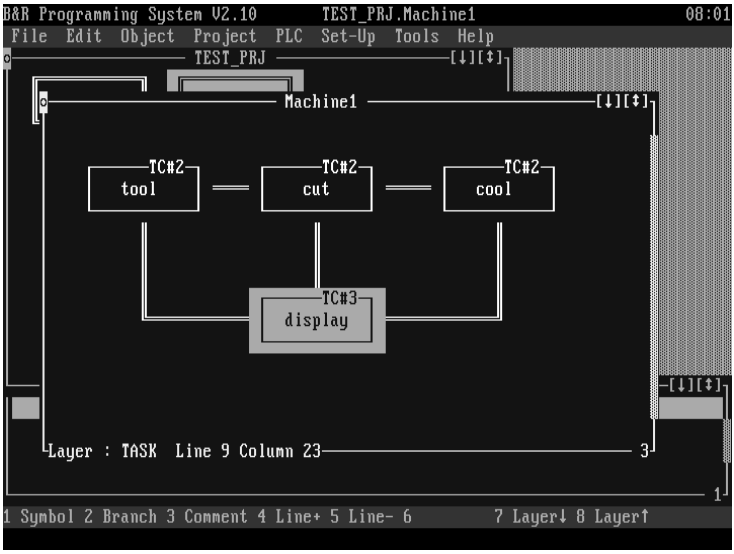
Advantages:

- ❍ Programming is hardware independent. The programmer does not have to take the hardware into consideration at all.
- ❍ The hardware assignments (I/O points) are not coupled with the source code.
- ❍ If a task is ported to various PLCs with different configurations, only a part of the variable declaration has to be changed, not the actual program code.

You can also import a variable declaration, as well as the manual hardware assignment using a table editor. This allows PG2000 to exchange I/O information easily with a CAD system using the import/export functions.

3.8 GRAPHIC DESIGN METHOD (GDM)

The GDM is a tool for planning and analysing automation projects. You can subdivide your automation task and setup links between the various parts. You can also plan everything with reference to the hardware.
There are three basic layers: PLC, processor (optional) and task. The PLC layer has all the PLCs shown as objects. Each PLC object "goes down" to a processor layer, where all the processors on this PLC are shown as objects. In turn, each processor object "goes down" to a task layer where all tasks are shown as objects. Each encapsulated object consists of other objects in a heireichary, where what is done to an object is done to all objects "inside" this one. The data flow between objects is shown by link lines between them.



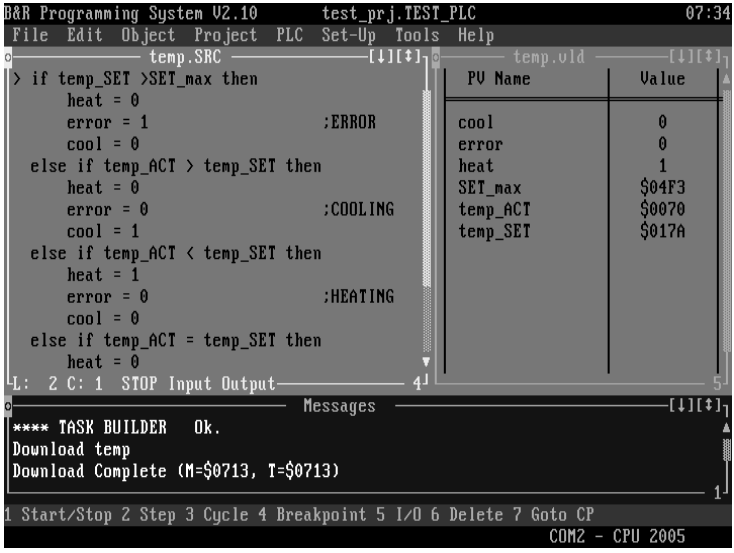
Graphic Design Method (GDM)

Advantages:

- The automation job is taken apart into little pieces and displayed in the GDM, providing a good overview.
- The data flow between the individual tasks is shown (especially important when several tasks are using a variable).
- All PG2000 functions can be directly called from the GDM. This provides you with a comfortable interface for working on your whole project.

3.9 SOURCE LEVEL DEBUGGER FOR PL2000 AND STL

You can find errors in tasks that are written in PL2000 or STL and loaded onto the PLC with the source level debugger. The task is taken out of the cyclic run-time system and run independent of the cycle time. You can control the execution speed of the task. Within the debugger, you can set up watches on variables to monitor their values in a non-cyclic Process Variable Monitor.



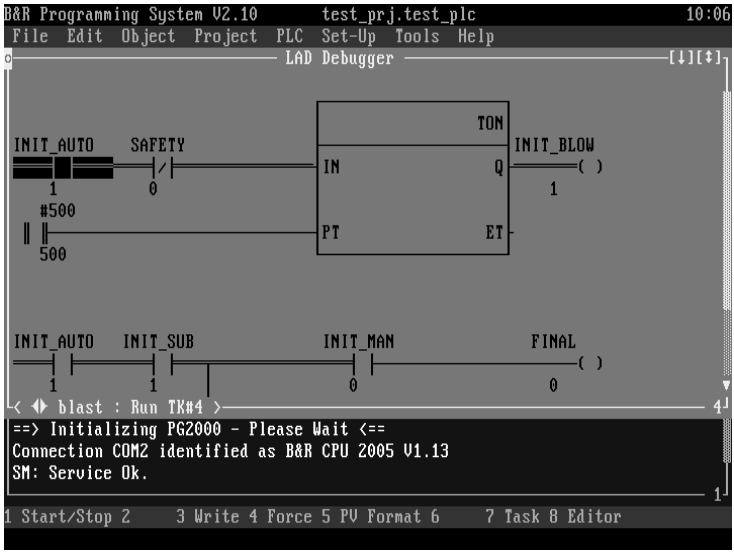
Source Level Debugger for PL2000

Advantages:

- The debugger lets you run a task step by step and see the influence of each program line.
- You can work with longer task thanks to the step, breakpoint and cycle functions.
- The Process Variable Monitor shows the current values of the variables.
- You can decide whether the task should change the actual outputs of the system while in the debugger or if the test should be run independent of the physical world. This is a feature to ensure the safety when using critical equipment.

3.10 LADDER DIAGRAM DEBUGGER

The Ladder Diagram Debugger (LAD Debugger) allows you to see the state of a task (that has be written in **LAD** and downloaded to the PLC) ONLINE. After starting the LAD debugger, the condition of the individual contacts, connection paths and function block contacts are cyclically read from the PLC and displayed. Working with several windows in parallel is possible even in the debugger due to the window based development environment of PG2000.



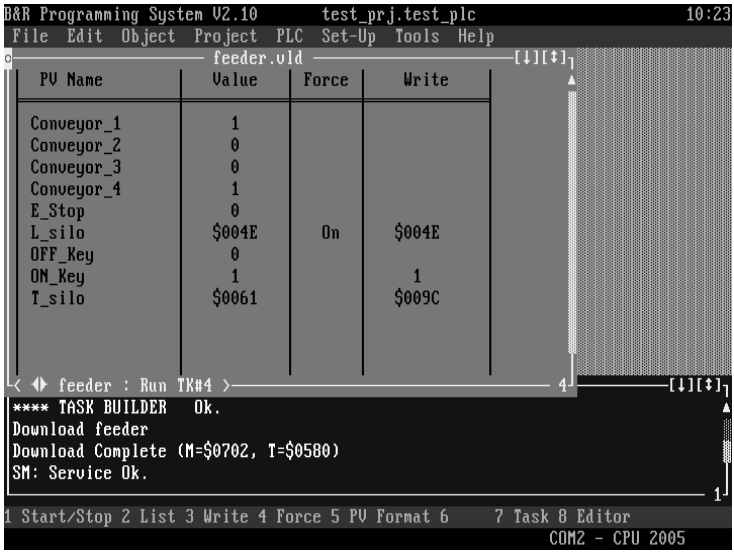
Ladder Diagram Debugger

Advantage:

- The task variables are shown directly on the LAD mask in the LAD debugger.

3.11 PROCESS VARIABLE MONITOR

The Process Variable Monitor (PV Monitor) can be used either in conjunction with the debuggers or on its own. It allows you to watch variables in a task running on the PLC. The list of variable is updated according to a user setting.



Process Variable Monitor

Advantage:

- The list of process variable can be saved and loaded again at a later date. This is especially useful for service and maintenance calls since the necessary variables are available without having to recreate the list each time.

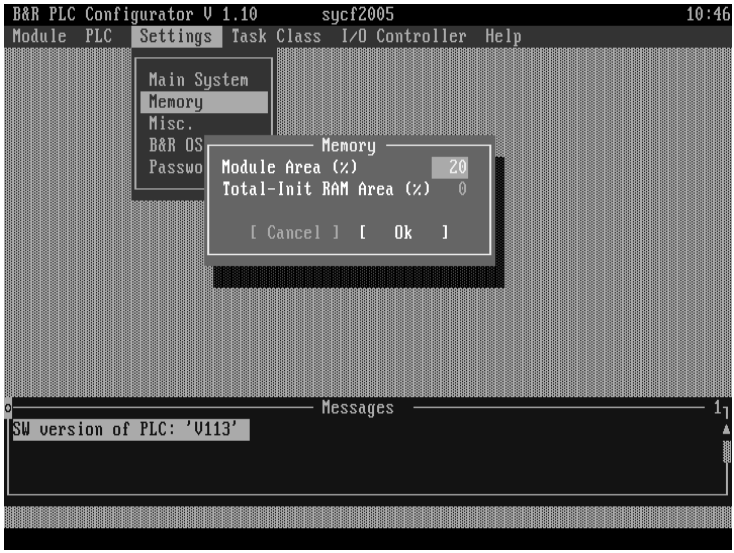
4 PLC CONFIGURATOR

The PLC Configurator allows you to configure the operating system PLCSW specifically for you application. By doing this, the capabilities of the PLCSW (e.g. number of task classes) can be limited for a particular application so that the processor power is used in the most efficient manner.

A small selection of the parameters that can be configured:

- Number of normal and high speed task classes
- Cycle time for each task class
- Allowed cycle time violations

The PLC configurator has all the normal features of modern software packages: mouse operation, dialog boxes, pop-up and pull-down menus. There is of course extensive online help available for all functions.



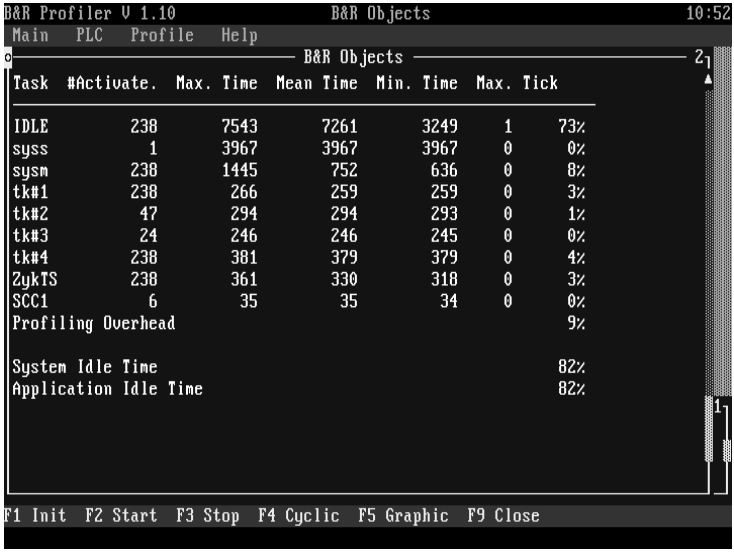
PLC Configurator

5 PROFILER

The profiler is a software package to measure the run-time of all PLC and system tasks. The run-time measurements provide you with information about how loaded your system is and the run-time behaviour of the individual task classes. This information makes it easier to select the correct task classes and optimize the system. The profiler has the uniform, easy-to-use menu driven environment.

A small selection of the functions of the Profiler:

- Measuring the run-time of the task classes
- Measuring the task run-times
- Cyclic evaluation of the run-time behaviour
- Displaying the system and PLC tasks that are running on the PLC
- Finding the required stack sizes



Run-time measurement with the Profiler

CHAPTER 2
OPERATING SYSTEM PLCSW

CONTENTS

1 Performance Characteristics 2-5

2 Memory Areas of the PLC 2-6
2.1 System RAM 2-6
2.2 Dual Ported RAM (DPR) 2-6
2.3 System ROM 2-7
2.4 User ROM 2-7
2.5 User RAM 2-7

3 How the Operating System Fits Together 2-8
3.1 Operating System Kernel 2-8
3.2 PLC Specific Software Packages 2-8
3.3 Application Programs 2-10

4 Application Programs 2-11
4.1 Cyclic Tasks 2-11
4.2 Non-cyclic Tasks 2-11

5 Task Classes 2-12
5.1 Normal Task Classes 2-12
5.2 High Speed Task Classes (HS Task Classes) 2-12
5.3 Task Class Priority 2-13
5.4 Cyclic Run-time Behaviour of the Task Classes 2-14
5.5 Execution Order of Tasks within a Task Class 2-15
5.6 System Loading 2-16

6 Exception Task Class 2-17

7 Interrupt Task Class 2-19

8 Boot Modes 2-20
8.1 TOTALINIT 2-22
8.2 INIT 2-23
8.3 RESET 2-24
8.4 RECONFIG 2-24
8.5 ERROR 2-25
8.6 DIAGNOSE 2-25
8.7 SERVICE 2-26

9 Power On Behaviour 2-27

10 Boot Behaviour 2-28
10.1 RESET with PG2000 2-28
INIT or TOTALINIT with PG2000 2-28
RESET or DIAGNOSE with PG2000 2-28
PLC Reset with the Reset key 2-29
10.2 RESET with the operation elements on the B&R 2010 CPU Module 2-29
PLC Reset with Boot Keys 2-29
PLC Reset with Boot and Reset Keys 2-30
10.3 RESET with the Operation Elements of the B&R 2005 CPU Module 2-31
PLC Reset with the TOTALINIT key 2-31
PLC Reset with the Reset key 2-31
PLC Reset with the TOTALINIT and Reset keys 2-31

11 Start Behaviour of Tasks after a TOTALINIT/INIT 2-32

12 I/O Data Transfer 2-34
12.1 Conventional I/O Data Transfer 2-34
12.2 I/O Data Transfer with B&R SYSTEM 2010 2-34
Normal Task Classes in the Boot-Up Phase 2-35
Normal Task Classes in Cyclic Operation 2-36
High Speed Task Classes in the Boot-Up Phase 2-37
High Speed Task Classes in Cyclic Operation 2-37
12.3 I/O Data Transfer on B&R SYSTEM 2005 2-38
12.4 Data Exchange between the CPU and I/O Processor 2-39
12.5 Multitasking of the I/O Processor 2-39

13 Monitoring Functions of the Operating System 2-40
13.1 Checksum of B&R Modules 2-40
13.2 System Structures 2-40
13.3 Stack Overflow 2-40
13.4 I/O Data Transfer 2-40
13.5 Cycle Time Monitoring 2-41
13.6 Hardware Watchdog 2-41

14 Error Handling 2-42
14.1 Fatal Errors 2-42
14.2 Exceptions 2-42

1 PERFORMANCE CHARACTERISTICS

The B&R SYSTEM 2000 is based on a multitasking operating system that has been specially adapted for use in control automation. With this operating system, B&R has successfully used the advantages of a real time multitasking system for a PLC. You can define the cycle time you want for your program. To do this there are up to 8 so-called "task classes" for you to use that can have different cycle times. The operating system takes care to ensure that all application programs are processed within the defined time frame. The operating system of the control generation B&R 2000 is a definable, deterministic (predictable), real time multitasking system.

Through this technology ...

... an extensive project can be divided up into smaller parts. This method increases the modularity and simplifies the maintenance of your project.

... each application program can be given the cycle time that it really needs.

... the reaction time for a particular event is not limited by the cycle time of the entire program.

... the cycle time for each application part can be set and therefore can be exactly predicted.

... a module is practically "several CPUs" with a variable power setting on the same I/O bus.

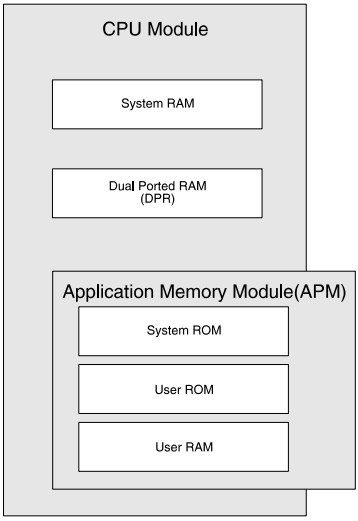
The operating system for the B&R SYSTEM 2000 will be called **PLCSW** (short for PLC Software)



This description is for version 1.10 of the operating system PLCSW.

2 MEMORY AREAS OF THE PLC

In order to understand the individual functions of the PLC software, you really need an overview of the memory areas of the PLC. Basically there are two different memory areas, one in the Application Memory Module (APM) and one in the CPU Module. Both the application data and the operating system are stored on the Application Memory Module. In this way, version conflicts on the B&R 2000 between the operating system and the application are minimized.



Memory	Type	Comments
System RAM	RAM	limited non-volatile memory (buffering when PLC is powered down)
DPR	RAM	limited non-volatile memory (buffering when PLC is powered down)
System ROM	ROM	non-volatile memory
User ROM	EPROM or Flash-PROM	non-volatile memory
User RAM	RAM	limited non-volatile memory (buffering when PLC is powered down)

2.1 SYSTEM RAM

The System RAM is used primarily by the operating system. It is also for saving all the system parameters. Free memory in the System RAM can be used and divided up into a module area and a temporary area using the PLC Configurator. The module area is used by the operating system for the download of B&R modules in the PLC. You can also use the module area for allocating buffered memory. The temporary area is for you to allocate fast access non-buffered (volatile) memory from within tasks (volatile memory must be reallocated after each boot up).

2.2 DUAL PORTED RAM (DPR)

The DPR contains all the data used for all variables used in application programs (both internal variables and I/O data). The data exchange between the CPU and the I/O processor is done with this memory. Both the CPU and the I/O processor can access the DPR.

2.3 SYSTEM ROM

The system ROM cannot be programmed (written to) or erased by the user. The operating system of the CPU is in this memory. Since the system ROM is in the Application memory module (APM) the whole CPU does not need to be exchanged to update the operating system.

2.4 USER ROM

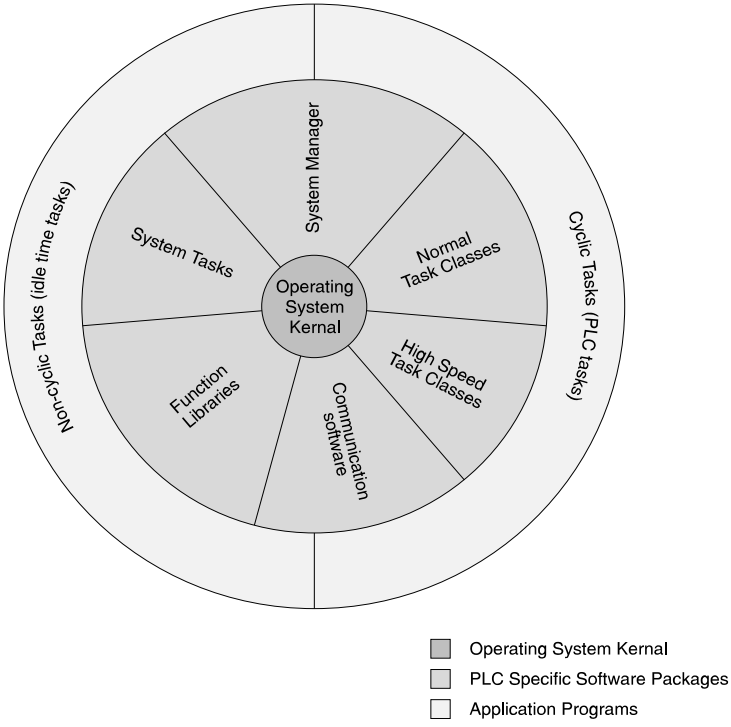
Application programs (cyclic and non-cyclic tasks, data modules, etc.) can be stored in the non-volatile User ROM (EPROM or Flash PROM). Programming (writing to) both types of memory (EPROM and Flash-PROM) is supported by PG2000. The hardware required to do this is on the APM.
The Flash PROM can be erased with software but the EPROM can only be deleted with a UV light (erasing window on APM).

2.5 USER RAM

The user RAM is available for application software (cyclic and non-cyclic tasks, data modules, etc.). You can download (transfer) a compiled application program into the RAM with the *Download* function of the PG2000 programming System.
You can configure a part of the user RAM so that it is not erased with a TOTALINIT like the User ROM (detailed information is in the section "Boot Modes"). This section of memory is called FIX RAM. This can be setup with the PLC Configurator or the PG2000 Programming System. Using the function *Fix* in PG2000 you can store a B&R module here (a B&R module is e.g. all modules that have been compiled into machine code).

3 HOW THE OPERATING SYSTEM FITS TOGETHER

The operating system of the B&R SYSTEM 2000 PLCs is a configurable, deterministic real-time multitasking system and is called the **PLCSW** (abbreviation for PLC SoftWare). The modular PLCSW consists of an operating system kernel and the PLC specific software packages.



3.1 OPERATING SYSTEM KERNEL

The operating system kernel is a multitasking-capable standard operating system that has the primary jobs of application resource management and multitasking.

3.2 PLC SPECIFIC SOFTWARE PACKAGES

Using the PLC specific software packages, the multitasking capable operating system has been adapted especially for use in control technology.

The jobs falling under this heading are split up into the following tasks:

a) System Manager

The system manager is activated every 10 msec. by the operating system clock. It has the following jobs:

- Monitoring the task class cycle time.
- Running the system checks (see section "Monitoring Functions of the Operating System").
- Setting up the input image for the individual task classes (see section "I/O Data Transfer").
- Managing the on-line communications to the PG2000 programming system or serial communications to panels.

b) Normal Task Classes

The normal task classes are activated by the system manager and perform the jobs:

- Controlling the I/O images for the task class (see section "I/O Data Transfer").
- Calling the tasks in the task class corresponding to the task execution order.
- Forcing of inputs and/or outputs (forced process variables are set to a defined value each cycle).
- Ordering the output image after calling all cyclic tasks.

c) High Speed Task Classes

The high speed task classes are activated by a hardware timer. The jobs of the high speed task classes are:

- Ordering and checking the I/O images for the high speed task class (see section "I/O Data Exchange").
- Calling the tasks in the high speed task class corresponding to the task execution order.
- Forcing of input and outputs (forced process variables are set to a defined each cycle).

d) Communication Software

The communications software is activated by the system manager or by communication interrupts.
Jobs of the communication software:

- Download/Upload of the B&R Modules.
- Writing/Reading of process variables.
- Manipulation of tasks (Start, Stop, Remove) and B&R Modules (Burn, Remove).
- PG2000 Communication Services.
- Driver for different protocol (downloadable firmware extensions).

e) Function Libraries

The function libraries can either be called by the system or by the application program. The function libraries fulfill the following services:

- Hardware, System and Module Access.
- Mathematics functions.
- etc.



Most functions libraries are firmware extensions and must be loaded into the PLC by the user.

f) System Tasks

System tasks are set up by the system or application programs.



System tasks are firmware extensions and must be loaded into the PLC by the user.

3.3 APPLICATION PROGRAMS

Application programs are managed by the operating system. Since the PLC SW is a multitasking operating system, multiple application programs (called "tasks") can be run on the CPU simultaneously. The available processor power is divided amongst the individual application programs by "task switching".

4 APPLICATION PROGRAMS

Application programs can be basically split into two groups: **cyclic tasks** (also called PLC tasks) and **non-cyclic tasks** (also called idle time tasks). PLC tasks are cyclically executed and non-cyclic tasks are only executed if both the operating system and also all PLC tasks require no CPU time.

4.1 CYCLIC TASKS

Cyclic tasks (PLC tasks) are typical PLC applications and are executed exactly once within a definable time (=cycle time). The PLCSW monitors if the cycle time is kept to. If a cycle time violation occurs, an exception is triggered. By default, an emergency stop of the system is done (the PLC goes into SERVICE mode). It is also possible to handle an exception in an application way by using an exception task (you can find detailed information about this in the section "Monitoring Functions of the Operating System").
To assign priorities, you can put tasks in different task classes (different cycle times). Tasks that perform important and time critical activities are put in a task class with a short cycle time and are therefore executed more often than tasks in task classes with a longer cycle time.

4.2 NON-CYCLIC TASKS

Non-cyclic tasks (idle time tasks) run without monitoring over the cycle time. They use the so-called idle time of the system. That means that non-cyclic tasks are only executed if both the operating system and the PLC tasks don't need any CPU time. Typical idle time tasks are statistical programs running in parallel to cyclic applications or activities for certain events that occur seldomly, e.g. a report printout from a printer.

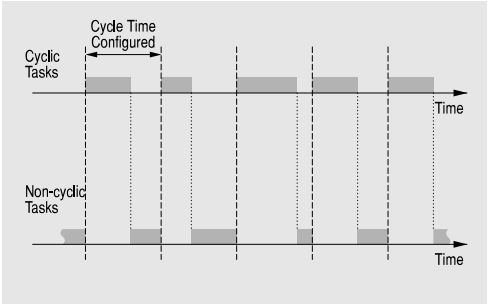


Idle time tasks can only be created with the C-Toolbox.

The simplified picture to the right should clarify the principle of **idle time** ! The CPU power required by the operating system has be left out of this diagram to keep things simple !



From this it is clear that cyclic tasks have a higher priority than non-cyclic tasks. PLC tasks interrupt the processing of idle time tasks.



The idle time is divided up among all existing non-cyclic tasks. As well as user created idle time tasks, there are also non-cyclic operating system routines (e.g. downloading tasks) and other tools (e.g. source level debugger).

5 TASK CLASSES

In order to set the priority of the individual PLC tasks with respect to one another, you can put tasks in different tasks classes. You can also set a different cycle time for each task class. All tasks in the task class are executed once within this time. Tasks that perform important or time critical duties are put in a task class with a short cycle time. These tasks are then executed more often than tasks in a task class with a longer cycle time.
If a large number of tasks have to processed in a task class with a short cycle time, this can cause the system to be overloaded. This can be avoided by correctly putting tasks in the correct task classes (you can find more detailed information in the section "System Loading"),
You have two different types of task classes: **normal task classes** and **high speed task classes**. Normal task classes are activated by the system manager (operating system) whereas high speed task classes are activated by a hardware timer.

5.1 NORMAL TASK CLASSES

Switching between tasks within normal task classes is done by the system manager.
The CPU module of the B&R SYSTEM 2000 offers you up to four normal task classes. You can set the cycle times of these task classes with the PLC Configurator or with the PG2000 Programming System (from version 2.10) from 10 msec. to 5 secs. in 10 msec. steps (refer to the relevant chapter for more information). The default settings of the normal task classes are listed in the following table:

Task Class	Abbreviation	Cycle Time (Default)	Processor
Task Class 1	TC#1	10 msec.	B&R 2010 and B&R 2005
Task Class 2	TC#2	50 msec.	
Task Class 3	TC#3	100 msec.	
Task Class 4	TC#4	10 msec.	

If, for example, the cycle time for the task class TC#1 is set to 10 msec. (default), this means that **each** task in TC#1 is executed once on the PLC **every** 10 msec.

5.2 HIGH SPEED TASK CLASSES (HS TASK CLASSES)

Tasks in high speed task classes are not called by the operating system, but via individual hardware timers (interrupts). Through this, they have the following differences to normal task classes:

- The time consistency of these tasks is better, since a hardware interrupt is more exactly executed than the switching by the system manager (due to the different monitoring activities, etc., the system manager has slight variations).
- The priority of the HS task classes is higher that the system manager and the normal task classes. HS task classes interrupt other CPU activities at an exact time and can only be interrupted by higher priority interrupts (e.g. higher priority HS task classes or interrupts from an I/O module).

You can set the cycle time of HS task classes can with the PLC configurator or the PG2000 Programming System (from Version 2.10) from 1 to 20.0 msec. in 0.5 msec steps (more information in the corresponding chapter). The B&R 2010 CPU modules offer 4 HS task classes and the B&R 2005 CPU module offers one HS task class. The default settings of the HS task classes are listed in the table below:

Task Class	Abbreviation	Cycle Time (Default)	Processor
HS Task Class 1	HS#1	3000 µsec	B&R 2010 and B&R 2005
HS Task Class 2	HS#2	5000 µsec	B&R 2010
HS Task Class 3	HS#3	7000 µsec	
HS Task Class 4	HS#4	9000 µsec	

HS task classes share there task class global data area with TC#1. This allows for faster access times and there is an easy way to exchange data between TC#1 and the HS task classes as well as between HS task classes,



If one or more HS task classes are configured, TC#1 must be configured (with PLC Configurator or PG2000).

5.3 TASK CLASS PRIORITY

To correctly share the processor power you assign task class priorities (higher priority task classes can interrupt lower priority task classes). The task class priority does not depend on the cycle time set but on the task class number as follows:

Priority	Task Class
high low	HS#1 HS#2 HS#3 HS#4 TC#1 TC#2 TC#3 TC#4

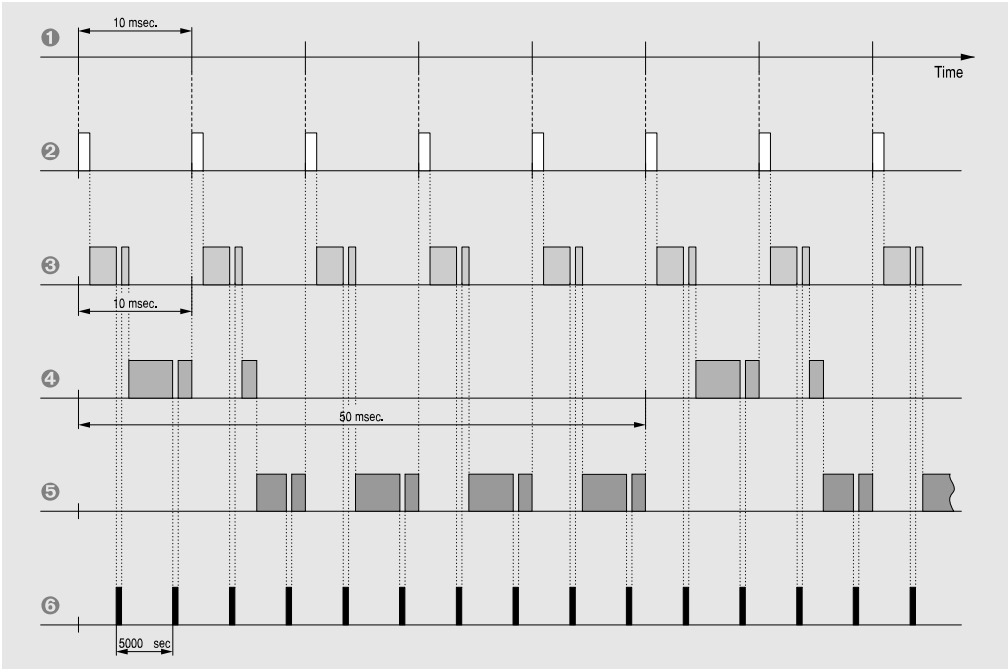
When you are changing the configuration, please take careful note of the individual task class priorities.

In general, the cycle time should be set so that it matches the priority (high priority, shorter cycle time). Otherwise cycle time violations can easily occur.

Both normal and high speed task classes can be interrupted by exception and interrupt tasks. You can find detailed information on exception and interrupt tasks in the sections "Exception Task Class" and "Interrupt Task Class".

5.4 CYCLIC RUN-TIME BEHAVIOUR OF THE TASK CLASSES

The execution order of the task classes is set by the task class priority (lower priority task classes are interrupted by higher priority task classes). The following diagram shows the basic run-time behaviour of task classes:



❶ Clock Tick (fixed):
10 msec operating system clock

❷ Operating System
(System manager)

❸ Task Class 1 (cycle time: 10 msec)
Task run-time: approx. 3 msec

❹ Task Class 2 (cycle time: 50 msec)
Task run-time: approx. 6 msec

❺ Idle Time Tasks
Use sremaining time

❻ HS Task Class 2 (Cycle time: 5000 µsec)
Task run-time: 500 µsec

Notes on the "Cyclic Run-time Behaviour of the Task Classes":

- 10% of the processor power is reserved for the operating system.
- The diagram shows an ideal task switching since the run-time of PLC tasks is not always constant due to program parts only being run depending on certain conditions.
- Cycle monitored multitasking guarantees that all tasks in a task class are executed once every cycle. In the previous example, all tasks in task class 1 were executed only once every 10 msec. You can set the execution order of the tasks within a task class with the PG2000 Programming System.

5.5 EXECUTION ORDER OF TASKS WITHIN A TASK CLASS

You can set the execution order of tasks within a task class using the PG2000 Programming System with the task parameter *Inst Number* (See section "GDM - Graphical Design Method" in the chapter "PG2000 Programming System"). The Inst Number defines the relative ordering of tasks to each other, their numerical value can be from 0 to 128 and is interpreted as follows:

- Tasks with the lowest "Inst Numbers" are processed first.
- Tasks with the same "Inst Number" are put in the correct order in reference to the other tasks. The order of these tasks is not set.
- Tasks with the Inst-Number 0 (⇒ “don't care”) are executed sometime within the cycle.

Example: The following tasks exist within a task class:

Task	A	B	C	D	E	F	G	H	I
Inst Number	0	0	1	10	25	25	25	3	128

The tasks are executed in the following order within a cycle:

Order	**Task**
1.	C
2.	H
3.	D
4.	E/F/G
5.	I



The order of the tasks E/F/G is not fixed. However there are all run between tasks D and I each cycle.
The order of the tasks A and B is not set. They are executed sometime within each cycle.

5.6 SYSTEM LOADING

By using the B&R multitasking, linearly programmed applications can be taken apart into smaller tasks and put into their correct task class depending on their speed requirements. Through purposeful choice of the required task classes, you can optimize the loading of your system.
In order to calculate the system loading, you measure the run-time of all the PLC and system tasks using the Profiler. After this you can calculate how much each task loads your system with the following formula:

$$\text{Loading [\%]} = 100 * \frac{\text{Task Runtime[msec]}}{\text{Cycle Time [msec]}}$$

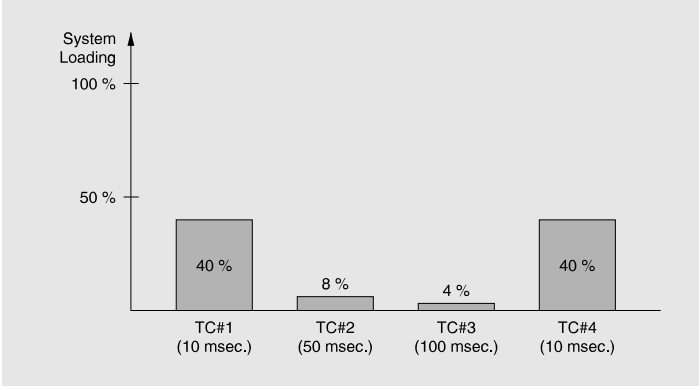
If you use this formula on all cyclic tasks and calculate the total, this results in the system loading of the CPU by all cyclic tasks.

Please note the following:

- Add approx. 10% to the total system load for the operating system.
- The I/O data transfer is performed by the I/O Processor (except on the Main Base Plate of a B&R SYSTEM 2005). This means that the CPU is not loaded with I/O transfer (You can find more detailed information in the section "I/O Data Transfer").
- When assigning tasks in individual task classes, please allow enough idle time for the non-cyclic tasks. The more the system is loaded by cyclic tasks, the less computing time is available for the non-cyclic tasks (communication).
- When the system is overloaded, a cyclic time violation is reported (You can find more detailed information in the section "Monitoring Functions of the Operating System).

The following example should help to show how a well though out and correctly divided up application can optimally use processor power.

Example: An application part has an average run-time of 4 msec (measured with the Profiler) loads the system differently depending on which task class it is assigned (values in brackets are the corresponding default cycle times):



6 EXCEPTION TASK CLASS

Exceptions are fatal errors that occur during PLCSW run-time and cannot be corrected by the operating system. By default an emergency stop of the system is performed (PLC goes into SERVICE mode) when an exception occurs. As well as the error reporting in the error module, the error is also shown on the status display of the B&R SYSTEM 2010 CPU.

As opposed to other errors that occur, the user has the possibility, not only to recognize that an error or exception occurred, but also to react to it. The operating system uses a so-called **Exception Handler** so that the most common exceptions can be handled in a user specific way.



The exception handler (B&R-Module EXERMO.BR) is a firmware extension and must be downloaded by you into the User ROM or FIX RAM.

The following exceptions are supported by the exception handler EXERMO.BR V1.12:

Exception Number (decimal)	Exception Name		Exception Type
	German	English	
2	Busfehler	Bus Error	Processor Exceptions
3	Adreßfehler	Address Error	
4	Illegaler Befehl	Illegal Instruction	
5	Division durch Null	Divide Zero	
6	Range Overflow	Range Overflow	
7	Null Pointer	Null Pointer	
8	Privilegverletzung	Privilege Violation	
10	Nicht implementierter Befehl	Unimplemented Instruction	
24	Falscher Interrupt	Spurious Interrupt	
128	IO-Exception	IO Exception	I/O Bus Exceptions
144	TK-Zykluszeitverletzung	TC Cycle Time Violation	Cycle Time Violations
145	TK-Max. Zykluszeitverletzung	TC Max. Cycle Time Violation	
146	TK-Input-Zykluszeitverletzung	TC Input Cycle Time Violation	
147	TK-Output-Zykluszeitverletzung	TC Output Cycle Time Violation	
160	SSTK-Max. Zykluszeitverletzung	HSTC Max. Cycle Time Violation	
161	SSTK-I/O-Zykluszeitverletzung [1)]	HSTC I/O Cycle Time Violation [1)]	
162	Systemzykluszeitverletzung	System Cycle Time Violation	

For each exception that is supported by the exception handler, you can create an **exception task**. This task is executed immediately after the corresponding exception occurs which is in effect the user specific exception handling. The exception task class has the highest priority, i.e. when an exception occurs, the respective exception task interrupts both normal and high speed tasks as well as interrupt tasks.



Before you install an exception task, you have to setup the exception task class (EXC task class) with the PLC Configurator .

[1)] The exception "HSTC-I/O Cycle time violation" is only supported from BUGFIX V1.11 (B&R Module BFX_V111.BR) of the operating system PLCSW Version 1.10.

Exception tasks can be created with the PG2000 Programming System, just like cyclic tasks. The task class has to be set to "EXC". The connection between the exception and the corresponding task is done via the task parameters. Where the Inst Number is usually entered, the exception number is entered (see table).

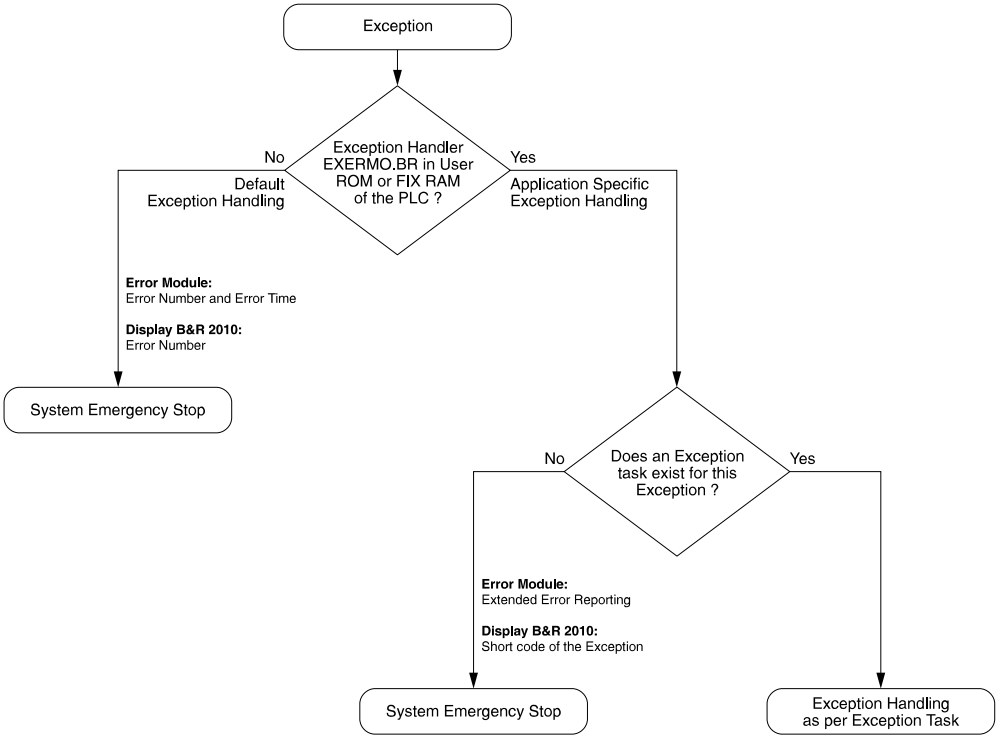
Only one exception task can be created for each exception supported by the exception handler.

The EXC Task class shares its task class global data area with TC#1. This allows for faster access times and provides a simple method (without PLC global variables) of exchanging information between TC#1 and the EXC task class.

If an EXC task class is configured with the PLC Configurator, the TC#1 must also be configured due to the common data area.

All exceptions that have **no exception task** assigned to them, the exception handler does extended reporting in the error module (turn to the chapter "PG2000 Programming System"). With the CPU module of the B&R System 2010, a short code of the exception is shown on the status display. Following this, an emergency stop of the system is done (the PLC goes into SERVICE mode). You can find detailed information on the exceptions supported by the exception handler as well as the display in the section "Error Handling".

Exception Handling:



7 INTERRUPT TASK CLASS

Interrupts are asynchronously occurring (hardware triggered) events that interrupt the cyclic program processing. Interrupt triggering of interrupts is only possible with special hardware. Interrupt capable modules are e.g. the digital input module DI400 and DI401 (System B&R 2010). Up to 16 digital channels on these modules can be used and CSI inputs (CSI = change of state interrupt).

You can create an interrupt task for each interrupt capable module. This task is executed immediately after the corresponding interrupts occur. The interrupt task class has the second highest priority, i.e. and interrupt task can be interrupted by an exception task.



Before installing interrupt tasks, the interrupt task class (IRQ task class) has to be setup with the PLC Configurator.

Interrupt tasks can be created with the PG2000 Programming System, just like a cyclic task. It is put in the "IRQ" task class. The connection between the interrupt and the appropriate task is done via the task parameters, where the *Inst Number* entry is usually given, the module address is to be typed in (address of the module that generates the interrupt).
If an interrupt capable module releases an interrupt that has no corresponding interrupt task, this interrupt is ignored. This means the cyclic system is not interrupted in this case.



Only one interrupt module can be created per interrupt capable module.
Since interrupt tasks are called asynchronous to the cyclic system, they can disturb the system when the frequency of the interrupts is to high or the system is already heavily loaded. If a part of the application is already running in a fast enough task class (e.g. HS#1), it is usually more efficient to check the CSI input in this task class, for the reasons previously given.

The IRQ task class shares its task class global data area with TC#1. This allows faster access times and provides a simple method of exchanging data between TC#1 and the IRQ task class.



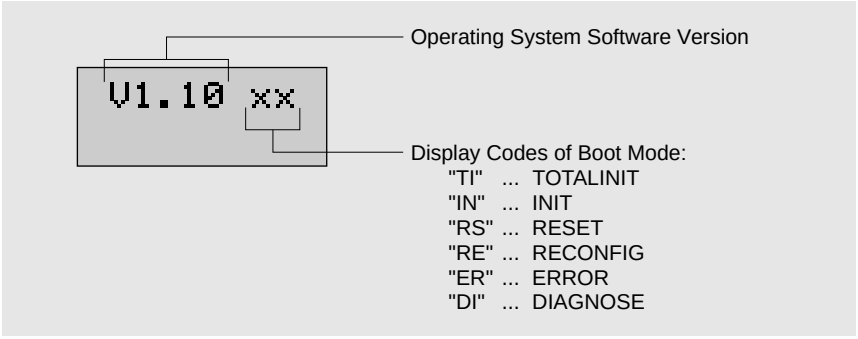
If an IRQ task class is configured with the PLC configurator, TC#1 has to be configured to because of the common data area.

8 BOOT MODES

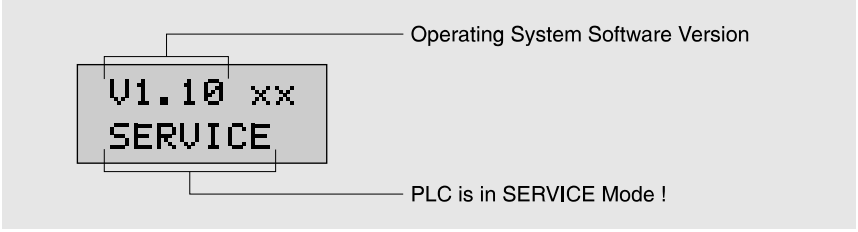
A boot is performed by either you, the user (e.g. with PG2000 or with the button on the CPU module) or by a fatal error. The CPU module if the B&R SYSTEM 2000 has the following bootmmodes:

- **TOTALINIT** [1)]
- **INIT**
- **RESET**
- **RECONFIG**
- **ERROR**
- **DIAGNOSE**

After booting, a two letter code for the boot mode is shown on the display of the B&R 2010 CPU module. Following a boot the PLC goes into a certain mode depending on several factors (e.g. boot mode, key switch, etc). One possible mode is **SERVICE Mode**. If an error occurs during boot-up, SERVICE mode is activated. After



booting in RESET, ERROR or DIAGNOSE mode, the PLC always goes in SERVICE mode. In SERVICE mode, all application programs are inactive and only the PLCSW runs.
If the PLC is in SERVICE mode, the display on the B&R 2010 CPU module looks like this:



[1)] This term for the boot mode is used from version 1.10 of the operating system PLCSW. In older versions of the PLCSW, the terms FIRST INIT or URINIT were used.

The Status LEDs on the CPU moulde are also set in SERVICE mode:

CPU Module of the B&R SYSTEM 2010

Status LED	PLC in SERVICE Mode (Application stopped)	Application running
“RUN”	LED off	LED on
“READY”	LED off	LED on
“ERROR”	LED on	LED off

CPU Module of the B&R SYSTEM 2005

Status LED	PLC in SERVICE Mode (Application stopped)	Application running
“RUN”	LED off	LED on
“READY”	LED on	LED off
“ERROR”	LED on	LED off

8.1 TOTALINIT

The TOTALINIT boot mode corresponds to the **FIRST initialization of the CPU**. The system is completely re-started, allocated memory is released and the system structures (all operating system information for orderly functioning of the CPU) is re-created. With a TOTALINIT, all system modules (Remote Master, Network Modules, etc.) are recognized and initialized. All the modules saved in the User RAM are deleted (except for those in FIX RAM) and all process variables (inputs, outputs and internals) are initialized with zero.

After a TOTALINIT, all tasks in the **User ROM or FIX RAM** are started. This includes running all attached Initialization sub-programs (INIT-SPs). You can find information about the execution order of the INIT-SPs after a TOTALINIT are in the section "Start Behaviour of Tasks after a TOTALINIT/INIT.

A TOTALINIT is triggered, if ...

... a fatal error occurs while testing a new PLC configuration module (see PLC Configurator).

... the APM is exchanged and something has changed (e.g. operating system).

... the standard function "SYSreset" (BURTRAP library) is called with mode "TOTALINIT".

... you press the reset button after you have already pressed it onceand the CPU is in the middle of booting up.

... you select boot mode TOTALINIT with the boot keys on the B&R 2010 CPU module (the key switch must not be in the RUN position). The function of the CPU operator elements (Reset key, Boot keys, etc.) are described in the "B&R 2000 Hardware User's Manual".

... you press the TOTALINIT key on the B&R 2005 CPU module (the key switch must not be in the RUN position).

... you activate the "Total Init" function in the PG2000 Programming System (see Menu option “Other”).

Since a TOTALINIT erased all RAM data (except modules in FIX RAM), you should only trigger a TOTALINIT during the programming phase (not in normal PLC operation!) or after plugging in a new system module (New system modules are only recognized and initialized with a TOTALINIT)

In the sections "Power on Behaviour" and "Boot-up Behaviour", the boot mode TOTALINIT is shown as the following symbol:



If the **Key Switch is in the Service position** with a TOTALINIT, the TOTALINIT is executed as described above. However, after a TOTALINIT the PLC goes into SERVICE MODE, i.e. the tasks in the User ROM and FIX RAM are not started. For this case, the following symbol is used in the "Power on Behaviour" and "Boot-up Behaviour" sections:



8.2 INIT

With an INIT, all battery backed RAM data (all tasks in the User RAM, values of the inputs, outputs and internals, etc) is kept. All task also keep their status (task active or inactive) they had before the INIT, after an INIT. Newly plugged in system modules (Remote Master, Network Modules, etc.) are not recognized with an INIT. A TOTALINIT must be done to initialize system modules.

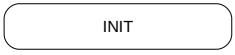
After an INIT, all tasks that **were active before the INIT** are started. This includes running the corresponding INIT-SPs (also the INIT-SPs of inactive tasks). You can find more information about the execution order of INIT-SPs after an INIT in the section "Start Behaviour of tasks after an INIT/TOTALINIT".

An INIT is triggered, if ...

- ... the PLC boots up after a power loss or power on.
- ... the standard function "SYSreset" (BURTRAP library) is called with the "INIT" mode in a task.
- ... you turn the key switch from the Service position to the Program position.
- ... you select the boot mode INIT with the boot keys on the B&R 2010 CPU module (the key switch must not be in the run position).
- ... you activate the function "Init" in the PG2000 Programming System (see Menu option "Other").

You would trigger an INIT yourself when, for example, you want to restart your application after a serious error has occurred.

In the sections "Power on Behaviour" and "Boot-up Behaviour", the boot mode INIT is shown as the following symbol:



If the**key switch is in the Service position** with a INIT, the INIT is executed as described above. However, after the INIT, the PLC goes into SERVICE MODE, i.e. the tasks in the User ROM and FIX RAM are not started. For this case, the following symbol is used in the "Power on Behaviour" and "Boot-up Behaviour" sections:



8.3 RESET

With an RESET (similar to INIT), all battery backed RAM data (all tasks in the User RAM, values of the inputs, outputs and internals, etc) is kept. All task also keep their status (task active or inactive) they had before the RESET, after an RESET. After a reset the PLC goes into SERVICE mode, i.e. application programs in the User ROM, User RAM and FIX RAM are not started. Only the PLCSW runs in SERVICE mode, all applcation programs are inactive.

A RESET is triggered, if ...

... the hardware watchdog does not report to the operating system within the 200 msec cycle (you can find more information in the section "Monitoring functions of the Operating System").

... you turn the key switch on the CPU module from the "Program" position to the "Service" position.

... you press the reset key on the CPU module (hardware reset).

... you activate the funtion "Reset" in the PG2000 Programming System (see menuoption "Other").

In the sections "Power on Behaviour" and Boot Up Behaviour", the boot mode RESET is shown as the following symbol:



8.4 RECONFIG

Using the PLC Configurator or PG2000, you can change the settings in the PLC Configuration module (e.g. size of the FIX RAM, number of normal and high speed task classes, etc.) and therefore configure the operating system PLCSW specifically for your application. There is a way to to prevent a incorrect PLC configuration module that may make the operating system not longer executable from being burned in the User ROM. You can download the new PLC configuration module into the User Ram with the trialing function "Test". This makes an erroneous attempt easy to remove. After downloading the new PLC configuration module into the User RAM, the PLC automatically boots up in RECONFIG mode.
The RECONFIG mode is a special type of TOTALINIT. As opposed to TOTALINIT, a boot in RECONFIG mode uses the new PLC configuration module in the User RAM not the one in the User ROM.
If a severe problem occurs with the new PLC configuration module, a TOTALINIT is triggered and the boot up uses the original configuration module in the User ROM. The cause of the error can be worked out by looking in the error module.

The PLC boots in RECONFIG mode, if ...

... you download a new PLC configuration module into the User RAM to test.

The boot mode RECONFIG is shown as the following symbol in the sections "Power On Behaviour" and "Boot Up Behaviour":



8.5 ERROR

If a fatal error occurs during boot-up or during runtime operation, the error is entered in the error module. The error number of fatal errors is also shown on the display of the B&R 2010 CPU module. Following this, the PLC boots automatically in ERROR mode and goes into SERVICE mode.
The boot mode ERROR corresponds to the boot mode RESET. Also after an ERROR mode boot, the error number of the fatal error is shown on the display (only B&R 2010) again. In this way, the cause of the error can be found not only from the error module, but also from the CPU display.

The PLC boots in ERROR mode if ...

... a fatal error occurs during boot-up or in cyclic operation.

In the sections "Power on Behaviour" and Boot Up Behaviour", the boot mode ERROR is shown as the following symbol:



8.6 DIAGNOSE

In boot mode DIAGNOSE, the PLC only boots up with the operating system softwware, i.e. **all application programs are ignored** and are therefore inactive. The PLC goes into SERVICE mode after the DIAGNOSE boot up.



After a DIAGNOSE boot-up, the PLC can only be booted with a TOTALINIT or in DIAGNOSE mode once again (an INIT boot-up is not possible).

Since all modules in User ROM and FIX RAM are installed with a TOTALINIT, a defect module in one of these memory areas can prevent a correct boot-up in TOTALINIT mode. In order to avoid having to delete the entire User ROM and/or FIX RAM, you can boot the PLC in DIAGNOSE mode. After you have done this, you can use the PLC Configurator function *Diagnose* to display existing B&R modules and **delete them singly** if necessary.

The PLC boots-up in DIAGNOSE mode if ...

... you press and hold both boot keys simultaneously on a B&R 2010 CPU while you press the reset key (the key switch must not be in the Run position).

... you, on a B&R 2005 CPU, press the Reset and TOTALINIT keys simultaneously and hold the TOTALINIT key for 5 seconds after you let go of the Reset key until onle the LEDs "READY" and "ERROR" are still lit (the key switch must not be in the Run position).

... you activate the "Diagnose" in the PG2000 Programming System (see menu option "Other").

In the sections "Power on Behaviour" and Boot Up Behaviour", the boot mode DIAGNOSE is shown as the following symbol:



8.7 SERVICE

The SERVICE mode is not a Boot mode, it is a possible result of a boot up. In SERVICE mode, only the PLC SW runs. All application porgrams are inactive.
If an INIT is done in SERVICE mode, the PLC boots up in INIT mode. This ends SERVICE mode and all tasks that were active before SERVICE mode are started. After the INIT, all tasks have the same state (active or inactive) as before the service mode. This means that if the PLC is, for example, in SERVICE mode after a fatal error, the application can be started again with a boot-up in INIT mode.

The PLC goes into SERVICE mode , if...

... an error occurs during boot-up.

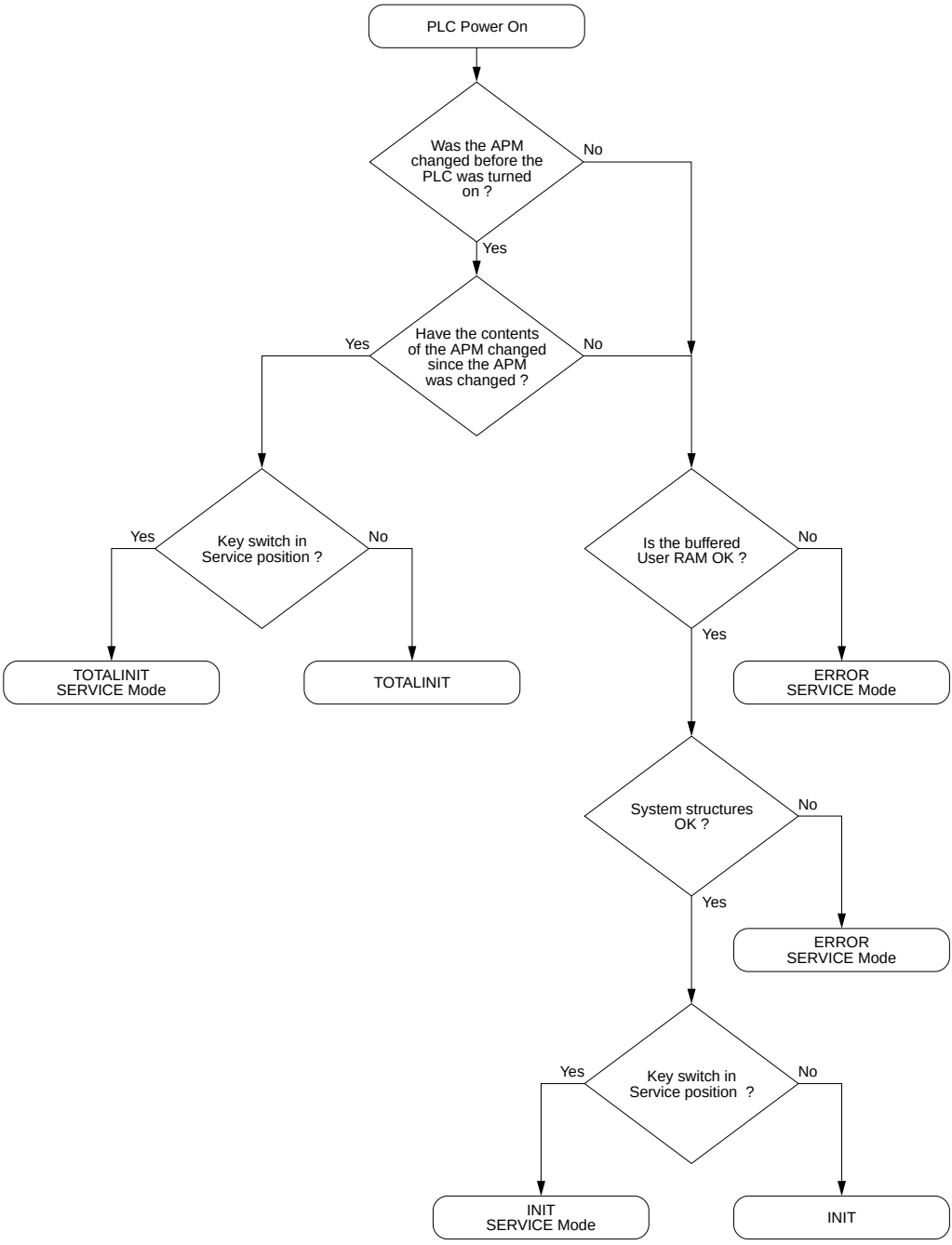
... the key switch is in the Service position at boot-up.

