PG2000
PROGRAMMING SYSTEM
QUICK START

Version: **1.0 (March 1996)**
Model No.: **MASYS2PGQS-E**

This manual is valid for **Version 2.23**
of the PG2000 Programming System

We reserve the right to change the contents of this manual without warning. The information contained herein is believed to be accurate as of the date of publication, however, Bernecker and Rainer Industrie-Elektronik Ges.m.b.H. makes no warranty, expressed or implied, with regards to the products or the documentation contained within this book. Bernecker and Rainer Industrie-Elektronik Ges.m.b.H. shall not be liable in the event of incidental or consequential damages in connection with or arising from the furnishing, performance or use of these products.

1. Introduction / Installation

2. The First Steps

3. Operation of PG2000

A. Model Numbers / Delivery Contents

B. Glossary

Table of Contents

Chapter 1 - Introduction / Installation 7

1. Information Concerning this Manual 9
2. Where do I find the answers to my questions? 10
   - 2.1. Online Help 10
   - 2.2. This Manual 10
   - 2.3. Information Concerning Current Software Version 10
   - 2.4. Additional Documentation 10
   - 2.5. B&R Support 11
3. Installation of PG2000 12
   - 3.1. System Requirements for PG2000 12
   - 3.2. PG2000 New Installation 13
   - 3.3. PG2000 Update 14
   - 3.4. Installation on a Network Server 15
   - 3.5. Editing your CONFIG.SYS 16
4. The First Program Start-Up 18

Chapter 2 - The First Steps 21

1. Preliminary Notes 23
2. Project, Task, Program, ...? 24
3. What is my goal? 27
   - 3.1. General Formulation of the Application 27
   - 3.2. Structured Formulation of the Application 27
   - 3.3. Formulating the Application in PL2000 28
4. How do I create a project? 29
5. My First Task (PL2000 Program) 31
   - 5.1. GDM 31
   - 5.2. Insert Task Symbol 34
   - 5.3. Enter Program Code 35
   - 5.4. Variable Declaration 38
   - 5.5. Creating a Task 42
6. How do I Connect PC and PCC? 44
   - 6.1. General Information 44
   - 6.2. Making the Connection 45
   - 6.3. Interface Driver 46
7. How do I transfer a task to the PCC? 48
8. How do I influence tasks in the PCC? 49
   - 8.1. PV Monitor 49
   - 8.2. Stopping, Starting, Removing Tasks 52
   - 8.3. Starting/Stopping Task Classes 53
9. Ladder Diagram - LAD 54
   - 9.1. Example 54
   - 9.2. Library Import 54
   - 9.3. Entering a Program 55
   - 9.4. Testing a Program 58
10. Statement List - STL 59
    - 10.1. Example 59
    - 10.2. Entering a Program 60
    - 10.3. Testing a Program 61

Chapter 3 - Operation of PG2000 63

1. The PG2000 Screen 65
   - 1.1. Project Management 65
   - 1.2. Programming Environment 66
2. Mouse / Keyboard Operation 67
   - 2.1. Mouse 67
   - 2.2. Keyboard 67