... the PLC is booted up in RESET, ERROR or DIAGNOSE mode.

In the sections "Power on Behaviour" and Boot Up Behaviour", the boot mode SERVICE is shown as the following symbol:



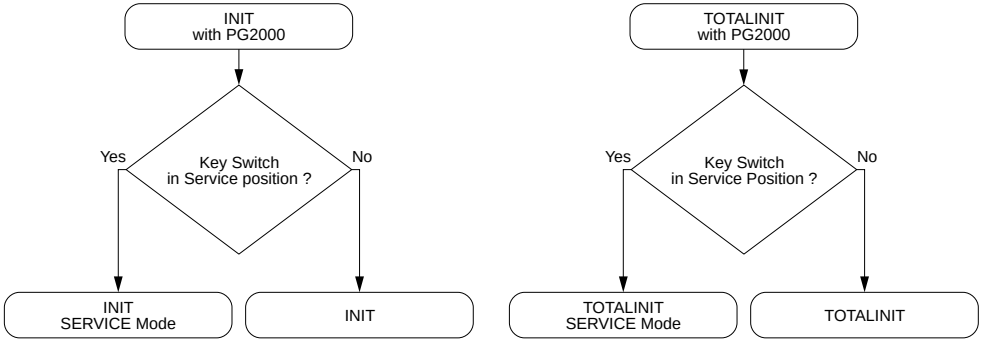
9 POWER ON BEHAVIOUR



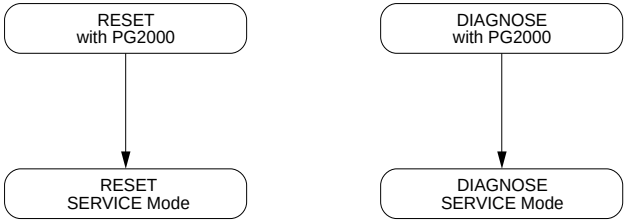
10 BOOT BEHAVIOUR

10.1 RESET WITH PG2000

INIT or TOTALINIT with PG2000

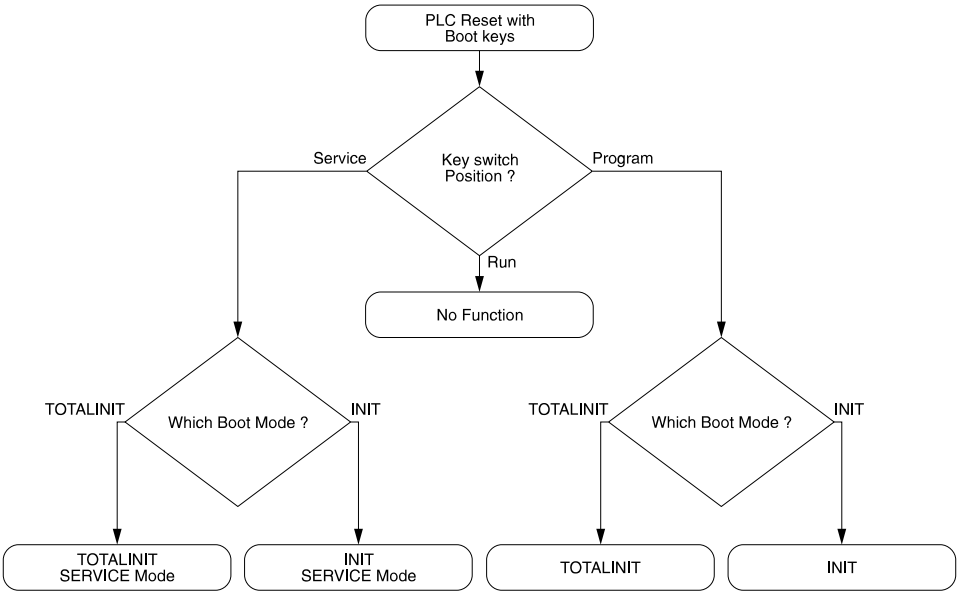


RESET or DIAGNOSE with PG2000

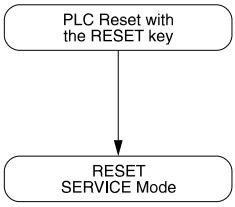


10.2 RESET WITH THE OPERATION ELEMENTS ON THE B&R 2010 CPU MODULE

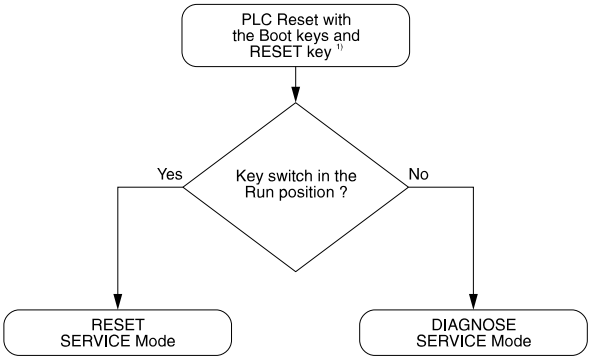
PLC Reset with Boot Keys



PLC Reset with the Reset key



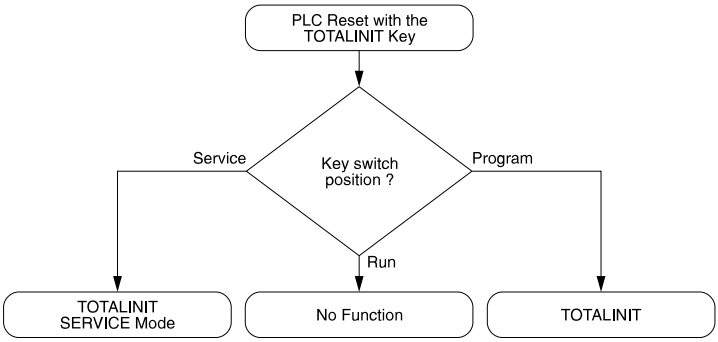
PLC Reset with Boot and Reset Keys



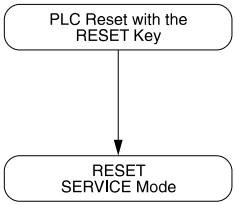
1) Press and hold the boot keys simultaneously and then press the reset key.

10.3 RESET WITH THE OPERATION ELEMENTS OF THE B&R 2005 CPU MODULE

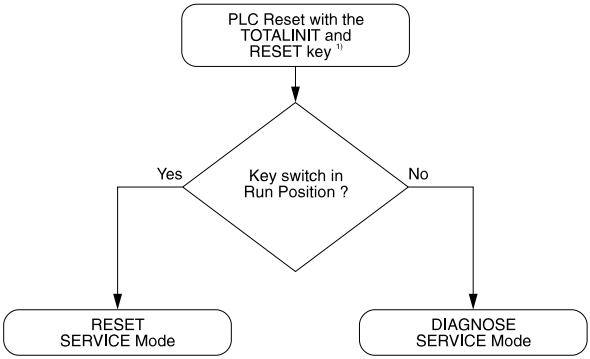
PLC Reset with the TOTALINIT key



PLC Reset with the Reset key



PLC Reset with the TOTALINIT and Reset keys

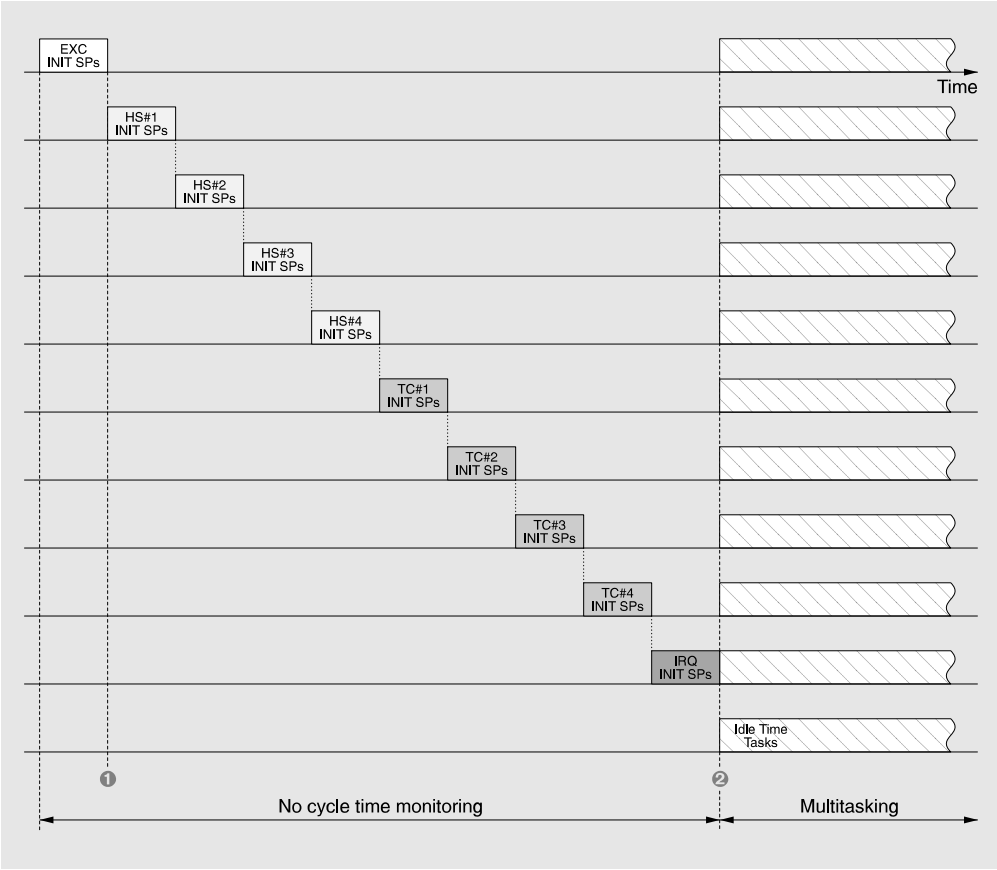


1) Press Reset and TOTALINIT keys simultaneously and then hold the TOTALINIT key 5 seconds after letting go of the Reset key, until onle the LEDs READY and ERROR are lit.

11 START BEHAVIOUR OF TASKS AFTER A TOTALINIT/INIT

A so-called Initialization Sub-program (Init-SP) can be created for each task (normal, high speed, interrupt and exception tasks) with the PG2000 Programming System. Init-SPs are **only executed in the boot-up of the PLC** (after a **TOTALINIT** or **INIT**) or **after downloading a task**. They are generally used for value initialization.

The following diagram shows the execution order of the tasks (Init-SPs) on a TOTALINIT/INIT:



1. Release EXC Task Class
2. Release the IRQ task class, release the high speed and normal task classes and activate cycle time monitoing.

Explanation to the diagram "Start Behaviour of the Tasks after a TOTALINIT/INIT":

1) After a TOTALINIT or INIT all INIT-SPs of the execption tasks (EXC tasks) are executed. Immediately after, the EXC task class is released(❶). From now on, all exceptions are handled with the optionally available user exception handler.

2) Next, all Init-SPs of high speed and normal task classes are executed in the order shown on the previous diagram (HS#1, HS#2, HS#3, HS#4, TC#1, TC#2, TC#3, TC#4). The order of the Init-SPs within the task classes is set with the task parameter Inst-Number (you can find more detailed information in the section "Execution Order of Tasks within a Task Class").

3) After processing all the Normal Task Class Init-SPs, the Init-SPs of the interrupt tasks (IRQ tasks) are run. Directly afterwards, the IRQ task class is released (❷). From now on, all interrupts are handled by the interrupt tasks.

4) Only after all the Init-SPs have been run, then multitasking operating starts. This guarantees that all Init-SPs are run before the first PLC, idle time or interrupt task starts.



Since exceptions could occur during the processing of the Init-SPs, the exception task class is released as soon as possible (❶). This allows exceptions to be handled during boot-up in a user specific way.



Initialization sub-programs run **without cycle time monitoring**, i.e. they can take any amount of time.

12 I/O DATA TRANSFER

With the B&R SYSTEM 2000, a new trail has been blazed in the area of I/O data transfer. CPU modules in this contro generation work with a memory image of the I/O channels. In order to relieve the CPU of haveing to read the I/O image before each cycle and writing the I/O data back to the peripherals afterwards, ...

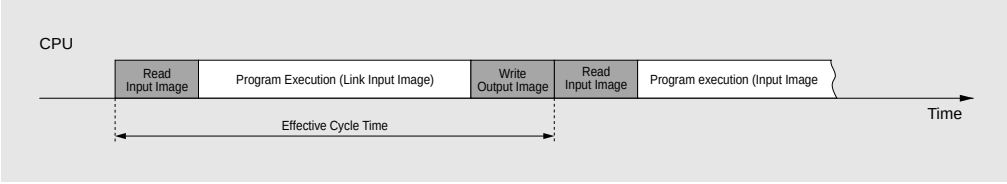
... the I/O bus is separated from the system bus (excpet on the main base plate of a B&R SYSTEM 2005).

... a separate I/O processor has been developed to take over the readign and writing of the I/O images.

Only one order is made by the CPU (operating system) to the optimized I/O processor. During the time that the I/O processor writes or reads the I/O image, the CPU is available for other jobs. This technique removes, for the first time, the disadvantage of a longer cycle time that plagued other PLCs with memory images. This allows for a system that has fast cycle times and efficient I/O transfer.

12.1 CONVENTIONAL I/O DATA TRANSFER

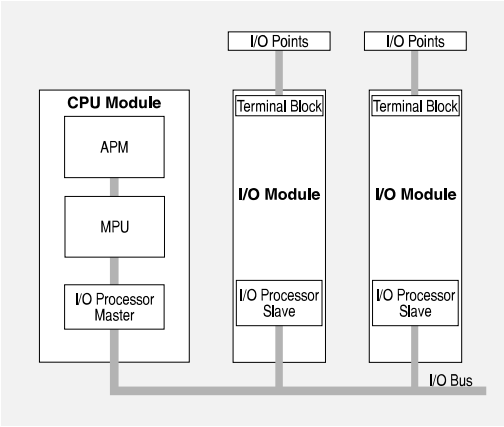
In conventional PLC systems, the CPU has to organize the input image itself. Also after the program has been processed, write the output image to the the peripherals. During the I/O transfer, the CPU cannot execute any application jobs.



12.2 I/O DATA TRANSFER WITH B&R SYSTEM 2010

The B&R SYSTEM 2010 has a bus system that is separated into system and I/O buses. The division of the bus systems provides the following advantages:

- In B&R 2010, reading and writing of I/O images is done by an component (I/O processor) optimized for this purpose. The CPU can completely concentrate on the linking of the I/O data and is not misused for data transfer.
- The data transfer on the I/O bus is protocol guaranteed. This guarantees that no incorrect I/O information is exchanged between the I/O Modules and the PLC CPU.
- The data throughput rate of the I/O bus is so high that even with a few thousand I/O points, time parallel creation of the I/O image is still guaranteed.

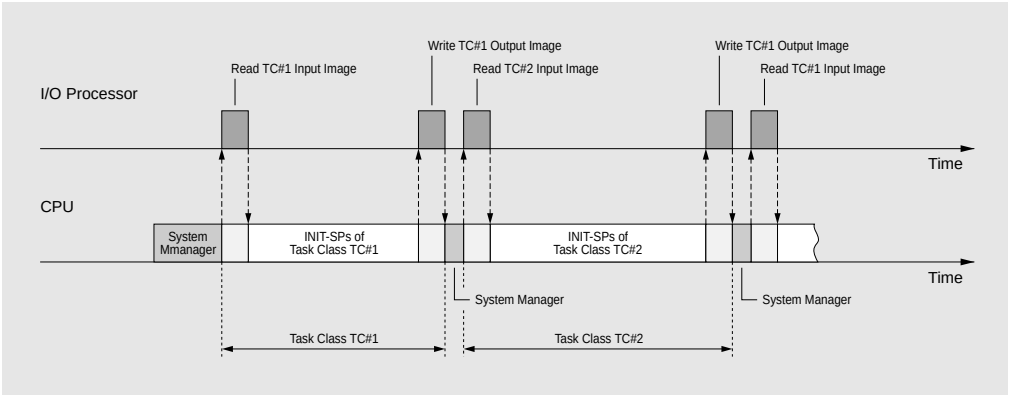


I/O-Data transfer in the B&R 2010 System

The transportation of I/O data is divided up into input and output images. Reading the input images or writing the output images is called "shovelling".
On the following pages, there is a description of the I/O data transfer for normal and high speed task classes. Also included are the differences between boot up phase and cyclic operation.

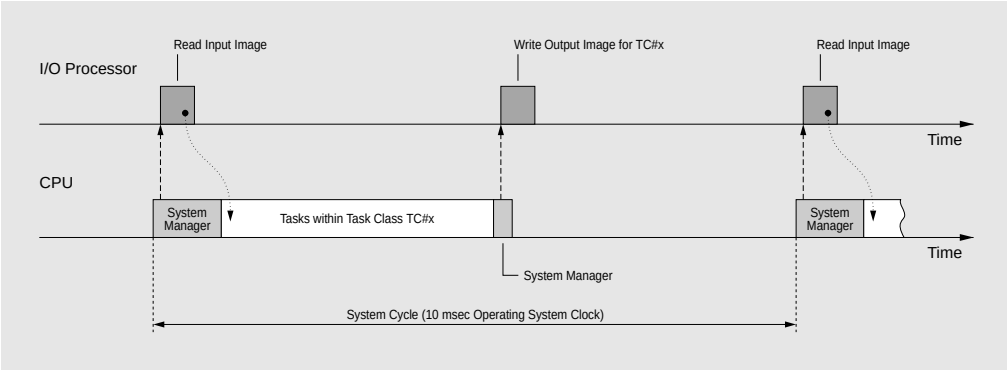
Normal Task Classes in the Boot-Up Phase

With normal task classes, input and output images are shovelled separately.



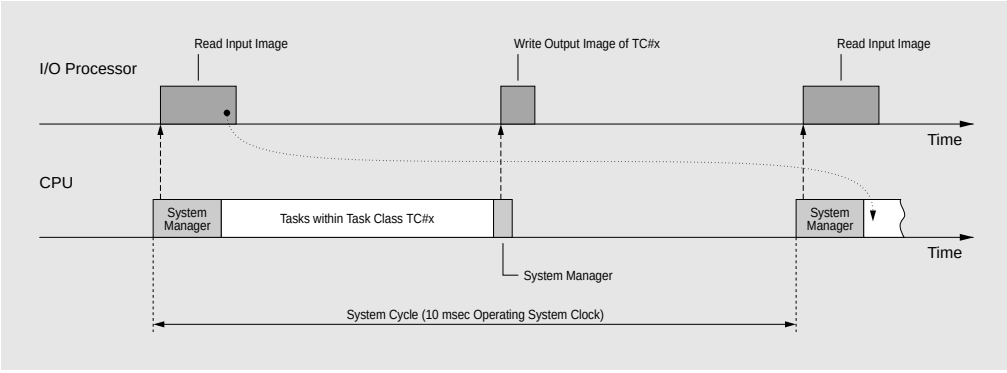
1) Before executing the first initialization subprogram of a normal task class, the output picture is ordered.
2) The first INIT-SP of a normal task class is started only when the input image for this task class is available.
3) If all task class INIT-SPs have been processed, the I/O processor writes the output image of the task class to the peripherals. When this process is finished, the next task class is activated.

Normal Task Classes in Cyclic Operation



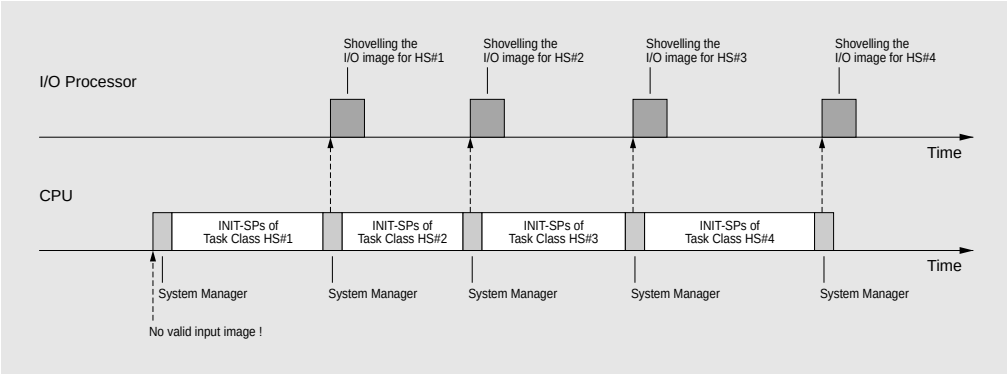
1) To start a new task class cycle, the operating system checks the I/O data transfer of the previous cycle. At the beginning of a task class cycle, the operating system checks if the input image ordered at the start of the previous cycle is ready. If this is not ready then a "TC Input cycle time violation" exception is set off. The same applies to the output image, but a "TC Output cycle time violation" exception is set off.
2) If both the input and output images of the previous cycle were correctly shovelled, the system manager orders the input images to be read by the I/O processor for all task classes that are to be started this cycle, i.e. if task classes 1 and 2 are to be started in this cycle, the input images are ordered for both these task classes.
3) Immediately before calling the first PLC task of a task class, the system manager checks whether the imput images ordered at the start of this system cycle are already available. Normally the current input images are already available and can be linked in by the CPU.
4) If all tasks in a task class have been executed, the I/O processor orders the output image for this task class to be written to the peripherals.

Special Case:
With large numbers of inputs, it is possibile that the input image ordered at the start of the system cycle is not available before the start of the first PLC task of this task class. In this case, the input image shovelled in the previous cycle is used (old input image) and the system flag PIO_OLD is set to "1" (system flags are predefined or known PG2000 flags).



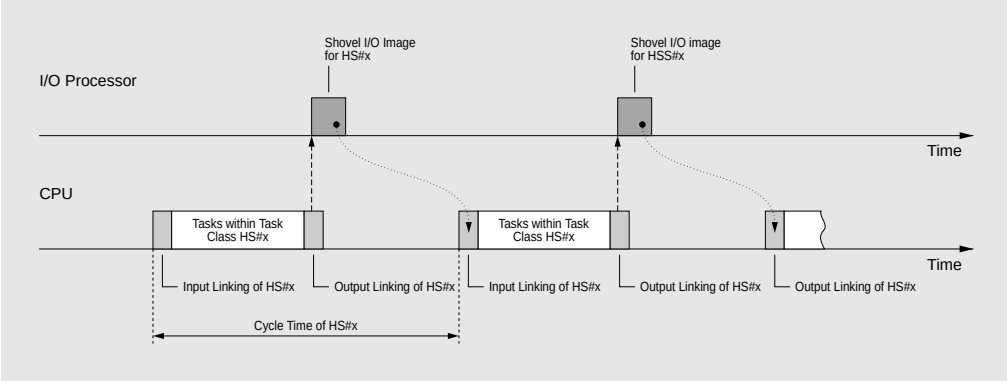
High Speed Task Classes in the Boot-Up Phase

With high speed task classes, input and output images are shovelled together (combined I/O image).



1) Due to the combined I/O image, there is no current input values available before the execution of the first initialization subprogram.
2) After executing the INIT-SPs of a HS task class, the system manager orders the I/O image for this HS task class.

High Speed Task Classes in Cyclic Operation

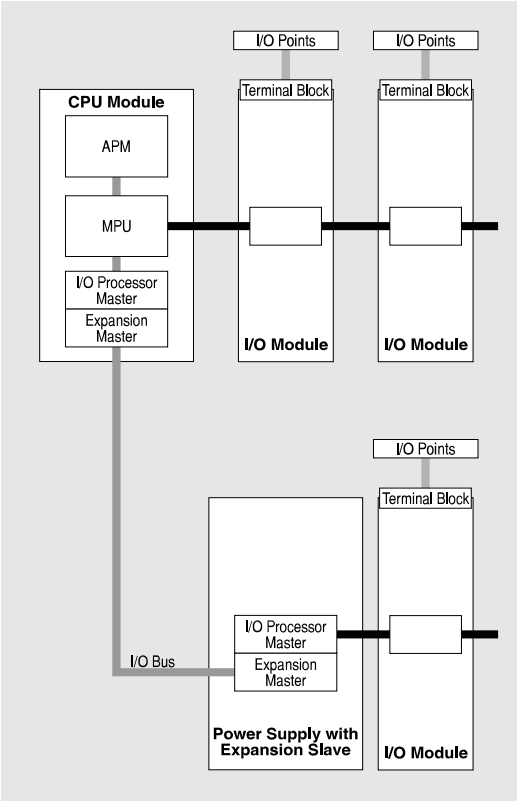


1) At the beginning of a HS task class cycle, the operating system checks whether the combined I/O image has been correctly processed. If the I/O image ordered at the end of the last HS task class cycle is not available yet, the exception "HSTC I/O cycle time violation" occurs.
2) Following this the tasks in the HS task class are executed. The input image from the last HS task class cycle is used.
3) After executing all tasks in the HS task class, the combined I/O image (HS task classes) is ordered by the operating system.

12.3 I/O DATA TRANSFER ON B&R SYSTEM 2005

On the B&R 2005, the system and I/O busses are not separated on the main base plate. This means that there is no I/O processor on the main base plate. The function of the I/O processor is emulated with software by the system. The data transfer to and from I/O modules on the main base plate costs time (like the conventional creation of images).

The data transfer between the CPU and the expansion base plates is done with additional I/O processors as previously described for the B&R 2010 system.



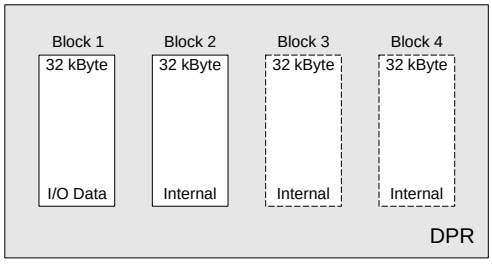
I/O Data Transfer on B&R SYSTEM 2005

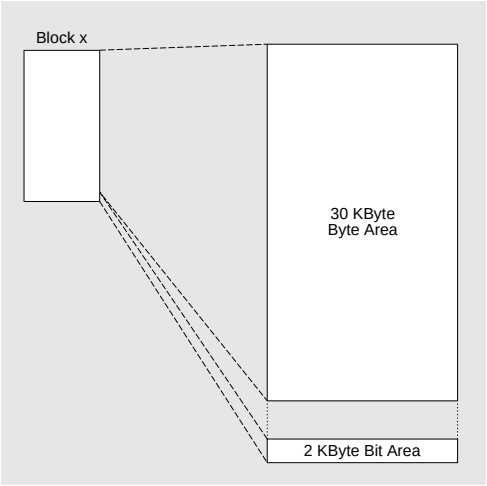
12.4 DATA EXCHANGE BETWEEN THE CPU AND I/O PROCESSOR

The data exchange between the CPU and I/O processor is done via the **D**ual **P**orted **R**AM (DPR). Both the CPU and the I/O processor can access this RAM. The CPU activates orders in the form of tables (called I/O or shovel tables) and the I/O processor finishes off the complete data transfer. The I/O processor takes over the preparation of all the data into the form used by the CPU.

Depending on the size (64 or 128 Kbyte), the DPR is divided into 2 or 4 blocks. Block 1 is reserved for I/O data. The remaining blocks are used for all PLC, task class global and/or local variables as well as function block structures.

Each block is divided into a bit and a byte area. The analog variables are stored in the byte area and the digital values are stored in the bit area.





The bit area of 2 Kbyte corrstponds to a memory area of 16 Kbit. The bit area is not directly addressed, but virtually arrdessed. Via these virtual addresses, you can access the individual bits in this 2 Kbyte range either normally or inverted.

With this technique, a byte area can be bitwise addressed and is therefore space savingly saved in memory. This bit/byte logic is realized as ASIC hardware-wise and replaces the usually practiced time consuming masking by the CPU. In spite of the memory saving, packed bit representation has no time overhead for linking of digital data.

12.5 MULTITASKING OF THE I/O PROCESSOR

The I/O processor works in multitasking operation like the main CPU in the CPU module. Shovel orders from tasks of higher priority have right of way over other shovel entries.

13 MONITORING FUNCTIONS OF THE OPERATING SYSTEM

The PLC software operating system provides different monitoring and safety functions that are **cyclically** performed. To guarantee a higher availability of the system:

There is a monitoring of ...
- ... the checksum of B&R modules
- ... system structures
- ... stack overflow
- ... I/O data transfer
- ... cycle times of the individual task classes
- ... the operating system via the hardware watchdog.

13.1 CHECKSUM OF B&R MODULES

Each software object on the PLC (system, PLC and idle time tasks) is called a B&R Module. All B&R modules are constructed in the same way which guarantees the management, identification and data security of the individual modules.
The checksum of the B&R module on the CPU is cyclically checked by the system manager (depending on the number of B&R modules, however at least once per minute).
This monitoring function ensures that there are no defective programs or data modules on the CPU.

13.2 SYSTEM STRUCTURES

The system structures of the operating system are cyclically monitored so that a disturbance by defective programs is recognized.

13.3 STACK OVERFLOW

Each task needs a stack (private memory area). The stack is used, for example, for exchanging data with subroutines (e.g. functions). Particular functions of the function libraries require the stack for their own internal calculations. If complex subroutines or stack intensive functions are called, it is possible that the stack is used up. This stack overflow is recognized by the operating system.
The size of the stack available to a task can be set with the PLC configurator for each individual task class.

13.4 I/O DATA TRANSFER

The data transfer that the I/O processor takes over from the CPU is monitored by the operating system (system manager).

13.5 CYCLE TIME MONITORING

During PLC operation, the system manager monitors whether the set cycle time is actually kept. A cycle time violation occurs if all the tasks in the task class cannot be executed within the cycle time. A cycle time violation can occur for various reasons:

1) The total run time of the tasks is longer than the cycle time set.

2) A higher priority task class is heavily loaded. This cuts down on the available processor time for the lower priority task classes.

If a cycle time violation occurs in a **high speed task class**, an exception is triggered.
For **normal task classes**, an **allowed cycle time violation** can be setup with the PLC configurator or PG2000 for tolerant cycle time monitoring. This allows a the cycle time to be exceeded by a preset amount (simple cycle time violation). However if this time is exceeded (maximum cycle time violation), an exception is triggered. By default, an emergency stop is performed (PLC goes into service mode). There is however the possibility to deal with an exception in an application specific way with an exception task.

13.6 HARDWARE WATCHDOG

The operating system has to report to the "hardware watchdog". If the operating system does not report with in 200 msec, a RESET is triggered. The CPU goes into SERVICE mode, all PLC and idle time tasks are stopped and all outputs are reset. This puts the PLC in a defined operation condition.

14 ERROR HANDLING

All errors that occur within the scope of an application (e.g. cycle time violation, bus error, etc.), are entered in the error module by the operating system with the time of occurrence and some additional information. The error module can be read with PG2000 or standard functions (BURTRAP library).

PLCSW errors are always reported with a number in the range from 0000 to 9999. The PLC error numbers are divided into several areas:

Error Number	Range
0000 - 1999	General Error
2000 - 2999	HW-Trap Library Error
3000 - 3999	SW-Trap Library Error
4000 - 4999	ALI-Trap Library Error
5000 - 5999	Task Class Error
6000 - 6999	System Task Error
7000 - 7499	Boot Error
7500 - 7999	Panel Trap Error
8000 - 8999	Installation Error
9000 - 9999	Miscellaneous Error

A list of all PLC error numbers and the corresponding error messages are in Appendix A "PLCSW Errors".

14.1 FATAL ERRORS

During run time of the operating system, fatal errors can also occur. These errors cannot be fixed by the operating system itself. If a fatal error occurs, an emergency stop of the system is preformed and the PLC goes into SERVICE mode. As well as the reporting in the error module, the error number is also shown on the display of the B&R 2010 CPU module.

14.2 EXCEPTIONS

By default, exceptions are treated like fatal errors. As opposed to other fatal errors, with exceptions the user has the possibility to recognize and react to this case. The operating system has an exception handler for this purpose. This allows the most frequently occurring exceptions to be handled in a user specific way.



The exception handler (B&R module EXERMO.BR) is a firmware extension and you have to download it into by the user ROM or FIX RAM yourself.

The following exceptions are handled by the B&R module EXERMO.BR (Version 1.12 or higher):

Exception Number (decimal)	Exception Name		Type of Exception
	German	English	
2	Busfehler	Bus Error	Processor Exceptions
3	Adreßfehler	Address Error	
4	Illegaler Befehl	Illegal Instruction	
5	Division durch Null	Divide Zero	
6	Range Overflow	Range Overflow	
7	Null Pointer	Null Pointer	
8	Privilegverletzung	Privilege Violation	
10	Nicht implementierter Befehl	Unimplemented Instruction	
24	Falscher Interrupt	Spurious Interrupt	
128	IO-Exception	IO Exception	I/O Bus Exceptions
144	TK-Zykluszeitverletzung	TC Cycle Time Violation	Cycle Time Vioaltion
145	TK-Max. Zykluszeitverletzung	TC Max. Cycle Time Violation	
146	TK-Input-Zykluszeitverletzung	TC Input Cycle Time Violation	
147	TK-Output-Zykluszeitverletzung	TC Output Cycle Time Violation	
160	SSTK-Max. Zykluszeitverletzung	HSTC Max. Cycle Time Violation	
161	SSTK-I/O-Zykluszeitverletzung [1]	HSTC I/O Cycle Time Violation [1]	
162	Systemzykluszeitverletzung	System Cycle Time Violation	

For all exceptions that have no exception task assigned to them, the exception handler performs the extended error reporting. On the CPU module of the B&R 2010, a short code of the exception is also shown. Following this an emergency stop of the system is done (PLC goes into SERVICE mode).

There is a short description of all exceptions handled by the exception handler EXERMO.BR Version 1.12 as well as a description of the short codes on the display:

Bus Error	Display: **B <Adr>** **E <PC>**
<Adr> .... Access address that caused the bus error. <PC> .... Program counter at time of the exception. A bus error occurs, for example, if a peripheral unit doesn't reply or an illegal memory access is made.	

Address Error	Display: **A <Adr>** **E <PC>**
<Adr> .... Access address that caused the bus error. <PC> .... Program counter at time of the exception. An address error occurs if a WORD or LONGWORD access to an odd address is made.	

[1] The Exception “HSTC I/O cycle time violation” is supported from BUGFIX Version 1.11 (B&R Module BFX_V111.BR) of the operating system version 1.10 and later.

Illegal Instruction	Display: **I <SR>** **I <PC>**

<SR> .... Status register at the time of the exception.

<PC> .... Program counter at the time of the exception. i.e. the program counter points to the instruction that is not a Motorola instruction.

If the CPU recognizes a (currently) unimplemented instruction, this exception is triggered.

Division by Zero	Display: **D <SR>** **Z <PC>**

<SR> .... Status register at the time of the exception.

<PC> .... Program counter at the time of the exception.

This exception is released if a division is done with a divisor of "0".

Range Overflow	Display: **R <SR>** **O <PC>**

<SR> .... Status register at the time of the exception.

<PC> .... Program counter at the time of the exception.

On task compiled with the compiler option "Range Check:On", each array access is checked during run time. If the CPU recognizes an access to a nonexistent array element, this exception is released. You can find more detailed information on this compiler option in the "PG2000 Programming System" chapter.

Null Pointer	Display: **N <SR>** **P <PC>**

<SR> .... Status register at the time of the exception.

<PC> .... Program counter at the time of the exception.

On task compiled with the compiler option "Null Ptr Check:On", each pointer access is checked during run time. If the CPU recognizes an access to an uninitialized pointer (points to the address 0), this exception is released. You can find more detailed information on this compiler option in the "PG2000 Programming System" chapter.

Privilege Violation	Display: **P <SR>** **V <PC>**

<SR> .... Status register at the time of the exception.

<PC> .... Program counter at the time of the exception.

This exception occurs if access is made to a protected data area while in User mode (application) or a command is executed that is only allowed when in supervisor mode.

Un-implemented Instruction	Display: **U <SR> I <PC>**

<SR>..... Status register at the time of the exception.

<PC>..... Program counter at the time of the exception.

If the CPU recognizes an un-implemented instruction, this exception is released.

Wrong Interrupt	Display: **S <SR> I <PC>**

<SR>..... Status register at the time of the exception.

<PC>..... Program counter at the time of the exception.

This Exception is triggered if a bus error occurs during an interrupt recognition cycle.

IO Exception		
	Display for I/O-Bus B&R 2010:	**IO xx yy Mzz abc**
	Display for I/O-Bus Extension B&R 2005:	**IO xx yy Mrzz abc**
	Display for Remote I/O-Bus B&R 2010 or B&R 2005:	**IO xx yy Rsszzabc**

xx ..... “TI” = Timeout, “RP” = Repeat

yy ..... Shovel table priority

zz ..... Module address that causes the error.

ss ..... Slave address of the Remote Slaves

r ..... I/O-Bus extension B&R 2005: base plate number (0 = main, 1 = extension base plate 1, 2 = extension base plate 2, etc.)

a ..... “R” = Read (Input image), “W” = Write (Output image)

b ..... “N” = Normal, “S” = Status are of the module

c ..... “M” = Module address, “K” = Code, “I” = Internal Address, “D” = Data

These exceptions are triggered by errors on the I/O bus (e.g. accessed module does not exist).

TC Cycle Time Violation	Display: **TC#x Err viCycle**

x ..... Number of the normal task class that caused the exception.

This exception is triggered when a cycle time violation occurs in a normal task class.



The cycle time monitoring for normal task classes is deactivated by default, i.e. cycle time violation are not recognized. The function "ZYKVL_en" (system library), you can turn on the cycle time monitoring for normal task classes.

TC Max. Cycle Time Violation	Display: **TC#x Err mxCycle**

x ..... Number of the normal task class that caused this exception.

For normal task classes, an allowed cycle time violation for tolerant cycle time monitoring can be set up with the PLC Configurator or PG2000. This means that the cycle time can be exceeded by a predefined amount (simple cycle time violation). If this time is also exceeded (maximum cycle time violation), this exception is triggered.

TC Input Cycle Time Violation	Display: **TC#x Err inCycle**

x ..... Number of the normal task class that caused this exception.

If the input image ordered by the operating system at the start of the previous task class cycle is not ready at the start of the task class cycle, this exception is released.

TC Output Cycle Time Violation	Display: **TC#x Err ouCycle**

x ..... Number of the normal task class that caused this exception.

If the output image ordered by the operating system at the end of the previous task class cycle is not ready at the start of the task class cycle, this exception is released.

HSTC Max. Cycle Time Violation	Display: **HS#x Err mxCycle**

x ..... Number of the high speed task class that caused this exception.

This exception is released when the run time of a high speed task class is longer than the set cycle time. This means that not enough calculation time is left for the other tasks.

HSTC I/O Cycle Time Violation	Display: **HS#x Err viCycle**

x ..... Number of the high speed task class that caused this exception.

If the I/O image ordered by the operating system at the end of the previous task class cycle is not ready at the start of the task class cycle, this exception is released.

System Cycle Time Violation	Display: **SYSM Err mxCycle**

This exception is triggered when there is a violation of the system cycle time.

As with normal task classes, an allowed cycle time violation can also be set for the system for tolerant monitoring (done with the PLC Configurator). This avoids a system cycle time violation occurring immediately when there is a short term bottle neck (e.g. if HS tasks and interrupt tasks need almost all the computing time). The allowed excess is exceeded, this exception is triggered.

CHAPTER 3
SOFTWARE INSTALLATION

CONTENTS

1 Contents of Delivery 3-5

2 Hardware Requirements 3-5

3 New Installation 3-6
3.1 New Installation of the PG2000 Programming Systems 3-6
3.2 New Installation of the Utility Box 3-10
3.3 New Installation of the PLC-OS OPTIONS 3-12
3.4 New Installation of the Function Libraries 3-12
3.5 Directory Structure 3-13

4 Replacing an Old Software Version 3-14

5 System Configuration and Memory Optimization 3-15
5.1 BIOS Setup 3-15
5.2 Settings in the CONFIG.SYS file 3-15
5.3 Settings in the Batch File AUTOEXEC.BAT 3-17

1 CONTENTS OF DELIVERY

The programming software for the B&R SYSTEM 2000 controller generation is delivered as a set (including documentation and on-line cable). There are two different versions of the PG2000 software package:

Product Name	Documentation	Model Number
PG2000 Programming Software	German	1A2000:01-110
PG2000 Programming Software	English	1A2000:01-120

The PG2000 software package consists of the following items:

- Cable PC-Controller 2000 RS232, Model#: 0G0001.01-090
  (On-line cable to connect the PLC and PC)
- PG2000 Programming Software, Model#: 1A2000.01
  (Installation disk for the PG2000 Programming System)
- PG2000 Accessory Utility Box, Model#: 1A2050.01
  (Installation Disk for Utilities like the PLC Configurator, Profiler, etc.)
- PG2000 Accessory P.L.C. OS-OPTIONS, Model#: 1A0200.01
  (Installation Disk for the operating system options like the exception handler, etc.)
- PG2000 Standard Software LIBRARY 1 (BASIC), Model#: 1A2200.01
  (Installation Disk for function libraries)
- Documentation

The software is delivered on 3½" HD disks.

2 HARDWARE REQUIREMENTS

Before you start the installation, please check that your system meets the following requirements:

- Minimal requirement for your PC: 386 SX, 25 MHz, 2 MBytes of RAM
- 3½" disk drive (1.44 MBytes)
- Hard disk (at least 7 MByte free for installation)
- 640 KByte RAM Memory; At least 500 KByte must be free. Memory resident programs are to be removed if necessary (You can find more detailed information on this topic in the "system configuration and memory optimization").
- PG Interface: Serial Interface (COM1 or COM2) or
  PROFIBUS
- MS-DOS Version 3.30 or higher

3 NEW INSTALLATION

3.1 NEW INSTALLATION OF THE PG2000 PROGRAMMING SYSTEMS

To install the PG2000 programming system, insert the installation disk with the label:

```
PG2000 Programming Software
     Model#: 1A2000.01
         Disk 1/2
```

in a 3½" disk drive (1.44 MByte, High Density Format). After you have done this, enter the drive letter of your 3½" drive (A: or B:) at the DOS command line:

```
C:\>a:setup
```

or

```
C:\>b:setup
```

After pressing the Enter key [↵], the installation procedure starts (all further steps are done using a menu). There is now a **Language Selection Menu** on the screen for the installation procedure on the PC:
You choose the language with this menu that is to be used for the installation procedure.



The standard language for the PG2000 programming system is set by the DOS country code. In the system file CONFIG.SYS, you can set the country code with COUNTRY command:

`COUNTRY=049`	049 ..... Country code for Germany
`COUNTRY=001`	001 ..... Country code for English (International)

You can find more detailed information on the COUNTRY command in your MS-DOS User Manual.



If the country code entry is missing from the CONFIG.SYS file or a country code other than 049 (Germany), 043 (Austria) or 041 (Switzerland) is set, English is used as the default language for the PG2000 Programming System.

Overview of the function keys in the Language selection menu:

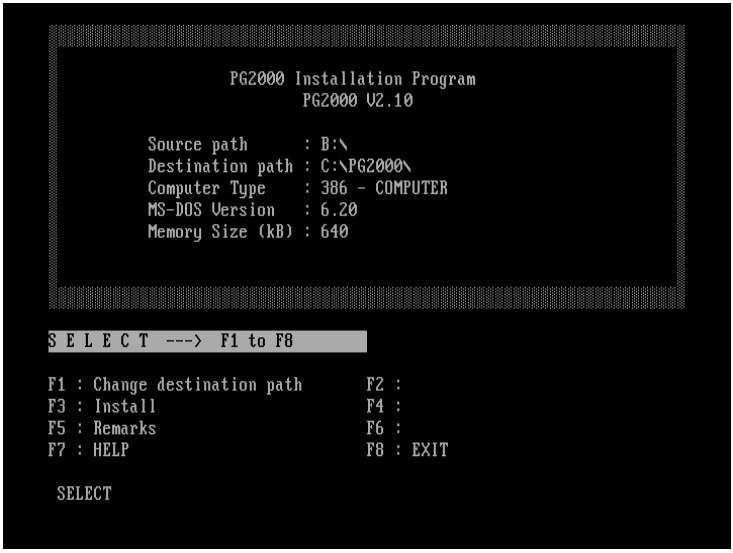
[F1 Deutsch] ..... German

[F2 English] ..... English

[F7 Hilfe - Help] ..... A help screen for the language selection is shown.

[F8 Beenden - Exit] ..... Quit and exit the installation procedure and return to DOS.

After you have selected the language you want, the **installation menu** appears on the screen:



Some information about the computer system being used as well as the source and destination paths of the software is shown with this menu.

Overview of the function keys of the installation menu:

[F1 Change Dest. Path] ..... Change the directory where the software will be installed.

[F3 Install] ..... Start PG2000 Installation.

[F5 Hints] ..... Display a few hints to help with the installation.

[F7 Help] ..... A short description of the function keys is displayed.

[F8 Exit] ..... The setup program ends.

Change Destination Path

The destination path is the directory where the software is to be installed. The destination path is set to C:\PG2000\ by default. If you want to change this, you have to press **[F1 Change Dest. Path]**. It is possible to enter a longer path name with several sub-directories. If the directories do not exist they are automatically created.

Start Installation

To start the installation, press **[F3 Install]**.
During the installation procedure, the setup program suggests changes to the CONFIG .SYS file. These changes are only made when confirmed by you:





The PG2000 software needs at least the suggested values for FILES and BUFFERS. If you enter smaller values in the CONFIG.SYS file, this can cause problems while running the PG2000 software.

After all this is done, the files are copied from the installation disk to the hard disk. Next, you are asked to insert the installation disk with the label:

PG2000 Programming Software
Model#: 1A2000.01
Disk 2/2

into the 3½" disk drive. The following message is displayed on the screen:



After you have inserted the second installation disk and pressed the enter key [↵] the installation procedure continues.
When all files from the programming system have been copied onto the hard disk (destination directory), the following message appears on the screen:



After pressing any key, the installation menu appears again.

Exit Installation

When you press the function key **[F8 Exit]**, the installation program ends. If the CONFIG.SYS file has been changed, the following message appears on the screen:



If this is the case, you should press the **[Y]** key so that the changes to your CONFIG.SYS file can take effect. If you press the**[N]**key the installation program ends without rebooting your computer and returns you to the DOS command line.

Batch File PG.BAT

During the installation procedure, a batch file called PG.BAT is created by the installation program. The given destination path of the PG2000 software is automatically entered in the batch file.
The PG.BAT file is saved in the root directory of your hard disk. When the software installation is complete, this batch file can be used to start the PG2000 programming system.

3.2 NEW INSTALLATION OF THE UTILITY BOX

To install the utilities like the PLC configurator, Profiler, etc. insert the installation disk with the label:

```
PG2000 Accessory Utility Box
     Model#: 1A2050.01
         Disk 1/1
```

in a 3½" disk drive (1.44 MByte, High Density Format). After you have done this, enter the drive letter of your 3½" drive (A: or B:) at the DOS command line:

```
C:\>a:setup
```

or

```
C:\>b:setup
```

After pressing the Enter key [↵], the installation procedure starts (all further steps are done using a menu). There is now a **Language Selection Menu** on the screen for the installation procedure on the PC:
You choose the language with this menu that is to be used for the installation procedure.
The standard language for the PLC Configurator and Profiler is set by the DOS country code.



If the country code entry is missing from the CONFIG.SYS file or a country code other than 049 (Germany), 043 (Austria) or 041 (Switzerland) is set, English is used as the default language for the PLC Configurator and Profiler.

Overview of the function keys in the Language selection menu:

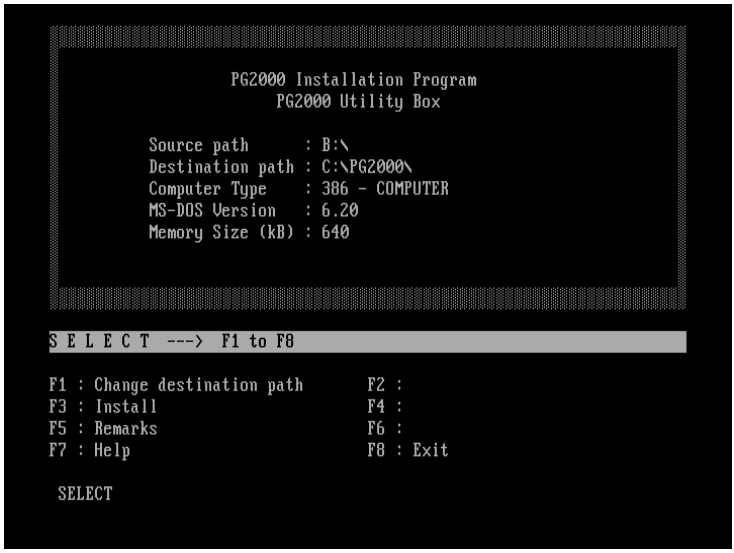
[F1 Deutsch] ..... German

[F2 English] ..... English

[F7 Hilfe - Help] ..... A help screen for the language selection is shown.

[F8 Beenden - Exit] ..... Quit and exit the installation procedure and return to DOS.

After you have selected the language you want, the **installation menu** appears on the screen:



Some information about the computer system being used as well as the source and destination paths of the software is shown with this menu.

Overview of the function keys of the installation menu:

[F1 Change Dest. Path] ..... Change the directory where the software will be installed.

[F3 Install] ..... Start PG2000 Installation.

[F5 Hints] ..... Display a few hints to help with the installation.

[F7 Help] ..... A short description of the function keys is displayed.

[F8 Exit] ..... The setup program ends.

Change Destination Path

The destination path must be the same as used for the PG2000 programming system installation!

Start Installation

To start the installation of the Utility Box, press [F3 Install]. When all files from the programming system have been copied onto the hard disk (destination directory), the following message appears on the screen:



After pressing any key, the installation menu appears again.

Exit Installation

When you press the function key **[F8 Exit]**, the installation program ends and returns you to the DOS command line.

3.3 NEW INSTALLATION OF THE PLC-OS OPTIONS

To install the Operating System Options, insert the installation disk with the label:

```
PG2000 Accessory P.L.C. OS-OPTIONS
        Model#: 1A0200.01
            Disk 1/1
```

in a 3½" disk drive (1.44 MByte, High Density Format). After you have done this, enter the drive letter of your 3½" drive (A: or B:) at the DOS command line:

```
C:\>a:setup
```

or

```
C:\>b:setup
```

After pressing the enter key [↵] the installation procedure starts. The following procedure is the same as for the installation of the Utility Box (see section on " New installation of the Utility Box").

3.4 NEW INSTALLATION OF THE FUNCTION LIBRARIES

To install the Function Library, insert the installation disk with the label:

```
PG2000 Standard Software LIBRARY 1 (BASIS)
            Model#: 1A2200.01
                Disk 1/1
```

in a 3½" disk drive (1.44 MByte, High Density Format). After you have done this, enter the drive letter of your 3½" drive (A: or B:) at the DOS command line:

```
C:\>a:setup
```

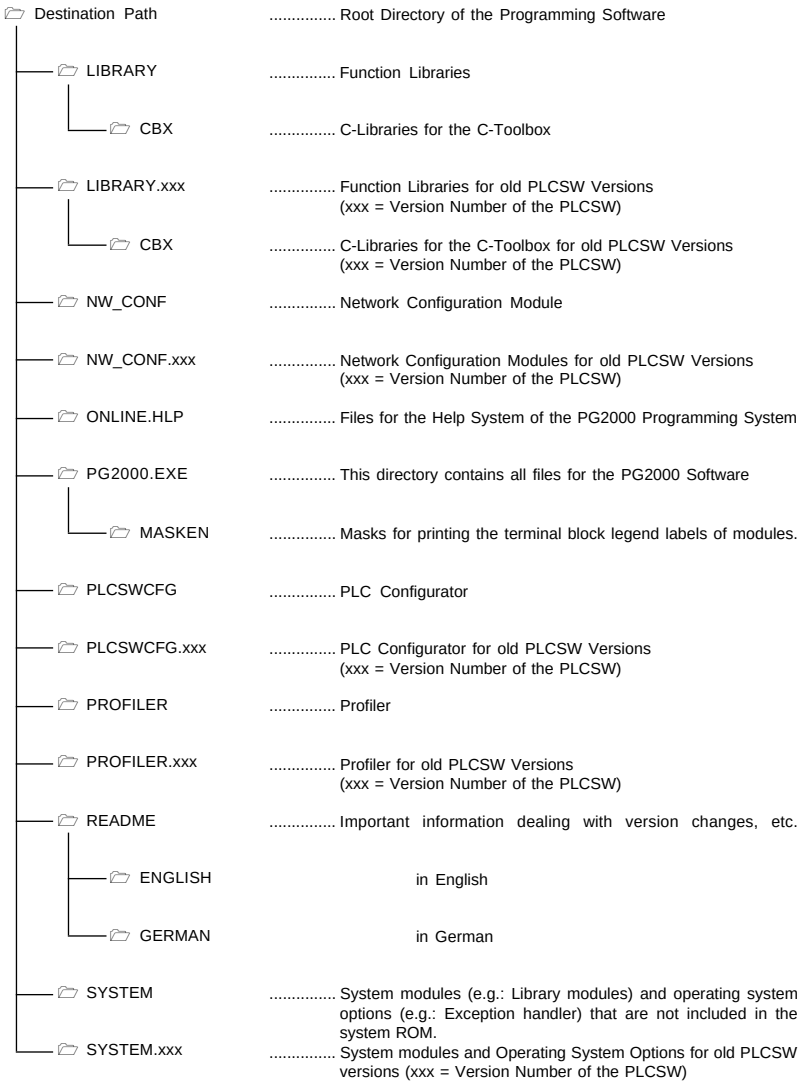
or

```
C:\>b:setup
```

After pressing the enter key [↵] the installation procedure starts. The following procedure is the same as for the installation of the Utility Box (see section on " New installation of the Utility Box").

3.5 DIRECTORY STRUCTURE

During the installation of the software, the following directory structure is set up:



4 REPLACING AN OLD SOFTWARE VERSION

To replace your old version with the new one, you do the same as for a new installation (see section "New Installation"). The old version is overwritten by the new files.

The setup program notices if there is already a version of the PG2000 programming software in the same destination path. In this case, it asks if the old PG.BAT file should be overwritten:



By pressing the [Y] key, the batch file PG.BAT is overwritten. If you press the [N] key you are given the option of giving the new batch file a new name. Apart from this the installation is identical to the new installation.

5 SYSTEM CONFIGURATION AND MEMORY OPTIMIZATION

Before you start the PG2000 programming software, you should take careful note of the following hints and recommendations for configuring your system and improving memory usage during the running of the PG2000 software.

5.1 BIOS SETUP

To run the PG2000 software problem free, the following settings in BIOS should be made:

- **CACHE**
  The memory range of a PROFIBUS card (if used) must not be in the cache region.

- **SHADOW**
  The shadow RAM for system and video ROM should be turned on.

5.2 SETTINGS IN THE CONFIG.SYS FILE

The following settings refer to the MS-DOS operating system. You can find hints and guidelines for other operating systems in the file OPTIMIZE.TXT in the directory <DEST.DIRECTORY>\README\ENGLISH.

Memory Management

To optimally use main memory, you should install one of the following memory management programs:

- **QEMM** (Quarterdeck) Version 5.12 or later
  The following lines must be entered in the CONFIG.SYS file to install this program:

  ```
  DEVICE=QEMM386.SYS RAM VREMS FRAME=C800
  DOS=HIGH
  ```

  You should load as many drivers as possible in the upper memory area with LOADHI.SYS.

- **MAX386** (Qualitas) Version 5.10 or later
  The following lines must be entered in the CONFIG.SYS file to install this program:

  ```
  DEVICE=386MAX.SYS EMS=2048
  DOS=HIGH
  ```

  You should load as many drivers as possible in the upper memory area with 386LOAD.SYS.

- **EMM386.EXE** (included in MS-DOS)
  The following lines must be entered in the CONFIG.SYS file to install this program:

  ```
  DEVICE=HIMEM.SYS
  DEVICE=EMM386.EXE 2048 RAM FRAME=C800
  DOS=HIGH,UMB
  ```

  You should load as many drivers as possible in the upper memory area with DEVICEHIGH.

Parameters

- **FILES and BUFFERS**
  The PG2000 software needs at least the values suggested in the installation procedure by the setup program for FILES and BUFFERS. If you use smaller values that those suggested, this could cause problems when running the PG2000 software. The following values are recommended by the setup program:

```
FILES=60
BUFFERS=20
```

- **STACKS**
  So as not to cause problems with the communication between the PLC and the PC and to avoid the operating system message "Internal memory overflow", the number and size of the stacks must be set with the STACKS command to "0":

```
STACKS=0,0
```

- **COUNTRY**
  The default language for PG2000, the PLC configurator and the Profiler is set using the DOS country code. You can set the country code with the COUNTRY command:

`COUNTRY=001, ,COUNTRY.SYS`	001 ..... Country code for International (English)
`COUNTRY=049, ,COUNTRY.SYS`	049 ..... Country code for Germany
`COUNTRY=043, ,COUNTRY.SYS`	043 ..... Country code for Austria
`COUNTRY=041, ,COUNTRY.SYS`	041 ..... Country code for Switzerland

Device Drivers

Only install the device drivers that your system needs (the ANSI.SYS device driver is not necessary for PG2000 operation). The required drivers should be loaded into upper memory (as many as possible) with DEVICEHIGH.

- **RAMDRIVE.SYS**
  When using a program with high memory requirements, is not sensible to use a part of memory as a virtual drive. For this reason, the device driver RAMDRIVE.SYS should not be used with PG2000.

- **NETWORK.SYS**
  When you have to use a network driver, leave out all components that are not absolutely necessary for normal work station operation (e.g.: EMAIL, MESSAGE, CHAT, REMOTE, etc).



For more detailed information on the CONFIG.SYS file, please refer to your MS-DOS User Manual.

5.3 SETTINGS IN THE BATCH FILE AUTOEXEC.BAT

The following settings refer to the MS-DOS operating system. You can find hints and guidelines for other operating systems in the file OPTIMIZE.TXT in the directory <DEST.DIRECTORY>\README\ENGLISH.

Keyboard Driver

Try to load your keyboard driver into the upper memory area.

Mouse Driver

Try to load your mouse driver into the upper memory area.

Disk Cache

A disk cache program can accelerate the compilation of a PLC tasks by a factor of six. It is highly recommended that you install a disk caching program, however the buffer should not be too large (maximum 512 KByte). After installation of the disk cache, there should be at least 2 MBytes of expanded memory (EMS) free.
Examples of disk cache programs:

- **PC-Cache** (Central Point Software) All Versions
- **SmartDrive** (included in MS-DOS) Version 4.0 and later

Try to load your disk caching program into the upper memory area.

Tools

Tools that have to be loaded as memory resident use up memory space that cannot be used during PG2000 operation. Included in tools are programs such as sidekick and programs to show the data and time. You should not load these tools unless absolutely necessary.



For more detailed information on the CONFIG.SYS file, please refer to your MS-DOS User Manual.

CHAPTER 4
PG2000
PROGRAMMING SYSTEM

CONTENTS

1 Highlights 4-7
1.1 Operation 4-7
1.2 Programming 4-7

2 Start Preparations 4-8
2.1 System Configuration 4-8
2.2 Use of a Pointing Device 4-12
2.3 Printing Requirements for PG2000 4-12
2.4 Starting the Programming System 4-12

3 General Operation information 4-13
3.1 Creating a Test Project 4-14
3.2 Screen Components 4-15
3.5 Conventions 4-16
3.3 Using the Mouse 4-16
3.4 Using the Keyboard 4-16
3.6 Main Menu Bar and Pull Down Menus 4-17
3.7 Working with Windows 4-19
3.8 Foot Line Menu 4-25
3.9 Context Sensitive Pop Up Menus 4-26
3.10 Dialog Boxes 4-27
3.11 File Selection Box 4-30
3.12 General Hot Keys 4-31
3.13 Help System 4-32

4 Introductory Example 4-35
4.1 Example Description 4-35
4.2 Hardware Requirements 4-35
4.3 Creating the Project 4-36
4.4 Calling the LAD Editor via the GDM 4-37
4.5 Creating the Ladder Diagram 4-39
4.6 Saving the Ladder Diagram 4-40
4.7 Variable Declaration 4-41
4.8 Connecting to the PLC 4-44
4.9 Compiling and Downloading to the PLC 4-45
4.10 Testing the Example Program with the LAD Debugger 4-45

5 Project Management 4-48
5.1 Creating a Project 4-49
5.1.1 Project Databank 4-51
5.1.2 Program Files 4-52
5.2 Opening a Project 4-54
5.3 Copying a Project 4-54
5.4 Including Projects in Project Management 4-55
5.5 Purging a Project 4-56
5.6 Deleting a Project 4-56

6 GDM - Graphical Design Method 4-57
6.1 General Information 4-57
6.2 Calling the GDM 4-58
6.3 Calling the Task Layer 4-59
6.4 Placing an Object in the Task Layer 4-60
6.5 Object Parameters 4-62
6.5.1 Parameters for Normal, High Speed, IRQ and EXC Tasks 4-62
6.5.2 Parameters for Data Modules and B&R Modules 4-65
6.5.3 PLC Configuration Module Parameters 4-66
6.6 Editing an Object 4-69
6.7 Ladder Diagram - LAD 4-70
6.7.1 General Information 4-70
6.7.2 Ladder Diagram Syntax 4-71
6.7.3 LAD Editor 4-74
6.8 Statement List - STL 4-84
6.8.1 General Information 4-84
6.8.2 Syntax of the Programming Language - STL 4-84
6.8.3 STL Editor 4-88
6.8.4 Configuring the STL Editor 4-94
6.9 PL2000 4-96
6.9.1 General Information 4-96
6.9.2 Syntax of the Programming Language - PL2000 4-96
6.9.3 PL2000-Editor 4-99
6.9.4 Configuring the PL2000 Editor 4-105
6.10 Variable Declaration 4-107
6.10.1 Symbolic Names 4-107
6.10.2 Scope 4-108
6.10.3 Data Type 4-109
6.10.4 Length 4-110
6.10.5 I/O Type 4-111
6.10.6 Init Value 4-119
6.10.7 Long Name 4-119
6.10.8 Configuring the Variable Declaration Editor 4-120
6.11 Initialization Sub-Programs 4-123
6.11.1 Creating an INIT-SP 4-123
6.11.2 Opening or Deleting an INIT-SP 4-123
6.12 Compiling Objects 4-124
6.12.1 Compiler Components 4-124
6.12.2 Compiler Options 4-125
6.13 Download object in User RAM 4-128
6.13.1 Download Function 4-128
6.13.2 Overload Function 4-129
6.14 Download Object in the User ROM or FIX-RAM 4-130
6.15 Delete Objects 4-131
6.16 Process Variable Monitor (PV Monitor) 4-132
6.17 Debugger 4-139
6.17.1 Ladder Diagram Debugger (LAD Debugger) 4-140
6.17.2 STL and PL2000 Debugger 4-144
6.18 Putting in a Second PLC 4-149
6.19 Foot Line Menu of the GDM 4-150
6.20 PLC Layer Pop Up Menu 4-154
6.21 Task Layer Pop Up Menu 4-158
6.22 GDM Keyboard Commands 4-162
6.23 Synchronizing the PLC and GDM 4-163

7 Functions 4-164
7.1 Standard Functions 4-164
7.1.1 General Information 4-164
7.1.2 Importing Function Libraries 4-164
7.1.3 Help on the Function Libraries 4-165
7.1.4 Deleting a Function Library 4-166
7.2 Application Specific Function Blocks 4-167
7.2.1 General Information 4-167
7.2.2 Creating An Application Specific Function Block 4-167

8 PLC Utilities 4-172
8.1 Task Management Utilities 4-172
8.2 Task Class Utilities 4-174
8.3 Module - Display and Manipulation 4-175
8.4 Error Analysis 4-178
8.5 Other Functions 4-180
8.6 Modem 4-184
8.7 PV Monitor 4-185

1 HIGHLIGHTS

The PG2000 Programming System is a powerful tool for programming the B&R 2000 PLC-PCC Family.

1.1 OPERATION

- You can use the mouse and the keyboard.

- User-friendly Integrated Development Environment (Menu bar, Pop-Up-Menus, window based desktop, dialog boxes).

1.2 PROGRAMMING

- Three different programming languages are available (Ladder Diagram, Statement List, PL2000).

- Combining all three programming languages in a project is possible.

- PL2000 is a high level (C-like) programming language with control constructs for loops, conditional branching (if then) and select structures (case).

- The "**G**raphic **D**esign **M**ethod" (GDM) supports structured programming.

- Symbolic names allow for hardware independent programming.



The description in this manual applies to version 2.10 of the PG2000 Programming System.

2 START PREPARATIONS

2.1 SYSTEM CONFIGURATION

PG2000 has to be suited to the system configuration of your computer before you actually start it. You can adapt it by editing the ASCII text file PG.INI (in the C:\PG2000\PG2000.EXE\ directory) with any text editor.
The settings that you can change in the configuration file PG.INI are described in the ASCII file PG_INI.TXT (in C:\PG2000\README\ENGLISH\):

```
********** PG - Initialization File PG.INI **********

Note:
========
PG.INI is a data base that stores the user defined settings for the PG.EXE program.
It is an ASCII text file so that you can edit it with any text editor. Basically,
this file should only be read from and written to by the program PG.EXE as it only
stores the settings that you changed and saved with the menu system. However, since
not all settings can be changed via the menu system, the entries are documented
here.

Warning:
========
If you use or change any entries in PG.INI, that are not listed here, marked as
"not to be changed", or if you exceed the limits for the entry, PG.EXE may not behave
as described in this manual or give incorrect results!

Syntax Rules:
=============
The file is based on line orientation. Each line corresponds to a single entry in
the data base. A line can have up to 255 characters. Continuation lines are not
allowed. Tabs and indents are not allowed. The first character in a line defines
the type of entry.

Comments:
===========
Starts in column 1 with the ";" character

Empty Lines:
===========
Are allowed and have no effect.

Sections:
===========
Are enclosed in [] and begin in column 1

Key words:
==========
Start in column 1 with any character except ";[=" and end at the first occurrence
of the "=" character.

Values:
======
Are strings come after the key word, immediately after the "=" character and go
until the end of the line.
```

The section names and keywords are not case sensitive, however the values following the "=" character are.

Each section is only allowed to appear in the file once.

Each keyword is only allowed to appear once in each section.

Semantics:
=========
A section [user] is to be setup for each user (user name 3-8 characters). Only alphabetic characters and numerals are allowed.

The section [default] must exist.

The following entries are required in the sections [user] and [default]:

RIGHTS=9
Every user has to have the entry RIGHTS=9. Other values are not allowed.

The sections [user] and [default] can have the following entries:

PG_CONFIG=xxxx
xxxx = PG-COL.CFG or PG-MON.CFG defines the colour set for the menu system according to your video card.
Other entries are not allowed.

NW_START=xxx
xxx = Yes or No. If this setting is set to "No", communication with the PLC is not automatically started. In this case, it has to be initialized using the menu option <Settings><Connection>.

NW_CONFIG=xxxxxxxx
xxxxxxxx = NW_P2_57.BR or a complete path including path, name and extension of a fie that contains the connection information for the PROFIBUS network or a serial connection to the PLC. The standard names are "NW_*.BR".
!! If this entry is missing, NW_START has to be set to "No" !!

LANGUAGE=xxx
xxx = Country Code (049 or 001). If the country code specified in the CONFIG.SYS file causes the wrong language to be displayed, you can set the language with this entry. ! Warning ! If the all the system files for the selected language are not available, the behaviour could be strange and unexpected !

COMPILER=xxxx
xxxx = Compiler Options. Not to be changed ! Can only be set using the menu system.

VD_OPTIONS=xxxxx
Editor options for the variable declaration. Not to be changed ! Can only be set using the menu system.

BE_OPTIONS=xxxxx
Editor options for the Offset Table. Not to be changed ! Can only be set using the menu system.

BR_OPTIONS=xxxxx
Editor options for the PL2000 Editor. Not to be changed ! Can only be set using the menu system.

TX_OPTIONS=xxxxx
Editor options for the Text Editor. Not to be changed ! Can only be set using the menu system.

IL_OPTIONS=xxxxx
Editor options for the STL Editor. Not to be changed ! Can only be set using the menu system.

PROJEKT=xxxxxxxx
xxxxxxxx is a complete including drive, path, name and extension of a project that is automatically opened when starting PG. If has to follow the operating system rules for a complete path. It must not be longer than 40 characters.

There is a section [PRJ:xxxxxxxx] for each project set up by PG. "xxxxxxxx" is the project name and has to follow the operating system rules for a file name (1-8 characters).
Projects can only be created by PG.EXE!

Entries possible in the [PRJ:xxxxxxxx] section:

SPS_STD=xxxx
Defines the PLC name that is used in the project as long as the GDM is not open.

PROC_STD=xxxx
Defines the processor name that is used in the project as long as the GDM is not open.

SPS_VERSION=xxx
xxx = 105 or 110. Sets the behaviour of PG.EXE to V1.05 or V1.10 of the PLCSW. Other values are not allowed.

GDM_SPS=xxxx
GDM_PROC=xxxx
Both these values are set when the entry layer of the GDM is set. Not to be changed ! Can only be set using the menu system.

REBUILD=xxx
xxx = Yes or No. The information for rebuilding or decompiling a loadable module back into source code is save if the setting "Yes" is set.

In the [default] section, the following project specific entries are allowed:

SPS_STD=xxxx
For all projects the do not have a PLC name defined yet, this PLC name is used. If this entry is missing, the name TEST_SPS is used.

SPS_VERSION=xxx
xxx = 105 or 110. For all project that have no PLC version assigned to them, this entry is used. If this entry is missing, version 1.10 is used.

REBUILD=xxx
xxx = Yes or No. The information for rebuilding or decompiling a loadable module back into source code is save if the setting "Yes" is set.

```
The section [PrinterSetup] is for the printer settings

Allowed entries in the section [PrinterSetup] :

DRIVER=xxxx
       xxxx = TTY , ISO , IBM , HPLJ , EPSON
Defined the printer driver.
       ISO:   No ESC sequences are output and only the control codes CR, LF, FF
              are used. Only the characters in the ISO646 character set are used
              to print block graphics. Conforms to the IEC 1131-3 standard.
       TTY:   The same as ISO, except that the character 124 is replaced with the
              character 33. This is so that printouts can also be made on devices
              that have the characters 123-126 replaced with national special
              characters.
       IBM:   No ESC sequences are output and only the control codes CR, LF, FF
              are used. The characters from code page 437 are used to print out
              block graphics. Intended for use in a text editor.
       HPLJ:  The same as IBM except that ESC sequences are used for block
              characters from code page 437 that can be printed out on a HP laserjet
              or compatible device.
       EPSON: The same as IBM except that ESC sequences are used for block
              characters from code page 437 that can be printed out on a EPSON or
              compatible device.

HeadLine=xxxxxxxxxxxxxxx
       Corresponds to the header given in the menu option <Settings><Documentation>.

FootLine=xxxxxxxxxxxxxxx
       Corresponds to the footer given in the menu option <Settings><Documentation>.

FormFeed=x
       Corresponds to the form feed setting given in the menu option
       <Settings><Documentation>. Not to be changed !

ProjectOptions=xxx
       Corresponds to the options given for project printout in the menu option
       <Settings><Documentation>. Not to be changed !

ObjectOptions=xxx
       Corresponds to the options given for file printout in the menu option
       <Settings><Documentation>. Not to be changed !

FileFormat=xxxxxxxxxx
       Corresponds to the page layout setting given in the menu option
       <Settings><Documentation>. Not to be changed !

TmpPath=xxxxx
       Corresponds to the path given for spooler files in the menu option
       <Settings><Documentation>. Not to be changed !
```

2.2 USE OF A POINTING DEVICE

To use the programming system with a mouse or other form of pointing device, you have to install a mouse driver before calling PG2000. Please refer to the user handbook for your mouse software.



You have to have a second serial interface on your PC to use a mouse.

2.3 PRINTING REQUIREMENTS FOR PG2000

The resident part of PRINT (MSDOS) has to be loaded before you start PG2000 if you want to print from the Programming System. You have to type in the following DOS command:

```
C:\>print /D:device
```

The interface that printer is connected to is given as "device".
Example for LPT1 interface:

```
C:\>print /D:lpt1
```

If the printer driver loads, the following message appears:

```
Residenter part of PRINT loaded
The printer queue is empty.
```

2.4 STARTING THE PROGRAMMING SYSTEM

After successfully installing and configuring PG2000, you can start it with the batch file PG.BAT. The batch file has the following command line options:

```
C:\>pg [User] [Options]
```

`User` ..... There is a section [user] for each user in the file PG.INI with a particular system configuration. To use the configuration under the section [user] you have to type in the corresponding section name.
If you do not specify a name the configuration under the section [default] is used.

`Options`

- `/P[+|-]` .....Pointing device (e.g. mouse) is used (`/P+`) / not used (`/P-`). Default `/P+`.
- `/A[+|-]` .....Expansion memory (EMS) is used (`/A+`) / not used (`/A-`). Default `/A+`.
- `/U[+|-]` .....The upper memory area (UMA) is used (`/U+`) / not used (`/U-`). Default `/U+`.
- `/H[+|-]` .....The high memory area (HMA) is used (`/H+`) / not used (`/H-`). Default `/H+`.
- `/X[+|-]` .....Extended memory (XMS) is used (`/X+`) / not used (`/X-`). Default `/X+`.
- `/?` .....The help text for the start options is displayed.

After you press the enter key [↵], the batch file is run and PG2000 starts.

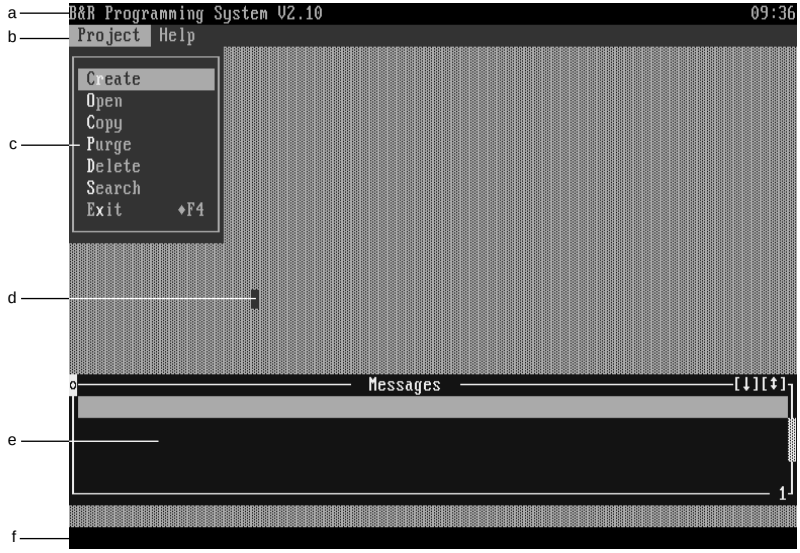
3 GENERAL OPERATION INFORMATION

The PG2000 programming system basically consists of two parts:

Project Management ..... To setup, open, delete, copy a project etc
Programming Environment ... To write the programs for a project (create, compile, download to the PLC, etc.)

The functions for project management allow you to work with several projects. For simplicity and good organization, you should create a new project for each application.

At the first call of the programming system, the project management is automatically called:



a) The following information is displayed in the **title bar**:
   - Program Name with version number: “B&R Programming System Vx.xx”
   - Current time

b) There are various pull down menus that can be opened from the **main menu bar**.

c) Different functions can be selected from the **Pull Down Menus**.

d) **Mouse Cursor:** The current mouse position is indicated with an inverted character at the relevant position.

e) **Message Window** cannot be closed, only erased.

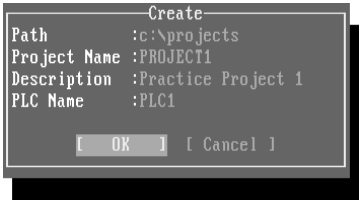
f) Additional information is shown in the **Message Line**.

3.1 CREATING A TEST PROJECT

The operation of the PG2000 Programming System **programming environment** is explained on the next page. This description is also for project management. To switch into the programming environment, you have to create and open a project in the project management. We will quickly go through creating and opening a project (a detailed description is in the section "Project Management").

After calling the programming, the pull down menu Project is already open in the project menu. To create a project, the function *Create* has to be highlighted with the cursor using the cursor keys then press the enter key. This opens a dialog box to create a project:





For our test project, enter this data in the dialog box:

Path:	c:\projects
Project Name:	PROJECT1
Description:	Practice Project 1
PLC Name:	PLC1

The data is accepted when you press the enter key. The given path is automatically created (if it doesn't already exist).
Before the project is created, you are asked to confirm this action:



When you press the enter key, the dialog box closes and the project is created. After this is finished, a dialog box appears, asking you if you want to open the project.



After pressing the enter key the project is opened and the programming system switches into the programming environment.

3.2 SCREEN COMPONENTS

After opening a project, you can call the individual functions of the **programming environment** with the pull down menus:



a) The following information is displayed in the **title bar**:
   - Program Name with version number: “B&R Programming System Vx.xx”
   - Project, PLC and possibly co-processor name in the following form: “<Project name>.<PLC Name>.<Processor Name>”
   - Current time

b) There are various pull down menus that can be opened from the **main menu bar**.

c) Different functions can be selected from the **Pull Down Menus**.

d) **Mouse Cursor:** The current mouse position is indicated with an inverted character at the relevant position.

e) **Message Window** cannot be closed, only erased.

f) There are some additional functions shown here as a **foot line menu bar** for particular windows (LAD editor, GDM, Debugger, PV Monitor).

g) Additional information is shown in the **Message Line**. (e.g. status of the connection between the PLC and PG2000).

3.3 USING THE MOUSE 

A mouse symbol is used throughout this manual to indicate which sections refer to using the mouse.

Mouse concepts:

Click ..... Put the mouse cursor on a particular position and press (quick tap) the left mouse button.

Drag ..... Put the mouse cursor on a particular position,
press and hold the left mouse button,
move mouse the mouse to the desired position,
release the mouse button.

3.4 USING THE KEYBOARD 

A keyboard symbol is used throughout this manual to indicate which sections refer to using the keyboard.

The following **syntax** is used for writing keys and key combinations:

[Key] ..... Keys are written in square brackets.

[Key 1] + [Key 2] ..... Keys that are to be simultaneously pressed are written with a "+".

[Key 1] [Key 2] ..... Keys that should be pressed one after the other but not simultaneously are separated with a space.

The following keys are used:

Engl.	Ger.	Description
[↑]	[↑]	Cursor up
[↓]	[↓]	Cursor down
[←]	[←]	Cursor left
[→]	[→]	Cursor right
[↵]	[↵]	Enter / Return
[PgUp]	[Bild↑]	Page up
[PgDn]	[Bild↓]	Page down
[Ins]	[Einfg]	Insert key
[Del]	[Entf]	Delete the cursor that in where the cursor is

Engl.	Ger.	Description
[Home]	[Pos1]	Cursor to the start of the line
[End]	[Ende]	Cursor to the end of the line
[Esc]	[Esc]	Escape (e.g. Exit a dialog box)
[Tab]	[Tab]	Cursor to the next tab stop
[Space]	[Space]	Space bar
[Ctrl]	[Strg]	Control key
[⇑]	[⇑]	⇑- shift key
[Alt]	[Alt]	Alternate key
[⇐]	[⇐]	Back Space

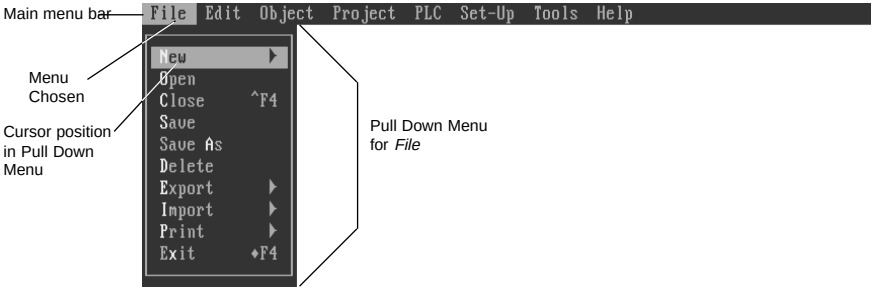
3.5 CONVENTIONS

In the menus and the help system of PG2000, special keys are represented with symbols. These conventions are as follows:

↑ ..... [⇑]
^ ..... [Ctrl]
◆ ..... [Alt]

3.6 MAIN MENU BAR AND PULL DOWN MENUS

After starting PG2000 and opening a project, the first pull down menu *File* is open:



Opening other pull down menus from the menu bar:

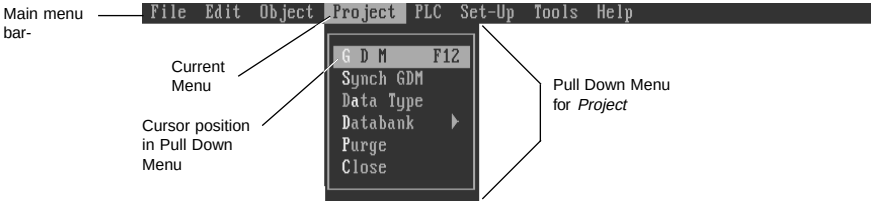


Click on the desired menu heading with the mouse.



With the [←] and [→] cursor keys, the neighbouring menus are opened.
The pull down menus can also be opened with the key combination [Alt] + [First Letter]:

[Alt] + [F] ..... *File*	[Alt] + [P] ..... *PLC*
[Alt] + [E] ..... *Edit*	[Alt] + [S] ..... *Settings*
[Alt] + [O] ..... *Object*	[Alt] + [T] ..... *Tools*
[Alt] + [P] ..... *Project*	[Alt] + [H] ..... *Help*



Selecting the function *G D M* from the pull down menu *Project*:



Click on the function *G D M*.



Mouse the cursor with [↑] and [↓] to the function *G D M* and press the [↵] key.
The *G D M* function can also be selected by pressing the highlighted letter [G].

To the right of some functions in the pull down menus, a key or key combination shown. By pressing this key or key combination, the corresponding function is called directly. Calling a function is possible whether the pull down menu is open or not. For example, the function *G D M* can be called directly by pressing [F12].

Closing an open pull down menu:

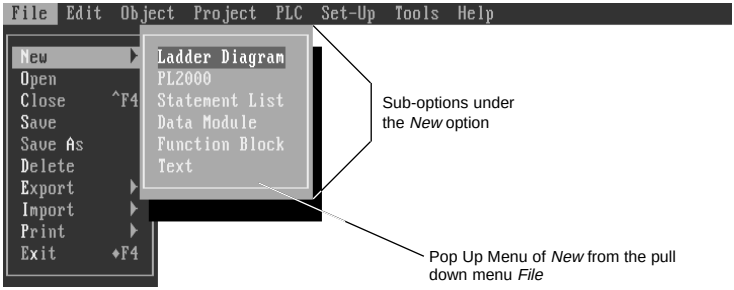


Select another object on the screen.
If the mouse cursor is moved, for example, to the message window and you press the left mouse button, the pull down menu that was open is closed.



The pull down menu is closed when you press the [ESC] key.

If there is an arrow (" ▸ ") on the right side of the pull down menu, this means that a pop up sub-menu is called when you select this function/option.
Pop up menu of the *New* option form the *File* pull down menu:



Selecting a function from a pop up menu:



Click on the function in the pop up menu.



Select the function with the keys [↑], [↓] or [First Letter] and call with [↵].

Closing a pop up menu without calling a function:



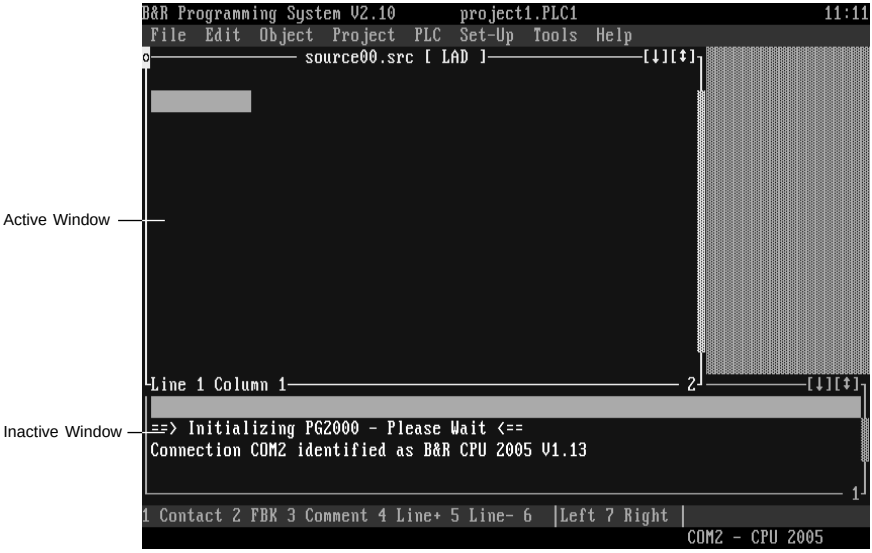
Clicking outside the pop up menu.



Press the [Esc] key.

3.7 WORKING WITH WINDOWS

This section shows you how to work with windows using the Ladder Diagram editor. In order to call the Ladder Diagram editor (LAD Editor), select the option New from the File main menu. A pop up menu appears listing all the editors. After you select Ladder Diagram, a window for the LAD editor opens:

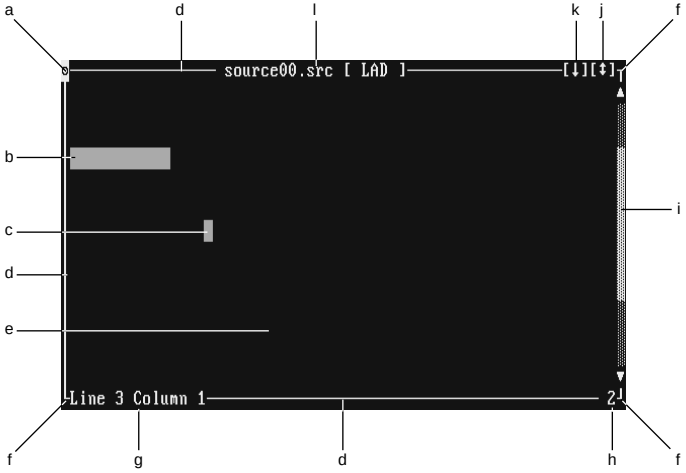


You can have **up to 10 windows open simultaneously**, but only one can be active at any one time. All other windows are inactive.

- ☞ The **active window** is recognized because it is completely in the foreground (front) and it's border is brighter that the inactive windows.

- ☞ The foot line menu shows the function key function for the currently active window. If you switch to another window, the foot line menu will change or be removed.

Window Elements



a) System menu button

b) Ladder Diagram Cursor

c) Mouse Cursor

d) Window Frame

e) Window contents

f) Window corner

g) Current cursor position or other information

h) Window number

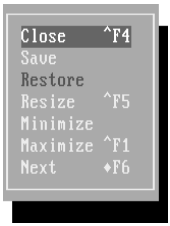
i) Scroll bars consist of a scroll bar cursor, area and arrows

j) "Maximize" button

k) "Minimize" button

l) Window or file name

System Menu



The system menu is a pop up menu that can be activated in every window. There are various window functions available in this menu (close window, resize window, etc.).

Activating the system menu of an active window:



Click on the "*System menu button (a)*".



Press the [Esc] key.

Selecting a function from the system menu:



Click on a function in the system menu.



Select a function with [↑], [↓] or [First Letter] and Call with [↵].

Closing the system menu without calling a function:



Click outside the system menu.



Press the [Esc] key.

Closing the Active Window



Select the function *Close* from the *File* pull down menu or the function *Close* from the system menu.



Press the key combination [Ctrl] + [F4].



The message window cannot be closed. Closing the message window only causes it to clear the window contents (all previous messages are deleted).

If changes were made in the open window, the following dialog box appears:



The meanings of the three buttons are as follows:

[Yes] ..... Changes will be saved. The window will be closed.

[No] ..... Changes will NOT be saved. The window will be closed.

[Cancel] ..... Changes will NOT be saved. The window will NOT be closed.

Resizing the Active Window



You can resize the window by dragging the *"window corner (f)"* to the size you want.



Select Resize from the system menu or press [Ctrl] + [F5]. The instructions are shown on the message line:

```
Cursor : move; SHFT Cursor : Resize; <ENTER> : end
```

By pressing the key combination [⇧] + [Cursor Key] the size of the windows is changed in steps. When the windows is the right size, you can exit resize mode with [↵] or [Ctrl] + [F5].

Moving the Active Window



Drag the window into position using the *"Window Frame (d)"*.



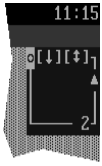
Select Resize from the system menu or press [Ctrl] + [F5]. The instructions are shown on the message line:

```
Cursor : move; SHFT Cursor : Resize; <ENTER> : end
```

The window is moved in steps with the cursor keys. When the window is in the right position, you can exit this mode with [↵] or the key combination [Ctrl] + [F5].

Minimizing the active window

This function minimizes (shrink as small as possible) the active window and moves it to the right hand side of the screen. This window is then made inactive (moved to background).



This function can only be selected when the desired window is active.



Click on the *"Minimize button (k)"*.
Pressing this button again restores the window to its previous size.



Select the function Minimize from the system menu.
The window is restored to its previous size with the *Restore* function from the system menu.

Maximizing the Active Window

The active window is maximized (made as large as possible) with this function.



Clock on the *"Maximize Button (j)"*.
Pressing this button again restores the window to its previous size.



Use the key combination [Ctrl] + [F1].
Pressing this key combination again restores the window to its previous size.



Choose the function Maximize from the system menu.
The window is restored to its previous size with the *Restore* function from the system menu.

Switching Windows

If there are several windows open at once, you can switch between them or put a particular window into the foreground.



Clicking on a window moves it to the foreground (it is then active). The mouse cursor has to be **inside** the active window frame.
If you click on the frame of an inactive window you can move and resize it - but it stays inactive.

The previously active window is made inactive and moved into the background (1 step).



You can switch windows with keys in the following ways:

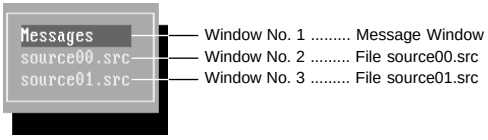
[Alt] + [Window No.] ..... The windows with the given number is activated.
(0 to 9; 0 ⇒ window number 10)

[Alt] + [F6] ..... The next window (next higher number) is activated. If there is no window with a higher number, window number 1 is activated.



Select the option Next from the system menu. It works like the [Alt] + [F6] key combination above.

Another way to switch windows is with the *Window List* in the pull down menu *Tools*. A pop up list of all open windows is shown. To get to the window you want, just select the corresponding entry from the list.
Example of a window list:



Scrolling the Window Contents

Since the entire contents of a window cannot always be shown, only a section as large as the window is displayed. It is possible to move the contents of the window with the keyboard or mouse so that a different section is shown. This moving of the currently displayed section of the window is called "scrolling".



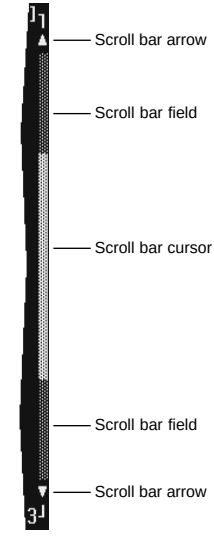
The section can be moved up or down with the scroll bars. There are two ways to do this:

1. Clicking on the **scroll bar arrows**, the window contents are scrolled up or down.

   The scroll bar arrows are only shown if there is more in that particular direction that isn't shown in the window.

2. By moving the scroll bar cursor, the desired section can be displayed.

   The size of the scroll bar cursor is proportional relationship of displayed contents and total contents. This means, the smaller the cursor, the larger the part is that is not shown.



Scroll bar



The section can be scrolled with the following keys and combinations:

[PgUp] ..... One page up

[PgDn] ..... One page down

[Ctrl] + [Home], [Home] [Home] . Go to the top (start of the window contents)

[Ctrl] + [End], [End] [End] ..... Goto the bottom (end of the window contents)

3.8 FOOT LINE MENU

There is an additional menu at the bottom of the screen (foot line menu) for some windows (LAD Editor, Function Block Editor, GDM, Variable Declaration, PV Monitor). The function key assignments for the active window are shown in the foot line menu. If you switch to another window, the foot line menu is changed correspondingly or removed.

The following picture shows the foot line menu for the Ladder Diagram editor:



Selecting a function from the foot line menu:



Click on the desired function in the foot line menu.



Call the desired function but pressing the corresponding function key [F1] to [F8].

3.9 CONTEXT SENSITIVE POP UP MENUS

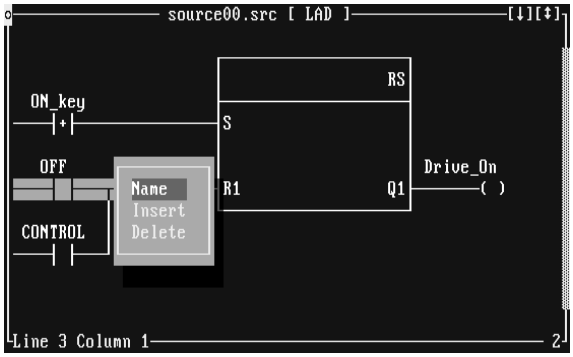
You can call up so called context sensitive pop up menus in some of the windows (LAD editor, GDM, PV Monitor):

The pop up menu appears after double clicking in the window.

[Ctrl] + [F3] calls the pop up menu of the active window.

There are different pop up menus in some windows (LAD editor, GDM) depending on the position of the cursor.

Example of a pop up menu in the LAD editor:



Selecting a function from a pop up menu:

Click on a function in the pop up menu.

Select a function with the [↑], [↓] keys or [First Letter] and call with [↵].

Closing a pop up menu without calling a function:

Click outside the pop up menu.

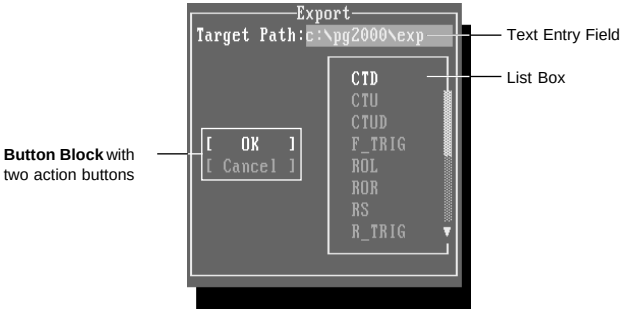
Press [Esc] key.

3.10 DIALOG BOXES

You are asked to fill out additional entries or to confirm an action for further operation via dialog boxes. A dialog box consists of several parts that do not all have to be there:

- Button Block with action buttons
- Text Entry fields
- List box (with scroll bar if necessary)
- Input box with pop up selection list

Example of a dialog box:



The effect of your entries is different depending on the dialog box. The following entries work in **all** dialog boxes:

[Ctrl] + [↵] ..... Exit the dialog box with the currently activated action button, e.g. if the [Cancel] button is active, [Ctrl] + [↵] has the same effect as [Esc].
[Esc] ..... Exit the dialog box without accepting any entries.
[Tab], [⇑] + [Tab], [↑], [↓] ..... Switch between the fields in the dialog box.

Selecting an Action button from the Button Block:



Click on the action button and you exit the dialog box with the selected action.



Select the desired action button with the cursor keys. [↵] confirms the selection, closes the dialog box and performs the selected action.

[Esc] has the same effect as the action button [Cancel].

Text Input Field

You can entry data with the keyboard if the cursor is in a text entry field.
Within the field, you can edit the text with the basic editing keys plus a few others:

[Home] ..... Move the text cursor to the start of the text
[End] ..... Move the text cursor to the end of the text (not line)
[←], [→] ..... Move the text cursor left/right
[↑], [↓], [↵] ..... Exit the text entry field. Cursor is moved to another field in the dialog box.
[Ins] ..... Toggle insert/overwrite mode
[Del] ..... Delete the character under the cursor
[⇐] ..... Delete the character to the left of the cursor

If the text is longer than the field, only a part of it is shown, but you can edit the whole text.

How to move the cursor into a text entry field so you can edit the text:



Click on the text entry field.



By pressing [Tab] several times, the cursor can be moved to the text entry field.

List Box

All entries that you can select are shown in a list box (e.g. file list box in File Selection Dialog Box). If there are more entries than can be seen in the list at once, there is a scroll bar on the right hand side of the list box.

Selecting an entry:



Click on the entry. If necessary, find the entry with the scroll bar.



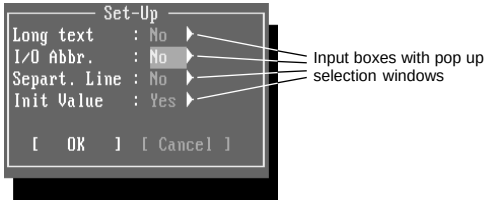
Press the [Tab] several times and the cursor is moved to the desired field.
When the cursor is in the list box, you can used the following key to select an entry:

[Home] ..... Cursor to the first entry
[End] ..... Cursor to the last entry
[↑], [↓] ..... Cyclic scrolling within the list box (wrap around)
[↵] ..... Choose the entry
[x] ..... Press a letter and the cursor is moved to the next entry that starts with this entry.

Input Box with a Pop Up Selection List



This entry field has an arrow (" ▸ ") on the right hand side.



When the cursor is in an Input box, you can select an entry from the selection list window.

Selecting an Entry:



A pop up window opens when you click on this field. You can choose an entry from this list.



Press the [Tab] several times and the cursor is moved to the desired field.
A pop up window opens when you press [Ctrl] + [F3], [Space], [←] or [→]. You can select an entry with [↑], [↓] and [↵].

Closing the Pop Up Selection List without selecting an Entry:



Click on the [Cancel] button.



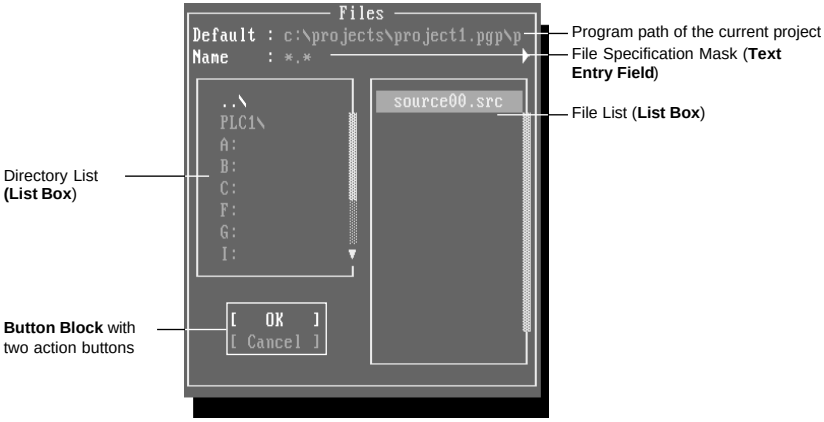
Click outside the selection window.



Press the [Esc] key.

3.11 FILE SELECTION BOX

The file selection box, one of the most frequently used, is called for all file activities (Open, Copy, Rename, ...):



Default: When the file selector box is opened, the contents of the program directory of the current project are displayed in the file list box. This default path cannot be changed while working with PG2000.

Name: A file selection mask (similar to DOS wild cards) can be entered in this text entry field. For example, if only files that start with the letter "D" should be shown, the mask is "d*.*".
You can also enter a complete file name (with or without complete path). If you enter a new mask or change into another directory, this field shows the complete path including the current mask.



If you have already done some file activities (Open,...), when you click on this field or press [Space] in the field, a pop up selection list appears. This pop up selection list contains a list of all previously used masks.

File List: All files in the current directory are shown in this list box. You can select the file you want from this list.

Directory List: All subdirectories of the current directory ("..\" is the directory "above" this one) as well as all drives are shown.
To change to another drive or directory and display the contents (according to the current mask) select the corresponding list entry.

[OK] The selected **file** (either marked in the list or typed in the *Name* field) is opened or copied (depending on operation). The file selector box is closed.
If a **directory** (or **drive**) marked, the display is changed to this directory (or drive).
If a new **mask** is entered, the corresponding file list is shown.

[Cancel] The dialog box is closed, the operation is cancelled.

3.12 GENERAL HOT KEYS

The basic keys and combination and their functions in PG2000 are as follows:

Key	Description
[F9]	Compile the selected object.
[F10]	Call the help screen that pertains to the active window or dialog box. This key can be pressed anytime.
[F11]	Call the PV Monitor for the selected object.
[F12]	Call the Graphic Design Method (GDM).
[Esc]	Call the system menu, cancel the actual function or exit from the active pull down menu or open dialog box.
[Ctrl] + [F1]	Maximize the active window. Calling the function again restores the window to its previous size.
[Ctrl] + [F3]	Calls the relevant pop up menu (within a window or input box of a dialog box).
[Ctrl] + [F4]	Close the active window.
[Ctrl] + [F5]	Turn on window resize/move mode.
[Alt] + [F4]	Exit PG2000.
[Alt] + [F6]	Switch to the next window.
[Alt] + [Window No.]	Activate the window with the given number.
[⇧] + [F5]	Search for the given mask in the selected direction.
[⇧] + [F9]	RUN - Compile selected object and download to the PLC.
[⇧] + [F11]	Call the selected object into the corresponding debugger (PL2000, STL, LAD).
[Ctrl] + [Ins]	Copy the marked area into the clipboard. Not deleted from the window.
[⇧] + [Ins]	Insert the contents of the clipboard to the editor.
[⇧] + [Del]	Move the marked area into the clipboard. Deleted from the window !

3.13 HELP SYSTEM

The PG2000 Programming System has an extensive help system so you can get information and help for the functions and operation of PG2000. There is also the possibility for you to add your own help files.

The help system is basically split up into three parts:

1) Information about the PG2000 Programming System
2) Additional help files
3) Function library help

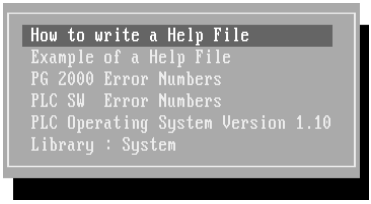
Calling with the Pull Down Menu



The help system is called with the menu options *Help*, *Additions* or *Functions* in the *Help* main menu:

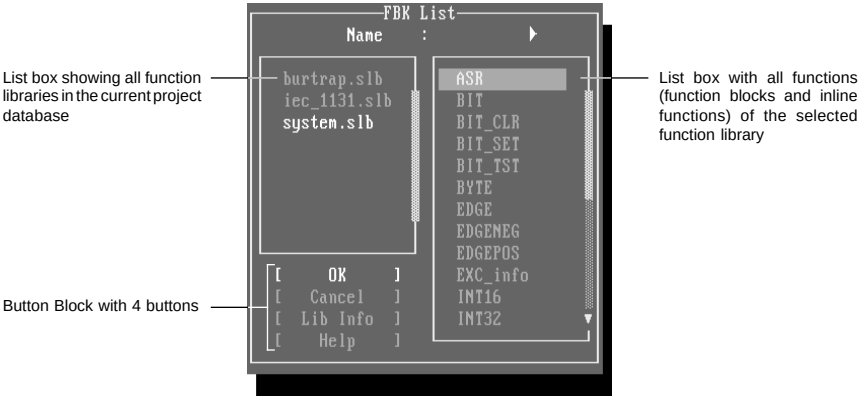
Help ..... Help System for PG2000.

Additions ..... A pop up menu appears with all additional help topics listed. You can also create and add help files to this menu.
After installing PG2000 the following topics are entered in this menu:



After choosing a topic, the corresponding help file is shown on the screen.

Functions ..... A dialog box showing all function libraries in the current project data base is displayed:



[OK] ..... Information about the selected function is shown (function block or inline function).
[Cancel] ..... The dialog box is closed.
[Lib Info] ..... A dialog box with information about selected function library.
[Help] ..... Help for the selected function library is shown.

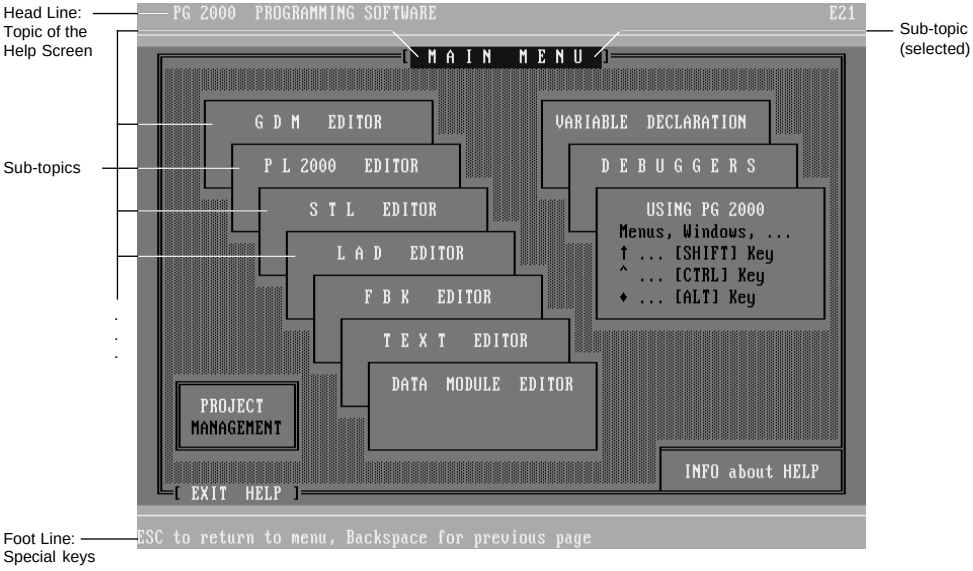
Calling with the Function Keys

You can directly call the PG2000 help system with the [F10] key at any time. This calls the "context sensitive" help, i.e. the help screen corresponding to what you are currently doing in PG2000.

Examples:
- If the PL2000 editor is active, information about the PL2000 editor is shown.
- If a dialog box is active, the instructions for how to work with dialog boxes is shown.
- If the variable declaration is active, information about the variable declaration is shown.

Using the Help System

Within the help system, you can go to other topics or go back to the last topic. When you select the Help option in the Help menu, the following screen appears:



Selecting A Topic

The bright positions in the text are topics or sub-topics that can be called up. You can choose them with the keyboard or mouse:



If the mouse cursor is moved over a bright topic button, it is marked and highlighted. The topic is selected by clicking on it and the corresponding topic is called.



You can select a topic button with the cursor keys or the first letter of the topic button name. The corresponding help text is called with [↵].

You can go back to the previous help screen by pressing the [⇐] key.

Exiting the Help System



If there is an "Exit HELP" button, you can click on it to leave the help system.



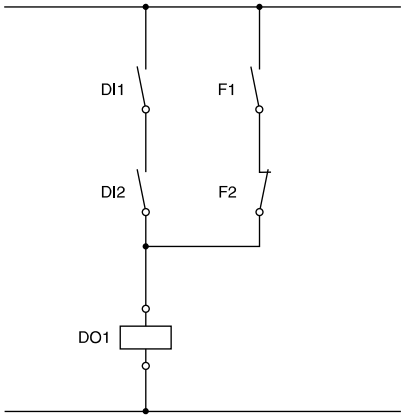
You can exit help with [Esc].

4 INTRODUCTORY EXAMPLE

To help you quickly grasp the basic functions of PG2000, we'll go through an example in the section.

4.1 EXAMPLE DESCRIPTION

Create a project called "TEST_PRJ" in the "C:\PROJECTS" with a default PLC called "TEST_PLC". After you have done this, open the project and create the circuit shown in the picture below in the programming language Ladder Diagram.



4.2 HARDWARE REQUIREMENTS

You need the following hardware components (independent of the system used) to do this example:

System B&R 2010:

- 1 Base plate module for the CPU module
- 1 Base plate for I/O modules
- 1 CPU Module (CPU Slots)
- 1 Digital input module - e.g. module with 32 digital inputs (in slot 1)
- 1 Digital output module - e.g. module with 16 digital outputs (in slot 2)
- 1 Power supply module (in the right most slot of the I/O bus)
- 1 ONLINE cable

System B&R 2005:

- 1 Base plate module
- 1 Power supply module (Slots 1 and 2)
- 1 CPU Module (Slots 3 and 4)
- 1 Digital input module - e.g. module with 16 digital inputs (in slot 5)
- 1 Digital output module - e.g. module with 16 digital outputs (in slot 6)
- 1 ONLINE cable

You can do the example without the hardware, but you can't download it to the PLC.

4.3 CREATING THE PROJECT

To create a new project, you have to be in the project management menu:

- The Programming System PG2000 is automatically in the project management menu if this is the first time you have used it.
- If you already have a project open (you are in the programming environment), this project has to be closed so you can switch to the project management menu. You can do this by selecting the option *Close* from the pull down menu *Project*.

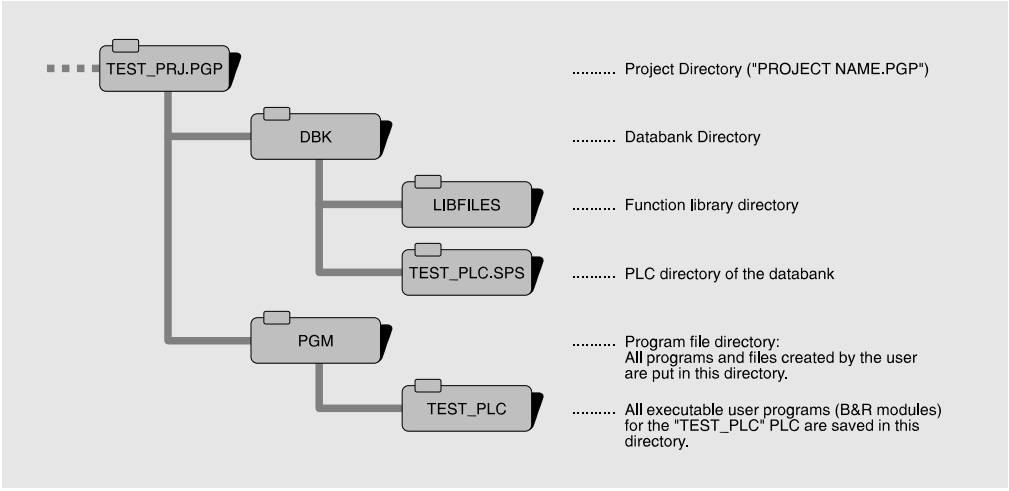
Now you are in the project management menu. Create a project with the Create function from the pull down menu project (you can find more detailed information in the "Project Management" section). You should enter the following information in the dialog box:

Path : c:\projects
Project Name : TEST_PRJ
Description : Test Project
PLC Name : TEST_PLC

When you press the [OK] button the data is accepted by the programming system. The following dialog box appears to confirm the entries:

Create project c:\projects\TEST_PRJ.pgp with PLC TEST_PLC ?

After confirming the project creation with the [Yes] button, the project is created. The programming system created the following directory structure under the given path (C:\PROJECTS):



Then a dialog box appears asking you if the project should be opened. When you press [Yes], the project "TEST_PRJ" is opened and the programming system switches from project management to the programming environment. In the programming environment, you can create, write, compile and download the ladder diagram to the PLC.

4.4 CALLING THE LAD EDITOR VIA THE GDM

The ladder diagram can be called directly (function New from the pull down menu File) or with the GDM (Graphic Design Method). However, it is recommended that you always use the GDM for programming. This is because it increases the clarity of the automation project and helps you program in a more structured way.

The GDM structures the project in several layers:

PLC Layer All control units (PLC) in this application (Project) are shown on this layer.

Processor Layer The processor layer is optional. For control units that have only one processor module (CPU), this layer is not necessary and does not appear. If there are several processors on this PLC (multiprocessors or intelligent I/O processors), the processor layer is inserted between the PLC and task layers.

Task Layer All application programs (PLC, idle time, exception and interrupt tasks) of this processor are shown.

On the following pages, we will only go through how to call the ladder diagram editor through the GDM. We don't go any further into the GDM than is necessary. There is a detailed description of the GDM in the section entitled "GDM - Graphic Design Method".

Calling the GDM

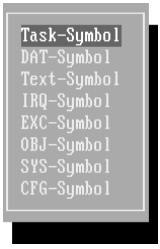
After selecting the function *G D M* from the pull down menu *Project*, a window opens. This window shows the PLC layer of the current project:



There is already a symbol for the current project's default PLC "TEST_PLC" entered and marked in the PLC layer. Since the control unit (PLC) has only one processor module (CPU) there is no processor layer.

Calling the Task Layer

By pressing the function key [F7 Layer↓], the task layer of the PLC "TEST_PLC" is called. A task symbol for the application program has to be put in the task layer to create it. Press [F1 Symbol] and a symbol cursor is shown at the current cursor position. The symbol cursor can be moved to the desired position with the cursor keys.
After selecting the position with [↵], a pop up menu appears to choose the task type. Select "Task symbol" from the available options for this example (the meanings are explained in the section "GDM - Graphic Design Method").



A window opens after you have chosen the task type, for you to enter the name.

Following this you have to enter the task parameters in a dialog box.

You can choose the programming language of the source file with a pop up menu in the "Source Type" input field. Select Ladder Diagram as we are going to write the program for the circuit diagram in this language. The settings for the other task parameters don't have to be changed for this example (the meanings of the other task parameters are explained in the section "GDM - Graphic Design Method").



By pressing the [OK] button, the task parameters are accepted by PG2000 and the dialog box is closed.

Calling the LAD Editor

In order to create the source file for the task, you have to go one more level down with [F7 Layer↓] so you can call the corresponding editor (in our case, the LAD editor).

4.5 CREATING THE LADDER DIAGRAM

The given circuit diagram should now be drawn in the open LAD editor

Setting a Contact

When you press the function key [F1 Contact], a pop up menu appears containing the various types of contact symbols. For the digital inputs and digital internal variables the 1 —| |— symbol has to be used (or 2 —|/|— for negated contact). For digital outputs the 4 —( )— symbol has to be used.

Choosing Symbolic Names

After you have selected the contact type, you have to enter the symbolic names of the contact.
For each variable used (internal variable, output, input, ...) in PG2000, you have to use a symbolic name. You can write (draw) the solution to a problem without any reference to the hardware. The assignment is done in the Variable Declaration before you create the executable program module. In our example, we'll use the following names for digital inputs, outputs and internal variables:

DI1 ..... Digital Input 1
DI2 ..... Digital Input 2
F1 ..... Digital Flag 1
F2 ..... Digital Flag 2

DO1 .... Digital Output 1



Symbolic names are case sensitive, i.e. **DI1** and **di1** and two completely different names !

Link Lines Between Contacts

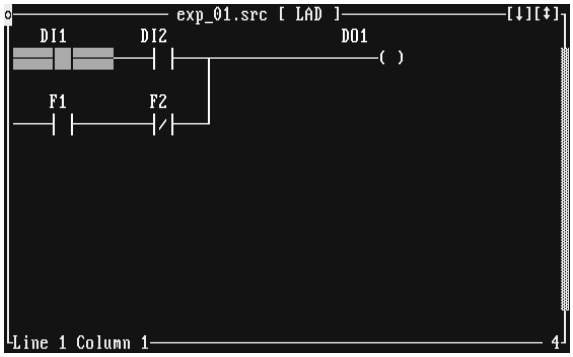
You can use the following keys to draw link lines between the contacts:

[F6 | left] ..... This function draws a line from the left of current cursor position up to the next symbol (line or contact) or deletes an existing vertical line.

[F7 right |] ..... This function draws a line from the right of current cursor position up to the next symbol (line or contact) or deletes an existing vertical line.

[Space] ..... The symbol at the current cursor position is deleted. If there is no symbol in this position, a horizontal link line is drawn.

The ladder diagram for the circuit diagram given looks like this:

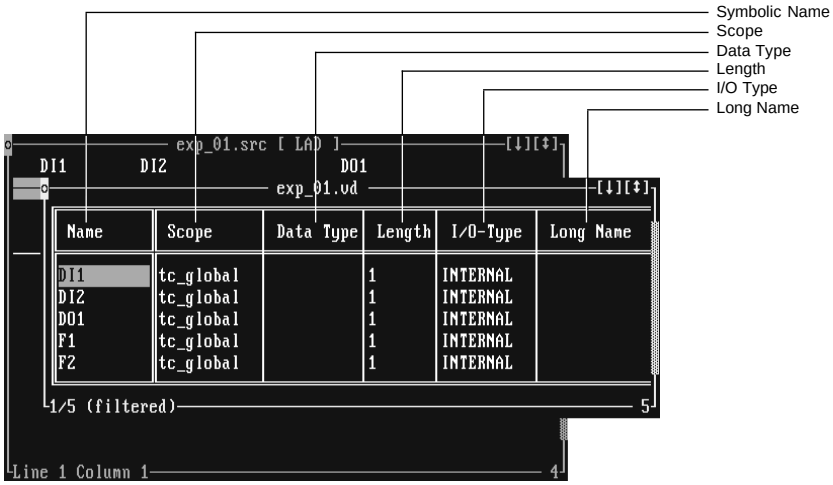


4.6 SAVING THE LADDER DIAGRAM

To save your ladder diagram source file with the name "EX_01.SRC", select the function *Save* from the *File* pull down menu. The LAD is parsed for errors (checked for correctness) when it is saved. After saving, a dialog box appears with a warning that there are still undeclared process variables (PVs). You should close this dialog box with the [OK] button and the variable declaration appears automatically.

4.7 VARIABLE DECLARATION

All variables that are used in the LAD are shown in a window ordered alphabetically. In this case, the symbolic names cannot be changed in the list.



The reference to the hardware is made in the variable declaration. There is a detailed description of the variable declaration in "^.10 Variable Declaration".
The cursor can be moved to the desired field with the [↑], [↓], [↵] and [Tab] keys. For our example, you should make the following entries:

Symbolic Name:

Only the names used in the source file are shown in this field. You cannot change names or add any new ones.

Scope:

The default "tc_global" does not have to be changed for this example.

Data Type:

When the cursor is in this field and you press [Space] or click with the mouse, a pop up menu opens showing the different data types. Use the data type "BIT" for all digital contacts.



Length:

The cursor moves automatically to the Length field after you have entered the data type. Do not change the default value, "1" for this example.

I/O Type:

When the cursor is in this field and you press [Space] or click with the mouse, a pop up menu opens showing the different I/O types.

B&R 2010 System

When you select the I/O type "2010 I/O" (B&R 2010 System), another window opens so you can enter the module address, module type and channel number for each variable.
To reference the hardware used, the following entries have to be made for the 5 variables:

	I/O Type	Module Adr.	Module Type	Channel No.
DI1	2010 I/O	1	Digit. In	1
DI2	2010 I/O	1	Digit. In	2
DO1	2010 I/O	2	Digit. Out	1
F1	INTERNAL	---	---	---
F2	INTERNAL	---	---	---

When the cursor is in the Module Type field, you can open a window with [Space]. This window contains all module types. In our example select the module type "Digit In" (for digital inputs) or "Digit Out" (for digital outputs).

System B&R 2005

When you select the I/O type "2005 I/O" (B&R 2010 System), another window opens so you can enter the base plate module address, module type and channel number for each variable.

To reference the hardware used, the following entries have to be made for the 5 variables:

	I/O Type	Base Plate	Module Adr. [1)]	Module Type	Channel No.
DI1	2005 I/O	Main	3	Digit. In	1
DI2	2005 I/O	Main	3	Digit. In	2
DO1	2005 I/O	Main	4	Digit. Out	1
F1	INTERNAL	---	---	---	---
F2	INTERNAL	---	---	---	---

When the cursor is in the Base Plate field, you can open a window with [Space]. This window contains all base plate types. In our example you don't have to change the default selection "Main".

When the cursor is in the Module Type field, you can open a window with [Space]. This window contains all module types. In our example, select the module type "Digit In" (for digital inputs) or "Digit Out" (for digital outputs).

[1)] The module address is defined by the slot (slot coding). Numbering of the module addresses on the B&R 2005 System starts in slot 3 with address 1.

Long Name:

You can enter a short description for the variable in this field.

When all variable are declared, save the variable declaration, e.g. with the *Save* function from the Pull down menu *File*. The variables and hardware assignment are stored in the project data bank when saving. You can now close the variable declaration with the function *Close* from the *File* pull down menu.

4.8 CONNECTING TO THE PLC

You have to make a connection between the Programming System (PC) and the PLC before you can download the example program to the PLC.:

- Connect the PC and PLC with the ONLINE cable

  PC: COM1 or COM2, as set in the configuration file PG.INI:
  nw_p1_19.br ..... Communication via COM1 with 19 200 Baud
  nw_p1_57.br ..... Communication via COM1 with 57 600 Baud
  nw_p2_19.br ..... Communication via COM2 with 19 200 Baud
  nw_p2_57.br ..... Communication via COM2 with 57 600 Baud

  PLC: B&R 2010 ..... With PLCSW version 1.10 and later, all interfaces on the CPU (or multiprocessor) can be used for programming. [1)]
  B&R 2005 ..... With PLCSW version 1.10 and later, all interfaces on the CPU can be used for programming. [1)]

- Connection is made (starting communications) by ...

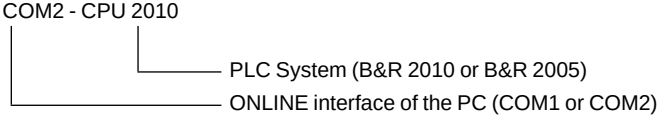
  ... either automatically by opening a project (if there is a PLC connected that is turned on and the option "NW_START = Yes" is set in the configuration file PG.INI) or automatically in one of the cyclic attempts (via the interface specified in PG.INI).

  ... manually selection the function *Connection* from the pull down menu *Set-Up.*

If a successful connection has been made, information about the active connection is shown in the message window and message line:

Connection COM2 identified as B&R CPU 10`0 V1.10

Connection Information Syntax on the Message Line:



[1)] You have to use an RS422/RS232 converter if you use the RS485/RS422 interface.

4.9 COMPILING AND DOWNLOADING TO THE PLC

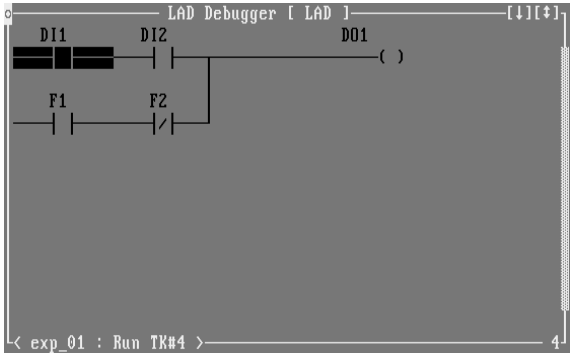
The ladder diagram and variable declaration have to be translated into the machine language of the processor. PG2000 has built-in a compiler to do this job. The compiler produces an executable program module (B&R module) in the machine language of the processor.
To compile our example and download it into the PLC, select the function *RUN* from the pull down menu *Object*. This compiles the program into a B&R module, downloads the executable module to the User RAM of the PLC and starts it.
If the compilation and download to the PLC are error-free, the following message appears in the message window:

```
**** LAD Parser    EX_01
**** LAD Parser    Ok
**** TASKBUILDER   EX_01
**** TASKBUILDER   Ok.
Download EX_01
Download complete (M=$070E, T=$0580)
```

The LAD is now running the User RAM. The digital outputs are set or cleared depending on the logical condition of the inputs and flags.

4.10 TESTING THE EXAMPLE PROGRAM WITH THE LAD DEBUGGER

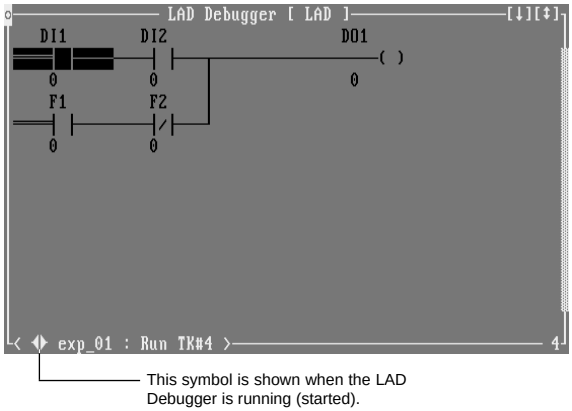
Select the *Debugger* function from the pull down menu *Object* to debug the ladder diagram that is running on the PLC:



You can directly monitor and write to variables in the ladder diagram on the PLC with the LAD Debugger. The LAD mask is shown on the screen.
When you start the LAD debugger it is stopped, i.e. the variable values are not displayed.

Starting the LAD Debugger

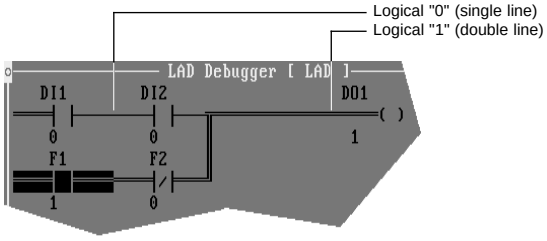
Press the function key [F1 Start/Stop] to start the LAD Debugger. The debugger is stopped if you press the [F1 Start/Stop] key again. The variable values shown are only updated when the LAD debugger is running (started).



When the LAD Debugger is running, the values of the individual variables are cyclically read from the PLC and displayed under their corresponding symbols.

Writing to a Variable

The variable (contact symbol) under the cursor can be written to with the function key [F3 Write]. Type in the desired value in the pop up window. The new value of the variable is displayed under the corresponding symbol in the LAD Debugger. Link lines (connection paths) that are logical "1" are shown as a double line.



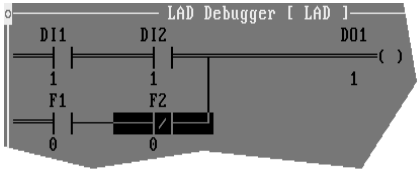
The state of the digital inputs DI1 and DI2 are read cyclically from the PLC (input image) and shown in the LAD debugger. By using the Force function of the LAD Debugger (function key [F4 Force]) you can change or set the state of inputs that are assigned to an input module with PG2000. There is detailed information on the "Force" function is in the sections "6.16 Process Variable Monitor" and "6.17.1 Ladder Diagram Debugger (LAD Debugger)".

Testing the Logic of the Ladder Diagram

To test the correctness of the LAD, simple change the values of the digital flags and inputs in the LAD debugger and watch the effects on the digital output. You should come up with the following results after testing all possibilities:

DI1	DI2	F1	F2	DO1
0	0	0	0	0
1	0	0	0	0
0	1	0	0	0
1	1	0	0	1
0	0	1	0	1
1	0	1	0	1
0	1	1	0	1
1	1	1	0	1
0	0	0	1	0
1	0	0	1	0
0	1	0	1	0
1	1	0	1	1
0	0	1	1	0
1	0	1	1	0
0	1	1	1	0
1	1	1	1	1

The following passage describes how to test one of the 16 variations:
First of all, set the digital inputs DI1 and DI2 to logical "1". Now write to the flags F1 and F2 with the value "0". This results in the digital output DO1 being set. This is shown when by the corresponding LED on the digital output module.



Exiting the LAD Debugger

There are two ways to exit the LAD Debugger:

- Close the Debugger window.
- Press the function key [F8 Editor]. This closes the debugger and switches to the LAD editor with the current ladder diagram.

5 PROJECT MANAGEMENT

The first time you start the PG2000 Programming System, you have to select a project. To make it absolutely clear that no project is selected, a reduced main menu (**Project Management**) with only the project functions are shown. With this menu you can search for existing projects, create new projects, duplicate existing projects and delete no longer needed ones:

PROJECT MENU
Create ..... Create a new project
Open ..... Open an existing project
Copy ..... Copy an existing project
Purge ..... Delete all backup files in a project
Delete ..... Delete a project (project directory and all projects are deleted)
Search ..... Search for and include all projects that are not in the project management.
Exit ..... Exit from the Programming System

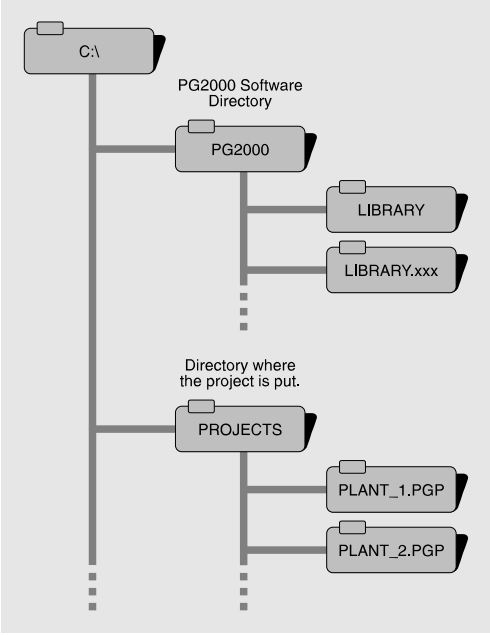
When you select a project (Open), the project menu is replace with the complete main menu (**Programming Environment**).

5.1 CREATING A PROJECT

Call the function **Create** from the pull down menu **Project** to create a new project. You can enter the required data for the creation of the project in a dialog box:

Path: **c:\projects**
Project Name: **Plant_1A**
Description: **Machine Plant 1A**
PLC Name: **2005**

Path ..... An individual directory structure is created on your hard drive for each new project. You can enter the path under which the new project is to be saved. To clearly separate the project data from the development tools (PG2000 Programming System), we recommend that the project directory structure is not put directly under the PG2000 directory:



Project Name ..... Name of the new project (8 character max.).
The project structure is put in the directory "path\project name.PGP".

Description ..... You can enter a short description (30 characters max.) for the new project.

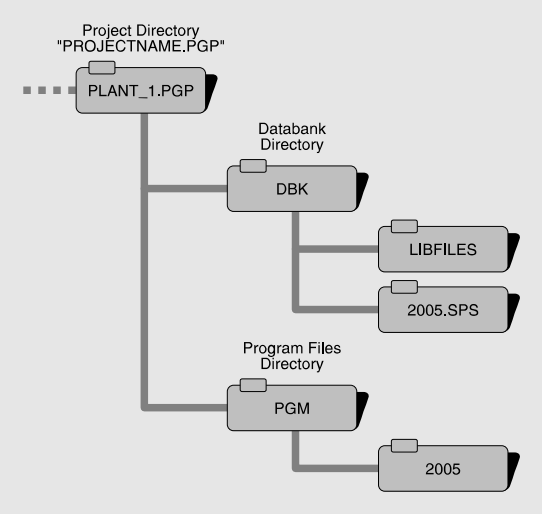
PLC_Name ..... Name of the default PLC of the new project (8 character max.).

This data is accepted by PG2000 when you press the [OK] button.

There is a confirmation question before the new project is created so you can check the information from the dialog box:

Create project c:\projects\plant_1.pgp with PLC 2005 ?

After confirming with the [Yes] button, the project "plant_1" is created. The following directory structure is created by PG2000 under the given path:



The data bank is then initialized (all internal data for the default PLC is setup and the system library is imported into the data bank). The data bank initialization allows you to open the new project and use the function in the system library.

Following the initialization of the data bank, you are asked if you want to open the project (in a dialog box):

Open Project Plant_1 ?

If you say [Yes] the project "Plant_1" is opened and PG2000 switches from project management to the programming environment.

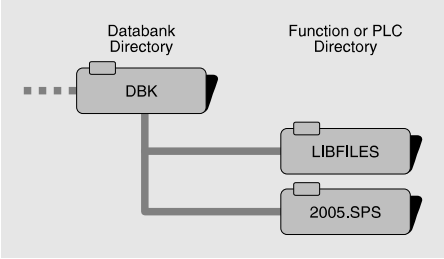
A project contains all information for an application, i.e. all files created by you (source files, executable program modules, etc.) with the corresponding data bank. Individual subdirectories are created for the project data bank and the program files when you create a project.

5.1.1 Project Databank

Each project has its own data bank. All project information that is not saved directly in the source files is stored in the data bank:

- A list of all variables used and their hardware assignment
- Data types
- A list of all tasks used in the project
- Parameters for each task (task class, Inst number, etc.)
- A list of the I/O modules used
- Cross reference list
- Function libraries of the project
- etc.

The project data bank is saved in the data bank directory. The following picture shows the data bank directory for the newly created project "Plant_1":



Data bank Directory (DBK)

All files that are required for the complete data bank are in this directory:

*.DAT ..... Data bank data files
*.IDX ..... Data bank indexes

When you open a project, these files and the information in them is checked for correctness. If there are missing files or incorrect information, PG2000 will attempt to correct the problem. If the problem cannot be corrected, the project is not opened and you are returned to the project management menu.

Function Directory (LIBFILES)

All functions of a data bank are stored in this directory.

*.BFU ..... User created function blocks (compiled source code and description)
*.BR ..... Function library runtime modules
*.SLB ..... Function libraries
*.049 ..... German help files for the function libraries
*.001 ..... English help files for the function libraries

Before you can use the functions of a library in a project, you have to import the library into the corresponding project data base. The function to do this is pull down menu *File*, function *Import* and option *Library* in the programming environment.



The system library is automatically imported into the project data bank when you create a project. This means you can use the system library functions immediately after you have created a project.

There is a detailed description of the function libraries in the "B&R SYSTEM 2000 Library Reference Manual".

PLC Directory (<PLC Name>.SPS)

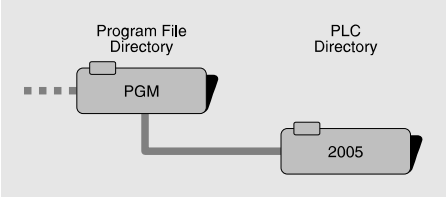
There is an individual PLC subdirectory underneath the data bank directory for each PLC used in a project. After creating a project, there is already a PLC directory for the default PLC. All information relating to the PLC is stored in this directory. This directory is called the data bank path in PG2000.

*.DAT ..... Data bank data files
*.IDX ..... Data bank index lists

The Graphical Design Method (GDM) is the way to effectively work with the data bank. The GDM shows the structure of the individual PLCs. You can also change the structure of the PLCs, change parameters and add new PLCs (including the PLC directories).

5.1.2 Program Files

The programs written and developed by you are not saved in the data bank. They are saved in the program files directory. The source files and the executable program modules are both saved in this directory. The program files directory for the newly created project "Plant_1" looks like this:



Program Files Directory (PGM)

All source and text files are saved in the PGM directory (when no other one is specified). All backup files are also put in this directory:

*.DAT ..... Data module source files
*.FUB ..... Function block source files (user created function blocks)
*.GDM ..... Graphical information of the GDM
*.SRC ..... PL2000, STL and LAD source files
*.TXT ..... Text files (from the text editor)
*.VLD ..... Variable lists
*.?_? ..... Back up files

This directory is also called the *Program File Path* in PG2000.

PLC Directory (<PLC Name>)

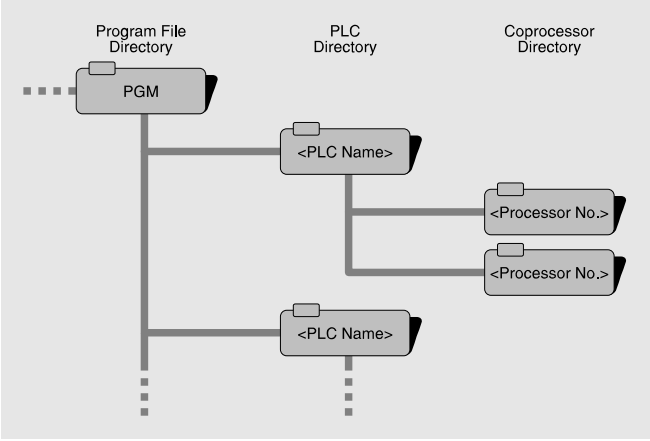
An individual PLC subdirectory under the program file directory is created for each PLC in the project. A PLC directory is created for the default PLC when the project is created. All executable program modules (B&R modules) of the corresponding PLC are stored in this directory:

*.BR ..... executable program modules (B&R modules)
*.BAK ..... Back up files

All system modules and operating system options for this PLC that are not on the system ROM are also put in this directory (e.g. extended exception handler, remote I/O firmware, etc.).

Co-Processor Directory (<Processor Name>)

If there are processors (multiprocessor or intelligent I/O processor) other than the CPU in a PLC, an individual co-processor directory is made under the corresponding PLC:



All executable program modules (B&R modules) that are for this co-processor (multiprocessor or Intelligent I/O processor) are stored in this directory:

*.BR ..... executable program modules (B&R modules)
*.BAK ..... Back up files

5.2 OPENING A PROJECT

To open a project, select the function Open from the pull down menu Project. This causes a pop up list of all projects in the project management to appear (project selection list).

After selecting a project, the project menu (project management) is replaced by the complete menu bar of the programming environment. The programming environment offers you an integrated programming, compiling and debugging tool.

5.3 COPYING A PROJECT

You can copy a project (including all project data) with the function *Copy* from the pull down menu *Project*. You might use this function to create a back up copy of an entire project or to duplicate a project to make a similar one. When you call this function, a pop up list of all project currently being managed appears. Select the desired source project from this list.
After this a dialog box appears, you can type in the additional information about the target copy - path, new name and new description:

Path ..... Enter the directory that a new project subdirectory will be created under. To clearly separate the project data from the development tools (PG2000 Programming System), we recommend that the project directory structure is not put directly under the PG2000 directory (see section "5.1 Creating a Project").

Project Name ..... Name for the target project (max. 8 characters).
The project structure is created under the directory "path\project name.PGP".

Description ..... You can enter a short description (up to 30 characters) for the target project.

This information is accepted by PG2000 when you press [OK] and the source project is copied to the target project. The new project is automatically included in the project management. You are then asked if the new project (target project) should be opened. Select [Yes] and the project is opened and the menu system changes from project management to the programming environment.

5.4 INCLUDING PROJECTS IN PROJECT MANAGEMENT

You can include "unmanaged" project in the project management with the Search function from the pull down menu Project (unmanaged projects are on the hard disk but are not shown in the project selection list and can therefore not be opened). You would use this function, for example, to include a project that was copied to the hard disk from a floppy in DOS.

When you call this function, a dialog box appears for you to enter the starting directory (or drive) that PG2000 should search through for "unmanaged" projects.

After pressing [OK], PG2000 searches the given search path (or drive) and all its subdirectories for projects that are not currently recognized.

If PG2000 finds an "unmanaged" project a dialog box with the projects basic data appears for you to accept or change:

Project Name ..... Name of the project found.

Description ..... You can enter a short description of the project here (max. 30 characters).

PLC_Name ..... Name of the default PLC in the project found.

PG2000 accepts the project information and includes the project in the project management. PG2000 then continues looking for more projects in the *search path* until it reaches the end.

5.5 PURGING A PROJECT

When you want to delete all back up files of a project (*.BAK and *.?_? files), select the function *Purge* from the pull down menu *Project*. You can select the project to clean up from a project selection list.
You are asked to confirm your selection :

Delete all backup files ?

When you agree to this with [Yes], all the back up files in the project are erased.

5.6 DELETING A PROJECT

To delete a project (including all project data and directory structure), select the function Delete from the pull down menu Project. A pop up list of all projects then appears for you to select the project.
After selecting the project you are asked for confirmation:

Really delete project c:\projects\plant_2.pgp ?

The selected project is deleted if you answer [Yes] to this question.



Warning: All project data and data bank information is lost when you delete a project. The directory structure is completely removed !

6 GDM - GRAPHICAL DESIGN METHOD

6.1 GENERAL INFORMATION

The GDM is an assistant for planning and project analysis of automation jobs. Using the GDM, you can split up an automation project into many small parts and define the relationship between these parts. You can plan you whole project without referencing the hardware.

The GDM structures a project in several layers:

PLC Layer ..... All control units (PLCs) in this application (project) are shown as objects (symbols) in this layer. If you select an object in the PLC layer, you can go to the layer under this one (child layer) with a function key. If this is a system with only one processor, you go straight from the PLC layer to the task layer of this PLC. If this is a multiprocessor system, you go to the processor layer.

Processor Layer ..... The processor layer is optional. If the PLC has only one processor module (Main CPU), this layer is not necessary and therefore nonexistent. If there is more than one processor (multiprocessor or intelligent I/O processor), the processor layer between the PLC and task layers is inserted. All processors of the corresponding PLC are shown as objects (symbols) in this layer.

Task Layer ..... All application programs (PLC, idle time, exception and interrupt tasks) for the corresponding processor are shown as objects (symbols).

This hierarchy structure is a great help for project clarity and structured programming.

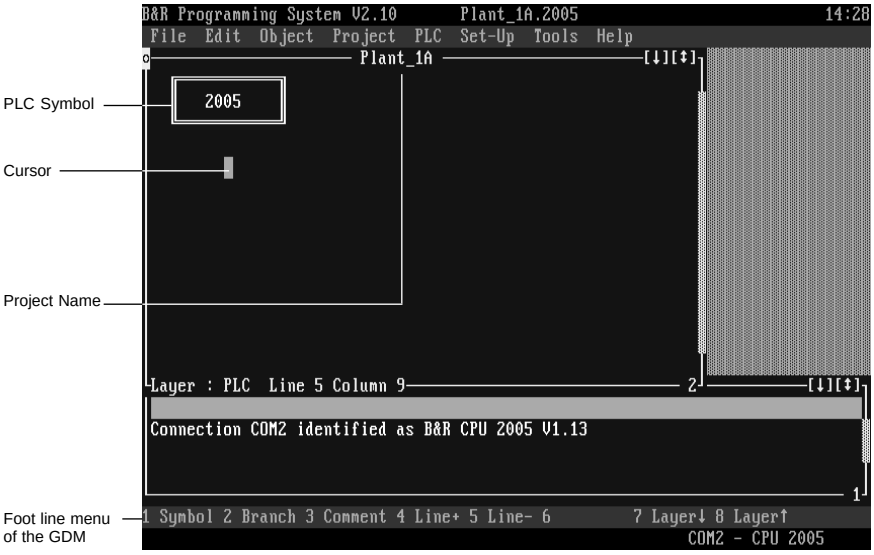


We recommend that you always use the GDM because it increases the clarity and understanding of the automation problem and helps you write more structured programs.

6.2 CALLING THE GDM

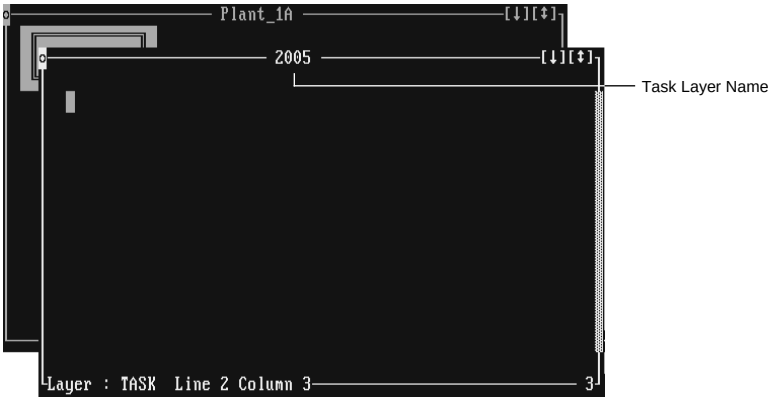
When you call the function *GDM* from the pull down menu *Project*, a window opens. This window shows the PLC layer of the current project ("Plant_1"). There is already one symbol for the default PLC ("2005") of the current project in the window. Behind this symbol, on the DOS level, there is the PLC directory "2005.SPS" (data bank path) with the information specific for this PLC.

The GDM window of the PLC layer always has the window name <Project name> and could look like this:



6.3 CALLING THE TASK LAYER

If the cursor is on a PLC symbol, press the function key [F7 Layer↓] to go one layer deeper. If this PLC has only one processor module (CPU) you switch straight to the task layer of this PLC. The first time you call the task layer, the following window appears:



If you successfully switched from the PLC symbol into the task layer, the following message is shown in the message window:

```
Databank path switched to PLC "<PLC Name>"
```

On a multiprocessor system, when you press [F7 Layer↓], you switch form the PLC layer to the processor layer. To activate the task layer for a processor, put the cursor on the processor you want and press the function key [F7 Layer↓] again.

You can create a multiprocessor system from a single processor system with the function Insert child layer from the pop up menu of the PLC layer.

PLC Layer Pop Up Menu

- GDM Check
- Object List
- Cross Reference
- Entry Layer
- Rename Symbol
- Move Symbol
- Optimize Object
- Download Object
- Delete Child Layer
- Insert Child Layer

6.4 PLACING AN OBJECT IN THE TASK LAYER

All application programs (PLC, idle time, exception and interrupt tasks), data modules, etc. of the corresponding processor (CPU module or co-processor) can be shown in symbol form in the task layer.
To place a task symbol, use the [F1 Symbol] function from the foot line menu. This causes a symbol cursor to appear at the current cursor position. The symbol cursor can be moved with the cursor or mouse click on the desired position. After confirming the position with [↵], a pop up menu appears for you to select the type of task:

Task Symbol ..... Set a task symbol for a normal or high speed task. The task class of the task is shown on the top right corner. (e.g. TC#1 = task class 1, HS#1 = high speed task class 1).
After selecting this task type, a window opens for you to type in the task name (up to 8 chars.). All capitals are converted to lower case. Within a task layer, each task name can only be used once.
If the given task name is not in the project data bank yet, the task parameter dialog box appears after the task is placed. You can find detailed information about task parameters in the section "6.5.1 Parameters for Normal, High Speed, IRQ and EXC tasks".

DAT Symbol ..... Place a data module symbol. After selecting this task type, a window opens for you to type in the data module name (up to 8 chars.). All capitals are converted to lower case. Within a task layer, each data module name can only be used once.

Text Symbol ..... Place a text symbol. After selecting this task type, a window opens for you to type in the text symbol name (up to 8 chars.). All capitals are converted to lower case.

IRQ Symbol ..... Set a task symbol for an interrupt task. The task class of the task is shown on the top right corner (IRQ = interrupt task class). After selecting this task type, a window opens for you to type in the task name (up to 8 chars.). All capitals are converted to lower case. Within a task layer, each task name can only be used once.
If the given task name is not in the project data bank yet, the task parameter dialog box appears after the task is placed. You can find detailed information about task parameters in the section "6.5.1 Parameters for Normal, High Speed, IRQ and EXC tasks".

EXC Symbol ..... Set a task symbol for an exception task. The task class of the task is shown on the top right corner (EXC = exception task class). After selecting this task type, a window opens for you to type in the task name (up to 8 chars.). All capitals are converted to lower case. Within a task layer, each task name can only be used once.
If the given task name is not in the project data bank yet, the task parameter dialog box appears after the task is placed. You can find detailed information about task parameters in the section "6.5.1 Parameters for Normal, High Speed, IRQ and EXC tasks".

OBJ Symbol ..... Place a symbol for a B&R module (e.g. compiled task, run time modules of the function libraries, exception handler, etc.). This symbol is put in when you perform a GDM synchronization (see section "6.23 Synchronizing the PLC and GDM"). If there is rebuild info in the B&R module, you can restore the source information (source code, INIT-SP, task parameters, variable declaration, etc.). of the task with the function *Task Rebuild* (pop up menu of the function *Import* in the pull down menu *File*) or with an *Object Rebuild* (pop up menu of the task layer).

SYS Symbol ..... Place a symbol for a B&R module (e.g. run time module of a function library, exception handler, etc.). After selecting this type of task, a pop up menu appears with a list of all system modules (e.g. burtrap library BURTRAP.BR, mathematics library MATHTRAP.BR, etc.) and operating system options (e.g. exception handler EXERMO.BR, etc.) installed in the system and not included in the task layer yet. When you make a selection, the symbol is placed in the GDM and the corresponding *.BR file is copied into the module directory of the current CPU (..\<ProjectName.PGP>\PGM\<PLC Name>\) or current processor (..\<Projectname.PGP>\PGM\<SPS-Name>\<ProcessorName>\).

CFG-Symbol ..... Places a symbol for the PLC configuration module. The name "sysconf" is automatically used.
You can find detailed information about the PLC configuration module in the section "6.5.3 Parameters of the PLC Configuration Module".

6.5 OBJECT PARAMETERS

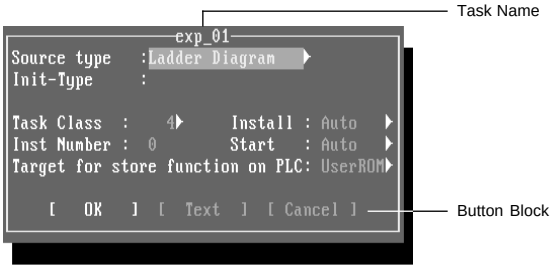
6.5.1 Parameters for Normal, High Speed, IRQ and EXC Tasks

So-called **Task Parameters** have to be defined for all task, interrupt task and exception task symbols used in a project. This means that if there is a task name of any type of task symbol that doesn't exist in the project databank, a dialog box is automatically opened after the symbol is placed for the relevant information to be entered (source code type, task class, etc.). The dialog box for the task parameters can also be called with the function *Parameters* from the *Object* menu.
The task parameters entered are only valid for the current task layer, i.e. you can set up other parameters in another task layer with the same source code.



It is not possible to have a source file with the same name and different source code in another task layer (task layer in another PLC or other processor) because all source files of a databank are saved in one directory (program file directory).

Dialog Box for entering the Task Parameters:



Source Type

You can select which programming language (Ladder Diagram, PL2000, Statement List) the source code of the task is from a pop up menu.
If there is already a source file with the corresponding task name, the source type is automatically entered.

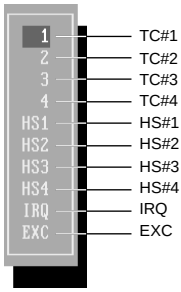
Init Type

If there is an INIT-SP (initialization sub-program) for the current task, you can select the programming language (PL2000, or Statement List) the INIT-SP is.

Task Class

To assign the priority of the individual task, you can put the tasks in different task class.
Different task classes are available depending on the system (B&R 2010 or B&R 2005) .

You can select the task class from a pop up menu:



Task Class	Priority	Cycle Time (Default)	System
EXC	High	---	B&R 2010 and B&R 2005
IRQ		---	B&R 2010
HS#1		3 msec	B&R 2010 and B&R 2005
HS#2		5 msec	B&R 2010
HS#3		7 msec	B&R 2010
HS#4		9 msec	B&R 2010
TC#1		10 msec	B&R 2010 and B&R 2005
TC#2		50 msec	B&R 2010 and B&R 2005
TC#3		100 msec	B&R 2010 and B&R 2005
TC#4	low	10 msec	B&R 2010 and B&R 2005

How task classes work is explained in detail in chapter 2, "Operating System PLCSW".

Inst Number

You can set the execution order of tasks within a task class with the Inst Number. The Inst Number defines the relative order of the tasks to one another. They can have a value from 0 to 128, which are interpreted as follows:

- Tasks with lower Inst Numbers are processed first.
- Tasks with the same Inst Number are put in the correct order in relation to the other tasks. The order of these particular tasks among themselves is no defined.
- Tasks with an Inst Number of 0 (= “don't care”) are executed sometime within the cycle.

The default value is 0 (= “don't care”).
Exception tasks have the **Exception No.** (exception number) here, and Interrupt tasks have the **Module Number** (module address of the module that generates the interrupts). Detailed information on this is in the sections "Exception Task Class" and "Interrupt Task Class" of chapter 2 "Operating System PLCSW".

Install

You can select the install Mode of the Task with a pop up menu.

auto ..... If install mode "auto" is active, the executable program module (B&R Module) of the task is automatically installed on the PLC after each download or boot up. This means that the task is started either automatically or manually depending on the entry in the start field.

manual ..... If install module "manual" is active, the task is on the PLC after a download but is not executable. The program module of the task has to be manually installed by you after each download or boot up with the function *Install* (see dialog box for the function *Module* in the pull down menu *PLC*). Then it is possible to manually start the task with the *Restart* function (see dialog box for the function *Tasks* in the pull down menu *PLC*). The install module manual is mainly used for testing.

Start

You can select the start mode of a task with a pop up menu.

auto ..... If start mode "auto" is active, the task is automatically started after each download or boot up.

manual ..... If start mode "manual" is active, you have to start the task manually after each download or boot up with the function *Restart* (see dialog box for the function *Tasks* in the pull down menu *PLC*).

Target for Store Function on the PLC

You can set the memory mode of the task with a pop up menu. The memory mode defines in which non volatile memory (User ROM or FIXRAM) the task is stored (programmed) in on the PLC when using the function *Store* (see pull down menu Object and GDM pop up menus).

USER ROM ..... The object is stored in the User ROM when you execute the function *Store*.

FIXRAM ..... The object is stored in the FIX RAM when you execute the function *Store*. Modules stored in the FIX RAM are not deleted by a TOTAL INIT. The FIX RAM can be configured with the PLC Configurator or PG2000.

Button Block

[OK] ..... The entries are accepted and saved in the databank. The dialog box is closed.

[Text] ..... This opens a dialog box so that you can enter a short description (max. 52 chars.) of the task.

[OK] ..... The text is accepted and the task description dialog box is closed.
[Cancel] ..... The text entered is ignored and the dialog box is closed.

[Cancel] ..... The entries are ignored and the dialog box is closed.

6.5.2 Parameters for Data Modules and B&R Modules

You can set the parameters for the selected data module or B&R module with the function *Parameters* from the pull down menu *Object*. This allows you to specify the memory target on the PLC.

Target for Store Function on the PLC

You can set the memory mode of the selected object (data module or B&R module) with a pop up menu. The memory mode defines in which non volatile memory (User ROM or FIXRAM) the task is stored (programmed) in on the PLC when using the function *Store* (see pull down menu Object and GDM pop up menus).

USER ROM ..... The object is stored in the User ROM when you execute the function *Store*.

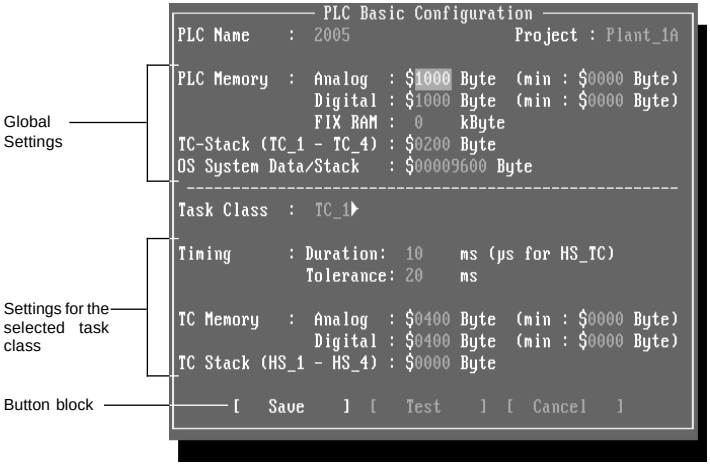
FIXRAM ..... The object is stored in the FIX RAM when you execute the function *Store*. Modules stored in the FIX RAM are not deleted by a TOTAL INIT. The FIX RAM can be configured with the PLC Configurator or PG2000.

Button Block

[OK] ..... The entries are accepted and saved in the databank. The dialog box is closed.
[Cancel] ..... The entries are ignored and the dialog box is closed.

6.5.3 PLC Configuration Module Parameters

To optimize the loading of your control system, you can set up the parameters of the PLC configuration module for your application. You can change all parameters in the PLC configuration module with the PLC configurator. In general, you only have to change very few parameters to match the operating system to your application. The parameters that are commonly changed can also be changed the PG2000 Programming System. To do this there is a symbol for the PLC configuration module (CFG symbol) that can be set in the task layer of the GDM. If this is selected and you call the function parameters from the pull down menu object the PLC configuration module is uploaded from the PLC and the main parameters are shown in a dialog box:



PLC Memory

Analog ..... This field allows you to change the size of the PLC global analog area. All PLC global process variables that are not type BIT are stored in the analog area.
The value in brackets is the minimum size required for the current project calculated by PG2000.

Digital ..... This field allows you to change the size of the PLC global digital area. All PLC global process variables that are type BIT (including BIT arrays) are stored in the digital area.
The value in brackets is the minimum size required for the current project calculated by PG2000.



Before you set the parameters in the PLC configuration module, you should delete all unused process variables from the databank and defragment the DPR with the function Optimize Object (see PLC Layer pop up menu). This is the only way to guarantee that the value in brackets is the real minimum value for the global analog and digital areas.

FIX-RAM ..... The entry defines the size of the FIX RAM. The size of the FIX RAM can only be "0" (no FIX RAM), "16" or multiples of "16".
The FIX RAM is a part of the user RAM that is not erased by a TOTALINIT, like the User ROM. This means that B&R modules in the FIX RAM are not deleted by a TOTAL INIT.

TC Stack (TC_1 - TC_4)

You can set the size of the user stack of the normal task classes with this entry (minimum value = $0040 bytes). The required size of the user stack for the normal task classes depends on the stack requirements of the functions used in the tasks. This can be found out with the Profiler.

Op. System Data/Stack

You can set the size of the memory area of the operating system data and stacks of the application programs (cyclic and non-cyclic tasks). If the error "3602 Not enough memory available in the OS Area" when you install an application program, you have to increase the size of this memory area.
You can determine the amount of free memory for the operating system data and stack of application programs with the function PLC Info (pop up menu from the function Other in the pull down menu PLC). This produces a dialog box, with the entry "PLC Free Memory" - "System".

Task Class

You can select the task class to be configured with a pop up menu. After selecting a task class, the bottom part of the dialog box shows the most important parameters for the corresponding task class.

Timing

Duration ..... You can set the cycle time of the selected task class in msec. (or µsex for HS task classes) with this entry.
For each task class, you can set an individual cycle time. Within this cycle time, all tasks in the task class are processed once. Tasks that perform important and time critical jobs should be put in a task class with a short cycle time. They are therefore executed more often that tasks in a task class with a longer cycle time. The cycle time has nothing to do with the actual execution time of the tasks on the PLC.
The CPU modules of the B&R 2000 PLC family offer you up to four normal task classes with definable cycle times of between 10 msec. and 5 sec. in steps of 10 msec. The cycle time of the high speed task classes can be set from 1 and 20 msecs. in 500 µsec steps. The B&R 2010 CPU modules offer four configurable HS task classes and the B&R 2005 CPU modules offer one.

Tolerance ..... You can set an allowed cycle time violation for the normal task classes to tolerantly monitor the cycle time. This means that the cycle time can be exceeded by a predefined amount (simple cycle time violation). If this tolerance time is exceeded (maximum cycle time violation) an exception is triggered.
You cannot set up an allowed cycle time violation for HS task classes. This means that when a cycle time violation occurs in a HS task class, an exception is triggered immediately.

The default settings for cycle time and allowed cycle time violation are in the table on the following page:

Task Class	Cycle Time (Default)	Allowed Cycle Time Violation (Default)	System
HS Task Class 1	3000 µsec.	No cycle time violation allowed	B&R 2010 and B&R 2005
HS Task Class 2	5000 µsec.	No cycle time violation allowed	B&R 2010
HS Task Class 3	7000 µsec.	No cycle time violation allowed	
HS Task Class 4	9000 µsec.	No cycle time violation allowed	
Task Class 1	10 msec.	20 msec.	B&R 2010 and B&R 2005
Task Class 2	50 msec.	50 msec.	
Task Class 3	100 msec.	100 msec.	
Task Class 4	10 msec.	30000 msec.	

TC Memory

Analog ..... This field allows you to change the size of the analog task class internal variable area (minimum value = $0010 bytes). All task class global process variables that are not type BIT are stored in the analog area.
The value in brackets is the minimum size required for the current project calculated by PG2000.

Digital ..... This field allows you to change the size of the digital task class internal variable area (minimum value = $0010 bytes). All task class global process variables that are type BIT (including BIT arrays) are stored in the digital area.
The value in brackets is the minimum size required for the current project calculated by PG2000.



Before you set the parameters in the PLC configuration module, you should delete all unused process variables from the databank and defragment the DPR with the function Optimize Object (see PLC Layer pop up menu). This is the only way to guarantee that the value in brackets is the real minimum value for the task class analog and digital areas.

TC Stack (HS_1 - HS_4)

You can set the size of the user stack of the selected high speed task class with this entry. The required size of the user stack for the normal task classes depends on the stack requirements of the functions used in the tasks. This can be found out with the Profiler.

Button Block

[OK] ..... The entries are accepted and saved. The dialog box is closed.

[Test] ..... To avoid having a defective PLC configuration module (where the operating system can't run) in the User RAM, new PLC configuration modules can be tested with this selection. The configuration module is downloaded to the User RAM and is easily removed if it is defective. After downloading the new configuration module in the User RAM, the PLC automatically boots up in RECONFIG mode. The RECONFIG mode is a special type of TOTAL INIT. As opposed to a TOTAL INIT, booting in RECONFIG mode uses the configuration module in the User RAM, not the one in the User ROM. If a fatal error occurs with the new PLC configuration module, a TOTAL INIT is triggered and the PLC boots up with the original configuration module. The reason for the error can be found by looking in the error module.

[Cancel] ..... The entries are ignored. The dialog box is closed.

6.6 EDITING AN OBJECT

To create or edit the source file of a task, you can call the corresponding editor (LAD, STL, PL2000, Data module or text editor) by pressing the function key [F7 Layer↓].
For text and data module symbols, a normal text editor is called as the child layer. For task, interrupt task, and exception task symbols, the editor as specified in the task parameters (LAD, STL or PL2000) is opened. You can find detailed information in the sections "6.7 Ladder Diagram - LAD", 6.8 Statement List - STL" and "6.9 PL2000".
If a source file already exists with the corresponding task name, it is loaded into the editor otherwise a new file is created.

6.7 LADDER DIAGRAM - LAD

6.7.1 General Information

The PG2000 Programming System allows you to write programs in a graphical programming language with the Ladder Diagram editor (LAD Editor). This type of programming is similar to drawing circuit diagrams.
Contacts are symbolically displayed in the LAD editor and called symbolic names. The relationship between symbolic names and hardware can be made at any time with the variable declaration.

Parallel to writing the LAD, it is checked for correctness:

- Outputs cannot be in column 1 of the LAD editor.
- Analog inputs must be in column 1 of the LAD editor.
- PG2000 allows you to set function symbols (symbols for function blocks or inline functions) only when the function symbol doesn't cover any other LAD component and it is not in column 1.
- Do the line and contact types match ? (analog/digital)
- Before the LAD is saved, it is completely checked for errors like missing outputs, unconnected lines, nameless contacts and function symbols without any input or any output parameters.

6.7.2 Ladder Diagram Syntax



This section gives you a quick overview of the syntax and symbols of the LAD programming language. For a detailed description of the LAD programming language, please refer to the "B&R SYSTEM 2000 Programming Languages Manual", Chapter 4 "Ladder Diagrams".

Contact Symbols

In a Ladder diagram, different logic structures can be constructed by using the contacts listed in the following table. Each contact has to be assigned to a process variable or constant.

Symbol	Contact Name	Limitations on position
─┤ ├─	Digital Input	No Limits
─┤/├─	Digital Input, negated	No Limits
‖ ‖─	Analog Input	Can only be put in column 1
─┤+├─	Positive One Shot Input	No Limits
─┤-├─	Negative One Shot Input	No Limits
─┤&├─	One Shot Input	No Limits
‖@‖─	Address Operator	Can only be put in column 1
─( )─	Digital Output	Cannot be put in column 1
─(/)─	Digital Output, negated	Cannot be put in column 1
─(( ))─	Analog Output	Cannot be put in column 1
─(+)─	Positive One Shot Output	Cannot be put in column 1
─(-)─	Negative One Shot Output	Cannot be put in column 1
─(&)─	One Shot Output	Cannot be put in column 1
─(L)─	Latch	Cannot be put in column 1
─(U)─	Unlatch	Cannot be put in column 1

Process Variables

Symbolic Names are used for process variable in the Ladder Diagram programming language. Theses names can use the following characters:

Capital Letters ..... A to Z
Lower Case Letters ..... a to z
Numbers ..... 0 to 9
Special Characters ..... _ and $

Symbolic names also have the following rules:

- A symbolic name can be a maximum of **10 characters** long.
- The first character of the symbolic name must be a letter.
- Symbolic names are case sensitive. This means that **SWITCH_1**, **Switch_1** and **switch_1** are three different variables !
- The symbolic name has to be a continuous character chain, i.e. **No spaces** are allowed in PV names !

LAD Constants

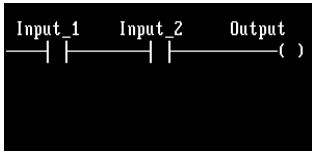
A constant is a value that is not changed for the run time of the program. LAD constants cannot be longer than 10 characters !

Constant	Constant Syntax	Examples
Decimal Constant	#[+|-]ddd	#34 #-26751 #+67013
Floating Point Constant	#[+|-]ddd.[ddd][E|e[+|-]dd]	#-47.23 #3.17E05 #+5.14e-02
Binary Constant	%ddd	%10101010 %1001 %00101001
Hexadecimal Constant	$ddd	$F9 $5A715 $E8B
String Constant	"sss"	"String" "a b c" "Textstring"

Syntax: Characters in square brackets are optional.

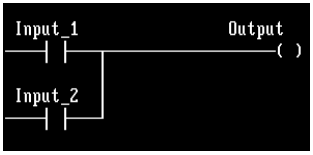
Simple Logical Structures

AND (Series connection of contacts):



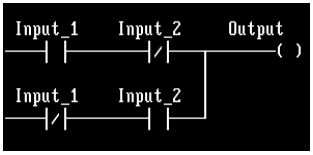
Logic Table				
Input_1	0	1	0	1
Input_2	0	0	1	1
Output	0	0	0	1

OR Structure (Parallel connection of contacts):



Logic Table				
Input_1	0	1	0	1
Input_2	0	0	1	1
Output	0	1	1	1

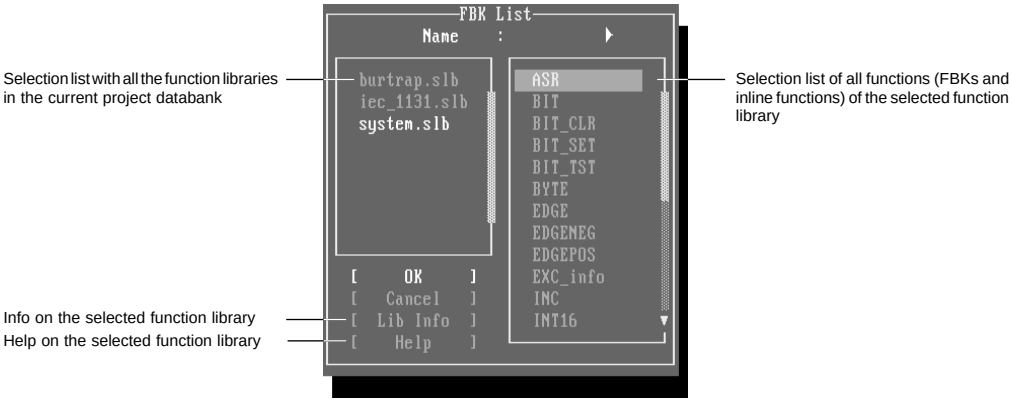
EXCLUSIVE OR Structure:



Logic Table				
Input_1	0	1	0	1
Input_2	0	0	1	1
Output	0	1	1	0

Calling Functions in LAD

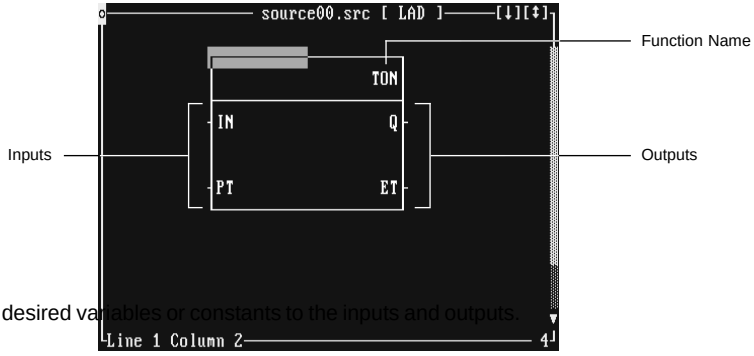
Both function blocks and inline functions are displayed in LAD with so-called function symbols. To place a function symbol, call up a list of all function blocks and inline functions in the current project databank with a function key of the LAD editor:



You can choose the desired function from the list.

How Functions Look in LAD

A function (function block or inline function) is shown as a block (function symbol) in the ladder diagram editor with inputs (on the left) and outputs (on the right):



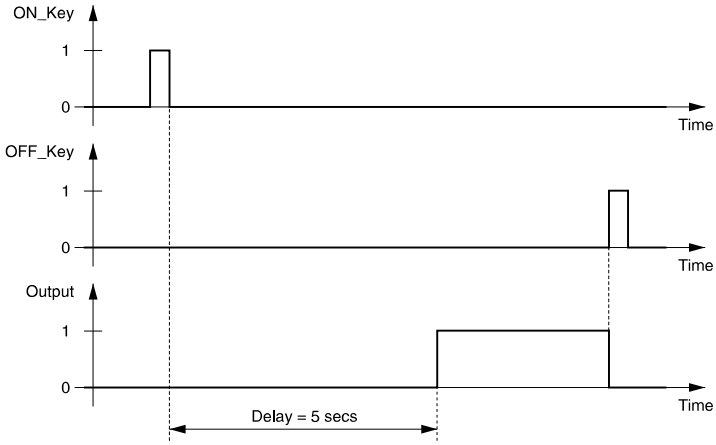
You can connect the desired variables or constants to the inputs and outputs.

6.7.3 LAD Editor

The most important functions of the LAD editor (set delete contacts, line and function symbols, block marking, etc.) are explained on the following pages by using a simple example.

Example Description

The output "output" should be turned on 5 seconds after you press the key (input) "ON_key" (delay time = 5 seconds). The output is reset to 0 with the input "OFF_key".



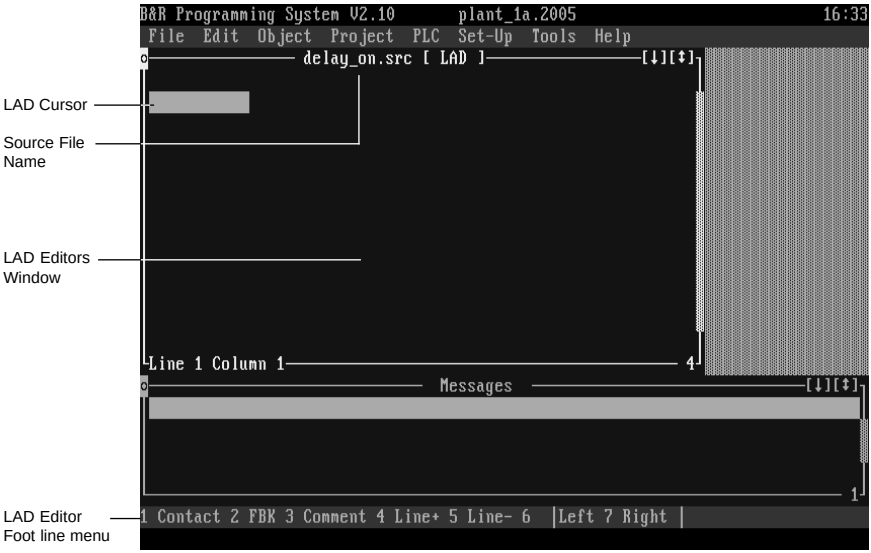
This "turn on delay" can be done with the "TON" function block (IEC 1131 library).
Before you can use the function blocks from the IEC 1131 library in your project, you have to import the library into the corresponding project databank. To do this, use the function *Library* (pop up menu from the *Import* function in the pull down menu *File*). You can find detailed information on importing libraries in the section "Functions".

Calling the LAD Editor

We recommend that you always call the LAD editor from the task layer of the GDM as this makes the whole thing clearer an easier to understand. To do this you have to put a task symbol for an application program on the task layer (see section "6.4 Placing an object in the Task Layer"). Since this example is done in the programming language LAD, select Ladder Diagram for the task parameter "Source Type". Afterwards you can open the LAD editor by pressing the function key [F7 Layer ↓] when the cursor is on the task symbol.

You can also call the LAD editor with the function *New* from the pull down menu *File*. A pop up menu appears for you to choose the type of editor. To create a LAD source file, select Ladder Diagram from this menu and a LAD editor window opens.

The LAD can now be drawn in this window:

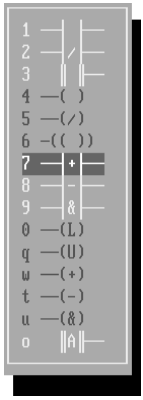


After opening the LAD editor from the GDM, the source file is automatically given the name <Task Name>.SRC. If the editor was called with the New function, by default the file is called SOURCExx.SRC, where xx is a the next available number in sequence.

The LAD editor can be operated with the mouse or the keyboard.

Place Contact

By pressing (clicking) the function key [F1 Contact], you can put a contact at the current cursor position. A pop up list of all contact types appears to choose from.



You can also set contacts without using the pop up menu of contact types. There are hot keys for the contact types given on the pop up menu shown on the right, e.g. the [3] key puts an analog input in the LAD.

After placing a contact, a small window opens to type in the process variable name (symbolic name) or constant value.

Delete Contact



If the LAD cursor is on a contact, you can open a contact functions pop up menu with [F1 Contact] or the combination [Ctrl] + [F3]. The selected contact is deleted if you choose *Delete* from this menu.



The selected contact can be deleted by pressing the [Del] key.

Drawing/Deleting Horizontal Lines

You can draw a horizontal line at the current cursor position with the [Space] bar. If there is already a symbol in the current position, it is deleted when you press [Space].

Drawing/Deleting Vertical Lines

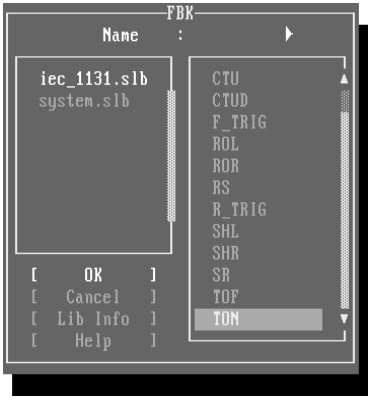
There are two function keys for you to draw vertical link lines:

[F6 | left] ..... A line up to the next symbol (line or contact) is drawn from the left hand side of the cursor position or if a line already exists, it is deleted.

[F7 right |] ..... A line up to the next symbol (line or contact) is drawn from the right hand side of the cursor position or if a line already exists, it is deleted.

Place a Function Symbol

A function (function block or inline function) is displayed as a so-called function symbol in LAD. You can place a symbol for a function at the current cursor position with [F2 FBK]. The FBK selector box appears for you to select a function block or inline function. For our example, select the function block "TON" from the IEC 1131 library (IEC_1131.SLB):





If a function symbol cannot be placed at the current cursor position, an error message is displayed.

After placing a FBK, you can give it an Alias name (up to 10 characters). This sets the name of the alias structure, which allows you to access this FBK from a PL2000 or STL program (Alias Call). You can find more detailed information about Alias calls in the "B&R SYSTEM 2000 Programming Language Manual".

Delete a Function Symbol

You can only delete a function symbol (FBK or inline) when the LAD cursor is on the top left corner of the corresponding function symbol. The following instructions apply only when this is the case.



You can open a pop up menu with [F1 Contact] or [Ctrl] + [F3]. Select *Delete* from the menu and the selected function symbol is deleted.



Press the [Del] key.

Block Marking

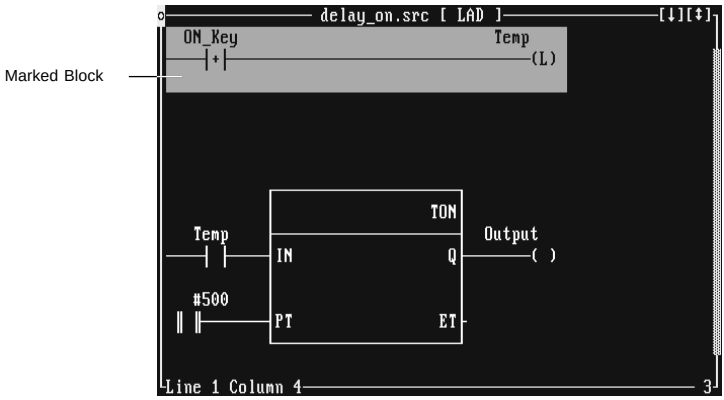


To make a block, put the cursor at the start of the area you want to mark. Press and hold the [⇧] key and move to the end of the area you want to mark with the cursor keys.



Put the cursor at the start of the area you want to mark. Select the function Mark for the pull down menu Edit. Move the cursor to the desired end position.
If you select the Mark function again, marking mode is turned off.

Text and symbols in the marked areas are either shown in inverse of grey, depending on you screen driver:



Copying a Marked Block



You can copy the marked block into the clipboard with [Ctrl] + [Ins] (block is not deleted from source file). You can insert this (the clipboard contents) at the current cursor position with [⇧] + [Ins]. It also stays in the clipboard.



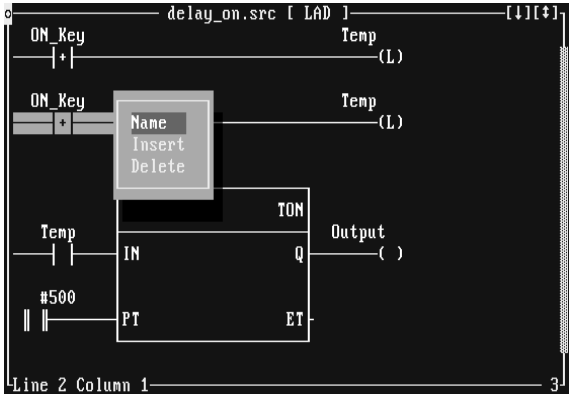
You can copy the marked block into the clipboard with *Copy* from the *Edit* menu (block is not deleted from source file). Marking mode is automatically turned off.
You can insert this (the clipboard contents) at the current cursor position with *Insert* from the *Edit* menu. It also stays in the clipboard.



The clipboard contents can be inserted into other open LAD editor windows.

Change Contact Names

When the cursor is on a contact, you can call the pop up menu with [F1 Contact] or [Ctrl] + [F3]. You can change the name of a contact by using the function name.



Change Contact Symbol

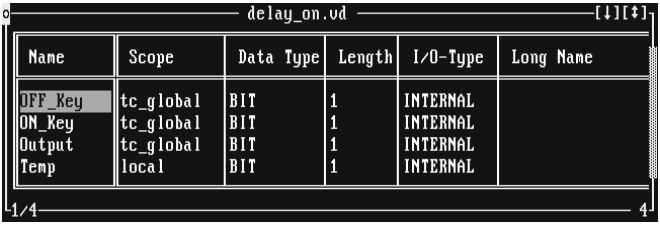
You can use the following keys to change the contact symbol:

[n] ..... You can invert a digital contact with the [n] key.
Example: ⇒

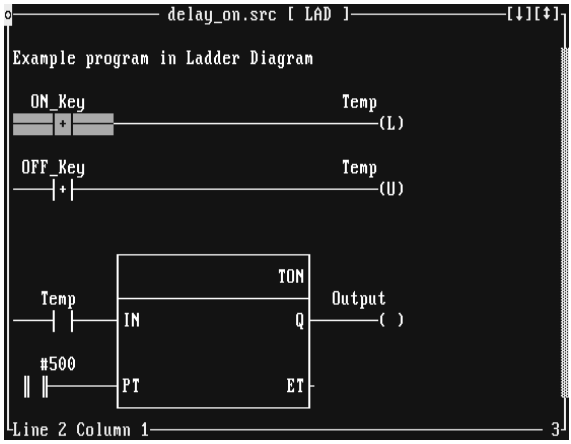
[-] ..... You can cycle through the contacts of the same type with the [-] key.
Example: (L) ⇒ (U)

Saving a Program

When you save a LAD, it is checked for errors like missing outputs, open lines, unnamed contacts and function symbols without inputs or outputs. If there are any undeclared variables, a dialog gives you the corresponding warning. If you close this box with [OK] the variable declaration appears:



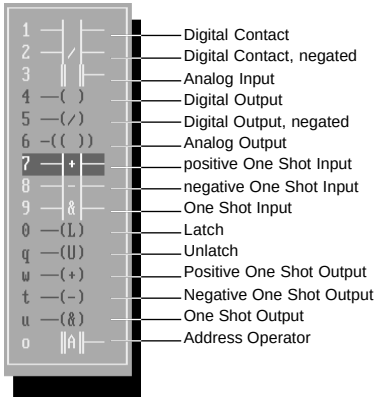
You can find detailed information on the variable declaration in section "6.10 Variable Declaration".
The required ladder diagram should look like this:



To save the LAD under the name <Task Name>.SRC, use the function *Save* from the *File* or *System* menu. The source file <Task Name>.SRC is put in the program files directory of the current project.

Foot Line Menu of the LAD Editor

[F1 Contact] This function puts a contact at the current cursor position. You can select a contact type from the pop up menu that appears:



After setting the contact, a window opens for you to type in a PV name or constant value for it.
If there is already a contact at the current cursor position, a different pop up menu is called with the following functions:

Name ..... You can change the name of the contact in a window.

Insert ..... You can insert a contact of line at the current position (existing structure is moved to the right).

Delete ..... The contact under the cursor is deleted. Excess lines are also deleted.

[F2 FBK] A function symbol is set at the current position. You can select the function block or inline function from the FBK selector box that shows all functions in the current databank.
If a function symbol cannot be put at the current position, an error message is output.
If the LAD cursor is in the top left corner of the function symbol, a pop up menu is called with the following options:

Alias Name ..... An alias name (name of the FBK alias structure) of up to 10 characters can be entered.

Delete ..... The function symbol is deleted.

[F3 Comment] You can put a comment in any free space in a LAD. The text is entered in a special dialog box. The size of the text entry window depends on the amount of free space in the LAD. If the cursor is on an existing comment field when this function is called, you can edit it.

[F4 Line+] A blank line is inserted above the cursor line.

[F5 Line-] The current cursor line is deleted. You cannot do this if there is part of a function symbol on this line.

[F6 | left] A line up to the next symbol (line or contact) is drawn from the left hand side of the cursor position or if a line already exists, it is deleted.

[F7 right |] A line up to the next symbol (line or contact) is drawn from the right hand side of the cursor position or if a line already exists, it is deleted.

LAD Editor Pop Up Menus

The pop up menus of the LAD editor (called with mouse double click or [Ctrl] + [F3]) are dependent on the current cursor position.

Pop Up Menu - Contact

These functions are available when the cursor is on a contact.

Name ..... You can change the name of the PV in a window.

Insert ..... You can insert a contact or line with this function (existing structures are move right). Another pop up menu appears with all the different contact types. If you don't select a contact and exit this pop up with [ESC], a line is inserted at the current position. This function *Insert* is only performed when there is enough space.

Delete ..... The contact under the cursor is deleted. Excess lines are also deleted.

Pop Up Menu - Line

These functions are available when the cursor is on a line.

Name ..... No function.

Insert ..... You can insert a contact or line with this function (existing structures are move right). Another pop up menu appears with all the different contact types. If you don't select a contact and exit this pop up with [ESC], a line is inserted at the current position. This function *Insert* is only performed when there is enough space.

Delete ..... The line under the cursor is deleted. Excess lines are also deleted.

Pop Up Menu - Function Symbol

These functions are available when the cursor is on the top left corner of the function symbol.

Alias Name ..... You can give a function symbol an Alias name (up to 10 characters). This sets the name of the alias structure, which allows you to access this FBK from a PL2000 or STL program (Alias Call). You can find more detailed information about Alias calls in the "B&R SYSTEM 2000 Programming Language Manual".

Delete ..... The function symbol is deleted.

LAD Editor Key Commands

The following keys and key combinations can be used in the LAD editor:

Key	Description
[⇑]	Cursor up
[⇓]	Cursor down
[⇐]	Cursor left
[⇒]	Cursor right.
[↵]	Cursor to the first column, next line.
[PgUp]	Page up
[PgDn]	Page down
[Home]	Cursor to the start of the line
[End]	Cursor to the end of the line
[Ctrl] + [Home]	Cursor to the start of the LAD
[Ctrl] + [End]	Cursor to the end of the LAD
[Del]	Delete line, contact, function symbol or comment. The remaining structure is moved on column left when you delete a line or contact.
[Ins]	A contact or line is inserted at the current cursor position and the existing structure is moved right.
[Space]	The symbol (line or contact) at the current cursor is deleted If there is nothing in the current position, a line is drawn.
[n]	Inverts a digital contact.
[-]	Cycle through the contacts of the same type, i.e. you can't change a digital contact to an analog one.
[⇑] + [Home]	The area from the current cursor position to the start of the line is marked.
[⇑] + [End]	The area from the current cursor position to the end of the line is marked.
[⇑] + [Cursor keys]	Mark any area.
[Ctrl] + [Ins]	The marked area is copied into the clipboard but not deleted from the LAD.
[⇑] + [Ins]	Insert the clipboard contents into the editor.
[⇑] + [Del]	The marked area is moved into the clipboard, i.e. it is deleted from the LAD.
[Ctrl] + [F3]	Call the position/symbol sensitive pop up menu.
[Esc]	Call the system menu or cancel the current function.

6.8 STATEMENT LIST - STL

6.8.1 General Information

The statement list (STL) is an assembler type programming language in alphanumeric form. The STL in PG2000 corresponds to the IEC 1131 standard. This allows application parts written in STL from other standard conformant programming systems to be reused.

You create or edit STL programs in the STL editor. The STL source code is checked for correct syntax when it is saved.

The variables used in STL are given symbolic names. The connection between the names and the hardware can be made at any time using the variable declaration.

6.8.2 Syntax of the Programming Language - STL



This section offers you a quick overview of the syntax and command set of STL. A detailed description of STL is in the "B&R SYSTEM 2000 Programming Languages Manual" in the "Statement List" chapter.

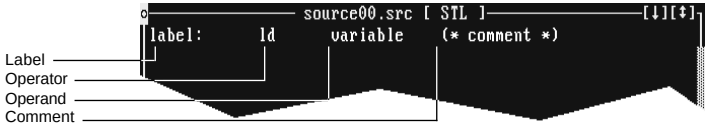
The STL compiler in PG2000 version 2.10 almost exactly corresponds to the IEC 1131 standard. The differences to the standard are due to the global concept of B&R programming languages:

- Uniform data types.
- Variable declaration with name management and data bank (not integrated in the source code) which is central and menu controlled as for all B&R programming languages.
- Extended FBK and inline function calls for compatibility for PL2000 .
- There are particular standard function blocks with certain characteristics described in the standard. These don't fit with the PG2000 concept and are therefore not implemented.
- Implicit data type conversion: the resultant data type of an operation always corresponds to the "larger" of the two operands. The "smaller" operand is automatically "up-converted" to the larger data type. Exception: with an assignment operation, the destination can not be converted to a larger data type ! (Explicit type conversion are always possible with inline functions)

The STL syntax is checked when the source code is saved. A compiler run in STL is the same as in PL2000, consisting of a parser, code generator and task builder. After the parser run, all variables are known in the name management and can be declared. All FBKs and inline functions in PL2000 can also be used in STL.

STL Line

An STL line consists of four possible fields that can be at any position in the line. You can set the positions of the fields with the menu option Editor for the pull down menu *Set Up* (see section "6.8.4 Configuring the STL Editor"). The fields have to be separated by at least one space. The order of the fields has to be kept.



Label field The name of the **Label** is followed by a colon ":" and marks a label (jump label). Labels are optional and can also be alone on a line.

Operator field An **operator** can be either a key word (STL command) or the name of an inline function.

Operand field An operator can have one or more **operands**. Basic operations can only have one operand.

Comment field **Comments** are text surrounded by **(*** and ***)**. They can be put anywhere where spaces are allowed (even between fields). There is a second way to specify comments, with a semicolon ";". In this case all text following the ";" up to the end of the line is a comment. Comments can not continue over several lines. Spaces are allowed !

STL Commands

There is only one register available in STL, the **accumulator** (A). The current result is stored in the accumulator. You have no address register and no extended addressing modes. Addressing is done the variables and constants in STL.
Key words in STL are not case sensitive, i.e. no difference between "LD" and "ld".

Overview of the STL commands:

Operator	Modifiers	Operand	Description
LD	N	V / K	Operand ⇨ A
ST	N	V	A ⇨ Operand
S		BIT V	If A≠ 0: 1 ⇨ Operand (Data type BIT)
R		BIT V	If A≠ 0: 0 ⇨ Operand (Data type BIT)
AND	N, (	V / K	A AND Operand ⇨ A
&	N, (	V / K	A AND Operand ⇨ A
OR	N, (	V / K	A OR Operand ⇨ A
XOR	N, (	V / K	A EXCLUSIVE OR Operand ⇨ A
ADD	(	V / K	A + Operand ⇨ A
SUB	(	V / K	A - Operand ⇨ A
MUL	(	V / K	A * Operand ⇨ A
DIV	(	V / K	A ÷ Operand ⇨ A
GT	(	V / K	If A > Operand: 1 ⇨ A; otherwise: 0 ⇨ A
GE	(	V / K	If A ≥ Operand: 1 ⇨ A; otherwise: 0 ⇨ A
EQ	(	V / K	If A = Operand: 1 ⇨ A; otherwise: 0 ⇨ A
LE	(	V / K	If A ≤ Operand: 1 ⇨ A; otherwise: 0 ⇨ A
LT	(	V / K	If A < Operand: 1 ⇨ A; otherwise: 0 ⇨ A
NE	(	V / K	If A ≠ Operand: 1 ⇨ A; otherwise: 0 ⇨ A
JMP	C, N	LABEL	Jump to label "LABEL"
CAL	C, N	NAME	Call Function block "NAME"
)			Resolve bracket operation

Explanation of the Table:

N ..... The operand is negated before the operand is executed.
Example: **`ANDN Operand`** → A = A & NOT Operand

C ..... The operation is only executed if A ≠ 0 (the result of the previous operation was not equal to 0).
Example: **`CALC fbk1`** → Call the function block fbk1, if A ≠ 0
`CALCN fbk1` → Call the function block fbk1, if A = 0

(..... The command is executed when the operations up to the operand ")" are executed.
Example:

```
LD      a
AND (   b
OR      c
)
```

→ A = a AND (b OR c)

V / K ... **V**ariable or **K**onstant (automatic data type recognition like in PL2000)
The operation is executed with the data type of the operand or the accumulator (implicit data type conversion). As in PL2000, the "larger" data of the two operands is used. The result is data type BIT for comparison operations. Implicit "down conversions" is not allowed (assignment operations).

Process Variables

Symbolic Names are used for process variables in STL. Theses names can use the following characters:

Capital Letters ..... A to Z
Lower Case Letters ..... a to z
Numbers ..... 0 to 9
Special Characters ..... _ and $

Symbolic names also have the following rules:

- The first character of the symbolic name must be a letter.
- Symbolic names are case sensitive. This means that **SWITCH_1**, **Switch_1** and **switch_1** are three different variables !
- The symbolic name has to be a continuous character chain, i.e. **No spaces** are allowed in PV names !
- The length of symbolic names is not limited, but only the **first 10 characters** are significant, i.e. **Temperature_set** and **Temperature_current** are treated as the same variable (the parser makes no distinction between these variables because the first 10 characters are the same).
- Process variables cannot be a keyword in STL syntax (LD, MUL, etc.) !

STL Constant

A constant is a value that is not changed for the run time of the program.

Constant	Constant Syntax	Examples
Decimal Constant	[+|-]ddd	#34 #-26751 #+67013
Floating Point Constant	[+|-]ddd.[ddd][E|e[+|-]dd]	#-47.23 #3.17E05 #+5.14e-02
Binary Constant	%ddd	%10101010 %1001 %00101001
Hexadecimal Constant	$ddd	$F9 $5A715 $E8B
String Constant	"sss"	"String" "a b c" "Textstring"

Syntax: Characters in square brackets are optional.



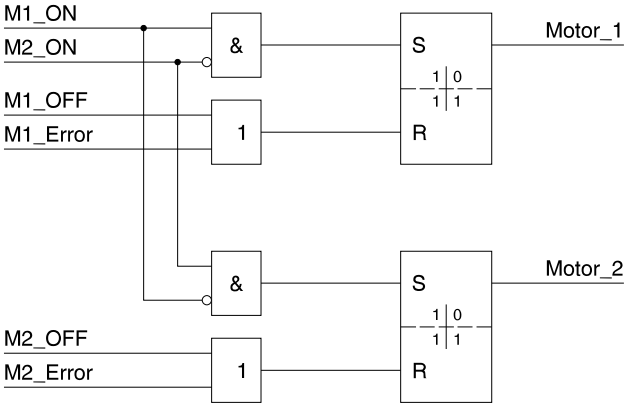
A constant is always interpreted as the smallest possible data type.

6.8.3 STL Editor

The most important functions of the STL editor (marking, copying, etc.) are explained on the following pages by using a simple example.

Example Description

The following Logic Plan (switching logic for the motors motor_1 and motor_2) should be written in the programming language STL:



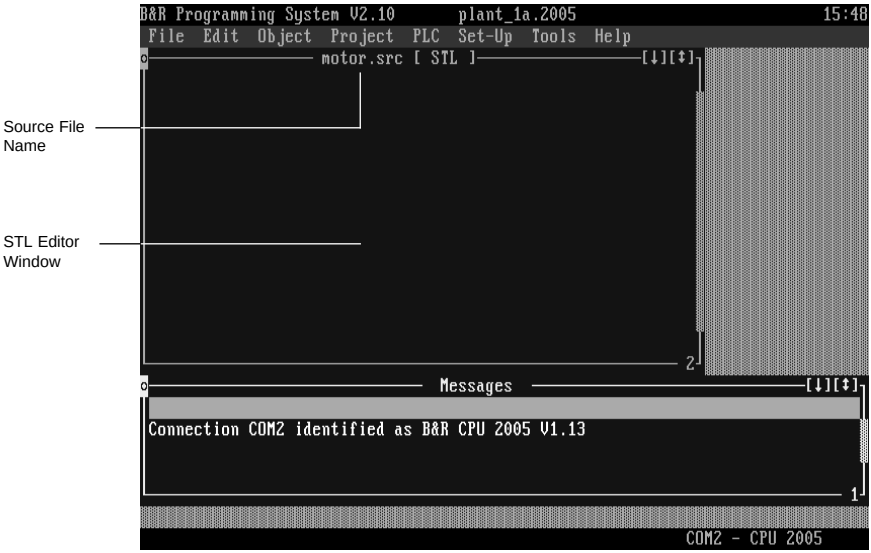
The bi-stable latch (reset dominant) can be realized with the function block "RS" (IEC 1131 library).
Before you can use the function blocks and inline functions of the IEC 1131 library in your project, you have to import the library into the corresponding project data bank. There is the function *Library* (pop up menu of the *Import* function from the *File* menu). You can find detailed information on importing libraries in the section "Functions".

Calling the STL Editor

We recommend that you always call the STL editor from the task layer of the GDM as this makes the whole thing clearer an easier to understand. To do this you have to put a task symbol for an application program on the task layer (see section "6.4 Placing an object in the Task Layer"). Since this example is done in the programming language STL, select Statement List for the task parameter "Source Type". Afterwards you can open the STL editor by pressing the function key [F7 Layer ↓] when the cursor is on the task symbol.

You can also call the STL editor with the function *New* from the pull down menu *File*. A pop up menu appears for you to choose the type of editor. To create a STL source file, select Statement List from this menu and a STL editor window opens.

STL source code can now be entered in this window:

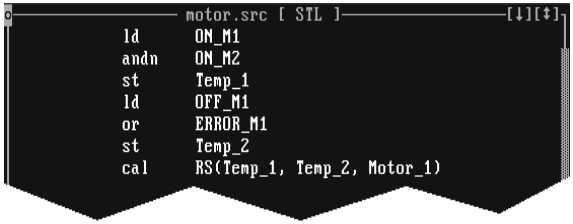


After opening the STL editor from the GDM, the source file is automatically given the name <Task Name>.SRC. If the editor was called with the *New* function, by default the file is called SOURCExx.SRC, where xx is a the next available number in sequence.

The STL editor can be operated with the mouse or the keyboard.

Entering the Source Code

Source code cannot be directly entered in the STL editor. The cursor is moved to the next field, next tab position, with [Tab].
Since the programming example consists of two identical parts except for the symbolic names (switching logic for the motors "Motor_1" and "Motor_2"), we recommend that you only type in the switching logic for one motor and then copy this part. After coping it you only have to change the symbolic names.
Switching logic for the motor "Motor_1":



Marking Text Blocks

You can choose between two different marking modes (line or column marking). The desired marking mode is set in the editor settings (see section "6.8.4 Configuring the STL Editor"). Line marking mode is set by default.

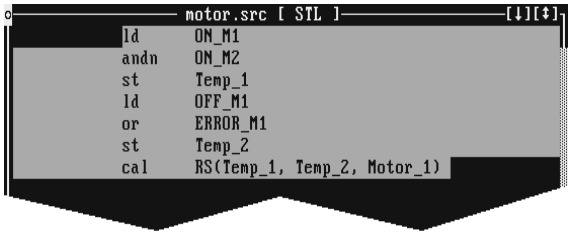


To make a block, put the cursor at the start of the area you want to mark. Press and hold the [⇧] key and move to the end of the area you want to mark with the cursor keys.



Put the cursor at the start of the area you want to mark. Select the function Mark for the pull down menu Edit. Move the cursor to the desired end position.
If you select the Mark function again, marking mode is turned off.

Text and symbols in the marked areas are either shown in inverse of grey, depending on you screen driver:



Copying a Marked Block



You can copy the marked block into the clipboard with [Ctrl] + [Ins] (block is not deleted from source file). You can insert this (the clipboard contents) at the current cursor position with [⇧] + [Ins]. It also stays in the clipboard.



You can copy the marked block into the clipboard with *Copy* from the *Edit* menu (block is not deleted from source file). Marking mode is automatically turned off.
You can insert this (the clipboard contents) at the current cursor position with *Insert* from the *Edit* menu. It also stays in the clipboard.



The clipboard contents can be inserted into other open STL editor windows.

Search/Replace Text

After doing the block copy, you have to change the variable names in the second program part. You can use the function *Replace* from the *Edit* pull down menu. A dialog box appears when you select this function, for you to enter the search and replace text as well as the direction and start position:
After entering the information and closing this dialog box with [OK], the search starts with the given parameters.



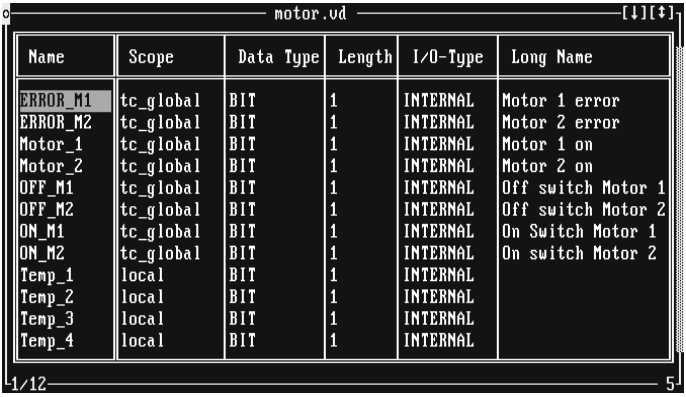
If a match is made you are given the following message on the message line:

Replace ? (Y)es (N)o (A)ll (E)nd

[Y] ..... Text found is replaced with the replace text. The search continues in the selected direction.
[N] ..... Text found is **not** replaced with the replace text. The search continues in the selected direction.
[A] ..... Text found is replaced with the replace text. The search continues in the selected direction and all matches found are replaced without asking for confirmation.
[E] ..... The *Replace* operation is terminated.

Saving a Program

When you save an STL program, it is checked for syntax. If there are any undeclared variables, a dialog gives you the corresponding warning. If you close this box with [OK] the variable declaration appears:

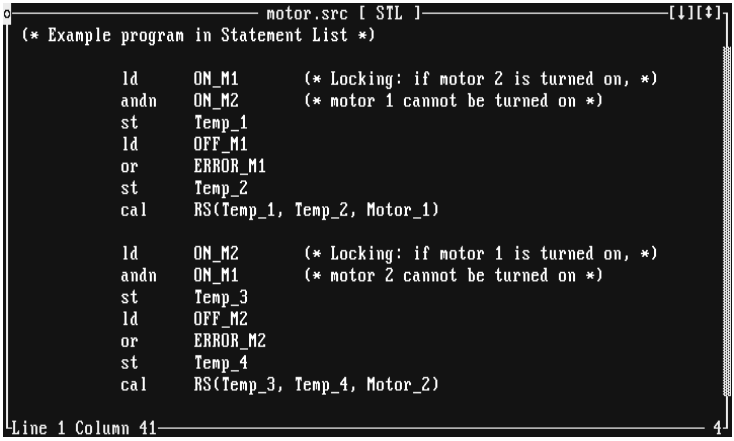

motor.vd

Name	Scope	Data Type	Length	I/O-Type	Long Name
ERROR_M1	tc_global	BIT	1	INTERNAL	Motor 1 error
ERROR_M2	tc_global	BIT	1	INTERNAL	Motor 2 error
Motor_1	tc_global	BIT	1	INTERNAL	Motor 1 on
Motor_2	tc_global	BIT	1	INTERNAL	Motor 2 on
OFF_M1	tc_global	BIT	1	INTERNAL	Off switch Motor 1
OFF_M2	tc_global	BIT	1	INTERNAL	Off switch Motor 2
ON_M1	tc_global	BIT	1	INTERNAL	On Switch Motor 1
ON_M2	tc_global	BIT	1	INTERNAL	On switch Motor 2
Temp_1	local	BIT	1	INTERNAL	
Temp_2	local	BIT	1	INTERNAL	
Temp_3	local	BIT	1	INTERNAL	
Temp_4	local	BIT	1	INTERNAL	

1/12 5

You can find detailed information on the variable declaration in section "6.10 Variable declaration".

The source code for the example STL program should look like this:



To save the STL program under the name <Task Name>.SRC, use the function *Save* from the *File* or *System* menu. The source file <Task Name>.SRC is put in the program files directory of the current project.

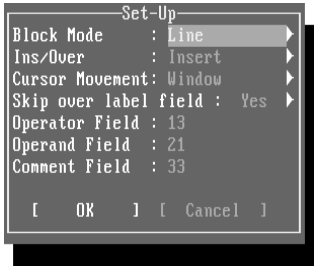
STL Editor Key Commands

The following keys and key combinations can be used in the STL editor:

Key	Description
[⇑]	Cursor up
[⇓]	Cursor down
[⇐]	Cursor left
[⇒]	Cursor right.
[↵]	Cursor to the first column, next line.
[PgUp]	Page up
[PgDn]	Page down
[Home]	Cursor to the start of the line
[End]	Cursor to the end of the line
[Ctrl] + [Home]	Cursor to the start of the code
[Ctrl] + [End]	Cursor to the end of the code
[Del]	Delete character under the cursor.
[Ins]	Switch between insert and overwrite mode.
[Tab]	Move cursor to the next tab position (field). The text on the right of the cursor is moved right with the cursor.
[⇑] + [Tab]	Move cursor to the previous tab position (field).
[↵]	Terminate the current line. In overwrite mode the cursor is moved to the start of the next line. In insert mode, a line is inserted. The text from the current cursor position to the end of the line is put in the new line.
[⇐]	Delete the character on the left of the cursor.
[Ctrl] + [⇐]	Delete the current line.
[⇑] + [Home]	The area from the current cursor position to the start of the line is marked.
[⇑] + [End]	The area from the current cursor position to the end of the line is marked.
[⇑] + [Cursor keys]	Mark any area.
[Ctrl] + [Ins]	The marked area is copied into the clipboard but not deleted from the source code.
[⇑] + [Ins]	Insert the clipboard contents into the editor.
[⇑] + [Del]	The marked area is moved into the clipboard, i.e. it is deleted from the source code.
[Ctrl] + [F3]	Call the position/symbol sensitive pop up menu.
[Esc]	Call the system menu or cancel the current function.

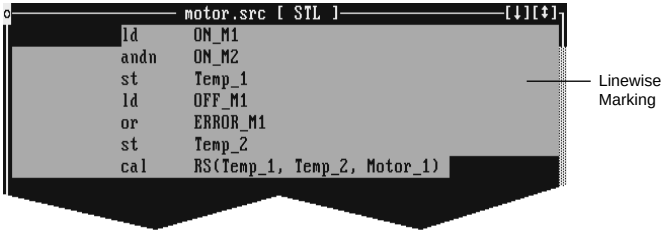
6.8.4 Configuring the STL Editor

By selecting the function *Editor* from the pull down menu *Set Up*, you open a pop up menu with a list of all the editors that can be setup. Select *Statement List* to change default settings of the STL Editor. This opens the following dialog box:

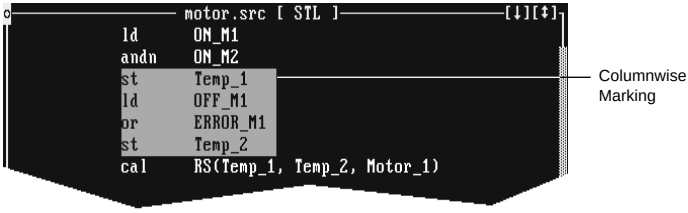


Block Mode Toggle marking mode:

Line ..... It is only possible to mark text linewise.



Column ..... It is only possible to mark text columnwise.



Ins/Over Toggle the insert mode:

Insert ..... Insert mode is the default mode.

Overwrite ..... Overwrite mode is the default mode.

Cursor Movement Toggle the cursor movement mode:

Text Area ..... The cursor can only be moved within the text area, i.e. where you have already typed code in.

Window..... The cursor can be moved anywhere within the window. It is not limited to the text area. The cursor colour changes depending on whether the cursor is inside or outside the text area. If the cursor is outside the text are and you type something in, the blank space between the cursor and the start of the line/text is filled with spaces.

Skip over label field

Yes ..... When you press [↵], the label field is skipped and the cursor goes straight to the operator position on the next line.

No ..... When you press [↵], the cursor goes to the label field on the next line.

Operator Field This is the column number for the start of the operator field.

Operand Field This is the column number for the start of the operand field

Comment Field This is the column number for the start of the comment field



To keep these settings permanently, use the Save function from the pull down menu Set Up. If you don't save the settings, they only apply to the current PG2000 session.

6.9 PL2000

6.9.1 General Information

PL2000 high level programming language that was developed for writing structured programs for automation tasks. Besides the standard elements of a high level language, there is also state machine functions that are required very frequently in the control automation area.
You create or edit PL2000 programs in the PL2000 editor. The PL2000 editor has a line syntax checker, that can check each line of the source code as it is written. The PL2000 source code is checked for correct syntax either online or when it is saved depending on how you have set it up.
The variables used in PL2000 are given symbolic names. The connection between the names and the hardware can be made at any time using the variable declaration.

6.9.2 Syntax of the Programming Language - PL2000



This section offers you a quick overview of the syntax and command set of PL2000. A detailed description of STL is in the "B&R SYSTEM 2000 Programming Languages Manual" in the "PL2000" chapter.

Operators

Operators in PL2000 are not case sensitive, i.e. no difference between "MOD" and "mod". Operands can be variables or constants.

Group	Operator	Usage	Meaning
Arithmetic Operators	-	- Operand1	unary minus
	+	Operand1 + Operand2	Addition
	-	Operand1 - Operand2	Subtraction
	*	Operand1 * Operand2	Multiplication
	/	Operand1 / Operand2	Division
	MOD	Operand1 MOD Operand2	Remainder of a whole number division
Logical Operators	NOT	NOT Operand1	unary negation
	AND	Operand1 AND Operand2	logical AND
	OR	Operand1 OR Operand2	logical OR
	XOR	Operand1 XOR Operand2	logical EXCLUSIVE OR
Comparison Operators	=	Operand1 = Operand2	Equal to
	<>	Operand1 <> Operand2	Not equal to
	>	Operand1 > Operand2	Larger than
	>=	Operand1 >= Operand2	Larger than or equal to
	<	Operand1 < Operand2	Less than
	<=	Operand1 <= Operand2	Less than or equal to
Other	( )	(Operation)	Bracket operations

The order of the execution of an operation is defined by the priority of the operators:

Priority	Operators
high low	() NOT, - (unary operators) *, /, MOD +, - AND XOR OR =, <>, <=, >=, <, >

Control Constructs

The key words for the control constructs are not case sensitive (no difference between "IF" and "if").

Construct	Key Words	Meaning
IF Statement	IF, THEN, ELSE, ENDIF	Condition (Question)
LOOP Statement	LOOP, ENDLOOP, EXITIF, TO	Repeat (Loop)
SELECT Statement	SELECT, NEXT, ENDSELECT STATE, WHEN, DO	State Blocks and Transition conditions
CASE Statement	CASE, OF ACTION, ELSEACTION ENDACTION, ENDCASE	Select an action (case/switch) from a list of options
Jump Statement	GOTO	Program flow change

Statements

The key words for the statements are not case sensitive (no difference between "ACCESS" and "access").

Statement	Key Words	Meaning
ACCESS Statement	ACCESS	Address Assignment
ADR Statement	ADR()	Find the address of a process variable
SIZEOF Statement	SIZEOF()	Find the size of a process variable

Comments

Comments are separated from the program code with a semicolon ";". The text between the semicolon and the end of the line is interpreted as a comment. Spaces are allowed !
If you put in meaningful comments, this helps you understand what is going on and orientate yourself in longer programs.

Process Variables

Symbolic Names are used for process variable in PL2000. Theses names can use the following characters:

Capital Letters ..... A to Z
Lower Case Letters ..... a to z
Numbers..... 0 to 9
Special Characters ..... _ and $

Symbolic names also have the following rules:

- The first character of the symbolic name must be a letter.
- Symbolic names are case sensitive. This means that **SWITCH_1**, **Switch_1** and **switch_1** are three different variables !
- The symbolic name has to be a continuous character chain, i.e. **No spaces** are allowed in PV names !
- The length of symbolic names is not limited, but only the **first 10 characters** are significant, i.e. **Temperature_set** and **Temperature_current** are treated as the same variable (the parser makes no distinction between these variables because the first 10 characters are the same).
- Process variables cannot be a keyword in PL2000 syntax (IF, ELSE, etc.) !

PL2000 Constants

A constant is a value that is not changed for the run time of the program.

Constant	Constant Syntax	Examples
Decimal Constant	[+|-]ddd	#34 #-26751 #+67013
Floating Point Constant	[+|-]ddd.[ddd][E|e[+|-]dd]	#-47.23 #3.17E05 #+5.14e-02
Binary Constant	%ddd	%10101010 %1001 %00101001
Hexadecimal Constant	$ddd	$F9 $5A715 $E8B
String Constant	"sss"	"String" "a b c" "Textstring"

Syntax: Characters in square brackets are optional.



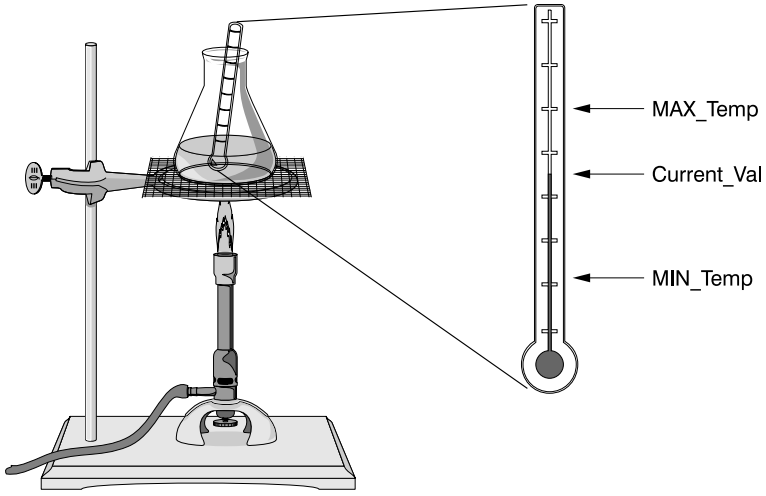
A constant is always interpreted as the smallest possible data type.

6.9.3 PL2000 Editor

The most important functions of the PL2000 editor (marking, copying, etc.) are explained on the following pages by using a simple example.

Example Description

The temperature of a liquid should be kept between two limits (Maximum > current value > minimum).



If the temperature of the liquid goes above the maximum value MAX_Temp, the alarm ALARM_MAX is triggered and the message "Temp>MAX" is shown on the display of the CPU (B&R 2010). If the temperature sinks below the minimum value MIN_Temp, the alarm ALARM_MIN is triggered.

	ALARM_MAX	ALARM_MIN	Status Display
Current_Val > MAX_Temp	1	0	“Temp>MAX” message
MAX_Temp > Current_Val < MIN_Temp	0	0	No Message
Current_Val < MIN_Temp	0	1	No Message

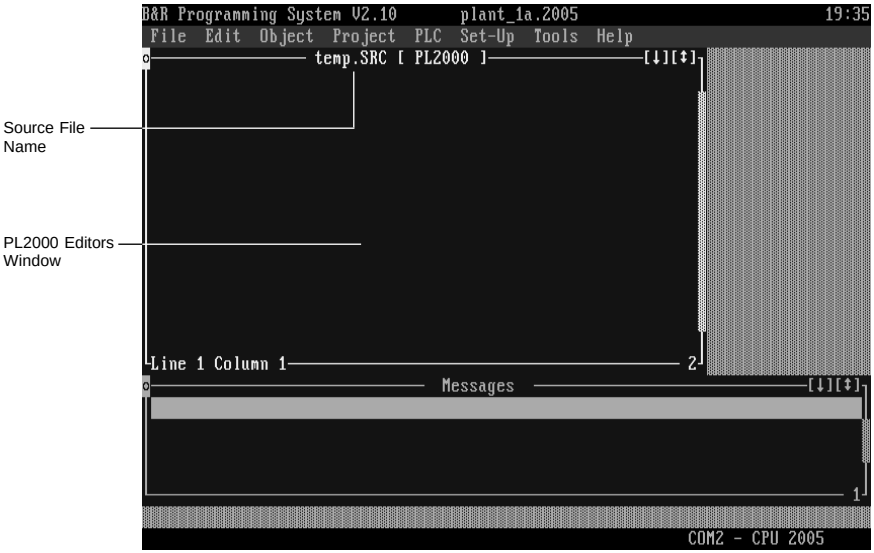
The message output on the standard display is done with the functions "DIS_str and "DIS_clr" (BURTRAP library).
Before you can use the function blocks and inline functions of the BURTRAP library in your project, you have to import the library into the corresponding project data bank. There is the function *Library* (pop up menu of the *Import* function from the *File* menu). You can find detailed information on importing libraries in the section "Functions".

Calling the PL2000 Editor

We recommend that you always call the PL2000 editor from the task layer of the GDM as this makes the whole thing clearer an easier to understand. To do this you have to put a task symbol for an application program on the task layer (see section "6.4 Placing an object in the Task Layer"). Since this example is done in the programming language PL2000, select PL2000 for the task parameter "Source Type". Afterwards you can open the PL2000 editor by pressing the function key [F7 Layer ↓] when the cursor is on the task symbol.

You can also call the PL2000 editor with the function *New* from the pull down menu *File*. A pop up menu appears for you to choose the type of editor. To create a PL2000 source file, select PL2000 from this menu and a PL2000 editor window opens.

PL2000 source code can now be entered in this window:



After opening the PL2000 editor from the GDM, the source file is automatically given the name <Task Name>.SRC. If the editor was called with the *New* function, by default the file is called SOURCExx.SRC, where xx is a the next available number in sequence.

The PL2000 editor can be operated with the mouse or the keyboard.

Entering Source Code

You directly enter source code in the PL2000 editor. The PL2000 editor has a line syntax checker, that can check each line of the source code as it is written. The PL2000 source code is checked for correct syntax either online or when it is saved depending on how you have set it up. If online syntax checking active, syntax errors are found and reported when the line is ended with the entry key[↵] . The message "Syntax Error" is displayed in the message window and the corresponding error description and number is shown in the message file.

Marking Text Blocks

You can choose between two different marking modes (line or column marking). The desired marking mode is set in the editor settings (see section "6.9.4 Configuring the PL2000 Editor"). Line marking mode is set by default.



To make a block, put the cursor at the start of the area you want to mark. Press and hold the [⇧] key and move to the end of the area you want to mark with the cursor keys.



Put the cursor at the start of the area you want to mark. Select the function Mark for the pull down menu Edit. Move the cursor to the desired end position.
If you select the Mark function again, marking mode is turned off.

Text and symbols in the marked areas are either shown in inverse or grey, depending on you screen driver:



Copying a Marked Block



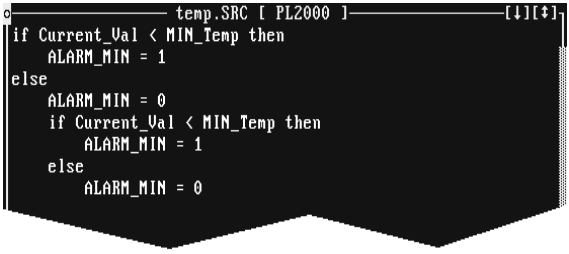
You can copy the marked block into the clipboard with [Ctrl] + [Ins] (block is not deleted from source file). You can insert this (the clipboard contents) at the current cursor position with [⇧] + [Ins]. It also stays in the clipboard.



You can copy the marked block into the clipboard with *Copy* from the *Edit* menu (block is not deleted from source file). Marking mode is automatically turned off.
You can insert this (the clipboard contents) at the current cursor position with *Insert* from the *Edit* menu. It also stays in the clipboard.



The clipboard contents can be inserted into other open PL2000 editor windows.



Search/Replace Text

After doing the block copy, you have to change the variable names in the second program part. You can use the function *Replace* from the *Edit* pull down menu. A dialog box appears when you select this function, for you to enter the search and replace text as well as the direction and start position:



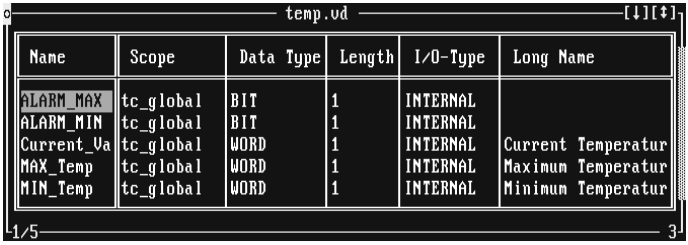
After entering the information and closing this dialog box with [OK], the search starts with the given parameters. If a match is made you are given the following message on the message line:

Replace ? (Y)es (N)o (A)ll (E)nd

[Y] ..... Text found is replaced with the replace text. The search continues in the selected direction.
[N] ..... Text found is **not** replaced with the replace text. The search continues in the selected direction.
[A] ..... Text found is replaced with the replace text. The search continues in the selected direction and all matches found are replaced without asking for confirmation.
[E] ..... The *Replace* operation is terminated.

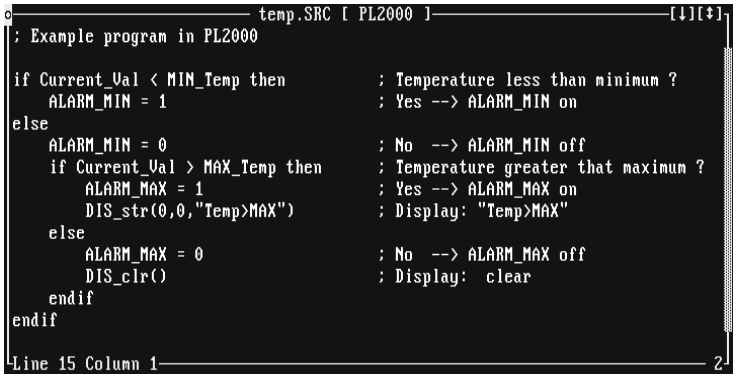
Saving a Program

The PL2000 editor has a line syntax checker, that can check each line of the source code as it is written. The PL2000 source code is checked for correct syntax either online or when it is saved depending on how you have set it up.
If there are any undeclared variables, a dialog gives you the corresponding warning. If you close this box with [OK], the variable declaration appears:



You can find detailed information on the variable declaration in section "6.10 Variable Declaration".

The source code for the example PL2000 program should look like this:



To save the PL2000 program under the name <Task Name>.SRC, use the function *Save* from the *File* or *System* menu. The source file <Task Name>.SRC is put in the program files directory of the current project.

PL2000 Editor Key Commands

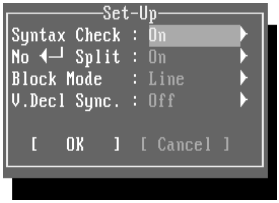
The following keys and key combinations can be used in the PL2000 editor:

Key	Description
[⇑]	Cursor up
[⇓]	Cursor down
[←]	Cursor left
[→]	Cursor right.
[↵]	Cursor to the first column, next line.
[PgUp]	Page up
[PgDn]	Page down
[Home]	Cursor to the start of the line
[End]	Cursor to the end of the line
[Ctrl] + [Home]	Cursor to the start of the code
[Ctrl] + [End]	Cursor to the end of the code
[Del]	Delete character under the cursor.
[Ins]	Switch between insert and overwrite mode.
[Tab]	Move cursor to the next tab position. In overwrite mode the text up to the next tab stop is deleted. In insert Mode the text on the right of the cursor is moved right with the cursor.
[⇑] + [Tab]	Move cursor to the previous tab position.
[↵]	End a line. [1] In overwrite mode, the cursor is moved to the start of the next line. In insert mode, a line is inserted and the cursor is moved to the start of the new line (see the section "6.9.4 Configuring the PL2000 Editor"). If the cursor is in the first column, a line is inserted before this line.
[⇐]	Delete the character on the left of the cursor.
[Ctrl] + [⇐]	Delete the current line.
[⇑] + [Home]	The area from the current cursor position to the start of the line is marked.
[⇑] + [End]	The area from the current cursor position to the end of the line is marked.
[⇑] + [Cursor keys]	Mark any area.
[Ctrl] + [Ins]	The marked area is copied into the clipboard but not deleted from the source code.
[⇑] + [Ins]	Insert the clipboard contents into the editor.
[⇑] + [Del]	The marked area is moved into the clipboard, i.e. it is deleted from the source code.
[Ctrl] + [F3]	Call the position/symbol sensitive pop up menu.
[Esc]	Call the system menu or cancel the current function.

[1] Action is only executed if the syntax checker reports no error

6.9.4 Configuring the PL2000 Editor

By selecting the function *Editor* from the pull down menu *Set Up*, you open a pop up menu with a list of all the editors that can be setup. Select *Statement List* to change default settings of the PL2000 Editor. This opens the following dialog box:



Syntax Check The editor has a line syntax checker, that checks each line for correct syntax. Syntax checking is either online when you enter the source code or when the source file is saved.

On ..... Each line of source code is checked when it is entered (online) for correct syntax. If there is a syntax error, the corresponding message appears in the message line and the cursor cannot be moved with [↑], [↓] and [↵] to another line. You can only leave the incorrect line after correction or by pressing [↑], [↓] or [↵] twice.

Off ..... The syntax check is only done when the file is saved. This means that if there is a syntax error while you are entering the text it is ignored (no message in the message line, you can leave the line). The message appears when the file is saved.

No [↵] Split

On ..... A line cannot be split [↵] (line break is not possible).

Off ..... In insert mode a line can be split into two with [↵] (line break is possible). The text from the current cursor position to the end of the line is put on the new line.

Block Mode Toggle marking mode (only has an effect when syntax checking is off):

Line ..... Only linewise marking is possible.

Column ..... Only columnwise marking is possible.

V.Decl Sync. This option only has an effect when syntax check is turned on.

Yes ..... New process variables are immediately shown in an open variable declaration window.

No ..... New process variables are only shown in a variable declaration window when it is reopened.



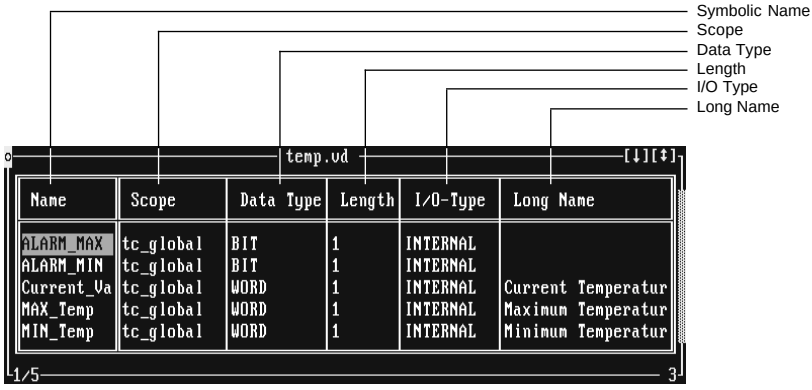
To permanently keep these settings, use the Save function from the pull down menu Set Up. If you don't save the settings, they only apply to the current PG2000 session.

6.10 VARIABLE DECLARATION

You can define the connection between the symbolic names and the hardware with the variable declaration. To call the variable declaration for the selected object, select the function *Variable* from the pull down menu *Object*. This function can be called directly from the corresponding editor or from all GDM layers:

PLC Layer ..... Call the variable declaration for all tasks in the selected PLC.
Processor Layer ..... Call the variable declaration for all tasks in the selected processor.
Task Layer or Editor ..... Call the variable declaration for the selected task.

After selecting the function Variable, the variable declaration editor appears with a list of all process variables (PVs) in the current object. The variable declarations can be completed or changed.
You can call a pop up menu with the function key [F1 Filter] to specify different display masks or filters.
The look of the variable declaration can be configured in the editor set up (see section "6.10.8 Configuring the variable Declaration Editor").



6.10.1 Symbolic Names

All symbolic names used for process variables in the source file that match the current filter are shown in this column. You cannot change or add new names here.

Symbolic Names are used for process variable in PG2000 programs. Theses names can use the following characters:

Capital Letters ..... A to Z
Lower Case Letters ..... a to z
Numbers ..... 0 to 9
Special Characters ..... _ and $

Symbolic names also have the following rules:

- The first character of the symbolic name must be a letter.
- Symbolic names are case sensitive. This means that **SWITCH_1**, **Switch_1** and **switch_1** are three different variables !
- The symbolic name has to be a continuous character chain, i.e. **No spaces** are allowed in PV names !
- The length of symbolic names is not limited, but only the **first 10 characters** are significant, i.e. **Temperature_set** and **Temperature_current** are treated as the same variable (the parser makes no distinction between these variables because the first 10 characters are the same).
- Process variables cannot be a keyword in a programming language syntax (IF, ELSE, etc.) !

6.10.2 Scope

You can set the scope of a variable with a pop up menu. This is where the variable can be used. Single processor systems can have one of the four options:



For scope *plc_global* and *local*, only the I/O Type *Internal* can be used !

tc_global ..... A variable with scope of *tc_global* can be used by all tasks within the **same** task class Task class global). They are always in the same physical memory location.



The high speed, exception and interrupt task classes share the task class global data area with TC#1. This means that variables in TC#1 declared as tc_global are also available in all high speed, exception and interrupt task classes !

plc_global ..... The corresponding variable is available for all tasks in all task classes on the current PLC. They are always in the same physical memory location.

local ..... Local variables are only available for a task. If the same symbolic name is used in another task, it is a new completely independent variable.

dynamic ..... A dynamic variable is similar to a pointer in the programming language C. It always takes up 4 bytes (LONG) of memory. The dynamic variables can be assigned during the run time of the program via this address and can therefore access any variable of any type. The address assignment is done with a so-called address operator. You can find detailed information on this in the "B&R SYSTEM 2000 Programming Languages Manual" in the "PL2000" chapter. Dynamic variables are **always local !**

6.10.3 Data Type

This allows you to a set the data type of a variable. The data type defines the amount of memory needed and the range of values possible. The data types available are: BIT, INT8, INT16, INT32, BYTE, WORD, LONG and FLOAT. There are two additional options typedef and FBK



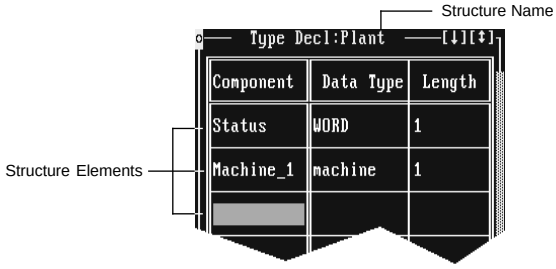
When addressing I/O modules, only particular data types are allowed. Please use the data types as specified in the hardware module descriptions (see "B&R 2000 Hardware User's Manual").

The following table lists the value ranges of the simple data types (BIT, INT8, INT16, INT32, BYTE, WORD, LONG and FLOAT):

Data Type	Bit Range	Numerical Range
BIT	1 Bit	0 1
INT8	7 Bit + Sign	-128 0 127
BYTE	8 Bit	0 255
INT16	15 Bit + Sign	-32,768 0 32,767
WORD	16 Bit	0 65,535
INT32	31 Bit + Sign	-2,147,483,648 0 2,147,483,647
LONG	32 Bit	0 4,294,967,295
FLOAT	32 Bit	-3.4 E+38 -1.18 e-38 0 1.18 e-38 3.4 E+38

typedef ..... You can declare structures with the data type typedef. You can find detailed information on "Structures" in the "B&R SYSTEM 2000 Programming Languages Manual" in the "PL2000" chapter.
If this data type is selected, you can enter the name of the structure. If there are already structures defined in the current project, there will be a selection list with these structure types in.

After entering the structure name, a dialog box opens for you to type in the components of the structure:



The data type **BIT cannot** be used in a structure.

Structure elements can also be arrays or structures. If a new structure is entered as a component, another structure definition box opens.

FBK_ALIAS .... Access to the inputs and outputs of a function block (FBK) via a structure variable (alias structure). You can find detailed information about alias calls of function blocks in the "B&R SYSTEM 2000 Programming Languages Manual".

6.10.4 Length

This entry is the number of array elements of a variable. If the entry is one (1), the variable is not an array, but just a normal variable. If the length is greater than 1, an array of the given data type is created that must be accessed with an index.
For example, if the WORD variable "word_array" is declared with a length of 10, 10 variables of type WORD can be accessed with the index 0 to 9 (word_array[0] to word_array[9]).
You can find detailed information on arrays in the "PL2000" chapter in the "B&R SYSTEM 2000 Programming Language Manual".

6.10.5 I/O Type

This entry defines the relationship between the variables and the physical hardware components of the PLC. The various types of I/O types (INTERNAL, 2010 I/O, REMOTE I/O, 2010 I/O, IP) are shown in a pop up window that are possible for the current variable (dependent on processor type, scope and data type).

Relationship between Scope and I/O Type:

Scope	I/O Type						
	INTERNAL	2010 I/O	REMOTE I/O	2005 I/O	IP	MP/PP	Reference
tc_global	✓	✓	✓	✓	✓		
plc_global	✓					✓	
local	✓						
dynamic							✓



Since the address assignment for dynamic variable is done only during run time, the I/O type *Reference* is automatically entered for the *Dynamic* scope.
For the I/O types *2010-I/O*, *REMOTE-I/O* and *2005-I/O* on the scope *tc_global* can be used.

INTERNAL ..... A variable defined as I/O type INTERNAL, is allocated a memory location in the dual Ported RAM (DPR) of the processor. Internals have no direct connection to the hardware, but only get their values by program assignments. An internal keeps its value until it is changed by another program assignment. If there is a power down, internals keep their value.

2010-I/O ..... I/O Module of the B&R 2010 system.
The symbolic name is assigned to a particular channel on a certain module.

The **Module Adr.** entry is the slot number of the particular module.
The module address (1 to max. 99) of the B&R 2010 I/O module is set by the slot (slot coding). Numbering for addresses begins at slot 1 of the I/O bus with module address 1. There is a 7 segment display on the I/O modules showing the module address.

The **Module Type** can be selected from a pop up menu. Only the module types that are allowed for the current variable are shown (dependent on processor type, scope and data type).

Module Type	Notes
Digit. In	Digit. In are input modules that allow external control components (relay, ...) to be read with a digital value.
Digit. Out	Digit. Out are output modules can write digital values to external control components (relay, ...).
Analog In	Analog In are modules that convert the measurement values (e.g. current, voltage, temperature,...) into numerical values and pass them to the CPU.
Analog Out	Analog Out are modules that convert the value given to them by the CPU into current or voltage amounts and pass this to the control components.
Transp. In	Transp. In is for modules that are not standard I/O modules (like digital or analog input modules). An other use is digital array access with analog data types to digital inputs.
Transp. Out	Transp. Out is for modules that are not standard I/O modules (like digital or analog output modules). An other use is digital array access with analog data types to digital outputs.
Status In	Status In allows you to access the status area of I/O modules (e.g. reading terminal block status).
Status Out	Status Out allows you to access the status area of I/O modules (e.g. resetting counter).



Information about which module type is to be used is in the hardware description (see "B&R 2000 Hardware Manual').

The channel number (0 to 255) defines a particular Input/output channel (digital or analog) that can be accessed with the symbolic name.

REMOTE-I/O ... This allows you to assign a symbolic name to a particular channel on a module on a Remote I/O bus.
The Remote Master system module it is possible to have up to 31 I/O buses, each of which can have up to 99 (B&R 2010) or 13 (B&R 2005) modules on them.

The setting in **Master** is only used in Remote I/O firmware version 4.0 and later.

Slave is the slave address of the remote slave station (1 to max. 99). The slave address is set with the two number switches (NODE #) on the remote slave module.

Slave Type is the control system type of the remote slave station (B&R 2010 or B&R 2005).

The **Module Address** of I/O modules is determined by the slot (slot coding) on both the B&R 2010 (1 to 99) and B&R 2005 (1 to 13).
B&R 2010: Numbering for addresses starts at slot 1 of the I/O bus with module address 1. There is a seven segment display on the module showing the module address.
B&R 2005: Numbering for addresses starts at slot 3 of the I/O bus with module address 1.

Module Type is selected from a pop up menu. Only the module types are shown that are allowed for the current variable (dependent on processor type, scope and data type). You can find a description of the individual module types in "2010 I/O".

The **channel number** (0 to 255) selects a particular input/output (digital or analog) that can be accessed with the symbolic name.

2005 I/O ..... I/O module of the B&R 2005 system.
The symbolic name is assigned to a particular channel of a module (using base plate, module address, module type and channel number).

The main base plate of the B&R 2005 system can have a maximum of 11 I/O modules. If a CPU module with an expansion master is used, you can connect up to 4 expansion base plates. You have to select which **base plate** the I/O module is in from a pop up menu (main, Exp. 1, Exp. 2, Exp. 3, Exp. 4).

Module Adr. defines which module the symbolic name refers to. The module address (1 to 13) of I/O modules is set by the slot (slot coding). Numbering of the addresses starts at slot 3 of the I/O bus with module address 1.

Module Type is selected from a pop up menu. Only module types that are allowed for the current variable are shown (dependent on processor type, scope, and data type). You can find a description of the individual module types under "2010 I/O".

The **Channel Number** (0 to 255) selects a particular input/output (digital or analog) that can be accessed with the symbolic name.

IP ..... Reserved for intelligent I/O processors (IPs).

MP/PP ..... Reserved for multiprocessors (MPs) and peripheral processors (PPs).

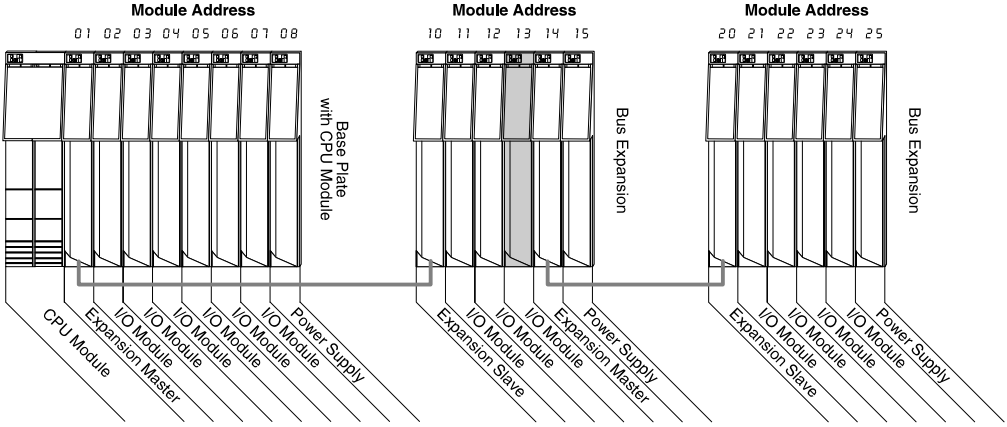
Reference ..... Since dynamic variables are assigned an address during run time, variables with dynamic scope are automatically given *Reference* I/O Type.

The following pages contain a few examples for the separation of the local I/O buses into several bus segments and the respective I/O type settings:

Bus Expansion B&R 2010:

It is possible to address up to 99 I/O modules on a 2010 I/O bus. Since a maximum of 20 I/O modules are allowed to be directly next to each other and there would be a space problem if 99 modules were put together (4m long I/O bus), you can split the I/O bus into segments with the expansion master and slave modules. The 2010 I/O bus can be divided into up to 10 segments. This is how you can reach the maximum of 99 I/O modules.
The module address of the expansion is to be set with the node number switch. The addresses on the slave base plate continue from this number. The slave address is set in steps of 10 (10, 20, ... 90), and please note that you can only use each address once.
You can find detailed information on topic in the "B&R 2000 Hardware User's Manual".

Example:



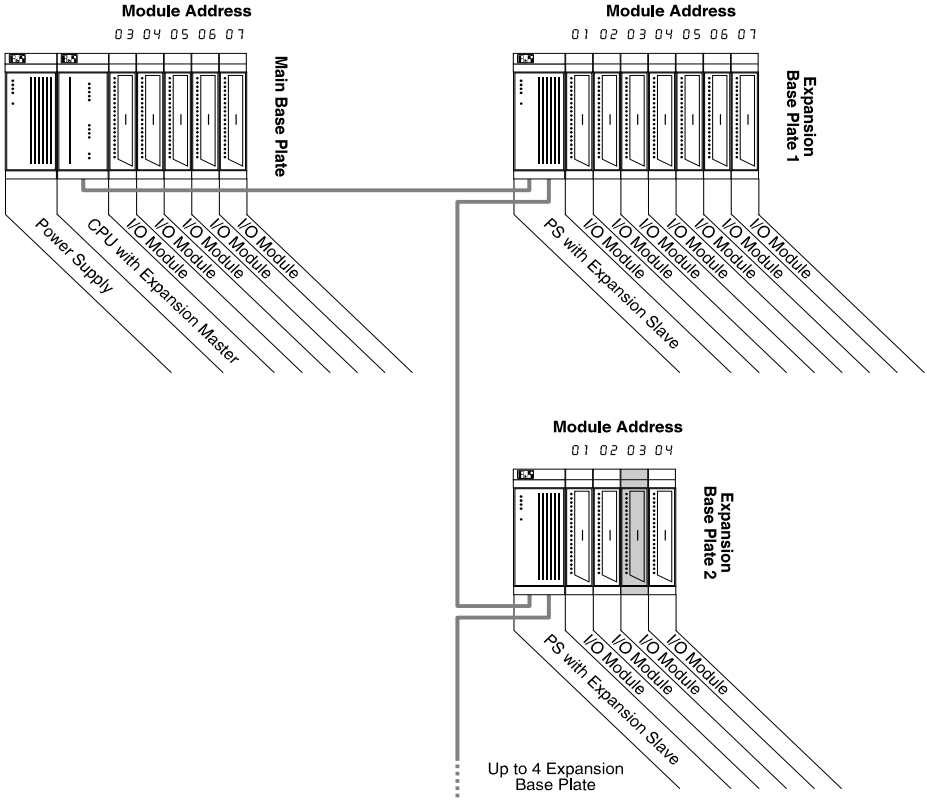
The I/O Type setting for the 3rd channel of the gray coloured I/O module (e.g. analog input module):

Module Adr. : 13
Module Type : Analog In
Channel No. : 3

Bus Expansion B&R 2005:

The main base plate can have a maximum of 11 I/O or system modules. You can expand the main base plate by using special modules (CPU module with expansion master and power supply with expansion slave). You can find detailed information on "B&R 2005 Bus Expansion" in the "B&R 2000 Hardware User's Manual".

Example:



The I/O Type setting for the 5th channel of the gray coloured I/O module (e.g. digital output module):

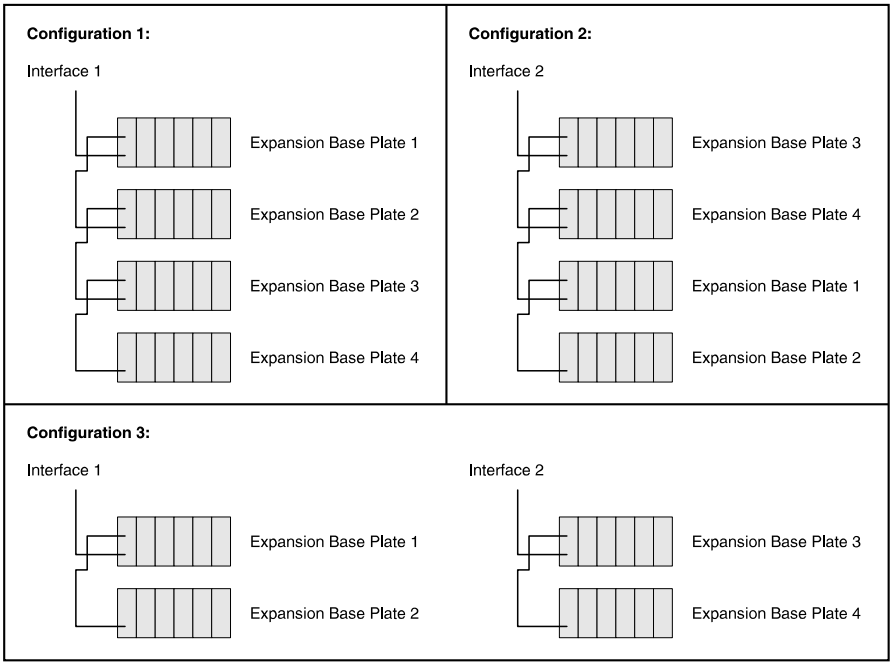
Base Pl.	:	Exp. 2
Module Adr.	:	3
Module Type	:	Digital Out
Channel No.	:	5

B&R 2005 Bus Expansion on a B&R 2010:

You can directly connect a 2005 expansion base plate as an expansion slave (max. 4 per 2010 system) to a 2010 expansion master. Both interfaces of the 2010 expansion master can be used for this type of expansion, but only the following configurations are allowed:

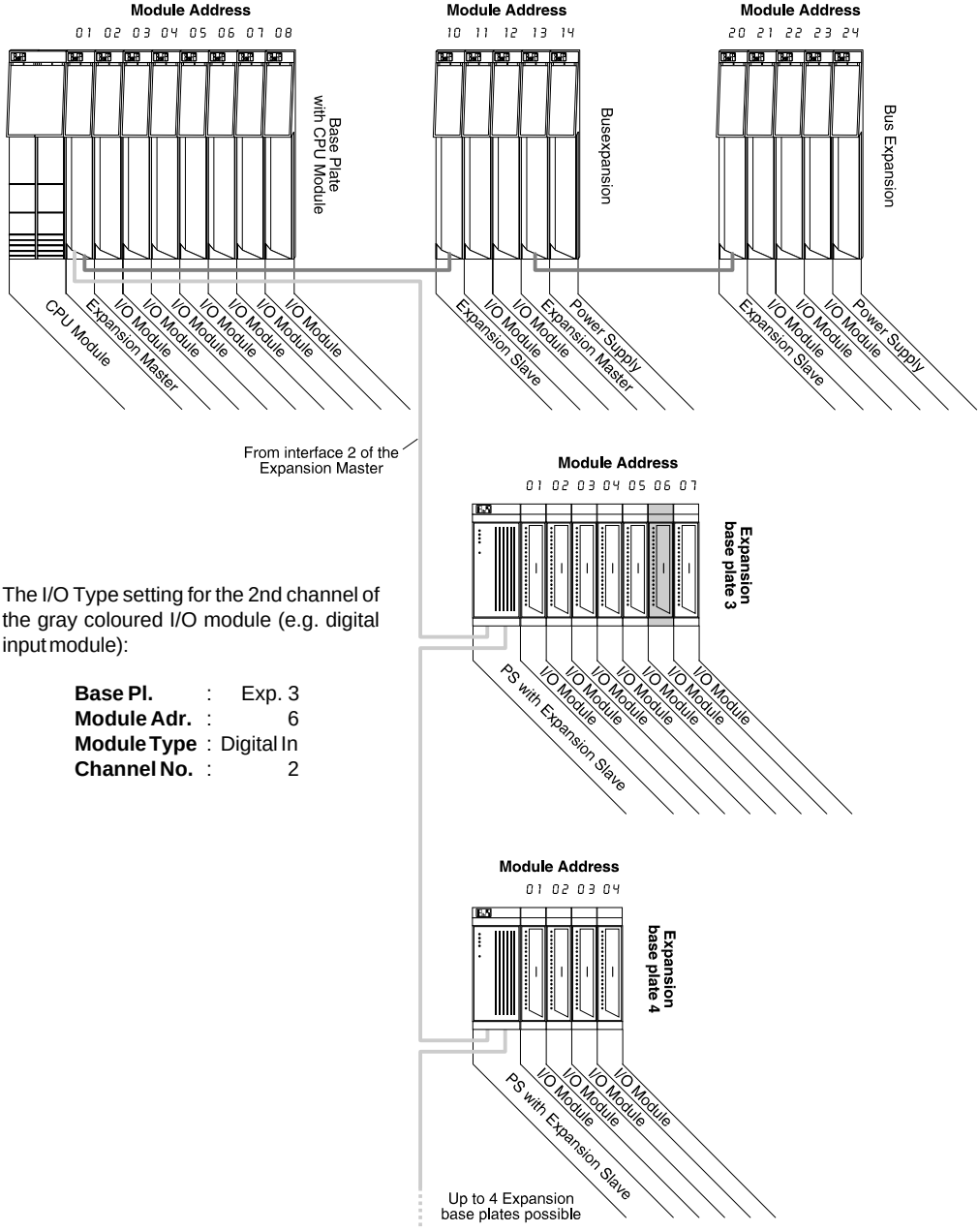
Expansion Master Interface	Number of 2005 Expansion Slaves		
	For configuration		
	1	2	3
Interface 1 (upper Interface)	4	0	2
Interface 2 (lower Interface)	0	4	2

Depending on the configuration chosen, the expansion base plates have the following numbers that have to be given in the variable declaration:



You can find detailed information on this in the "B&R 2000 Hardware User's Manual".

Example for configuration 2 (Expansion Slave 2005 on interface 2 of the 2010 expansion master):



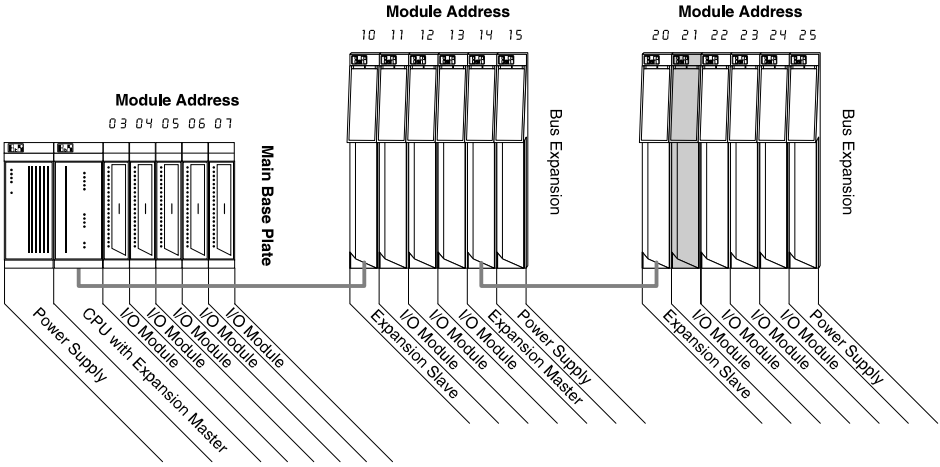
The I/O Type setting for the 2nd channel of the gray coloured I/O module (e.g. digital input module):

Base Pl.	:	Exp. 3
Module Adr.	:	6
Module Type	:	Digital In
Channel No.	:	2

B&R 2010 Bus Expansion on a B&R 2005:

B&R 2010 I/O modules can be connected with an expansion slave to either a 2005 expansion master in the CPU module or a 2005 expansion slave.
The module address of the 2010 expansion slave is set with a node number switch in steps of 10 (00,10, ... 90). The addressing of modules on this expansion base plate continues from this number. Up to 20 I/O modules can be put on a bus segment. It is possible to expand the system to its maximum of 99 modules with other 2010 expansion masters.

Example:



The I/O Type setting for the 8th channel of the gray coloured I/O module (e.g. analog output module):

Module Adr. : 21
Module Type : Analog Out
Channel No. : 8

6.10.6 Init Value

You can set the initialization value of a variable in this column. After each INIT or TOTALINIT of the PLC, the variables are initialized with the init value. Following this, the initialization sub-program (INIT-SP) are executed. You can assign a new value to the variables here.
The Init Value column can be shown or hidden with an option under pull down menu *Set Up*, *Editors* function (you can find detailed information about this in the section "6.10.8 Configuring the Variable Declaration Editor"). This column is not shown by default.



The specified initialization value is only supported in BUGFIX version 1.15 or later (B&R module BFXV115.BR) of the operating system PLC SW V1.10. This means that the column Init Value is only evaluated when the BFX_V115.BR is on the PLC (User ROM).

6.10.7 Long Name

You can enter an additional description for a variable in *Long Name*. This entry is optional, i.e. you don't have to have a long name for every process variable.
The *Long Name* column can be shown or hidden with an option under pull down menu *Set Up*, *Editors* function (you can find detailed information about this in the section "6.10.8 Configuring the Variable Declaration Editor").

Long names can use the following characters:

Capital Letters ..... A to Z
Lower Case Letters ..... a to z
Numbers ..... 0 to 9
Special Characters ..... _ and $

Long names also have the following rules:

- The first character of the symbolic name must be a letter.
- Long names are limited to 35 characters.
- The long name does not have to be a continuous character chain, i.e. **Spaces** are allowed.

6.10.8 Configuring the Variable Declaration Editor

By selecting the *Editor* function from the *Set Up* pull down menu, a pop up menu opens with a list of all editors that can be configured. You can change the default settings of the variable declaration editor by choosing *Variable* from this list. This opens a dialog box to configure the variable declaration editor:



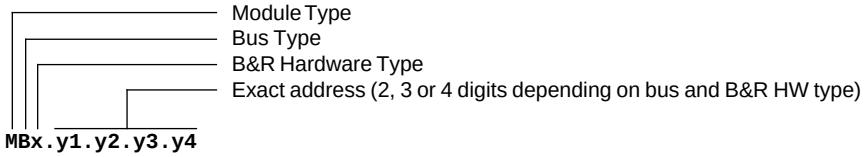
Long Name

Yes ..... A column for the long name is shown in the variable declaration (default). This allows you to enter a short description (35 characters) for a variable.

No ..... The long name column is not shown in the variable declaration editor.

I/O Abbr.

Yes ..... In the I/O Type column, the hardware (HW) assignment of the variable is shown. When this is turned on, you can directly see and enter the HW assignment of the process variables (without using the pop up windows.
HW assignment syntax:

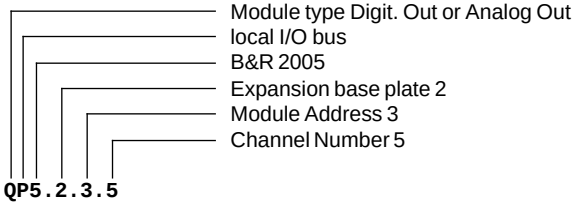


Module Type:	I ..... Digit. In or Analog In Q ..... Digit. Out or Analog Out IT ..... Transp. In QT ..... Transp. Out IS ..... Status In QS ..... Status Out
Bus Type:	P ..... Local I/O bus R ..... Remote I/O bus
B&R Hardware Type:	1 ..... B&R 2010 Control system 5 ..... B&R 2005 Control system

Exact address : (2, 3 or 4 digits depending on bus and B&R HW type)

	Local I/O Bus B&R 2010	Local I/O Bus B&R 2005	Remote I/O Bus B&R 2010	Remote I/O Bus B&R 2005
y1	Module Adr.	Base Plate [1]	Master	Master
y2	Channel No.	Module Adr.	Slave	Slave
y3	---	Channel No.	Module Adr.	Module Adr.
y4	---	---	Channel No.	Channel No.

Example:



No ..... In the I/O Type column only the I/O type of the variable (INTERNAL, 2010 I/O, REMOTE I/O, 2005 I/O or Reference) is shown. When this option is selected, you have to enter the HW assignment with the pop up selection windows (default).

Separt. Line

Yes ..... The individual PV entries in the variable declaration are separated with a line.

No ..... There is no separating line between the individual PV entries in the variable declaration (default).

[1] 0 .... Main Base Plate
1 .... Expansion Base Plate 1
2 .... Expansion Base Plate 2
3 .... Expansion Base Plate 3
4 .... Expansion Base Plate 4

Init Value

Yes ..... A column for the init value (initialization value) of the variables is shown. You can set different initialization value for each variable. After each INIT or TOTALINIT of the PLC, the variables are set to their init values.
The specified initialization value is only supported in BUGFIX version 1.15 or later (B&R module BFX_V115.BR) of the operating system PLC SW V1.10. This means that the column Init Value is only evaluated when the BFX_V115.BR is on the PLC (User ROM).

No ..... The column for the initialization value is not shown (default).



These settings are only made permanent after you have called the function Save form the pull down menu Set Up. If you don't use this function, the settings are only for the current PG2000 session.

6.11 INITIALIZATION SUB-PROGRAMS

You can create a so called initialization sub-program (INIT-SP) for every task. This is a component of the executable program module. INIT-SPs are **only executed in the boot up phase** of the PLC (after a TOTALINIT or INIT) or **after downloading the task** to the PLC. They are generally used for value initialization or running program functions that only have to be done once.



An INIT-SP is not an individual task. It is a component or part of a task. This means that when a task is deleted, compiled, downloaded, etc. the corresponding INIT-SP is also deleted, compiled, downloaded, etc.

6.11.1 Creating an INIT-SP

Call the function *INIT-SP* from the *Object* pull down menu from the editor of the corresponding task to create an INIT-SP, i.e. you can only call this function from the LAD, STL or PL2000 editor. This opens a pop up menu for you to choose a language for the INIT-SP (Ladder Diagram, Statement List or PL2000). Currently, you cannot create a LAD INIT-SP.
After you have selected the programming language, the corresponding editor (STL, PL2000) opens to write the INIT-SP.

6.11.2 Opening or Deleting an INIT-SP

If there is already an INIT-SP for the current task, when you call the *INIT-SP* function from the *Object* pull down menu, a pop up menu appears with the options to either open or delete the existing INIT-SP.
If you choose Open INIT-SP, the existing INIT-SP for the current task is opened in the corresponding editor.
If you choose Delete INIT-SP, the existing INIT-SP is deleted from the data bank.

6.12 COMPILING OBJECTS

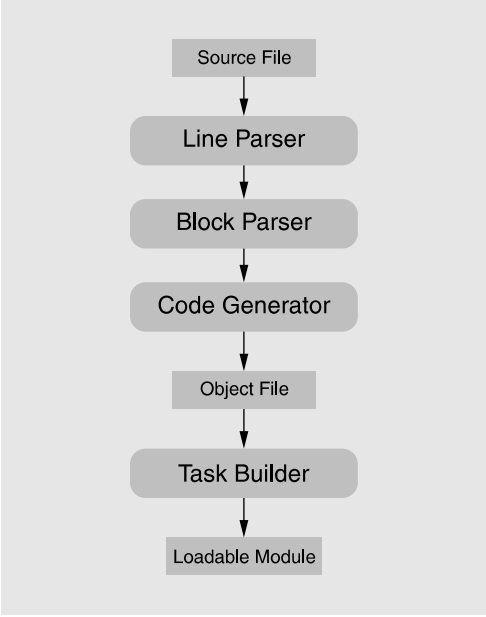
PL2000, STL and LAD Application programs as well as data modules have to be translated into the machine language for the processor of the PLC. A compiler is built in to PG2000 for this translation job. The compiler produces an executable program module (B&R module) that the PLC processor understands and that can be downloaded into the CPU or multiprocessor module.
Each compiled and downloadable modules is saved in the "..\<ProjectName.PGP>\PGM\<PLCName>\" directory under the name <Task-Name>.BR.

To compile the selected object according to the current compiler options, choose the function *Compile* from the *Object* pull down menu. This function can be called directly from the editor and also from all layers in the GDM:

PLC Layer ..... All tasks in the selected PLC are compiled.
Processor Layer ..... All tasks in the selected processor are compiled.
Task Layer or Editor ..... The selected task is compiled.

6.12.1 Compiler Components

The compiler process is influenced by compiler options (see section "6.12.2 Compiler Options") and can be divided into several sub processes:



Complete Compiler Process

The **line parser** checks each individual line of the source code for correct syntax. In the PL2000 editor, the syntax check is either done online or when the file is saved (editor set up). The syntax check in the LAD and STL editors is performed when the file is saved. All variables used in the source code are entered into the same management of the data bank (if they were not already there).

The **block parser** checks if all block constructs (control structures, loops) are correct (e.g. missing END statements) when the PL2000 source file is saved.

The **code generator** creates an object file that contains the machine code and symbolic names. It also checks it all variables used were declared in the data bank of the current databank path.

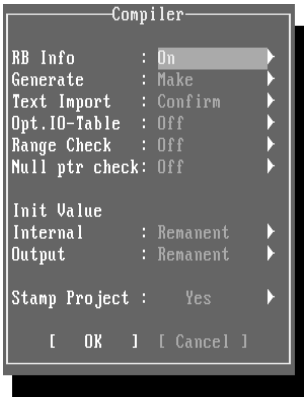
The **task builder** creates an executable program module (B&R module) from the object files and the information in the data bank (hardware assignment of the variables, task parameters, ...).



If you only made changes to the variable declaration (I/O type, scope, length) or task parameters, you don't have to do a complete compiler run through (see "6.12.2 Compiler Options").

6.12.2 Compiler Options

To change the default settings for the compiler, select the *Compiler* function from the *Set Up* pull down menu. This opens a dialog box with the various compiler options:



All entries in this dialog box affect the way the compile procedure runs !

RB - Info (Rebuild Info)

On ..... If you created a B&R module with this option on, you can "decompile" it back to its complete source information (source code, INIT-SP, task parameters, variable declaration). This can be done with the function *Task Rebuild* (pop up menu of the function *Import* from the pull down menu *File*) or the function *Rebuild Object* (task layer pop up menu).

Off ..... No rebuild information is included in the B&R module, i.e. you cannot decompile the module.

Generate

Make ..... Only the necessary parts of the compilation are done on the object:

1. The line parser is only called if the source file of a PL2000 program has a more recent date than the program module.
2. The code generator (PL2000, LAD, STL) is automatically called if the source file or variable declaration has been changed since the last compilation.
3. If neither of the above cases are true, only the task builder is called.

Compiler ..... The code generator and task builder are automatically called for each compile.

Parser ..... A complete compiler process is performed (line parser, block parser, code generator and task builder).

Text Import

You can create ASCII files from source code with the function *Editor->ASCII* (pop up menu of the function *Export* from the pull down menu *File*). At compile time you can, if the file is newer that the corresponding source file, choose to import this file into the data bank and save as the new source file.
The following ASCII files can be imported:

*.AWT files ..... ASCII files of STL programs
*.AWI files ..... ASCII files of INIT-SPs in STL
*.BRT files ..... ASCII files PL2000 programs
*.BRI files ..... ASCII files of INIT-SPs in PL2000

Confirm ..... ASCII import files that are newer than the corresponding source files are imported and saved as new source files before the compilation process begins only if you confirmation it.

Automatic ..... ASCII import files that are newer than the corresponding source files are automatically imported and saved as new source files before the compilation process begins.

No Import ..... Existing ASCII import files care not imported.

Opt.IO Table

On ..... The I/O tables (shovel tables, see section "I/O Data Transfer" in chapter 2 "Operating System PLCSW") are optimized. Only one I/O table is created for each task class that is downloaded in a project download with the function *Object list RUN* (pop up menu PLC Layer). This option is very useful with time critical applications, since I/O variables are only read once within a task class. If you compile without optimized I/O tables variables that are used in several tasks are read in for each task.



Tasks compiled with this options are not executable program modules (extra shovel table) and can only be downloaded with the function *Object list RUN* (see section "6.20 Pop Up Menu of the PLC Layer"). If the tasks are downloaded singly, conflicts can occur on the PLC.

Off ..... No optimized I/O tables are created during compilation. Each task compiled with this option is a stand alone executable program module and an individual I/O shovel table.

Range Check

On ..... During run time, the array access in the task is monitored. If the CPU notices access to a nonexistent element in an array, a "Range Overflow" exception is triggered. You can find detailed information on exceptions in chapter 2 "Operating System PLCSW".

Off ..... Array access is not checked.

Null Ptr Check

On ..... The pointer access is monitored during run time. If the CPU recognizes access to an uninitialized pointer (points to address 0), a "Null Pointer" exception is triggered. You can find detailed information on exceptions in chapter 2 "Operating System PLCSW".

Off ..... Pointer access is not monitored.

Init Value

You can define an initialization value for each process variable in the variable declaration. Along with this you can set a concrete value for every single variable, a **global default initialization value** for **all INTERNAL variables** with the compiler options "Internal".

Remanent ..... No default initialization value exists for INTERNALs.

0 ..... By default, all INTERNALS are given the initialization value "0".

Output

You can define an initialization value for each process variable in the variable declaration. Along with this you can set a concrete value for every single variable, a **global default initialization value** for **all outputs variables** (digital or analog) with the compiler options "Output".

Remanent ..... No default initialization value exists for all digital and analog outputs.

0 ..... By default all digital and analog outputs are given the initialization value "0".

Stamp Project

Yes ..... You can stamp a version number on a project when you compile it. This is so you can keep track of which version are on the PLC and in PG2000.

No ..... Projects are not stamped.



To permanently keep these settings, use the Save function from the pull down menu Set Up. If you don't save the settings, they only apply to the current PG2000 session.

6.13 DOWNLOAD OBJECT IN USER RAM

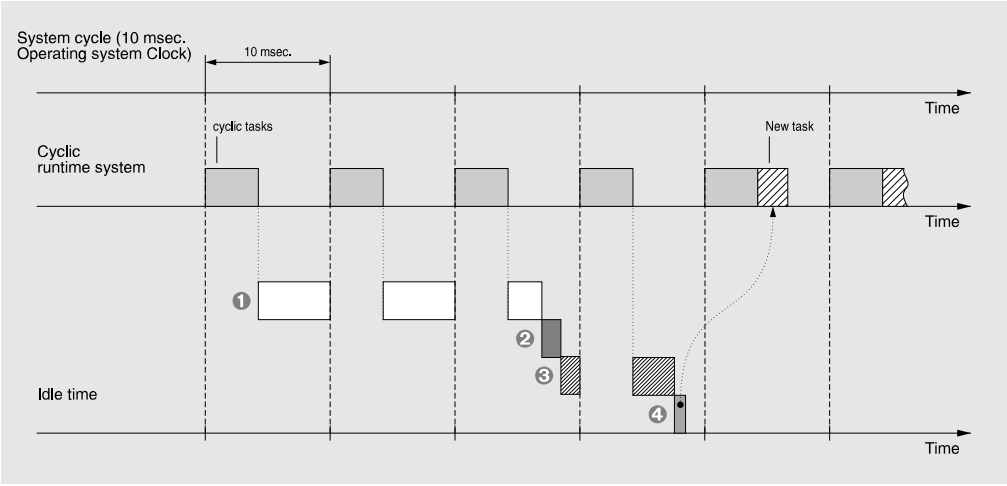
To compile the selected object according to the current compiler options and download it to the User RAM of the current PLC, choose the function *R U N* from the *Object* pull down menu. This function can be called directly from the editor and also from all layers in the GDM (prerequisite: active connection between PLC and PG2000):

PLC Layer ..... All tasks in the selected PLC are compiled and downloaded into the User RAM of the current PLC.
Processor Layer ..... All tasks in the selected processor are compiled and downloaded into the User RAM of the current PLC.
Task Layer or Editor ..... The selected task is compiled and downloaded into the User RAM of the current PLC.

When you download a task into the user RAM, there are two different functions: "Download" and "Replace". If there is already an old version of the task on the PLC (user RAM, FIX RAM or user ROM), the function Replace is used. If the task doesn't already exist on the PLC, Download is used.

6.13.1 Download Function

The download function servers to transfer the task in the user RAM on the PLC. The function can only be used if the task to be downloaded doesn't exist in the user RAM, FIX RAM or user RAM of the PLC.
Download Function Sequence:

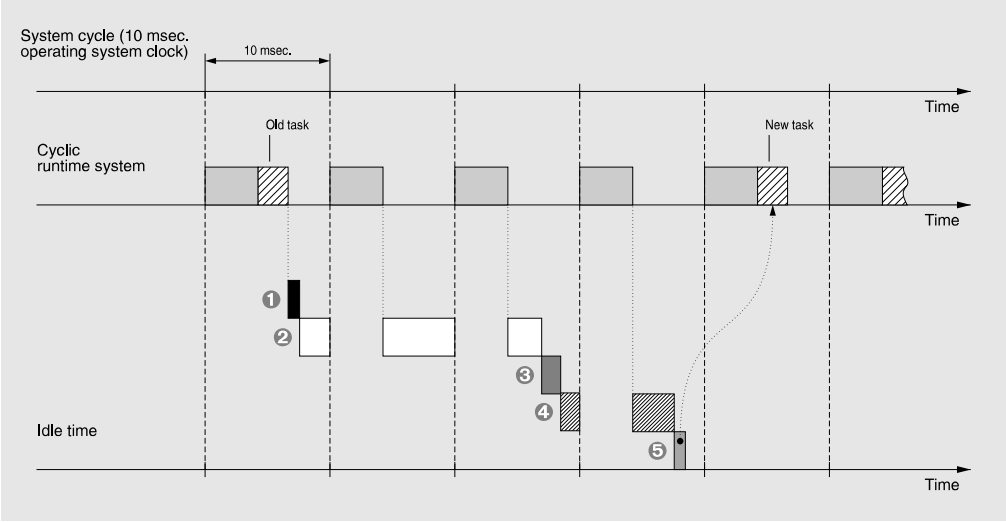


1. Download the new task to the PLC (user RAM)
2. Set up the local memory area
3. Execute the initialization sub program of the new task
4. Start the new task

Next the cyclic tasks are processed in the order they are downloaded. The execution order set with "Inst Number" is only taken in consideration after the next INIT or TOTALINIT.

6.13.2 Overload Function

The overload is a special case of the download function. This function is used if there is already an old version on the PLC (user RAM, FIX RAM or user ROM).
Overload Function Sequence:



1. Delete (deinstall) the already active task
2. Download the new task to the PLC (user RAM)
3. Set up the local memory area
4. Execute the initialization sub program of the new task
5. Start the new task

Next the cyclic tasks are processed in the order they are downloaded. The execution order set with "Inst Number" is only taken in consideration after the next INIT or TOTALINIT.

6.14 DOWNLOAD OBJECT IN THE USER ROM OR FIX-RAM

To compile the selected object according to the current compiler options and download (program) it directly[1] to the selected nonvolatile memory (User ROM or FIX RAM depending on the task parameter "Store Target") of the current PLC, choose the function *Store* from the *Object* pull down menu. This function can be called directly from the editor and also from all layers in the GDM (prerequisite: active connection between PLC and PG2000):

PLC Layer ..... All tasks in the selected PLC are compiled and downloaded into the nonvolatile memory (task parameter "Store Target") of the current PLC.

Processor Layer ..... All tasks in the selected processor are compiled and downloaded into the nonvolatile memory (task parameter "Store Target") of the current PLC.

Task Layer or Editor ..... The selected task is compiled and downloaded into the nonvolatile memory (task parameter "Store Target") of the current PLC.



When storing in the user ROM, the Application Program Memory write protect switch has to be in the write position ! You are not allowed to change the position of the write protect switch during the programming (burning) of B&R modules in the user ROM !

[1] The direct download of B&R modules into the FIX RAM or user ROM of the PLC is only possible with PLCSW V1.10 and later. With older version of the PLCSW, only B&R modules that are already in the user RAM can be programmed into the FIX RAM or User ROM.

6.15 DELETE OBJECTS

Select the function Delete from the Object menu to delete the selected object. This function can be called from all GDM layers ! After calling this function, a dialog box opens with the various options:

Delete module from the PLC

Yes ..... All B&R modules (e.g. compiled tasks, runtime modules of the function libraries, exception, etc.) of the selected object that are in the user RAM, FIX RAM or user ROM of the PLC are deleted from the PLC. There is a confirmation dialog box.



B&R modules that are in the EPROM (user ROM) can only be logically deleted, i.e. these modules are marked as being invalid and are therefore no longer activated. The actual ROM memory is kept until the EPROM is physically deleted (with a UV light).

No ..... If there are B&R modules of the selected object on the PLC (user RAM, FIX RAM or user ROM), they are not deleted.

Delete objects in the GDM

Yes ..... The selected object is deleted from the GDM. There is a confirmation dialog box.
If there are more layers under the active layer (so-called child layers), another dialog box with a warning that these child layers are about to be deleted too.

No ..... The selected object symbol is not deleted.

Delete src and br modules

Yes ..... All source files (*.SRC files) and B&R modules (*.BR) of the selected object are deleted. There is a confirmation dialog box.

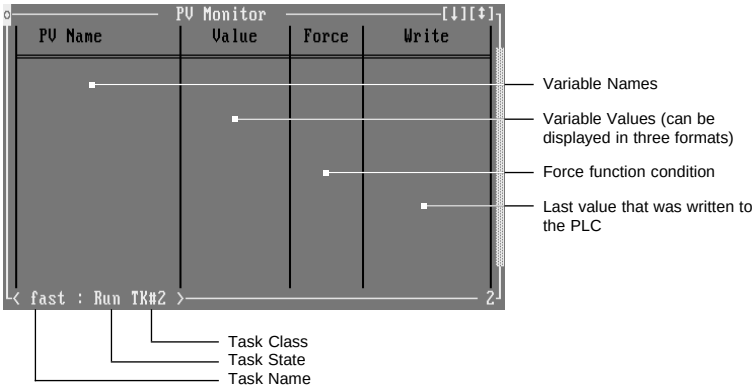
No ..... All source files (*.SRC files) and B&R modules (*.BR) of the selected object are not deleted.

6.16 PROCESS VARIABLE MONITOR (PV MONITOR)

You can look at and write to process variables of a task running on the PLC with the PV Monitor. The variable values are updated at the sampling rate (this can be set with the function *PLC Refresh Rate* from the pull down menu *Set Up*).
To use the PV Monitor on a task running on the PLC, select the function *PV Monitor* from the *Object* menu. This function can only be called from the task layer of the GDM. The cursor must be on the corresponding task symbol when you call this function.
Requirements for trouble free operation of the PV Monitor:

1) Correct (active) connection between the PLC and PG2000.
2) Task is defined in the current project data bank.
3) Task was compiled and downloaded to the PLC error free.

If all the above requirements are met, a PV Monitor window opens:



If there is already a variable list for the selected task available (file <task-name>.VLD), this list of variables is automatically loaded into the PV Monitor window.

You can do the following in the PV Monitor:

- In the above window, any list of variables from the task can be put together. The list can be saved (*.VLD file) and used again at a later date. The symbolic names of the variables are shown in the **PV Name** column.
- After starting the Monitor, the values of the variables are cyclically read from the PLC and displayed in the **Value** column.
- The variables on the PLC can be written to in two ways:
  1) Once off writing any value to a variable. The task can change the variable afterwards.
     The last value written to the PLC is shown in the **Write** column.
  2) Forcing a value on a PV. A variable is given a certain value and held to this value. If the task writes to a forced variable, the changed value stays there during the current task cycle. At the end the variable is overwritten with the force value. The force function is active until it is deactivated from PG2000. This force condition is shown with an LED on the CPU (FORCE).
     This state is indicated in the PV monitor with an "on" in the **Force** column.
- The task can also be stopped, started or run through cyclically (the number of cycles can be set).

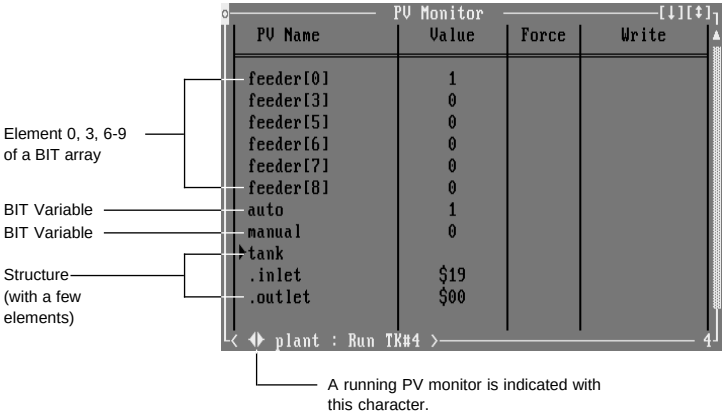
Foot Line Menu of the PV Monitor

[F1 Start/Stop] Toggle function to start/stop the PV Monitor.
When you first call this function the PV Monitor is started. While the PV Monitor is running (started), the variable values are updated at the sample rate (set with the function PLC sampling Rate in the Set Up menu). If you press [F1] again the PV Monitor is stopped and the values are no longer updated.



To start the PV monitor, you have to have at least one variable in the list. To insert a process variable in the list, use the *Insert* function from the pop up menu of the PV Monitor.

This is what a running PV Monitor looks like:



[F2 List] This function calls a pop up menu to load or save a variable list:

Load ..... You can select a saved variable list from the program file directory with the file selector box showing the *.VLD files. The selected file is loaded into the PV Monitor.

Save..... The current variable list is saved as an ASCII file (*.VLD) in the program file directory of the current project. You can type in the file name in a window (default <task name>.VLD).
The variable list name is shown as the window title of the PV monitor.

[F3 Write] The variable that the cursor is on is written to with a value. The value is entered in a window. Please note the following points:

- Put a "$" in front of hexadecimal values (e.g.: $1B0E).
- Put a "%" in front of binary values (e.g.: %1011).
- Whole number floats must have a decimal point (e.g.: 15.0E+4)

[F4 Force] After selecting this function a dialog box appears to turn forcing on or off:

[On] The variable that the cursor is on, is written to with a value and keeps this value. If a task writes to this variable, the values is only changed until the end of the cycle, then it is overwritten with the force value again. Forced variables have "On" in the **Force** column of the PV Monitor. The force value is shown in the **Write** column.

[Off] Force is deactivated for the selected variable.



You can turn forcing off for all variables on the PLC with the function *Force Off* (pop up menu of the *PLC* menu function *Other*. After this function has been called, the PLC can change the values of the previously forced variables.

[Cancel] The entries are ignored and the dialog box is closed.

[F5 PV Format] The display format of the variable can be changed with a small pop up menu:

decimal ..... Decimal number format.

hex ..... Hexadecimal format. A hexadecimal value is indicated in the list with a "$".

binary ..... Binary number format (only for 1 byte variables). A binary value is indicated with a "%".

[F7 Task] A pop up menu appears with the various task functions. The current task state is shown at the bottom left corner of the window.

Start..... Start a stopped task. A running (active) task is indicated with "Run".

Stop ..... Stop a running task. A stopped (inactive) task is indicated with "Stop".

Cycle ..... Cyclic processing of a task. The number of cycles to run through is set with the *Step size* function. The state indicator on the bottom left of the window shows "Step".

Step Size .... You enter the number of cycles to run through with Cycle in a small window. The step size is shown on the bottom left corner of the window frame.

[F8 Editor] The corresponding editor (LAD, PL2000, STL) is called. The source file can be edited.

PV Monitor Pop Up Monitor

You can call the pop up menu in PV Monitor with a mouse double click or by pressing [Ctrl] + [F3]. It provides the functions: insert, delete, write, force and format.

Insert ..... Inserts a variable in the PV Monitor. You can select a variable from a list of all variables in the task.

[OK] The selected variable is entered into the PV Monitor list.

[Cancel] Close the dialog box.

[Name] Another dialog box is opened so that you can enter the name of another variable that is not is the current task (only variables with tc_global or plc_global scope). The given variable is entered into the PV Monitor list.

If the selected process variable is an array or structure, another dialog box opens do that you can enter the index or structure element.

Array:

You can enter individual indices with a comma "," (e.g. 0,3,6,7,8,9) or ranges with a dash "-" (e.g. 0,3,6-9).

[OK] The variables with the defined indexes are entered in the list.

[Cancel] The entries are ignored and the dialog box is closed.

Structure:

All structure elements are shown in a selection list.

[OK] The selected structure element is entered into the PV Monitor list. If this structure element is a structure or array, the corresponding dialog box appears.

[Cancel] The entries are ignored and the dialog box is closed.

Delete ..... The variable under the cursor is deleted from the list.

Write ..... The variable under the cursor is written to with a value. The variable values is entered in a window. Please note the following points:

- Put a "$" in front of hexadecimal values (e.g.: $1B0E).
- Put a "%" in front of binary values (e.g.: %1011).
- Whole number floats must have a decimal point (e.g.: 30.0 or 15.0E+4)

Force ..... After selecting this function a dialog box appears to turn forcing on or off:

[On] The variable that the cursor is on, is written to with a value and keeps this value. If a task writes to this variable, the values is only changed until the end of the cycle, then it is overwritten with the force value again. Forced variables have "On" in the **Force** column of the PV Monitor. The force value is shown in the **Write** column.

[Off] Force is deactivated for the selected variable.



You can turn forcing off for all variables on the PLC with the function *Force Off* (pop up menu of the *PLC* menu function *Other*. After this function has been called, the PLC can change the values of the previously forced variables.

[Cancel] The entries are ignored and the dialog box is closed.

Format ..... The display format of the variable can be changed with a small pop up menu:

decimal ..... Decimal number format.

hex ..... Hexadecimal format. A hexadecimal value is indicated in the list with a "$".

binary ..... Binary number format (only for 1 byte variables). A binary value is indicated with a "%".

PV Monitor Keyboard Commands

You can call various functions directly in the PV Monitor with keys and key combinations:

Key	Description
[↑]	Cursor up.
[↓]	Cursor down.
[PgUp]	Page Up.
[PgDn]	Page Down.
[Ins]	Insert a new variable in the PV Monitor list.
[Del]	The selected variable is deleted from the list.
[w]	The selected variable is written to with a value.
[f]	Change the number format (decimal, hex, binary). You select with a pop up menu.
[s]	Set the step size for the cyclic processing of a task.
[Ctrl] + [F3]	Call the pop up menu.
[Esc]	Call the system menu or cancel the current function.

6.17 DEBUGGER

As well as using the PV Monitor to test tasks running on the PLC, you can call a powerful source level debugger. In PG2000 there is a source level debugger for all three programming languages (Ladder Diagram, Statement List and PL2000):

LAD ..... Ladder Diagram Debugger. Test the ladder diagram with the ladder diagram debugger by calling the function *Debugger* from the *Object* pull down menu. You can find detailed information on this diagnosis tool "6.17.1 Ladder Diagram Debugger (LAD Debugger)".

STL ..... STL and PL2000 Debugger. Test the STL program with the source level debugger by calling the function *Debugger* from the *Object* pull down menu. You can find detailed information on this diagnosis tool "6.17.2 STL and PL2000 Debugger".

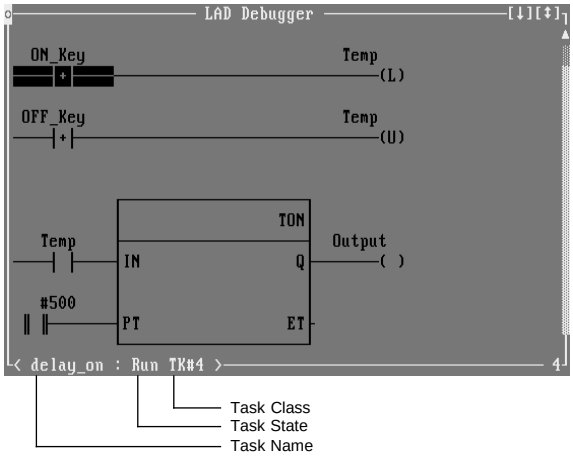
PL2000 ..... STL and PL2000 Debugger. Test the STL program with the source level debugger by calling the function *Debugger* from the *Object* pull down menu. You can find detailed information on this diagnosis tool "6.17.2 STL and PL2000 Debugger".

6.17.1 Ladder Diagram Debugger (LAD Debugger)

You can view and write to variables in a task running on the PLC directly in a Ladder Diagram mask with the LAD debugger. The variable values are updated at the sampling rate (this can be set with the function *PLC Refresh Rate* from the pull down menu *Set Up*).
To use the LAD debugger on a task running on the PLC, select the function *Debugger* from the *Object* menu. You can call this function directly from the LAD editor as well as from the task layer of the GDM. When you call it from the task layer of the GDM, the cursor must be on the corresponding task symbol.
Requirements for trouble free operation of the LAD debugger:

1) Correct (active) connection between the PLC and PG2000.
2) Task is defined in the current project data bank.
3) Task was compiled and downloaded to the PLC error free.

If all the above requirements are met, a LAD debugger window opens:



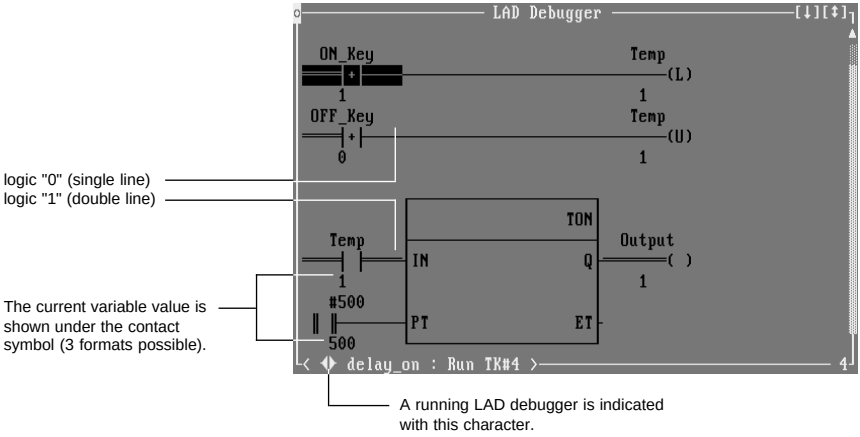
You can do the following in the LAD debugger:

- After starting the LAD debugger, the state of the individual contacts, link lines (connection paths) and function symbol contacts are cyclically read from the PLC (at the set sampling rate) and displayed.
- The variables on the PLC can be written to in two ways:
  1) Once off writing any value to a variable. The task can change the variable afterwards.
  2) Forcing a value on a PV. A variable is given a certain value and held to this value. If the task writes to a forced variable, the changed value stays there during the current task cycle. At the end the variable is overwritten with the force value. The force function is active until it is deactivated from PG2000. This force condition is shown with an LED on the CPU (FORCE).
- The task can also be stopped, started or cyclewise run through (with settable number of cycles).

LAD Debugger Foot Line Menu

[F1 Start/Stop] Toggle function to start/stop the LAD Debugger.
When you first call this function the LAD Debugger is started. While the PV Monitor is running (started), the individual contacts, link lines and function symbol contacts are updated at the sample rate (set with the function PLC sampling Rate in the Set Up menu). If you press [F1] again the LAD Debugger is stopped and the values are no longer updated.

This is what a running LAD debugger looks like:
In a running LAD debugger, the variables values are cyclically read from the PLC and displayed under the contact symbols. Link lines that have a logical condition "1" are shown with a double line:



[F3 Write] The variable that the cursor is on is written to with a value. The value is entered in a window. Please note the following points:

- Put a "$" in front of hexadecimal values (e.g.: $1B0E).
- Put a "%" in front of binary values (e.g.: %1011).
- Whole number floats must have a decimal point (e.g.: 15.0E+4)

The new value is shown under the corresponding LAD symbol in the LAD debugger.

[F4 Force] After selecting this function a dialog box appears to turn forcing on or off:

[On] The variable that the cursor is on, is written to with a value and keeps this value. If a task writes to this variable, the values is only changed until the end of the cycle, then it is overwritten with the force value again. The force value is shown under the corresponding contact symbol in the LAD debugger.

[Off] Force is deactivated for the selected variable.



You can turn forcing off for all variables on the PLC with the function *Force Off* (pop up menu of the *PLC* menu function *Other*. After this function has been called, the PLC can changed the values of the previously forced variables.

[Cancel] The entries are ignored and the dialog box is closed.

[F5 PV Format] The display format of the selected contact can be changed with a small pop up menu:

decimal ..... Decimal number format.

hex ..... Hexadecimal format. A hexadecimal value is indicated in the list with a "$".

binary ..... Binary number format (only for 1 byte variables). A binary value is indicated with a "%".

[F7 Task] A pop up menu appears with the various task functions. The current task state is shown at the bottom left corner of the window.

Start ..... Start a stopped task. A running (active) task is indicated with "Run".

Stop ..... Stop a running task. A stopped (inactive) task is indicated with "Stop".

Cycle ..... Cyclic processing of a task. The number of cycles to run through is set with the *Step size* function. The state indicator on the bottom left of the window shows "Step".

Step Size .... You enter the number of cycles to run through with Cycle in a small window. The step size is shown on the bottom left corner of the window frame.

[F8 Editor] This functions switches you from the LAD debugger to the editor with the current LAD. You can edit the source file in the editor.

LAD Debugger Keyboard Commands

You can call various functions directly in the LAD Debugger with keys and key combinations:

Key	Description
[↑]	Cursor up.
[↓]	Cursor down.
[←]	Cursor left one column.
[→]	Cursor right one column.
[PgUp]	Page Up.
[PgDn]	Page Down.
[Home]	Cursor to the start of the line.
[End]	Cursor to the end of the line.
[w]	The selected variable (contact) is written to with a value.
[c]	A dialog box appears to turn forcing on or off.
[+]	The key on the *numerical keypad* starts a stopped task.
[-]	The key on the *numerical keypad* stops a running task.
[Tab]	Cyclic processing of a task. The number of cycles is set with step size.
[s]	Set the step size for the cyclic processing of a task.
[y]	Switch to the Editor with the current LAD. You can edit the source file if necessary.
[Esc]	Call the system menu or cancel the current function.

6.17.2 STL and PL2000 Debugger

You can run through a program written in STL or PL2000 stepwise or cyclewise with the source level debugger. It is also possible to set break points and watch and alter the variable values in a separate window (Process Variable Monitor).

The Debugger for STL and PL2000 is an additional task that runs on the PLC. If the debugger is not already on the PLC (user RAM, FIX RAM or User ROM, it is automatically downloaded into the user RAM when you use the *Debugger* function (*Object* pull down menu).



You shouldn't store the debugger (B&R module DEBUGGER.BR) in the user ROM or FIX RAM because a new debugger is always delivered with the new PG2000 version. If there is already a debugger on the PLC (user RAM, FIX RAM or user ROM, the debugger is not downloaded again, even if the debugger on the PLC is an older one. In this case there can be a conflict between the debugger on the PLC and in PG2000. The only way to resolve this is to remove the debugger on the PLC.

To use the source level debugger on a task, select the function *Debugger* from the *Object* menu. You can only call this function from the task layer of the GDM. When you call it from the task layer of the GDM, the cursor must be on the corresponding task symbol.
Requirements for trouble free operation of the debugger:

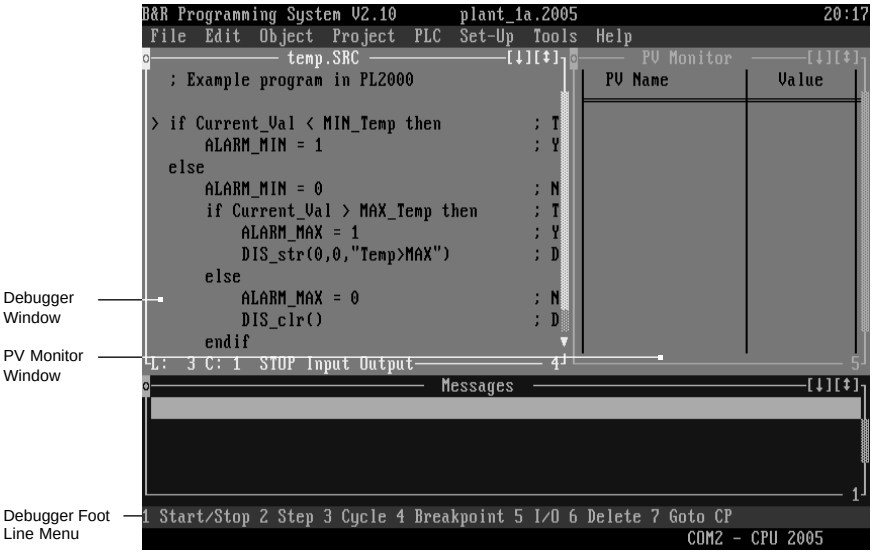
1) Correct (active) connection between the PLC and PG2000.
2) Task is defined in the current project data bank.
3) Task was compiled and downloaded to the PLC error free.

If all the above requirements are met, the debugger and then the selected task are downloaded into the user RAM on the PLC.



If the debugger and the task to be tested are already on the PLC (user RAM, FIX RAM or user RAM) they are not downloaded again !

When the debugger for STL and PL2000 is called, the PV Monitor is automatically started too:



After you call the debugger, the task is stopped. i.e. the task is not processed.

You can do the following in the STL/PL2000 debugger:

- Start and stop a task.
- Step by step processing of a task. This allows you to see the effect of each single program line immediately.
- Cyclic processing of a task.
- The values of variable on the PLC can be shown and changed with the PV monitor.
- You can decide if the task can change or is totally isolated from the outputs of the system during debugging. This allows you to test critical program parts with the most safety for the equipment.

This section only deals with the debugger for STL and PL2000. The Process Variable Monitor is described in "6.16 Process Variable Monitor (PV Monitor).

Foot Line Menu for STL and PL2000

[F1 Start/Stop] Toggle function to start/stop the LAD Debugger.
When you first call this function the Debugger is started and runs to the next breakpoint or until you press the [F1 Start/Stop]. The variable list in the PV Monitor is only updated when the task is running !

The task state is shown in the bottom left corner of the debugger window.

Task Status: RUN ..... task running (started)
STOP ..... task stopped
Line and column position are also shown.

[F2 Step] One program line of the task is executed. The variable values are updated afterwards in the PV Monitor.
The program line that will be executed when you press [F2 Step] is indicated in the debugger window with a ">" character. After executing the line, this so-called code pointer points to the next line that will be executed:



Moving the cursor does not affect the code pointer. A task is always processed from the current code pointer when you press [F2 Step].

[F3 Cycle] The task is executed once to the end of the program or the next breakpoint. The code pointer indicates the first program line will be executed.



Moving the cursor does not affect the code pointer. A task is always processed from the current code pointer.

[F4 Breakpoint] A break point is set on the current cursor line. If there is already a breakpoint on this line, it is removed. Breakpoints are indicated with a "=" character on the left hand side of the window.
If the task is started with [F1] or [F3], tha task runs to the next break point and stops. The code pointer and cursor are moved to the line with the breakpoint. A line is marked with "**»"** when the program stops on a breakpoint.

[F5 I/O] This function calls a pop up menu. You can turn on or off the automatic reading of input images (or writing of output images) for this task.

Input ..... This function toggles the automatic shovelling of the input image for the task being tested. .
The current status is shown on the bottom left corner of the debugger window. If the word "input" is displayed, automatic shovelling of the input image is turned on, otherwise it is turned off.

Output ..... This function toggles the automatic shovelling of the input image for the task being tested. You can choose whether the task should change the actual outputs of the system during debugging or if it should be isolated from the machine physically. This lets you test the program with upmost safety for you equipment. ....
The current status is shown on the bottom left corner of the debugger window. If the word "output" is displayed, automatic shovelling of the output image is turned on, otherwise it is turned off.

Manual ..... This function has not been realized yet.



Turning off automatic shovelling of the input and output images disconnects the task from the external hardware. The inputs have no effect and the outputs don't change the output modules.

[F6 Delete] All breakpoints are deleted.

[F7 Goto CP] Move the cursor to the line with the code pointer on it.

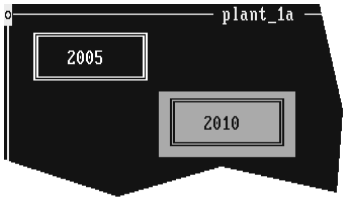
STL/PL2000 Debugger Keyboard Commands

You can call various functions directly in the Debugger with keys and key combinations:

Key	Description
[↑]	Cursor up.
[↓]	Cursor down.
[PgUp]	Page Up.
[PgDn]	Page Down.
[Home]	Cursor to the start of the line.
[End]	Cursor to the end of the line.
[Ctrl] + [Home]	Cursor to the start of the code
[Ctrl] + [End]	Cursor to the end of the code
[Esc]	Call the system menu or cancel the current function.

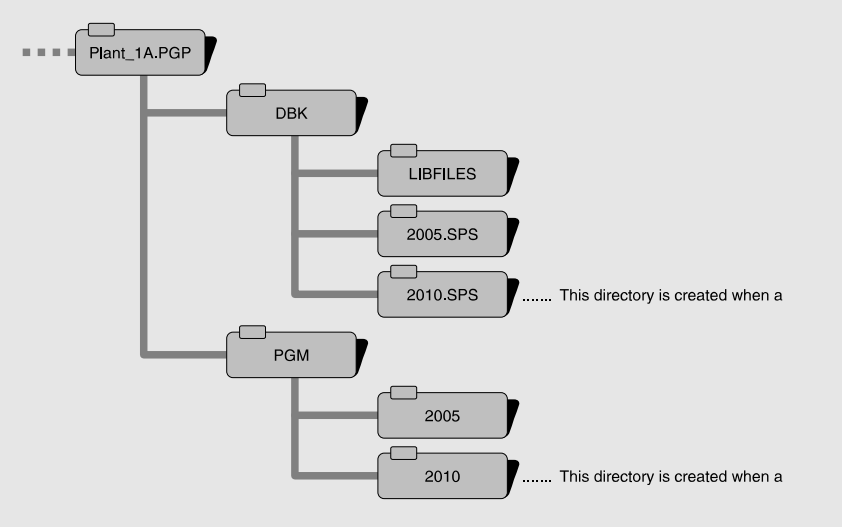
6.18 PUTTING IN A SECOND PLC

To put in a second PLC, you have to be in the PLC layer of the GDM and insert a new PLC symbol. Press [F1 Symbol] and a symbol cursor appears. The symbol cursor can be moved to the desired position with the cursor keys or mouse click.
After confirming the position with [↵], a dialog box opens to enter the PLC names (max. 8 characters). If the symbol cursor is to close to another symbol, the error message "`Positioning here is not possible !`" appears in the message line.



After entering the PLC name, the PLC symbol appears with the given name.
PLC names must be unique, i.e. you cannot use the same PLC name twice within a project !

When you place a PLC symbol, the two directories <PLC-Name>.SPS (in the databank directory) and <PLC-Name> (in the program files directory) are created:



The following message appears in the message:

```
Databank path to PLC "<PLC Name>" created
```

To save the PLC layer of the GDM, use the function Save from the *File* pull down menu or *System* menu.

6.19 FOOT LINE MENU OF THE GDM

The foot line menu of the GDM is the same for all layers (PLC, Processor and Task):

Some function keys have different (or no) functions depending on the active layer and the current cursor position.

[F1 Symbol] Place a new symbol.
A symbol cursor appears at the current cursor position. The symbol cursor can be moved with the mouse and keys to the desired position:



Click on the desired position.



Move the symbol cursor to the desired position in steps with the cursor keys.

After confirming the position with [↵] you have to enter a name and other information (depending on the active layer):

PLC Layer:	PLC Name
Processor Layer:	Processor type and processor name
Task Layer:	Task type, task name and task parameter (for all task, interrupt task and exception task symbols)



If the symbol cursor is too close to another symbol, the new symbol cannot be put here. If this is the case the following error is shown on the message line:

```
Positioning here is not possible !
```

You can also confirm the desired symbol position with [Space] in the task layer. A pop up list appears with all tasks in the current project that are not included in the GDM, but source code exists. After selecting a task, the task symbol is set.
To take full advantage of the GDM, all tasks in the application should be put in the GDM.

[F2 Branch] A pop up menu appears with all the data branch function listed (Draw, Erase, Draw/Erase Off, Process Variables.
A data branch is a symbolic representation of common data points between two or more objects (symbols) on the same level. The common data points of these objects have to be declared with the same scope (tc_global, plc_global).

Draw ..... Turn on drawing mode.



By clicking on any point in the active layer, a data branch (double line) is drawn the current and new cursor position. Only straight horizontal or vertical lines can be drawn.

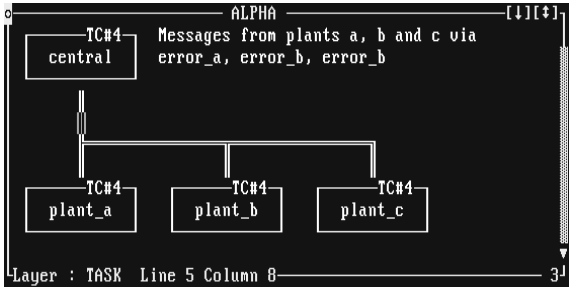


You can draw the data branch lines with the cursor keys.



There must be at least a two character spacing between two symbols to draw a data branch. Data branches must connect at least two symbols together.

Example:



Erase ..... Turn on erase mode.



If the cursor is on a data branch line, a straight section (horizontal or vertical) can be deleted by clicking on it.



When you move the cursor with the cursor keys over a line sections of it are deleted.

Draw/Erase Off ..... The current mode (Drawing or Erasing) is turned off. The cursor can be moved normally again.

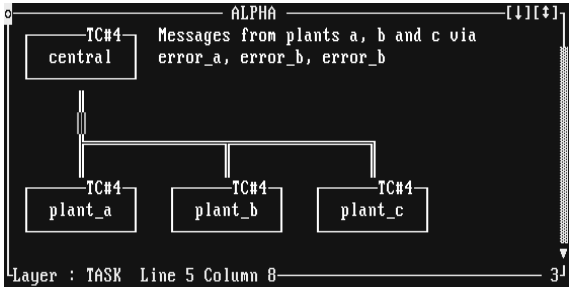
Process Variables .. If the cursor is on a data branch in the task layer, this function shows all common process variables of the tasks connected with this data branch.



The cross reference information from the data branches is entered in the data bank during compilation (see section "6.12.1 Compiler Components"). This means that the *Process Variable* function only provides correct information if all the objects connected by the data branch have been compiled.

The cursor has to be on a data branch between two tasks or between a task and a junction point (intersection of data branches).
You can also call this function with [Ctrl] + [F3] or a double click on the data branch.

Example:



The tasks in the example contain the following variables:

central:	plant_a:	plant_b:	plant_c:
error_a	error_a	error_b	exit
error_b	xchange_ab	xchange_ab	error_c
error_c	xchange_ac	xchange_bc	xchange_ac
	entry		xchange_bc

If the cursor is on the data branch between "central" and the junction point and you call the Process Variable function is called, a dialog box opens. There is a list of all process variables that are also used in "plant_a" and/or "plant_b" and or "plant_c":

error_a
error_b
error_c

The dialog box has two buttons to exit it:

[OK] Save the variable list in an ASCII file (*.VLD file). You can select the name to save it with the file selector box (existing file would be overwritten).
Such a file can be used for the PV Monitor, amongst other things.

[Cancel] Close the dialog box.

[F3 Comment] You can enter a comment in the window on any level of the GDM. The text is entered in a special dialog box. The width of the dialog box is dependent on the amount of free space in the window.
If the cursor is already on a comment field when you call this function, you can edit the existing comment.

[F4 Line+] A line is inserted above the current cursor line. Symbols are moved down and data branches are extended to accommodate the new line.

[F5 Line-] The current cursor line is deleted. This function can only be executed if there is no symbol in the current line.

[F7 Layer↓] If the cursor is on a symbol, you go one layer deeper. If no window exists for this layer, a new window is opened, otherwise the corresponding window is moved to the foreground.

Depending on which layer is active, you switch to the next layer deeper:

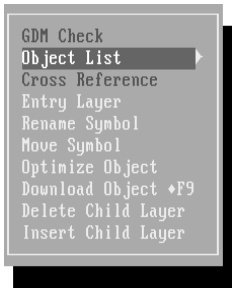
PLC Layer	→	Processor or Task Layer
Processor Layer	→	Task Layer
Task Layer	→	PL2000, LAD, STL, Data Module or Text Editor

[F8 Layer↑] You switch to one layer higher. If there is no open window for this layer, one is opened, otherwise the corresponding window is moved to the foreground.

Depending on which layer is active, you switch to the next layer up:

PLC Layer	→	No function
Processor Layer	→	PLC Layer
Task Layer	→	Processor or PLC Layer

6.20 PLC LAYER POP UP MENU



The pop up menus of the GDM (called with mouse double click or [Ctrl] + [F3]) depend on the current layer and the position of the cursor.
Only the functions are active that are available for the current cursor position. Inactive functions cannot be selected.

GDM Check ..... This function is not active in the PLC layer.

Object List ..... This function calls a pop up menu with various object list functions:

Create ..... Using this function, you can create a list of all tasks used in the GDM (object list). The object is saved as an ASCII file in the program file directory.
You can select or type in the file name with the file selector box (default: TASK.PRJ). The *.PRJ files can be edited with a text editor (*Open* from the *File* menu). If the file is to be used for object list functions like *Compile*, *RUN*, etc. you have to keep the format intact. If only particular tasks are to be compiled when compiling a project (object list function *Compile*), remove the unwanted tasks from the list.

Compile ..... All tasks in the select object list (*.PRJ file) are compiled. The compiler process depends on the compiler settings (see "6.12.2 Compiler Options. You can select the desired object list with the file selector box. The project (all tasks, in the selected object list) is then compiled. The progress of the compilation is shown in the message window.

Download ..... All tasks in the select object list (*.PRJ file) are downloaded into the user RAM of the corresponding PLC. You can only call this function when the corresponding tasks are already compiled. The object list is selected form the file selector box (*.PRJ). After selecting the object, a dialog box appears to ensure that PG2000 is connected to the correct PLC:

Please reconnect to "<PLC Name>"

After selecting [Yes], all tasks in the selected object list are downloaded to the PLC. The progress is shown in the message window.
If there are other tasks in the object list for another PLC, after the downloaded to the first PLC, a dialog box opens for you to change the connection again (only if using the serial interface). If you are using ONLINE programming via a network (PROFIBUS), PG2000 automatically switches to the correct PLC.

RUN ..... All tasks in the selected object list (*.PRJ file) are compiled and downloaded into the user RAM of the corresponding PLC. If there are tasks in the object list that are already on the PLC, theses tasks are deleted and replaced.
You can select the object list from the file selector box (*.PRJ). After selecting the object list, the project (all tasks in the selected object list) is compiled. Next, a dialog box appears to ensure that PG2000 is connected to the correct PLC:

Please reconnect to "<PLC Name>"

After selecting [Yes], all tasks in the selected object list are downloaded to the PLC. The progress is shown in the message window.
If there other tasks in the object list for another PLC, after the downloaded to the first PLC, a dialog box opens for you to change the connection again (only if using the serial interface). If you are using ONLINE programming via a network (PROFIBUS), PG2000 automatically switches to the correct PLC.

Store ..... All tasks in the selected object list (*.PRJ files) are compiled and stored in the nonvolatile memory (user ROM or FIX RAM) as specified in the task parameters (target).
You can select the object list from the file selector box (*.PRJ). After selecting the object list, the project (all tasks in the selected object list) is compiled. Next, a dialog box appears to ensure that PG2000 is connected to the correct PLC:

Please reconnect to "<PLC Name>"

After selecting [Yes], all tasks in the selected object list are downloaded to the PLC. The progress is shown in the message window.
If there are other tasks in the object list for another PLC, after the downloaded to the first PLC, a dialog box opens for you to change the connection again (only if using the serial interface). If you are using ONLINE programming via a network (PROFIBUS), PG2000 automatically switches to the correct PLC.

Print ..... Starts a project print-out.
You can select the object list from the file selector box (*.PRJ). A project printout is printed with the object list according to the documentation configuration (function *Documentation* in pull down menu *Set Up*).
The object list can be manipulated so that only the desired tasks are printed out.

Cross Reference ..... The function is not active on the PLC Layer.

Entry Layer ..... The current GDM Layer is saved as the GDM entry layer. This means that next time you call the GDM for this project, the GDM entry layer is opened.

Rename Symbol ..... You can change the name of a symbol under the cursor.

Move Symbol ..... You can change the position of a symbol under the cursor within this layer.



The symbol cursor can be moved with the mouse click. By clicking the mouse, the new position of the top left corner is set and the symbol is moved.



The symbol cursor can be moved with the cursor keys. Press [↵] to select the position.



If the symbol cursor is to close to another symbol, the symbol cannot be moved. In this case the following error message is displayed on the message line:

```
Positioning here is not possible!
```

Optimize Object ..... All process variables of the selected PLC that are no longer used are deleted from the databank (optimization of the data bank). A confirmation dialog box appears with a warning:

All unused PVs and task entries will be deleted. Optimize object anyway ?

The following order is followed for databank optimization:

- All process variables in the databank of the selected PLC are marked as "deletable".
- Compilation of the individual tasks of the selected PLC. During compilation of the tasks, all process variables used are marked as "reserved" in the databank (not "deletable").
- All process variables of the selected PLC that are still marked as "deletable" are deleted. This means that only the process variables that are actually required are kept (marked as "reserved").

Download Object ..... All tasks of the selected object are downloaded into the user RAM of the corresponding PLC. This function can only be executed is the relevant tasks are already compiled.

Delete Child Layer ..... Delete the child layer (next layer down) of the selected symbol. A confirmation dialog box opens just to make sure:

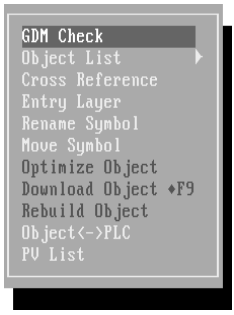
Really delete all information of child GDM of > "PLC Name" < ?



If there is a processor layer and **only one** task layer under that, only the processor layer is deleted (the task layer stays).
If there is a processor layer under the PLC symbol and several task layers under that, the *Delete Child Layer* cannot be executed (interrupted with an error message).

Insert Child Layer ..... If the selected PLC symbol doesn't have a processor layer yet, it is created. An already existing task layer is not deleted, but put one layer deeper.

6.21 TASK LAYER POP UP MENU



The pop up menus of the GDM (called with mouse double click or [Ctrl] + [F3]) depend on the current layer and the position of the cursor.
Only the functions are active that are available for the current cursor position. Inactive functions cannot be selected.

GDM Check ..... The current task layer is checked for completeness and correctness. The following points are monitored:

1. Is there a source file for each task ?
   Error message : NM List for "<Task-Name>" does not exist or file being used
   Solution: Write the source code for the program
2. Are all task parameters defined ?
   Error message: Task "<Task-Name>" not in the data bank
   Solution: Define task parameters
3. Are there empty data branches (no common variables) ?
   Error message: <Task-Name> <-> <Task-Name> : Empty data branch (no common PVs)
   Solution: Delete the corresponding data branch
4. Are there common global variables in tasks that are not connected with a data branch ?
   Error message: <Task-Name> <-> <Task-Name> : Common PVs but no data branch
   Solution: Draw in corresponding data branch

Object List ..... This produces another pop up menu with various object list functions (see PLC Layer Pop Up Menu).

Cross Reference ..... You can check to see if two tasks have common variables. Theses are select from pop up lists of available tasks.



Since the cross reference information is only entered into the databank during compilation, Cross Reference only gives correct information if the tasks are already compiled.

The result is a list of common variable in a dialog box. There are two buttons to exist the dialog box:

[OK] Save the variable list in an ASCII file (*.VLD). You can select or enter in a name in the file selector box.
This file can be used for things like the PV Monitor to see the values of these variables.

[Cancel] Close the dialog box.

Entry Layer ..... The current GDM Layer is saved as the GDM entry layer. This means that next time you call the GDM for this project, the GDM entry layer is opened.

Rename Symbol ..... You can change the name of a symbol under the cursor.

Move Symbol ..... You can change the position of a symbol under the cursor within this layer.



The symbol cursor can be moved with the mouse click. By clicking the mouse, the new position of the top left corner is set and the symbol is moved.



The symbol cursor can be moved with the cursor keys. Press [↵] to select the position.



If the symbol cursor is to close to another symbol, the symbol cannot be moved. In this case the following error message is displayed on the message line:

```
Positioning here is not possible!
```

Optimize Object ..... This function is inactive in the task layer.

Download Object ..... All tasks of the selected object are downloaded into the user RAM of the corresponding PLC. This function can only be executed is the relevant tasks are already compiled.

Object Rebuild ..... If the cursor is on an executable program symbol (OBJ symbol), this can be "decompiled" with the *Object Rebuild* function. The rebuild restores the complete source information (source code, task parameters, variable declaration, etc.) of the program module.



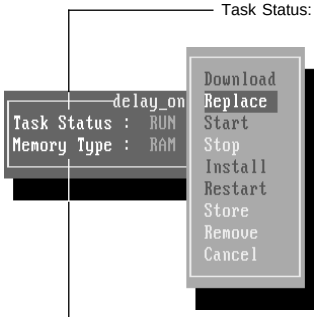
Only program modules (B&R modules) that were compiled with the compiler option "BR Info" turned on can be decompiled (see section "6.12.2 Compiler Options").

The source code of the original source type (LAD, STL, PL2000) is restored with a rebuild and the task in entered in the project data bank. The OBJ symbol is automatically replaced with the corresponding task symbol (TC#1-TC#4, HS#1-HS4, IRQ or EXC).



Program modules (B&R modules) that are only available on the PLC can be fetched from the PLC into PG2000 and entered in the GDM (task layer) with the *GDM Synch.* function (*Project* pull down menu) before doing an *Object Rebuild*.

Object<->PLC ..... The status of a task on the PLC and the memory type of the task are displayed. There is also an additional pop up menu (you need a correct connection between the PLC and PG2000):



NON ..... Task is not on the PLC.
RUN ..... Task running (task active)
STP ..... Task stopped (task inactive)
RDY ..... Task is on the PLC but can only be manually installed with the *Install* function (task inactive)
IDL ..... Task is installed but can only be manually started with the *Restart* function (task inactive)

NON ..... Task is not on the PLC
RAM ..... User RAM (also FIX-RAM)
EPR ..... User ROM

Download ..... The selected task is downloaded to the PLC (user RAM) according to the task parameters. This function can only be executed when the task is already compiled.

If the task is already on the PLC, this function is inactive.

Replace ..... The selected task is deleted from the PLC and downloaded again. This function is used, for example, if a program module on the PLC should be replaced with a new version.

Start ..... A stopped task (task status "STP") can be started.

Stop ..... A running (active) task (task status "RUN") can be stopped.

Install ..... A task downloaded with the option "Install: manual" (task status "RDY") can be manually installed with this function and started with the *Restart* function.

Restart ..... A task that has to be manually started (task status "IDL") can be activated with this function.
This is either task that are downloaded with the option "Start: manual" or manually installed tasks (see Task Parameters).

Store ..... The selected task is stored in the nonvolatile (set with the task parameter "memory target") memory (user ROM or FIX RAM). This function can only be executed if the selected task is already compiled.

Remove ..... The selected task is deleted from the PLC.
A task that is in the EPROM (user ROM) can only be logically deleted , i.e. the task is marked as invalid and is therefore no longer activated. The ROM memory used remains unavailable until the EPROM is physically erased (UV light).

Cancel ..... Close the pop up menu without a change.

PV List ..... All variables of the selected task are shown in a dialog box. The two buttons on the dialog box are:

[OK] Save the variable list in an ASCII file (*.VLD). You can select or enter the name with the file selector box.
This file can be used for things like the PV Monitor to show the values for these variables.

[Cancel] Close the dialog box.

6.22 GDM KEYBOARD COMMANDS

The following keys and key combinations can be used on all GDM layers:

Key	Description
[⇑]	Cursor up a line or a symbol.
[⇓]	Cursor down a line or a symbol.
[←]	Cursor left one column or symbol.
[→]	Cursor right one column or symbol.
[Tab]	Select the next symbol. Order: left → right; top → bottom
[⇑] + [Tab]	The previous symbol is selected. Order: right → left; bottom → top
[PgUp]	Page Up
[PgDn]	Page Down
[Home]	Cursor to the start of the line.
[End]	Cursor to the end of the line.
[Ctrl] + [Home]	Cursor to the top of the GDM
[Ctrl] + [End]	Cursor to the bottom of the GDM
[Del]	Delete the selected symbol or comment field.
[Ins]	Place a new symbol.
[⇑] + [Home]	Mark the line from the current cursor position to the start of the list.
[⇑] + [End]	Mark the line from the current cursor position to the end of the line.
[⇑] + [Cursor key]	Mark an area.
[Ctrl] + [Ins]	Copy the marked area into the clipboard. The area is not deleted from the GDM.
[⇑] + [Ins]	Insert the clipboard contents at the current cursor position.
[⇑] + [Del]	Move the marked area to the clipboard. The area is deleted from the GDM.
[Ctrl] + [F3]	Call the symbol sensitive pop up menu.
[Esc]	Call the system menu or cancel the current function.

6.23 SYNCHRONIZING THE PLC AND GDM

Using the GDM Synch. function (Project pull down menu), all B&R modules (e.g. compiled tasks, function library runtime modules, exception handler, etc.) in the user ROM, user RAM or FIX RAM of the **current PLC** [1] but not in the GDM, are automatically entered in the appropriate layer of the GDM and shown as symbols.

This function is generally used to upload projects that are only stored on the PLC back into the GDM. You can then archive the project in the project management (*Copy* from the pull down menu *Project*).

If the modules uploaded with GDM Synch. were compiled with Rebuild Info (see "6.12.2 Compiler Options"), you can "decompile" these tasks. This means that using the function Task Rebuild or Object Rebuild, the complete source information (source code, INIT-SP, task parameters, variable declaration) of the program module can be restored.

[1] The current PLC name is shown on the title line of the programming environment.
Title Line Information:
- Program Name and version number: “B&R Programming System Vx.xx”
- Project, PLC Name and possible Co-Processor Name in the following form: “<Project Name>.<PLC-Name>.<Processor-Name>”
- Current time

7 FUNCTIONS

7.1 STANDARD FUNCTIONS

7.1.1 General Information

Functions are program parts, that - once written - can be used in again and again in any application program (LAD, statement list or PL2000). They function like subroutines in other programming languages. You can treat a function like a block box. They ask for information at the input end, perform particular operations with this information and provide a result at the output. You don't have to know or understand the program code behind the function, only the input information and what to do with the result.
The PG2000 software includes a large number of standard function that are put together in so-called system libraries. An exact description of the function libraries is in the "B&R SYSTEM 2000 Library Reference Manual".



Before you can use library functions in your project, you have to import the library into the corresponding project data bank. If you use a function and don't import the corresponding library, the compiler doesn't recognize the function and returns an error message.

There are two types of functions (function blocks and inline functions):

Function Blocks ..... A function block (FBK) is a function written in a PLC programming language that returns one or more values. The PG2000 software contains a large number of function blocks (function libraries). You can also create function blocks yourself (see "7.2 Application Specific Function Blocks").
You can find detailed information on calling and using function blocks in different programming languages in the "B&R SYSTEM 2000 Programming Languages Manual".

Inline Functions ..... An inline function (INLINE) only returns a data element (can be an array or structure). The result of an inline function can be used as an operand in an operation. The PG2000 software contains a large number of inline functions (function libraries). You cannot create inline functions yourself.
You can find detailed information on calling and using inline functions in different programming languages in the "B&R SYSTEM 2000 Programming Languages Manual".

7.1.2 Importing Function Libraries

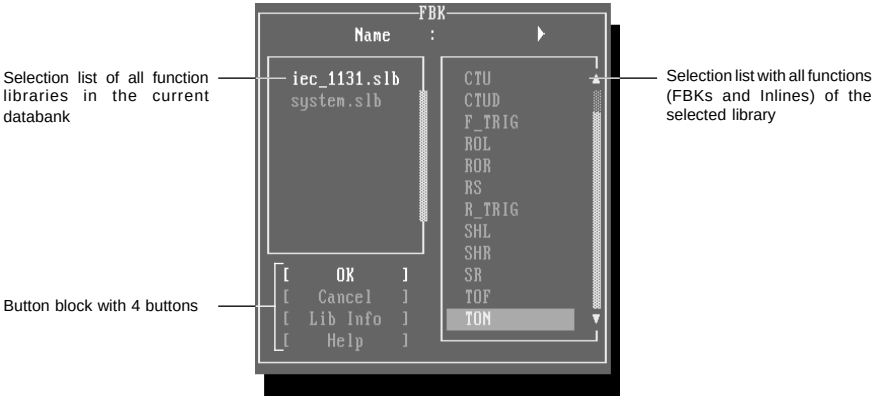
Before you can use function blocks or inline functions from a library in your project, you have to import the library into the corresponding project databank. Use the *Library* function (pop up menu of the *Import* function in the *File* menu) to do this. After calling this function, the file selector box appears for you (path PG2000\LIBRARY\) to select the required library (*.SLB) file.
After selecting the desired library, it is imported into the appropriate project data bank. The corresponding *.SLI file is put in the functions directory of the current project.



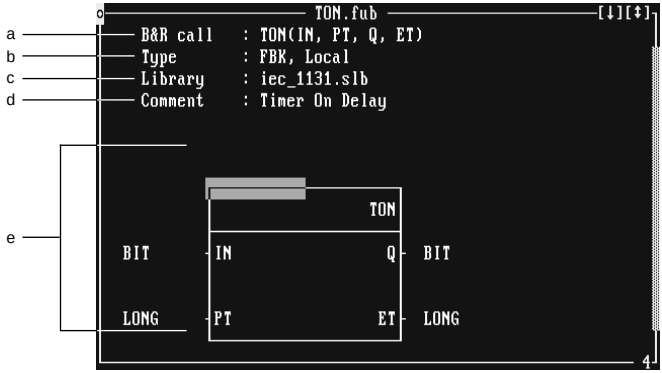
The System library is automatically imported into the data bank when you create a project, i.e. you can use the functions of the system library immediately after creating a project.

7.1.3 Help on the Function Libraries

You can find detailed information on calling and using inline functions in different programming languages in the "B&R SYSTEM 2000 Programming Languages Manual". In addition to this you can get information on each library and its functions via the PG2000 help system. Select *Functions* from the *Help* pull down menu. This opens a dialog box with a selection list of all libraries in the current databank:



[OK] ..... All information about the selected function (FBK or Inline) is shown in a window:



a) Function call in PL2000
b) Function Type: FBK, Local ..... Local function block [1)]
FBK, AVT ..... AVT function block [1)]
FBK, Inline ..... Inline function
c) Function Library
d) Short description of the function
e) Graphical representation of the function with the inputs (left) and outputs (right)

[Cancel] ..... The dialog box is closed.

[Lib Info] ..... A dialog box with information about the selected function library appears.

[Help] ..... The help file for the selected function library is shown.

[1)] The difference between local function blocks and AVT function blocks is explained in chapter 5 "The Real World" of the "B&R SYSTEM 2000 Programming Languages Manual".

7.1.4 Deleting a Function Library

To delete a function library from the databank of the current project, use the Delete Lib (pop up menu from Databank, Project menu). You can select the library to be deleted from a pop up list of currently used function libraries. After choosing a library, it is deleted from the databank of the current project/ There is a confirmation dialog box to check you selection.

7.2 APPLICATION SPECIFIC FUNCTION BLOCKS

7.2.1 General Information

If a program section is going to be used several times, you can create an application specific function block (also known custom or user FBK) with PG2000 that contains this section of code. To do this you write the program for the function block in one of the three programming languages (LAD, STL or PL2000) and define the inputs and outputs with the function block editor.
When creating your own function blocks, there are a few things to keep in mind:

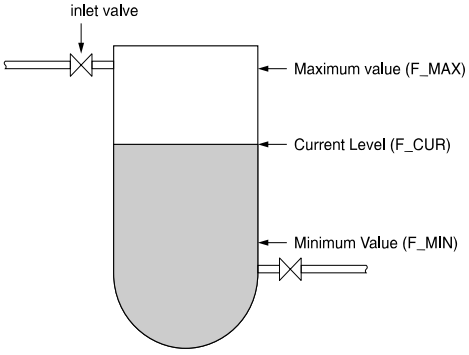
- You can use inline functions and other function blocks in your source code.
- Function blocks cannot be recursive, i.e. a function block cannot call itself. Also function block A cannot call function block B whitch calls function block A, etc.

7.2.2 Creating An Application Specific Function Block

We'll use a simple example to go through how to create an application specific function block:

Example Description

The level of fluid in a tank has to be kept between to limits (minimum < current < maximum).



If the level in the tank exceeds the maximum F_MAX, the inlet value is closed (I_Valve = 0). If the level in the tank goes below the lower limit F_MIN, the inlet valve opens (I_Valve = 1).

Create the Source Code for the User FBK

When you create an application specific function block, you have to write the FBK source code in one of the three programming languages (Ladder Diagram, Statement List, PL2000). Call the function *Function Block* (pop up menu of *New* from the *File* menu). Select the desired programming language from the pop up menu to write the source code in. Select PL2000 for this example and a PL2000 editor window opens for you to enter the source code:

```
; Source code for the user FBK "TANK"
; Simple tank level control

if F_CUR < F_MIN then           ; level < minimum ?
   I_Valve = 1                  ; Yes -> open valve
else if F_CUR > F_MAX then      ; level > maximum ?
   I_Valve = 0                  ; Yes -> close valve
endif
```

You can find a detailed description of the PL2000 editor in "6.9 PL2000".

Saving the FBK Source Code

The FBK source file has the name FUNCT_xx.FUB by default, where "xx" is the next number in sequence. To save the FBK source file as "TANK.FUB", use *Save As* function from the *File* menu. This opens a dialog box for you to enter the name (without file extension).
The name of the source file also sets the name with which you can call the FBK later.

Variable Declaration

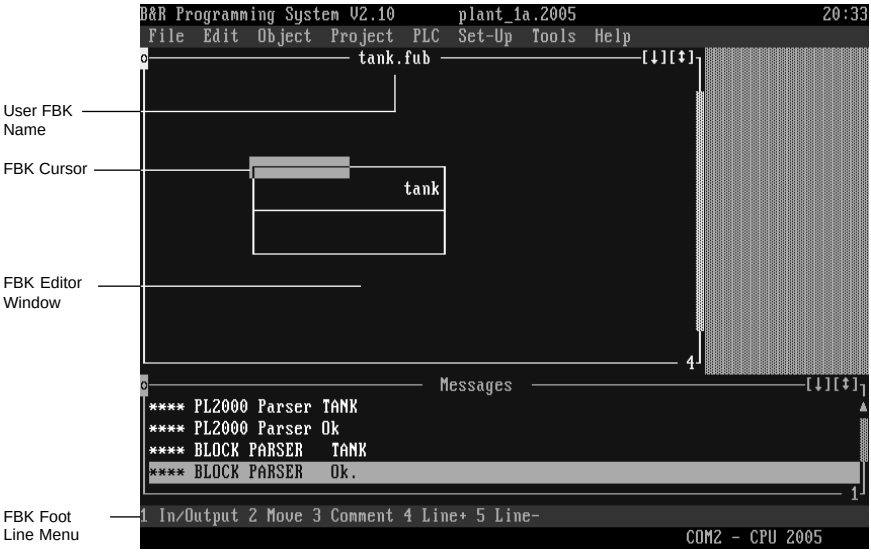
If there are still variables that are not declared when the file is saved, a dialog box warning is given. If you close this dialog box with [OK], the variable declaration is automatically called. Enter the following in for our example:

Name	Scope	Data Type	Length	I/O Type
F_CUR	local	WORD	1	INTERNAL
F_MAX	local	WORD	1	INTERNAL
F_MIN	local	WORD	1	INTERNAL
I_Valve	local	BIT	1	INTERNAL

Please note that for the variables of a FBK source code, only **local** and **dynamic** can be used for scope !
For detailed information on the variable declaration, please refer to "6.10 Variable Declaration".

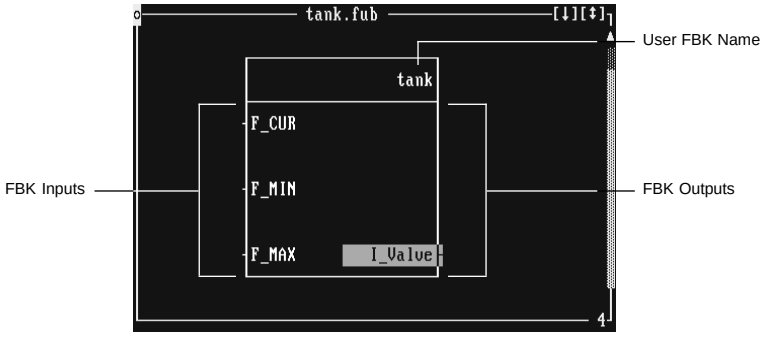
Calling the Function Block Editor

If the source code of a user FBK is open in the editor, you can define the inputs and outputs of the FBK with the function *Parameter* from the *Object* pull down menu:



Foot line menu of the FBK Editor

[F1 In/Output] An input or output can be set at the current cursor position. The variables that haven't been used yet as inputs or outputs are listed in a pop up menu.
If you cannot put an input or output at the current cursor position, an error message appears.



[F2 Move] You can move the input or output under the cursor to somewhere else.



By clicking on [F2 Move], move mode is activated. After you click on another position the selected input or output is moved. If there is already an input or output at the new position, the FBK is expanded and the input/output is inserted.



By pressing [F2 Move], move mode is activated. After you move the cursor to another position and press [↵], the selected input or output is moved. If there is already an input or output at the new position, the FBK is expanded and the input/output is inserted.

[F3 Comment] You can enter a comment of up to 60 characters.
The comment is a short description of the function and is saved with the user FBK.

[F4 Line+] A new line is inserted at the current cursor position. If there is already an input /output here, it is moved down.

[F5 Line-] The current cursor line is deleted. Input/Outputs are also deleted without a warning.

FBK Editor Keyboard Commands

The following keys and combinations can be used in the FBK editor:

Key	Description
[↑]	Cursor up.
[↓]	Cursor down.
[←]	Cursor one column left (only within the FBK).
[→]	Cursor one column right (only within the FBK).
[PgUp]	Cursor, one page up.
[PgDn]	Cursor, one page down.
[Ins]	An input or output is placed at the current cursor position. The variables that haven't been used in the FBK yet are shown in a pop up menu.
[Del]	Delete the selected input or output
[Esc]	Call the system menu or cancel the current function.

Save the FBK Mask

After defining the FBK mask (inputs and outputs of the user FBK), you have to save it. Use Save from either the *File* or *System* menu. The FBK mask is saved in the source file with the FBK source code in the program files directory of the current project.

Compiling the User FBK

To include the user function block in the data bank of the current project, call the function *Compile* (*Object* menu) from within the FBK editor.



You can only use the user function block like a standard function in LAD, STL and PL2000 programs AFTER it is compiled !

The compiled user FBK (*.BFU file) is saved in the function directory of the current project.

8 PLC UTILITIES

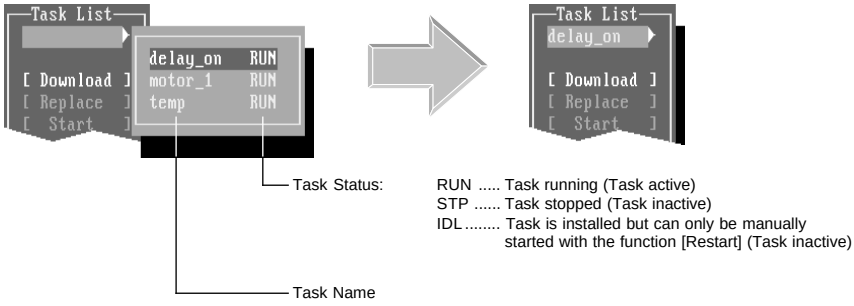
All the functions that can be called via the *PLC* pull down menu are described in this section. An active connection between the PLC and PG2000 is necessary for most of these functions.

8.1 TASK MANAGEMENT UTILITIES

After selecting Tasks from the PLC menu, a dialog box appears with various functions:

If the input field of the dialog box is selected, you can open a pop up menu ([Space] or [Ctrl] + [F3]) of all tasks installed (by the user) on the PLC. Tasks that were downloaded with the option "Install: manual" (see task parameters) and have not been manually installed yet, are not shown in this list.

After selecting a task, it is shown in the input field:



The various menu buttons perform their operations on the task in the input box:

Download You can select a B&R module from the file selector box. The selected task is downloaded to the PLC in the memory specified with the *DL Target* function (*PLC* menu, *Other* function). This function can only be executed when the task is already compiled. If the download is successful, the following message appears:

```
Download <Task-Name>
Download complete (M=$070F, T=$0581)
```



The task to be downloaded must not already be on the PLC !

Replace The selected task is deleted from the PLC and downloaded again. This function is used, for example, if a program module on the PLC should be replaced with a new version. The corresponding message is output.

Start A stopped task (task status "STP") can be started. The corresponding message is output.

Stop A running (active) task (task status "RUN") can be stopped. The corresponding message is output.

Restart A task that has to be manually started (task status "IDL") can be activated with this function. This is either task that are downloaded with the option "Start: manual" or manually installed tasks (see Task Parameters).
The corresponding message is output.

Remove The selected task is deleted from the PLC. The corresponding message is output.
A task that is in the **EPROM** (user ROM) can only be logically deleted , i.e. the task is marked as invalid and is therefore no longer activated. The ROM memory used remains unavailable until the EPROM is physically erased (UV light).

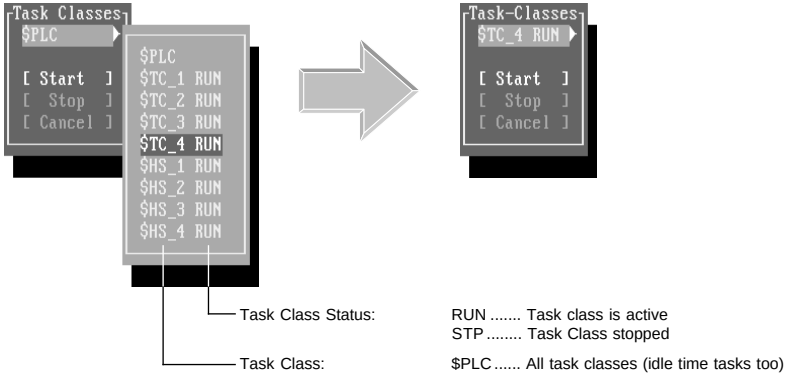
Cancel Close the pop up menu without a change.

8.2 TASK CLASS UTILITIES

After selecting the function Task Classes from the PLC menu, a dialog box appears with various functions.

You can either start or stop individual task classes of the entire PLC. The task class can be selected in the input box via a pop up menu.

After selecting a task class, it is shown in the input box:



Start Processing of all tasks in the selected task class is continued. The corresponding message is output in the message window.



Tasks that were individually stopped (PLC, task, stop) before the start, stay stopped even when the task class is started.

Stop Processing of all tasks in the selected task class are stopped. The corresponding message is output in the message window.

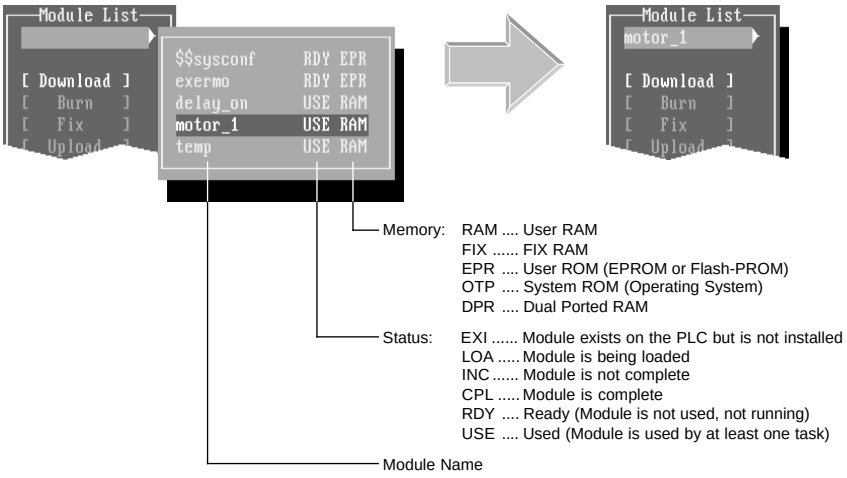
Cancel The dialog box is closed without manipulating the task classes.

8.3 MODULE - DISPLAY AND MANIPULATION

After selecting the function *Module* from the *PLC* menu, a dialog box appears with various functions.

All modules on the PLC can be listed and manipulated. You select the desired module from a pop up menu.

After selecting the module it is shown in the input box of the dialog box:



Download

You can select a B&R module (e.g. compiled tasks, library runtime module, exception handler, etc.) from the file selector box. The selected module is downloaded to the PLC in the memory specified with the *DL Target* function (*PLC* menu, *Other* function). This function can only be executed when the task is already compiled. If the download is successful, the following message appears:

```
Download <Module Name>
Download complete (M=$070F, T=$0580)
```



The module to be downloaded must not already be on the PLC !

Burn

B&R Modules (compiled tasks, library runtime module, exception handler, etc.), that are in the user RAM of the PLC can be programmed (burned) into the user ROM (EPROM or Flash-PROM).



Before you burn a module into the user ROM with this function, the PLC (all tasks) has to be stopped (*Stop* from dialog box from *Task Class* function of *PLC* menu).

After calling the *Burn* function, a dialog box appears:

Is PLC <PLC NAME> stopped ?

[Yes] ..... The selected module is burned into the user ROM. The appropriate message appears in the message window.

[No] ..... The process is terminated.

Fix

B&R Modules (compiled tasks, library runtime module, exception handler, etc.), that are in the user RAM of the PLC can be programmed (burned) into the FIX RAM. The appropriate message appears in the message window.
Modules saved in the FIX RAM are not deleted by a TOTAL INIT. The FIX RAM can be configured with PG2000 or the PLC Configurator.



To delete all module saved in the FIX RAM, use the function *Del FIX RAM* (pop up menu from *Other* of the *PLC* menu).

Upload

The module selected from the pop up menu is uploaded from the PLC to PG2000 and put in the corresponding PLC directory (directory "..\project-name.PGP\PGM\<PLC-Name>) as a *.BR file. If there is already a module in this directory with the same name, it is overwritten. The following message appears in the message window if the upload was successful:

```
Upload <Module name> in progress
Upload complete
```

If the modules (B&R modules) were compiled with the compiler option "BR Info" turned on (see section "6.12.2 Compiler Options"), they can be decompiled with *Task Rebuild* or *Object Rebuild*. The source code of the original source type (LAD, STL, PL2000) is restored with a rebuild and the task in entered in the project data bank.

Install

Tasks compiled with the option "Install: manual" (see task parameters) and downloaded to the PLC (marked as "RDY") can be manually installed on the PLC with this function. They have to be started with *Restart* (*PLC* menu, *Tasks* function) to be activated. If this is successful, the following message appears in the message window:

```
Task created, index = $0580
```

Remove

The selected module is deleted from the PLC. If the module was successfully deleted, the following message appears:

```
PLC Task Action Ok.
Module deleted.
```

A module that is in the **EPROM** (user ROM) can only be logically deleted , i.e. the task is marked as invalid and is therefore no longer activated. The ROM memory used remains unavailable until the EPROM is physically erased (UV light).

Info	A dialog box with information about the selected module appears (size, version number, creation date, etc.).
Cancel	The dialog box is closed. No function is executed.

8.4 ERROR ANALYSIS

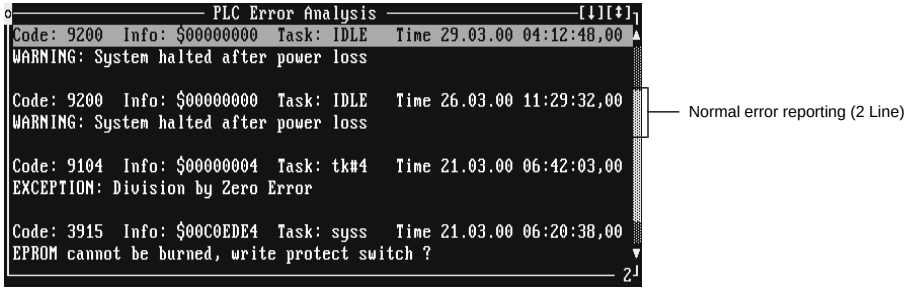
All error that occur within the scope of an application (e.g. cycle time violations, bus errors, errors for incorrect usage), are entered in the error module by the operating system with the time and some additional information. The maximum number of entries in the error module is set with the PLC configurator.

You can read the entries in the error module with this function. Calling this function opens a small dialog box for you to enter what you want to read from the error module:

Entry ..... Start entry to read. Enter "0" to start at the most recent entry.

Number .... The number of entries to read from the error module.

After confirming with [OK], a window with the desired error module entries is opened:



Normal Error Reporting:

Normal (2 line) error reporting is performed by default for all errors and exceptions that occur in the scope of an application:

Line 1: **`Code: 3915 Info: 00C12B3C Task: syss Time 23.01.95 08:14:12,21`**

Code: PLC error number
Info: Error detail
Task: Task class of the task that caused the error
Time: Time of the error

Line 2: **`EPROM cannot be burned, write protection switch ?`**
This is a description of the error.

If the module "EXERMO.BR" (Exception handler) in the user ROM for all exceptions that have no exception task assigned to them, an extended (3 line) error reporting is done.

Extended Error Reporting



Extended error reporting is only not possible if the Exception handler (module "EXERMO.BR") in the user ROM.

For all exceptions that have no exception task assigned to them, an extended (3 line) error reporting [1] is done:

Line 1: **`Code: 9104  Info: 00000005  Task: tc#1   Time 23.01.95 08:26:00,70`**

Code: PLC error number
Info: Error detail
Task: Task class of the task that caused the error
Time: Time of the error

Line 2: **`Module:  measure1  Line:  8`**

Module: Program module that caused the exception
Line: Program line that caused the problem

Line 3: **`EXCEPTION: Divide by Zero`**

This is a description of the error.

You can find detailed information on "Exception Handling" in chapter 2 "Operating System PLCSW".

[1] This form of error reporting is in version 1.12 of EXERMO.BR and later.

8.5 OTHER FUNCTIONS

After selecting Other from the PLC menu a pop up menu appears with the following functions:

Reset PLC You can reset the PLC in any reset mode. A dialog box opens for you to enter the password and choose a boot mode.



You have to have an active connection between the PLC and PG2000 to use the *Reset PLC* function.

Password ... You have to have the correct password to be able to reset the PLC. The default password "bur". If it is changed with the PLC configurator, you have to type in the new password in this text input field and confirm it with [Passwd].

Mode ..... When this field is selected, you can use the pop up menu to select a boot mode. After selecting a boot mode and pressing the [OK] button, the corresponding boot mode is triggered.

Init:
With an INIT, all battery backed RAM data (all tasks in the User RAM, values of the inputs, outputs and internals, etc.) is kept. All task also keep their status (task active or inactive) they had before the INIT, after an INIT.
Newly plugged in system modules (Remote Master, Network Modules, etc.) are not recognized with an INIT. A TOTALINIT must be done to initialize system modules.
After an INIT, all tasks that were active before the INIT are started. This includes running the corresponding INIT-SPs (also the INIT-SPs of inactive tasks).
You would use an INIT, for example, to start an application again after a fatal error occurred.
You can find more information about the boot mode INIT in chapter 2 "Operating System PLCSW", section "Boot Modes".

Total-Init:
The TOTALINIT boot mode corresponds to the FIRST initialization of the CPU. The system is completely restarted, allocated memory is released and the system structures (all operating system information for orderly functioning of the CPU) is recreated. With a TOTALINIT, all system modules (Remote Master, Network Modules, etc.) are recognized and initialized. All the modules saved in the User RAM are deleted (except for those in FIX RAM) and all process variables (inputs, outputs and internals) are initialized with zero.
After a TOTALINIT, all tasks in the User ROM or FIX RAM are started (if task parameters *Install* and *Start* are "auto"). This also includes running all attached Initialization sub-programs (INIT-SPs). You can find more information about the boot mode INIT in chapter 2 "Operating System PLCSW", section "Boot Modes".

Reset:
With an RESET (similar to INIT), all battery backed RAM data (all tasks in the User RAM, values of the inputs, outputs and internals, etc.) is kept. All task also keep their status (task active or inactive) they had before the RESET, after an RESET. After a reset the PLC goes into SERVICE mode, i.e. application programs in the User ROM, User RAM and FIX RAM are not started. Only the PLCSW runs in SERVICE mode, all application programs are inactive. You can find more information about the boot mode INIT in chapter 2 "Operating System PLCSW", section "Boot Modes".

Diagnose:
In boot mode DIAGNOSE, the PLC only boots up with the operating system software, i.e. all application programs are ignored and are therefore inactive. The PLC goes into SERVICE mode after the DIAGNOSE boot up.
Since all modules in User ROM and FIX RAM are installed with a TOTALINIT, a defect module in one of these memory areas can prevent a correct boot-up in TOTALINIT mode. In order to avoid having to delete the entire User ROM and/or FIX RAM, you can boot the PLC in DIAGNOSE mode. After you have done this, you can use the PLC Configurator function *Diagnose* to display existing B&R modules and **delete them singly** if necessary. You can find more information about the boot mode INIT in chapter 2 "Operating System PLCSW", section "Boot Modes".



Hot Key:
You can switch the PLC into SERVICE Mode at any time with [Ctrl] + [Break] (when communication is active). In SERVICE mode, only the operating system is running, all application programs (cyclic and non-cyclic) are inactive. If the PLC is in SERVICE mode, "!Service!" is on the message line.
You have to have entered the correct password for the hot key function to work. The default password "bur". If it is changed with the PLC configurator, you have to type in the new password in this text input field and confirm it with [Passwd].
The hot key function is not a replacement for a EMERGENCY STOP key, since an active connection cannot be guaranteed.

Read PLC Clock Read the real time clock. The date and time are shown in a window.

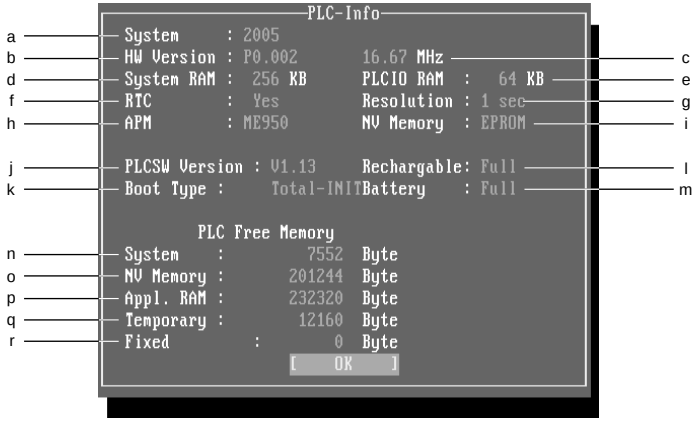
Set PLC Clock You can set the date and time of the PLC real time clock. The current values are shown as default.

Init.RIO To start a remote I/O system, you have to perform a first initialization with Init RIO. The module address of the remote master is entered in a dialog box. After pressing [OK], the slaves are initialized.
If the initialization worked OK, the required operation data is sent to the slave stations and the appropriate message appears in the message window.



From version 3.0 of Remote I/O, you can use a remote system without doing a first initialization, i.e. the *Init RIO* function has no function !

PLC Info Information about the versions and memory of the current PLC is displayed:



a) PLC System (B&R 2010 or B&R 2005)
b) CPU Hardware Revision
c) Clock Frequency
d) Size of the CPU system RAM
e) Size of the CPU Dual Ported RAM (DPR)
f) Real Time Clock (Yes/No)
g) Real time clock resolution
h) Model of Application Program Memory
i) Type of non volatile memory (EPROM or Flash PROM)
j) PLC SW Operating System Version
k) (Boot Mode)
l) Status of the NiCd rechargeable (Full/Dead)
m) Status of the Lithium Battery in APM (Full/Dead)
n) Free memory in System RAM (for operating system data and stacks for application programs)
o) Free memory in User ROM (EPROM or Flash-PROM)
p) Free memory in the User RAM
q) Free temporary area of the system RAM
r) Free memory in FIX RAM

Parameters This function downloads the PV initialization values to the PLC. You can select the desired variable list from the file selector box (*.VLD) that contains the initialization values.



The variable list can only have task class global and PLC global variables. All other variables are ignored !

A variable list can be created in different ways:

1) Since the variable list is an ASCII file, it can be created with the text editor. The file looks like this:

```
Array[0]           0    500
Structure.Element  1    $100F
Variable1          -1   3.15
Variable2          3    %1001
```

The initialization value of the variables can be various formats:
- 500 ..... decimal value
- $100F ..... hexadecimal value (indicated with a "$")
- 3.15 ..... floating point value (FLOAT)
- %1001 .... binary value (indicated with a "%")

Variable Scope:
- -1 ..... local or dynamic (Variable is ignored)
- 0 ..... PLC global
- 1 ..... Task class global (Task class 1 and high speed task classes)
- 2 ..... Task class global (Task class 2)
- 3 ..... Task class global (Task class 3)
- 4 ..... Task class global (Task class 4)

Variable name

2) Creating and saving the variable list in PV Monitor or in the task layer of the GDM (PV List function). The initialization values have to be inserted in the file with the text editor.

Force Off You can turn off forcing on **all variables on the PLC** that were forced with *Force* to a defined value. After this function is successfully completed, the values of these variables can change again.

Delete FIX RAM You can delete all modules that are in the FIX RAM or FLASH PROM of the current PLC.



This function can only be executed after booting up in DIAGNOSE mode !

RAM ..... You can delete the entire contents of the FIX RAM. After this you have to perform a TOTAL INIT.

FLASH ..... You can delete the entire contents of the Flash PROM (User ROM). After this you have to perform a TOTAL INIT.

DL Target This function determines which memory area of the PLC (user RAM, User ROM or FIX RAM) you download a task or module to with the *Download* function (see dialog box of the function *Task* or *Module* from the *PLC* menu) [1].



This setting is only kept until you exist PG2000.

RAM ..... The selected task or module is downloaded to the **User RAM** when you use the *Download* function (see dialog box of the function *Task* or *Module* from the *PLC* menu).

ROM ..... The selected task or module is downloaded to the **User ROM** when you use the *Download* function (see dialog box of the function *Task* or *Module* from the *PLC* menu).

FIX ..... The selected task or module is downloaded to the **FIX RAM** when you use the *Download* function (see dialog box of the function *Task* or *Module* from the *PLC* menu).

8.6 MODEM

After selection the Modem function from the PLC menu, a pop up menu appears with the following functions:

Dial You can dial up a modem connection.

Hang Up You can hang up and disconnect the from the modem.

Command You can perform a modem specific command.

Save Save the modem settings.

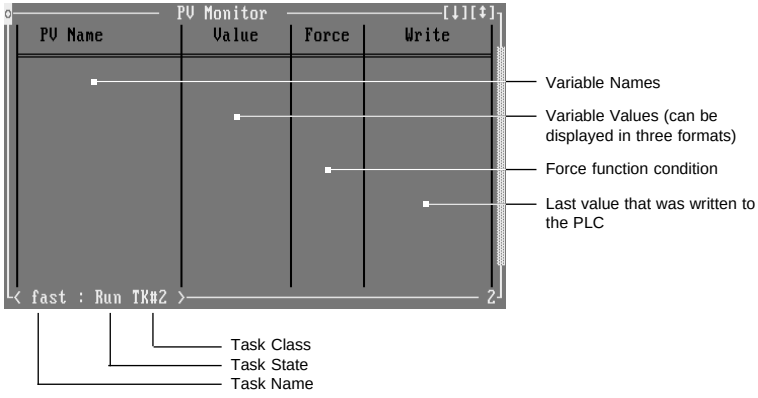
[1] Downloading directly into the User ROM or FIX RAM with *Download* is only possible with V1.10 of the PLCSW Operating system and later. With older versions of PLCSW, tasks or modules can only be downloaded into the User RAM.

8.7 PV MONITOR

You can look at and write to process variables of a task running on the PLC with the PV Monitor. The variable values are updated at the sampling rate (this can be set with the function *PLC Refresh Rate* from the pull down menu *Set Up*).
To use the PV Monitor on a task running on the PLC, select the *PV Monitor* function from the *Object* menu. This function can only be called from the task layer of the GDM. The cursor must be on the corresponding task symbol when you call this function.
Requirements for trouble free operation of the PV Monitor:

1) Correct (active) connection between the PLC and PG2000.
2) Task is defined in the current project data bank.
3) Task was compiled and downloaded to the PLC error free.

After selecting the function PV Monitor, a pop up menu appears with a list of all tasks installed (by the user) on the PLC. Select a task from this list to use in the PV Monitor. If all the above requirements are met, a PV Monitor window opens:



APPENDIX A
PLCSW ERROR NUMBERS

0000 - 1999 GENERAL ERRORS

0001 SYSTEM: Time Out with Service handling
0005 SYSTEM: Object no longer exists
0006 SYSTEM: Invalid ObjectID
0008 SYSTEM: No more objects can be created (PLCSWCFG)
0015 SYSTEM: Not enough Stack area -> PLCSWCFG
0016 SYSTEM: Stack is too Small
0020 SYSTEM: Task is already suspended
0021 SYSTEM: Task is already running
0032 SYSTEM: Attempt to allocate no memory
0033 SYSTEM: Too much memory allocated at once
0034 SYSTEM: Too much memory allocated at once
0035 SYSTEM: Memory classes do not match
0036 SYSTEM: The memory returned doesn't correspond to the allocation
0051 SYSTEM: No enough queues configured -> PLCSWCFG
0052 SYSTEM: Too little message buffer is available
0072 WARNING: The RTC or system time is not set
0073 WARNING: The RTC or system time is not set
0257 SYSTEM: Incorrect device number
0258 SYSTEM: No driver available
0259 SYSTEM: Driver function is not supported

2000 - 2999 HARDWARE TRAP LIBRARY ERRORS

2000 Incorrect access to hardware component
2001 Incorrect access to hardware component
2010 Incorrect access to hardware component
2011 Incorrect access to hardware component
2020 Incorrect access to hardware component
2021 Incorrect time at timer activation
2022 Timer does not support interrupt
2023 Timer does not support this time
2030 Incorrect access to hardware component (CRM)
2031 Incorrect offset for access to CRM
2032 Incorrect length for access to CRM
2040 Incorrect access to hardware component (IRC)
2041 Mode is not supported by IRC
2042 Incorrect interrupt number transmitted
2043 Incorrect call by IRC-Routine
2050 Error during watchdog activation
2060 Incorrect access to hardware component (DIS)
2061 Incorrect line number for display (2010)
2062 Incorrect column number for display (2010)
2063 Invalid character number for display (2010)
2064 Incorrect access to display (2010)
2070 Incorrect access to hardware component (RTC)
2071 Incorrect time set to RTC
2072 RTC does not support Hold function
2073 RTC is not initialised yet
2080 Incorrect access to hardware component
2090 Incorrect access to hardware component (DMA)
2091 Incorrect access to hardware component (DMA)
2100 Incorrect access to hardware component (AS1)

2101 Incorrect access to hardware component (AS1)
2110 Incorrect access to hardware component (PAP)
2111 Incorrect access to hardware component (PAP)
2501 Error initializing the IO-Controller's PLCIO Bus
2502 Error initializing the Remote IO Master
2900 No interrupt handler installed for IRL1 Interrupt

3000 - 3999 SOFTWARE TRAP LIBRARY ERRORS

3000 System Structure Configuration Error
3001 System Structure Configuration Error
3002 ERR_SS_NOMEM
3003 Invalid Pointer in the System Structure
3004 Invalid length for the System Structure
3005 System Structure Configuration Error
3006 ERR_SS_WRONG_ENT
3007 ERR_SS_WRONG_FL
3008 Incorrect Search criteria for the system structure
3010 Invalid Address for a B&R Module
3011 Offset Error in a B&R Module
3012 Invalid Section in a B&R Module
3013 Invalid Data length accessing a B&R Module
3014 ERR_MO_NOBLOCK
3015 Module data in the contents directory module and B&R module is different
3016 ERR_MO_NOINSTALL
3017 ERR_MO_WRONG_CS
3018 Invalid name length for a B&R Module
3019 B&R-Module not available
3020 ERR_SIO_NOSIOTAB
3021 ERR_SIO_WRONGFLAG
3022 ERR_SIO_NOSIO
3023 ERR_SIO_WRONGPARM
3027 ERR_TRY_NEXT_BITMAP
3028 Invalid Memory Class
3029 Not enough Memory Available
3030 No more memory available
3031 Invalid memory class
3032 ERR_SM_WRONG_DATPTR
3033 ERR_SM_NOSYSTIMER
3034 ERR_SM_NOSSTIMER
3035 ERR_SM_BUSY
3036 AVT Link count Error
3037 AVT already exists
3038 ERR_AVT_UPDATE
3039 AVT Reference not available
3040 ERR_CR_WRONG_ID
3041 ERR_CR_ATTACHED
3042 ERR_CR_NOATTACH
3045 Checksum in B&R Module Header is wrong
3050 No Comm Area configured for a communication task
3051 No entry found in the connection list of the Comm Area
3052 Invalid state of a connection in the Comm Area
3053 No more entries are free in the connection list of the Comm Area
3054 Invalid Index for KA_del_entry

3060 ERR_HI_NOMOD
3061 ERR_ER_NOMOD
3062 Multiple run through of the ring buffer
3070 ERR_FOC_WRONGFLAG
3071 ERR_FOC_NOPTR
3072 Invalid Task Class for a PV to be forced
3073 Invalid data type for a PV to be forced
3074 Invalid Base pointer for a PV to be forced
3075 Invalid B&R physical address (Extension bit) for a PV to be forced
3076 Invalid Index in the Force system functions
3077 This PV is already Forced
3080 ERR_xx_WRONGOFF
3085 No User Exception task found
3089 Error 'info' in the communication
3090 Invalid function number in a SYSTRAP-Call
3091 Invalid function number in a INSTTRAP-Call
3092 Invalid function number in a BURTRAP-Call
3100 Size of a bit map unit is too large or 0
3101 Attempt to generate a bitmap with 0 units
3102 The pointer in the bitmap management data is not WORD aligned
3103 A large enough memory area could be found
3104 The passed data pointer points to the management area.
3105 The requested data length is larger than the total memory area
3300 BURLIB: Wrong TC Number
3301 BURLIB: Wrong Offset
3302 BURLIB: Wrong state of an object
3303 BURLIB: No message sent/received
3305 BURLIB: Object already exists
3304 BURLIB: Timeout Error
3306 BURLIB: Entry already exists
3307 BURLIB: Linkcnt <> 0 or Linkcnt > 127
3308 BURLIB: Error programming a module
3309 BURLIB: Invalid mailbox
3310 BURLIB: Invalid base pointer
3311 BURLIB: Invalid PLC IO Type
3312 BURLIB: Invalid extension bit
3320 BURLIB: Object is already deleted
3324 BURLIB: Object does not exist
3328 BURLIB: Wrong object type
3332 BURLIB: No entry exists
3336 BURLIB: Wrong Ident
3560 BURLIB: No more semaphore buffer
3564 BURLIB: Semaphore not available
3568 BURLIB: Semaphore was deleted
3584 BURLIB: Time not available
3600 Not enough memory in User Area
3601 Not enough memory in PLCSW AREA
3602 Not enough memory in OS Area
3603 Not enough memory in Dig. System Area
3604 Not enough memory in Ana. System Area
3605 Not enough memory in Debug Area
3610 Not enough memory in Dig. IO Area
3611 Not enough memory in Ana. IO Area
3612 Not enough memory in IO Management Area

3620 Not enough memory in Dig. RIO Area
3621 Not enough memory in Ana. RIO Area
3622 Not enough memory in RIO Management Area
3623 Not enough memory in loc. RIO Area
3699 Invalid memory class
3700 Invalid parameter with MEM_free in User Area
3701 Invalid Parameter with MEM_free in PLCSW AREA
3702 Invalid Parameter with MEM_free in OS Area
3703 Invalid Parameter with MEM_free in Dig. System Area
3704 Invalid Parameter with MEM_free in Ana. System Area
3705 Invalid Parameter with MEM_free in Debug Area
3710 Invalid Parameter with MEM_free in Dig. IO Area
3711 Invalid Parameter with MEM_free in Ana. IO Area
3712 Invalid Parameter with MEM_free in IO Management Area
3720 Invalid Parameter with MEM_free in Dig. RIO Area
3721 Invalid Parameter with MEM_free in Ana. RIO Area
3722 Invalid Parameter with MEM_free in RIO Management Area
3723 Invalid Parameter with MEM_free in loc. RIO Area
3799 Invalid memory class
3800 Invalid memory management structure in User Area
3801 Invalid memory management structure in PLCSW AREA
3802 Invalid memory management structure in OS Area
3803 Invalid memory management structure in Dig. System Area
3804 Invalid memory management structure in Ana. System Area
3805 Invalid memory management structure in Debug Area
3810 Invalid memory management structure in Dig. IO Area
3811 Invalid memory management structure in Ana. IO Area
3812 Invalid memory management structure in IO Management Area
3820 Invalid memory management structure in Dig. RIO Area
3821 Invalid memory management structure in Ana. RIO Area
3822 Invalid memory management structure in RIO Management Area
3823 Invalid memory management structure in loc. RIO Area
3899 Invalid Memory Class
3912 Not enough EPROM memory
3913 EPROM not erased
3915 EPROM cannot be burned, write protect switch ?
3922 EPROM with programming voltage cannot be erased
3923 EPROM Verify not OK

4000 - 4999 ALI-TRAP LIBRARY ERRORS

4000 ALI: The referenced ALI-Device (SAP-List) doesn't exist
4001 ALI: The referenced Telegram Queue (Layer7-SAP) doesn't exist
4002 ALI: All communication buffers of the Telegram queue are full
4003 ALI: Wrong transmission direction of the selected Telegram queue (In/Out)
4004 ALI: The version of the NW configuration module is not supported
4010 ALI: Not enough memory available
4011 ALI: The given connection number is unplanned
4012 ALI: Invalid operation on an inactive connection
4013 ALI: Invalid operation on an inactive connection
4014 ALI: Invalid connection type
4015 ALI: The given connection name is unplanned
4016 ALI: Warning: Executing Hidden Abort
4017 ALI: Connection is not a client connection

4040 ALI: INIT not OK
4041 ALI: Error accessing to the communication module
4042 ALI: The planned serial driver is not supported
4043 ALI: Overrun of the communication buffer for serial communication
4044 ALI: The SIO-Timer has too little us/TICK
4045 ALI: Timer not suitable for cyclic operation or IRQ
4050 ALI: The maximum telegram length was exceeded during coding
4051 ALI: The FRAME length was exceeded
4052 ALI: The telegram is incorrectly coded and can't be decoded
4053 ALI: The list available is too small
4060 ALI: Non-allowed type of ALI Device (SAP-List)
4061 ALI: The configured number of the communication buffer is not enough
4062 ALI: The configured maximum telegram length is not enough
4063 ALI: The Profibus Service is not supported
4064 ALI: OV version is incompatible with its communication partners
4065 ALI: The SAP List is not available on this device
4080 ALI: The selected function is not implemented
4081 ALI: The ALI client module must be loaded for client services
4900 NW: Not enough memory available on the NW module (-> NWCFG)
4901 NW: Layer2 LSAP Number too high (-> NWCFG)
4902 NW: Connection attribute for a VB is invalid (-> NWCFG)
4903 NW: Dev_kbl_index of a VB is wrong (-> NWCFG)
4904 NW: Error occurred during Layer2 SAP Configuration
4905 NW: A Layer2 LSAP is being accessed by several connections (-> NWCFG)
4906 NW: Error while initializing Layer2
4907 NW: Error with Start the Layer2
4908 NW: Checksum Error during Download
4910 NW: Meminit does not deliver a DPR_struct_ptr
4912 NW: IN FIX KBL planned connection is not supported (-> NWCFG)
4920 NW: No valid FMA7 Service
4921 NW: FMA7 Service can not be executed

5000 - 5999 TASK CLASSES ERRORS

5000 No PLC task in the Task Class
5010 Task class stack destroyed
5011 Task class numbers do not match
5020 No input data during shovelling for the InitSP
5021 No output data during shovelling for the InitSP
5030 No input available for the Interrupt task class
5100 Task class cycle time violation
5200 The HS task class stack was destroyed
5201 The HS task class data area was destroyed
5202 Hs task class cycle time violation
5203 There is no Task Class #1
5500 This I/O controller is not implemented
5501 This I/O controller is not implemented
5900 The PLC task/HS task chaining is destroyed

6000 - 6999 SYSTEM TASK ERRORS

6000 The system manager is not correctly configured
6001 The cyclical activities of the system manager are missing
6002 Total time bottle-neck in system
6009 B&R Library or mathematic library not installed
6010 System Manager internal Error
6011 System Manager internal Error
6015 Incorrect parameter transmitted to system manager
6016 Communication channel is already being used
6017 Communication channel is not used
6020 Cyclic system manager action is not implemented
6021 Module checksum destroyed
6022 System Manager Checksum destroyed
6023 Checksum of the system anchor destroyed
6024 Checksum of the basis management table destroyed
6025 Checksum of the system management table destroyed
6030 System Structure No. 0 destroyed
6031 System Structure No. 1 destroyed
6032 System Structure No. 2 destroyed
6033 System Structure No. 3 destroyed
6034 System Structure No. 4 destroyed
6035 System Structure No. 5 destroyed
6036 System Structure No. 6 destroyed
6037 System Structure No. 7 destroyed
6038 System Structure No. 8 destroyed
6039 System Structure No. 9 destroyed
6040 System Structure No. 10 destroyed
6041 System Structure No. 11 destroyed
6042 System Structure No. 12 destroyed
6043 System Structure No. 13 destroyed
6044 System Structure No. 14 destroyed
6045 System Structure No. 15 destroyed
6046 System Structure No. 16 destroyed
6047 System Structure No. 17 destroyed
6048 System Structure No. 18 destroyed
6049 System Structure No. 19 destroyed
6050 System Structure No. 20 destroyed
6051 System Structure No. 21 destroyed
6052 System Structure No. 22 destroyed
6053 System Structure No. 23 destroyed
6054 System Structure No. 24 destroyed
6055 System Structure No. 25 destroyed
6056 System Structure No. 26 destroyed
6057 System Structure No. 27 destroyed
6058 System Structure No. 28 destroyed
6059 System Structure No. 29 destroyed
6060 B&R Module No. 0 destroyed
6061 B&R Module No. 1 destroyed
6062 B&R Module No. 2 destroyed
6063 B&R Module No. 3 destroyed
6064 B&R Module No. 4 destroyed
6065 B&R Module No. 5 destroyed
6066 B&R Module No. 6 destroyed

6067 B&R Module No. 7 destroyed
6068 B&R Module No. 8 destroyed
6069 B&R Module No. 9 destroyed
6070 B&R Module No. 10 destroyed
6071 B&R Module No. 11 destroyed
6072 B&R Module No. 12 destroyed
6073 B&R Module No. 13 destroyed
6074 B&R Module No. 14 destroyed
6075 B&R Module No. 15 destroyed
6076 B&R Module No. 16 destroyed
6077 B&R Module No. 17 destroyed
6078 B&R Module No. 18 destroyed
6079 B&R Module No. 19 destroyed
6080 B&R Module No. 20 destroyed
6081 B&R Module No. 21 destroyed
6082 B&R Module No. 22 destroyed
6083 B&R Module No. 23 destroyed
6084 B&R Module No. 24 destroyed
6085 B&R Module No. 25 destroyed
6086 B&R Module No. 26 destroyed
6087 B&R Module No. 27 destroyed
6088 B&R Module No. 28 destroyed
6089 B&R Module No. 29 destroyed
6090 B&R Module No. 30 destroyed
6091 B&R Module No. 31 destroyed
6092 B&R Module No. 32 destroyed
6093 B&R Module No. 33 destroyed
6094 B&R Module No. 34 destroyed
6095 B&R Module No. 35 destroyed
6096 B&R Module No. 36 destroyed
6097 B&R Module No. 37 destroyed
6098 B&R Module No. 38 destroyed
6099 B&R Module No. 39 destroyed
6100 APPLICATION LOADER: Invalid System Message with SY_Event
6101 APPLICATION LOADER: Invalid Service ID in Download
6102 APPLICATION LOADER: Invalid MI-Code in Download
6103 APPLICATION LOADER: Invalid State of Connection
6104 APPLICATION LOADER: Invalid data type in service
6105 APPLICATION LOADER: KA-Ptr = NULL
6200 VARIABLE ACCESS TASK: DATA TYPE - Error
6201 VARIABLE ACCESS TASK: No access rights with PV-read
6202 VARIABLE ACCESS TASK: No access rights with PV-write
6203 VARIABLE ACCESS TASK: Index Error
6204 VARIABLE ACCESS TASK: PV not found
6205 VARIABLE ACCESS TASK: VL not found
6206 VARIABLE ACCESS TASK: Access Error
6207 VARIABLE ACCESS TASK: Invalid system message with SY_Event
6208 VARIABLE ACCESS TASK: Invalid connection state
6209 VARIABLE ACCESS TASK: Invalid MI-Code
6210 VARIABLE ACCESS TASK: KA-Ptr = NULL
6211 VARIABLE ACCESS TASK: Invalid Base pointer
6212 VARIABLE ACCESS TASK: No valid B&R Address
6213 VARIABLE ACCESS TASK: Index points to a descriptor
6300 PROGRAM INVOCATION: Invalid Event Type

6301 PROGRAM INVOCATION: Invalid System Message
6302 PROGRAM INVOCATION: PI_Index error
6303 PROGRAM INVOCATION: Invalid MI-Code
6304 PROGRAM INVOCATION: Invalid Object
6305 PROGRAM INVOCATION: Invalid state of the Objects
6306 PROGRAM INVOCATION: KA-Ptr = NULL
6307 PROGRAM INVOCATION: No more connections free
6308 PROGRAM INVOCATION: OS Error - t_start
6310 KA-Ptr = NULL
6311 Event-Routine = NULL-Ptr
6350 MPServer: Not a valid instruction code
6351 MPServer: Multiprocessor not available
6352 MPServer: Wrong instruction data length
6353 MPServer: Timeout in Handshake to multiprocessor
6400 Checksum of the system table management table is wrong
6401 Checksum of the system table is wrong
6402 Checksum of a B&R Module is wrong
6404 No system manager exists
6406 The RIO Master does not report
6407 Timeout with RIO-Life list Creation
6408 There is an invalid I/O type in the system bus table
6409 Referenced RIO-Slave does not exist
6410 Empty RB Section in configuration module
6412 Warning: Network is configured in NWCFGMOD, but is not in the system bus
6413 Warning:Unknown NW Type
6414 Warning:NW Module Number 0, but more than 1 NW module configured
6415 Memory management in User RAM corrupted, possible battery problem
6420 Multiprocessor doesn't answer to 1st handshaking part
6422 Multiprocessor doesn't answer to 2nd handshaking part
6424 MP: Timeout waiting for CPU_INIT
6426 MP: Timeout waiting for CPU_READY
6428 MP: Timeout waiting for DPR-OK-Signal of the MP-Server
6429 MP: Timeout waitint fro Bootup_OK from the PLC CPU
6430 MP-Warning: more than 1 HS task class configured
6432 MP-Warning: IRQ Task class configured
6433 APM PROM changed:- TOTAL INIT required.
6500 TASK-TASK-COMMUNICATION: Invalid Event
6501 TASK-TASK-COMMUNICATION: Invalid MI-Code
6502 TASK-TASK-COMMUNICATION: Invalid Instruction number
6503 TASK-TASK-COMMUNICATION: Invalid Comm. ID
6504 TASK-TASK-COMMUNICATION: Invalid data type
6505 TASK-TASK-COMMUNICATION: KA-Ptr = NULL
6506 TASK-TASK-COMMUNICATION: No more connections free
6507 TASK-TASK-COMMUNICATION: ALI Error
6520 DEBUGGER: Invalid MI-Code
6521 DEBUGGER: No more connections free
6522 DEBUGGER: Invalid TT Msg Type
6523 DEBUGGER: Invalid Instruction number
6524 DEBUGGER: Invalid Event Info
6600 The master spooler sends no event to the instruction sender
6610 Command for the master spooler is not implemented
6620 The ASIC#2 is not finished
6621 Spooling: Timeout with ASIC#2-Init after Periell Error
6622 Periell Error during Spooling

6623 Periell Interrupt during Spooling
6624 AMA Error during spooling
6625 Timeout during spooling a data block
6626 Spool-SP can only be called in the INIT phase
6627 Spooler: Wrong I/O type
6628 Spooler: System module does not exist
6630 The Master spooler doesn't exist
6650 The Slave spooler does not report
6652 The Frame is not completely sent
6654 Incorrect command to the slave spooler
6656 A module with an invalid module type was spooled
6658 Incorrect Module info. All Modules must be installed and started
6659 Spooler: The queue for the drum sequencer doesn't exist
6660 Error that makes further spooling impossible
6662 Spooler: Timeout waiting for bootup of intelligent peripheral
6666 Info: Spooler still working
6699 Info: Reseting the Spool FBK
6700 Stack of Task No. 0 destroyed
6701 Stack of Task No. 1 destroyed
6702 Stack of Task No. 2 destroyed
6703 Stack of Task No. 3 destroyed
6704 Stack of Task No. 4 destroyed
6705 Stack of Task No. 5 destroyed
6706 Stack of Task No. 6 destroyed
6707 Stack of Task No. 7 destroyed
6708 Stack of Task No. 8 destroyed
6709 Stack of Task No. 9 destroyed
6710 Stack of Task No. 10 destroyed
6711 Stack of Task No. 11 destroyed
6712 Stack of Task No. 12 destroyed
6713 Stack of Task No. 13 destroyed
6714 Stack of Task No. 14 destroyed
6715 Stack of Task No. 15 destroyed
6716 Stack of Task No. 16 destroyed
6717 Stack of Task No. 17 destroyed
6718 Stack of Task No. 18 destroyed
6719 Stack of Task No. 19 destroyed
6720 Stack of Task No. 20 destroyed
6721 Stack of Task No. 21 destroyed
6722 Stack of Task No. 22 destroyed
6723 Stack of Task No. 23 destroyed
6724 Stack of Task No. 24 destroyed
6725 Stack of Task No. 25 destroyed
6726 Stack of Task No. 26 destroyed
6727 Stack of Task No. 27 destroyed
6728 Stack of Task No. 28 destroyed
6729 Stack of Task No. 29 destroyed
6800 Stack of a HS-Task destroyed

7000 - 7499 BOOT ERRORS

7000 Error while searching for the Sync bytes of the boot module
7001 Error while searching the contents module
7002 Error while searching for a B&R Module with the contents module
7004 Checksum of both anchors is wrong
7006 Incorrect Checksum of the PSOS- Node-ConfTab
7007 PSOS-Module not found or incorrect checksum
7008 PROBE-Module not found or incorrect checksum
7009 Module $$probeIO not found
7010 Im Init:CS of the Basis VWTab is incorrect
7012 Im Init: CS of the configuration module is incorrect
7015 HW-Traplib was not found
7016 SYS-Traplib was not found
7017 Install trap library not found
7018 Configuration module not found
7019 Configuration module not found
7020 Startup module not found
7022 The CS of the error module was destroyed
7023 Fatal operating system error
7024 The check sum 1. contents was destroyed
7025 Operating system cannot be started
7101 Illegal boot-up reason
7102 Checksum of the contents in user RAM 1 is corrupted
7104 Checksum of the contents in user RAM 1 is corrupted
7108 MP: Handshake with the main CPU doesn't work. No RECONFIG
7110 APM was exchanged :- TOTAL INIT required.

7500 - 7999 OPERATOR PANEL TRAP ERRORS

7500 Incorrect Identifier
7501 OP Ctrl not connected
7502 Cannot wait for GETKEY
7503 General error flag
7504 No communication to the selected OP
7505 Impossible Window
7506 RAM-Error
7507 ROM-Error
7508 Display not connected
7509 Incorrect Display
7510 Incorrect Configuration
511 Keyboard is defect
7512 Impossible wait time repeat mode
7513 Impossible repeat frequency
7514 Blinking character outside the window
7550 KeyPad is not available
7600 Incorrect number of panels in config module
7601 Incorrect name in config module
7602 Incorrect device name in config module
7603 Incorrect device type in config module
7604 Incorrect display number in config module
7605 Incorrect number of keys in config module
7606 Incorrect number of lines in config module
7607 Incorrect number of lines in config module

7608 Incorrect number of pictures in config module
7609 Incorrect version name in config module
7650 BTSERV C200 : Error with EV_receive
7651 BTSERV C200 : Key buffer empty

8000 - 8999 INSTALLATION ERRORS

8001 Attempt to install a PLC task for an unconfigured task class
8002 PLC task stack requirements exceed total reserved for task class stack
8003 The PLC task requested cycle time < cycle time of associated task class
8004 Analog data area requested by PLC task > analog data area of task class.
8005 Digital data area requested by task > digital data area of task class.
8006 Error allocating memory
8007 No more room in the system table
8008 The requested I/O data length by the PLC task is too large.
8009 Install Status of a B&R Module is wrong
8010 Pointer does not point to a valid object
8011 Deinstall a running PLC task
8012 A PLC task requested a global analog data area > the entire task class
8013 A PLC task requested a global digital data area > the entire task class
8014 Number of PLCIO types is configured too small for task classes
8015 Incorrect module type installing a system task or user task
8016 Referenced System bus module missing
8017 Variable PB-IX not configured
8018 IRQ-Task not supported by 2005 systems
8019 IRQ-Task not supported by 2005 systems
8020 B&R module checksum is wrong
8021 The B&R Module already exists in the system
8022 System table is full
8023 Pointer does not point to a B&R Module
8024 2010-CPU-Version can't handle I/O-interrupts
8030 Trap number not implemented
8031 The B&R module type is not defined for a trap install
8032 The hardware trap library version does not match the CPU version
8033 Too many expansion trap libraries installed
8034 Version Inconsistency BURTRAP.br
8040 An AVT Library with the same name already exists
8041 Not enough AVT entries configured (PLCSWCFG)
8045 Configuration module version number doesn't match current version number
8046 No valid type ??
8048 Wrong INSTALL-Mode
8050 Debug task for Interrupt task class or Exception task class
8052 The I/O controller type requested by the PLC task doesn't exist
8054 The I/O controller type requested by the PLC task doesn't exist
8056 Invalid IO-Controller-ID
8060 Install attempt of interrupt handler for non-existant system bus module
8062 Installing the Process Variable Table
8063 Shovel processor does not exist
8070 Attempt to install an unconfigured device driver
8071 No buffer available ??
8072 Incorrect buffer transmitted ??
8073 Command is not supported ??
8074 Device doesn't exist ??

8075 The System manager does not command the RIO-Master
8076 Not enough internal memory ??
8077 Not enough internal memory
8078 Transmission active
8079 Device receive error
8080 RIO Master has not finished configuring the instruction list.
8081 Error during I/O Transfer to the remote station.
8082 Error during Init Remote IO. RIO master is not at given module address
8083 RIO in HS-Task classes is not valid
8084 RIO First initialization not complete after 40 seconds
8090 RIO Master has not completed output copying
8092 Interrrupt task or Exception task is not allowed for the Debugger
8095 Error in Config.Module HS section does not match the RK section
8096 Max. number of interrupt tasks exceeded
8097 More than 1 Exception task class configured
8098 IRQ-Task for IO-Module or EXC-Task for Exception no already exists
8099 Installation failure
8100 Deleteing a B&R-Module: No PI Object exists for this module
8101 Ptr Error while deleting a B&R-Module
8111 Ptr Error whil deleting a system task
8112 System task has not been stopped before deletion
8115 Ptr Error while deleting a user task
8116 User task has not been stopped before deletion
8120 Still an active reference to AVT Module while deleting (linkcnt != 0)
8121 Ptr Error while deleting an AVT Library Module
8150 ID is not in a table
8151 Code for Error Message
8160 There is no I/O timer
8161 The given timer value is wrong
8162 Unknown I/O Timer distributor
8205 Object pointer for starting a System/User Task is invalid
8206 No enough memory configured (MEMCL_OS/MEMCL_PLCSW) when starting a task
8300 RIO Exception : No error entered
8301 RIO Exception : Master Error
8302 RIO Exception : Slave Error
8401 RIO Exception : Wrong block number
8402 RIO Exception : Unknown command
8403 RIO Exception : Input already configured
8404 RIO Exception : Shovel table is too long
8407 RIO Exception : Module address too high
8408 RIO Exception : Internal Address too high
8409 RIO Exception : Block already configured
8413 RIO Exception : Output already configured
8414 RIO Exception : Error not repairable, despite multiple repeats
8451 RIO Exception : Invalid slave number configured
8452 RIO Exception : Shovel offsets do not suit or overlap
8453 RIO Exception : Block contains no data
8454 RIO Exception : Block number for slave too large
8455 RIO Exception : Block configured without Input or Output
8456 RIO Exception : Too many instruction lists configured
8500 RIO Exception : Slave does not report
8501 RIO Exception : Slave address invalid (0 or too large)
8502 RIO Exception : Error on slave
8503 RIO Exception : Slave does not report despite multiple repeats

9000 - 9999 VARIOUS

9001 No more space in CSwrite table
9002 Error converting ASCII -> BRSTR
9003 The end of the MSPS region in the Config.Module is incorrectly set
9004 The given shovel table is empty
9005 The I/O instruction has not been executed yet
9006 Incorrect SAP in the SAPLI_TAB (SAP_check_rights)
9008 The checksum of a Continuous IVZ is destroyed
9009 Checksumm length not equal to 4 is not supported
9100 EXCEPTION: Bus Error
9101 EXCEPTION: Address Error
9102 EXCEPTION: Illegal Instruction
9103 EXCEPTION: Privilege Violation
9104 EXCEPTION: Division by Zero Error
9105 EXCEPTION: Spurious Interrupt
9106 EXCEPTION: Unimplemented Instruction
9107 EXCEPTION: Uninitialized Interrupt
9110 EXCEPTION: Perieller Error
9111 EXCEPTION: Perieller Error
9112 HS#1 cycle time error
9113 HS#2 cycle time error
9114 HS#3 cycle time error
9115 HS#4 cycle time error
9116 HS#1 maximum cycle time violation
9117 HS#2 maximum cycle time violation
9118 HS#3 maximum cycle time violation
9119 HS#4 maximum cycle time violation
9120 TC#1 cycle time error
9121 TC#2 cycle time error
9122 TC#3 cycle time error
9123 TC#4 cycle time error
9124 TC#1 maximum cycle time violation
9125 TC#2 maximum cycle time violation
9126 TC#3 maximum cycle time violation
9127 TC#4 maximum cycle time violation
9128 EXCEPTION: NMI-Handler
9129 Exception released by RIO-Master.
9130 EXCEPTION: Default handler
9131 IRQ-TC maximum cycli time violation
9132 EXC-TC maximum cycle time violation
9199 Default-EXCEPTION-Handler called -> download EXERMO
9200 WARNING: System halted after power loss
9210 WARNING: Bootup by Watchdog or manual RESET
9301 INTELLIGENT MIXED MODULE: Not enough data for congfiguration
9302 INTELLIGENT MIXED MODULE: Not enough memory for configuration
9303 INTELLIGENT MIXED MODULE: The calibration factor is invalid
9304 INTELLIGENT MIXED MODULE: The sampling rate is invalid
9800 Trap-Expandsion library not installed (RIO, ...)
9900 General Bugfix error
9999 This function is not implemented yet

B&R SALES AND SUPPORT

B&R SALES AND SUPPORT

Services:

All B&R subsidiaries and representatives hold ***customer training sessions*** which provide extensive coverage of the B&R product range as well as regular PLC programming courses. For information about current training programs your B&R sales consultant will be glad to help out.

Our worldwide staff of trained sales consultants are also ***available to support any projects*** where a well founded knowledge of hardware or software oriented ***project planning*** is needed. Software can be developed by one of our specialized application departments and our technicians are on hand for fast and efficient installation or service upon request. Support staff hourly rates and travel expenses are also available through your local sales consultant.

Any questions or problems which may arise can be answered or solved through the use of B&R's ***global sales and service telephone network*** which is available for technical support almost 24 hours each and every day. Telephone numbers of all B&R subsidiaries and business partners are listed on the following pages.

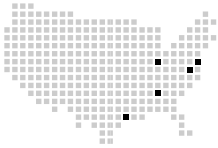


1 USA

B&R Subsidiary (1987)
Sales, Applications, Training

B&R Industrial Automation Corp.
1325 Northmeadow Parkway Suite 130
Roswell, Georgia 30076

Tel: 404 / 772-0400
Fax: 404 / 772-0243



2 CYPRUS

B&R Representative
Sales, Applications, Training

TONIS G. PAVLIDES
Electrical & Mechanical Contractor
75 c, Makarios III Ave.,
Limassol, Cyprus

Tel: 5-337906, 346907
Fax: 5-339322

3 AUSTRALIA

B&R Representative
Sales, Applications, Training

PROCON INSTRUMENT TECHNOLOGY PTY. LTD.
365 Montague Rd., West End,
Brisbane, Queensland, Australia 4101
P.O. Box 1539 Coorparoo, Qld. 4151

Tel: 07-846 3511
Fax: 07-846 1588

4 UKRAINE

B&R Representative
Sales, Applications, Training

SKIF ENGINEERING Ltd.
ul. Vinnichenka 9a
274028 Chernivzi

Tel: 03722 / 45207, 44024
Fax: 03722 / 47332

5 INDONESIA

B&R Representative
Sales, Applications, Training

P.T. SRIKAYA PUTRA MAS
P.O. Box. 218 SBS, Surabaya-Selatan

Tel: 031 / 813109, 830403, 839360
Fax: 031 / 839870

6 BELARUS

B&R Representative
Sales, Applications, Training

ENTAS
220113 Minsk, P.O. Box 500

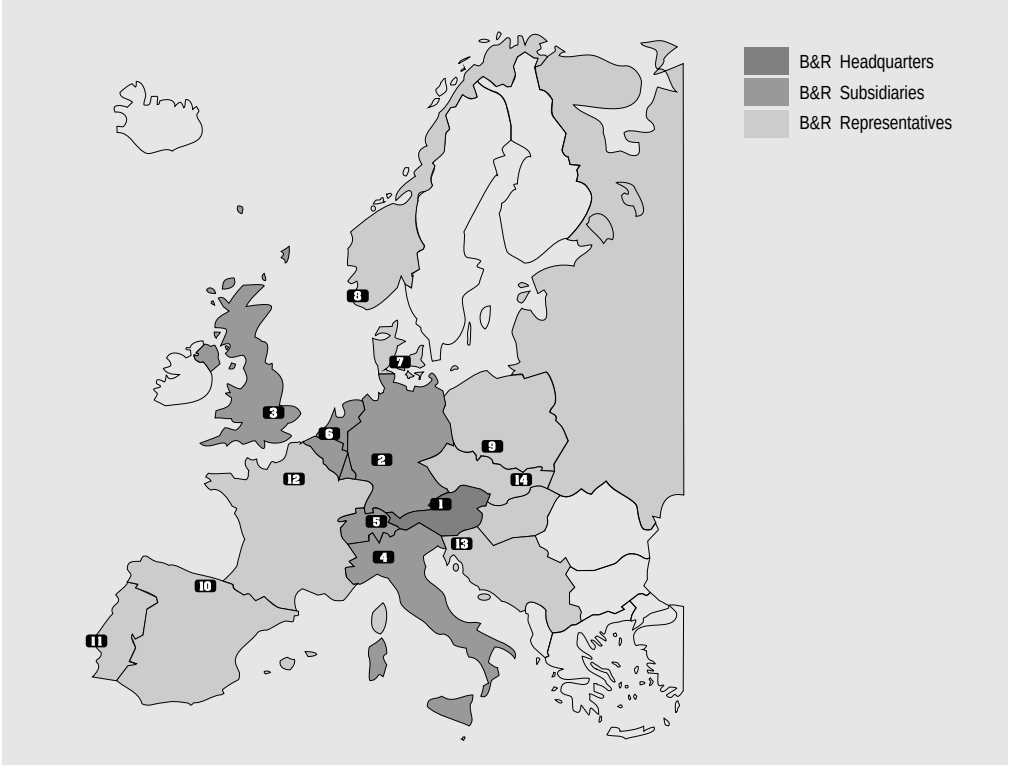
Tel: 0172 / 686537
Fax: 0172 / 686537

7 PAKISTAN

B&R Representative
Sales, Applications, Training

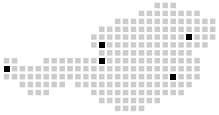
INTECH
6 Tipu Block, New Garden Town
Lahore

Tel: 242 / 5838562
Fax: 242 / 5880052



1 AUSTRIA

B&R Headquarters (1979)
Development, Production,
Sales, Applications, Training

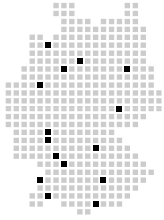


Bernecker + Rainer
Industrie-Elektronik Ges.m.b.H.
A-5142 Eggelsberg 120

Tel: 07748 / 6586-0
Tbx: 27477
Fax: 07748 / 6586-26

2 GERMANY

B&R Subsidiary (1983)
Sales, Applications, Training



B&R Industrie-Elektronik GMBH
Ernst-Wiss-Str. 18
D-65933 Frankfurt 80

Tel: 069 / 380315-0
Fax: 069 / 397019

TECHNICAL OFFICES IN GERMANY

Sales, Applications, Training

T.O. Bayern
Freisinger Str. 34
D-85737 Ismaning
Tel: 089 / 996554-0
Fax: 089 / 996554-99

T.O. Southwest
Friedrich-Dürr-Str. 70
D-74074 Heilbronn
Tel: 07131 / 5971-0
Fax: 07131 / 5971-71

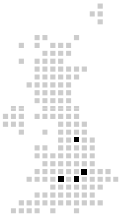
T.O. West
Kimpler Str. 294
D-47807 Krefeld
Tel: 02151 / 3334-5
Fax: 02151 / 3334-60

T.O. Central
Ernst-Wiss-Str. 18
D-65933 Frankfurt 80
Tel: 069 / 380315-0
Fax: 069 / 380315-73

T.O. Southeast
Calvisius Str. 38
D-04177 Leipzig
Tel: 0341 / 44646-0
Fax: 0341 / 4419100

T.O. North
Rotenburger Str. 3
D-30659 Hannover 51
Tel: 0511 / 614077
Fax: 0511 / 612337

3 U.K.



B&R Subsidiary (1988)
Sales, Applications, Training

B&R Industrial Automation Ltd.
Milnyard Square,
Orton Southgate
Peterborough, PE2 6GX

Tel: 0733 / 371320
Fax: 0733 / 371306

4 ITALY

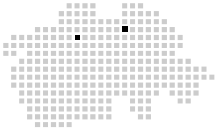


B&R Subsidiary (1987)
Sales, Applications, Training

B&R Automazione Industriale S.p.A.
Viale Certosa 191
I-20151 Milano

Tel: 02 / 3085844, 3083822, 3084358
Fax: 02 / 38001864

5 SWITZERLAND



B&R Subsidiary (1987)
Sales, Applications, Training

B&R Industrie Automation AG
Langfeldstr. 88
CH-8500 Frauenfeld

Tel: 054 / 217425
Fax: 054 / 217424

6 BENELUX



B&R Subsidiary (1989)
Sales, Applications, Training

B&R Industrièle Automatisering B.V.
Hoge Schouw 1
4817 BZ Breda

Tel: 076 / 715303
Fax: 076 / 715306

7 DENMARK

B&R Representative
Sales, Applications, Training

HANO Elektroteknik A/S
Kallerupvej 60
DK-5230 Odense M

Tel: 66190600
Fax: 66190700

8 NORWAY

B&R Representative
Sales, Applications, Training

Kverneland Elektriske A/S
Bedriftsv. 6
N-4062 Klepp St

Tel: 423840
Fax: 423577

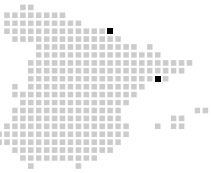
9 POLAND

B&R Representative
Sales, Applications, Training

CONSTEL Sp.zo.o.
40-952 Katowice
ul. Jordana 25

Tel: 031 / 578770
Fax: 032 / 511292

10 SPAIN

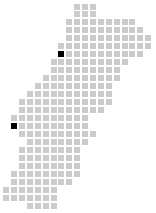


B&R Representative
Sales, Applications, Training

LARRAIOZ
Larraioz Etxea Ind. (Apartado 193)
E-20800 Zarautz (Gipuzkoa)

Tel: 943 / 140139
Fax: 943 / 140327

11 PORTUGAL

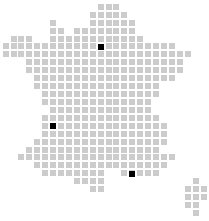


B&R Representative
Sales, Applications, Training

TECNILAB
Av. Columbano Bordalo
Pinheiro 97 -2° Dto
P-1000 Lisbon

Tel: 1 / 7269129
Fax: 1 / 7264550

12 FRANCE



B&R Representative
Sales, Applications, Training

GROUPE JEAMBRUN
BALLUFF Dpt. ABR
2, rue du Vallon
F-94440 Marolles en Brie

Tel: (1) 45692332
Fax: (1) 45990864

13 SLOVENIA

B&R Representative
Sales, Applications, Training

ISKRA ROBOTEC
Stegne 13
61000 Ljubljana

Tel: 061 / 575054
Fax: 061 / 572247

14 SLOVAKIA

B&R Representative
Sales, Applications, Training

URAP AUTOMATIZACIA
Po Box B152
02141 Zilina

Tel: 089 / 622070
Fax: 089 / 42340

More contact addresses
available upon request

INDEX

A

Allowed cycle time violation 2 41, 4 67
Application Program 1 7, 2 10, 2 11
Application specific function block 4 167
  Compile 4 171
  Create 4 167
Array 4 110
Array variable 4 110
AVT function block 4 165

B

B&R Module 2 40, 4 124
Back up files, delete 4 56
Batch file PG.BAT 3 9
Block parser 4 124
Boot behaviour 2 28
Boot Modes 2 20, 4 180
  DIAGNOSE 2 25, 4 181
  ERROR 2 25
  INIT 2 23, 4 180
  RECONFIG 2 24
  RESET 2 24, 4 181
  TOTALINIT 2 22, 4 181
Boot up 2 20
Bus expansion B&R 2005 ..... 4 115
Bus expansion B&R 2005 on a B&R 2010 ..... 4 116
Bus expansion B&R 2010 ..... 4 114
Bus expansion B&R 2010 on a B&R 2005 ..... 4 118
Button block 4 27

C

C 1 14
C Toolbox 1 7, 1 14, 2 11
Code generator 4 124
Compiler 4 124
Compiler options 4 125
Compilier sequence 4 124
CONFIG.SYS 3 8, 3 15
Configuration File PG.INI 4 8
Connection between PG and PLC 4 44
Contact symbols 4 71
Coprocessor directory 4 53
Cross reference 4 158
Cyclic runtime system 2 14
Cyclic tasks 1 7, 2 11
Cycle time 1 7, 2 11, 4 67
  configuring 4 67
Cycle time monitoring 2 41
Cycle time violation 2 41, 4 67
  allowed 2 41, 4 67
  maximum 2 41
  simple 2 41

D

Databank 4 51
Databank directory (DBK) 4 51
Databank path 4 52
Data branch 4 151
Data module 4 60
Data type of variables 4 109
Debugger 4 139
Debugger for STL and PL2000 ..... 4 144
Delivery contents of the PG2000 Software 3 5
DIAGNOSE 2 25, 4 181
Dialog boxes 4 27
Display codes for exceptions 2 43
Download function 4 128
Dual Ported RAM (DPR) 2 6, 2 39
dynamic 4 108

E

Editor
  Function block 4 169
  Ladder Diagram 4 74
  PL2000 ..... 4 99
  Statement List 4 88
  Variable Declaration 4 107
ERROR 2 25
Error analysis 4 178
Error handling 1 9, 2 42
Error module 2 42, 4 178
Error reporting
  extended 2 18, 2 43, 4 179
  normal 4 178
Execution order of the task classes 2 14
Execution order of the tasks 2 15
EXC task class 2 17
Exception 2 17, 2 42
Exception handler 2 17, 2 42
Exception number 2 17, 4 63
Exception task 2 17
Exception task class 2 17
Executable program module 4 124
EXERMO.BR 2 17, 2 42
Extended error reporting 2 18, 2 43, 4 179

F

Fatal Errors 2 42
FBK 4 164
FBK Editor 4 169
Files
*.AWI 4 126
*.AWT 4 126
*.BFU 4 171
*.BR 4 124
*.BRI 4 126
*.BRT 4 126
*.FUB 4 168
*.PRJ 4 154
*.SLB 4 164
*.VLD 4 153, 4 159, 4 161
File selector box 4 30
FIX RAM 2 7
configuring 4 66
delete 4 183
Flash PROM
deleting 4 183
Foot Line Menu 4 25
Debugger for STL and PL2000 ..... 4 146
FBK Editor 4 169
GDM 4 150
LAD Editor 4 80
LAD Debugger 4 141
PV Monitor 4 131
Force Function 4 132, 4 140
activating 4 134, 4 137, 4 142
deactivating 4 134, 4 137, 4 142, 4 183
Function block 4 164
AVT 4 165
local 4 165
Function block Editor 4 169
Function directory (LIBFILES) 4 51
Function Library 1 13, 4 164
Delete 4 166
Help 4 165
Import 4 164
Functions 1 13, 4 164

G

GDM 1 16, 4 57
GDM Synch. 4 163
Graphic Design Method 1 16, 4 57

H

Hardware requirements 3 5
Hardware watchdog 2 41
Help system 4 32
High speed task classes 1 7, 2 12
HS task classes 2 12

I

Idle time 2 11
Idle time tasks 1 7, 2 11

I/O image 2 34
I/O Data transfer 2 34
System B&R 2005 ..... 2 38
System B&R 2010 ..... 2 34
I/O Processor 2 6, 2 34
I/O Type of variables 4 111
Importing function Libraries 4 164
INIT 2 23, 4 180
Init values of variables 4 119
INIT SP 2 32, 4 123
Create 4 123
Delete 4 123
Open 4 123
Initialization subprograms 2 32, 4 123
Initialization value 4 119
Inline Function 4 164
Input box with pop up selection window 4 29
Input image 2 35
Install mode of a task 4 63
Installation of PG2000 ..... 3 6
Interrupt 2 19
Interrupt capable module 2 19
Interrupt task 2 19
Interrupt task class 2 19
IRQ task class 2 19

K

Keyboard commands
Debuggers for STL and PL2000 ..... 4 148
FBK Editor 4 170
GDM 4 162
LAD Editor 4 83
LAD Debugger 4 143
PL2000 Editor 4 104
PV Monitor 4 138
STL Editor 4 93

L

Ladder Diagram 1 13, 4 70
Ladder Diagram Editor 4 74
Ladder Diagram Debugger 1 18, 4 140
LAD 1 13, 4 70
LAD Debugger 1 18, 4 140
Length of variables 4 110
Library 4 164
Line parser 4 124
local 4 108
Local function block 4 165
Long name of variables 4 119

M

Memory areas of a PLC
  User RAM 2 7
  User ROM 2 7
  Dual Ported RAM (DPR) 2 6
  System RAM 2 6
  System ROM 2 7
Memory mode of a Task 4 64
Memory optimization 3 15
Menu bar 4 17
Message line 4 13, 4 15
Message window 4 13, 4 15
Modem 4 184
Module
  Display and Manipulation 4 175
Monitoring functions of PLCSW 1 9, 2 40
  Check sum of B&R Modules 2 40
  Cycle time monitoring 2 41
  Hardware Watchdog 2 41
  I/O Data transfer 2 40
  Stack overflow 2 40
  System structures 2 40
Mouse, use of 4 12
Multiprocessor system 4 59
Multitasking 2 8
  of the I/O processor 2 39

N

New installation of PG2000 ..... 3 6
Non cyclic tasks 1 7, 2 11
Normal error reporting 4 178
Normal task classes 1 7, 2 12
Null ptr check 4 127

O

Object
  Compile 4 124
  Delete 4 131
  Download into user RAM 4 128
  Download in User ROM or FIX RAM 4 130
  Edit 4 69
  INIT SP 4 123
  Optimize 4 156
  Parameters 4 62
  Rebuild 4 160
  Variable Declaration 4 107
Object list 4 154, 4 158
Operating system kernal 2 8
Optimized I/O Tables 4 126
Output Image 2 35

P

Parameters
  Data module and B&R Modules 4 65
  PLC Configuration module 4 66
  Task 4 62
PG.BAT 3 9
PG.INI 4 8
PL2000 ..... 1 12, 4 96
PL2000 Editor 4 99
PL2000 Editor configuration 4 105
PLC Clock
  read 4 182
  set 4 182
PLC Configurator 1 20
PLC Configuration module 2 24, 4 66
PLC directory 4 52
PLC_global 4 108
PLC Layer 4 57
PLC Software 1 6, 2 5
PLC Specific software package 2 8
PLC SW 1 6, 2 5
PLC SW Operating system 1 6, 2 5
PLC Symbol, position 4 149
PLC Tasks 1 7, 2 11
PLCSW 1 6, 2 5
Pop Up Menus 4 26
  LAD Editor 4 82
  PLC Layer 4 154
  PV Monitor 4 136
  Task Layer 4 158
Power on behaviour 2 27
Printing with PG2000, pre requisites 4 12

Priority
Task classes 2 13
Tasks 2 12
Process variable monitor 1 19, 4 132
Process variables 4 107
Delete 4 156
Profiler 1 21
Program files 4 52
Program files directory (PGM) 4 52
Program files path 4 52
Programming Environment 4 13
Programming in C 1 14
Programming System PG2000 ..... 1 10
Project
Copy 4 54
Create 4 49
Delete 4 56
Include in project management 4 55
Open 4 54
Purge 4 56
Project back up copy 4 54
Project data bank 4 51
Processor layer 4 57
Project management 4 13, 4 48
Project name 4 49
Project print out 4 156
Pull Down Menus 4 17
PV (process variables) 4 107
PV Monitor 1 19, 4 132

R

Range Check 4 126
RB Info 4 125
Rebuild (decompile) Info 4 125
RECONFIG 2 24
Replace Function 4 129
RESET 2 24, 4 181

S

Scope of variables 4 108
Selection Field 4 27
Selection List 4 28
SERVICE 2 26
SERVICE Mode 2 20, 2 26
Shovelling the I/O image 2 35
Single processor system 4 59
Source Level Debugger 1 17, 4 139
Standard Functions 4 164
Start behaviour of the tasks 2 32
Start modes of a task 4 64
Starting PG2000 ..... 4 12
Statement list 1 12, 4 84
STL 1 12, 4 84
STL Editor 4 88
STL Editor configuration 4 94
Structure 4 109
Structure element 4 110
Structure variable 4 109
Symbolic names of variables 4 107
Synchronizing the PLC and GDM 4 163
Syntax Checking 4 124
System file CONFIG.SYS 3 8, 3 15
System load 2 16
System manager 2 9
System menu 4 21
System modification 3 15
System RAM 2 6
System ROM 2 7

T

Task builder 4 124
Task classes 1 7, 2 11, 2 12
normal 1 7, 2 12
high speed 1 7, 2 12
Task class utilities 4 174
Task layer 4 57
Task management services 4 172
Task parameters 4 62
Task symbol, place 4 60
Tasks
non cyclic 1 7, 2 11
cyclic 1 7, 2 11
tc_global 4 108
Text entry field 4 28
Title bar 4 13, 4 15
TOTALINIT 2 22, 4 181

U

User FBK 4 167
User RAM 2 7
User ROM 2 7
EPROM 2 7
Flash PROM 2 7
Using the programming system 4 13
with the keyboard 4 16
with the mouse 4 16

V

Variable Declaration 1 15, 4 107
Data type 4 109
I/O Type 4 111
Init value 4 119
Length 4 110
Long name 4 119
Scope 4 108
Symbolic name 4 107
Variable editor 4 107
Variable editor configuration 4 120
Variable list 4 153, 4 159, 4 161

W

Windows 4 19
Changing windows 4 23
Closing 4 21
Elements 4 20
Maximizing 4 23
Minimizing 4 23
Moving 4 22
Resizing 4 22
Scrolling contents 4 24
System menu 4 21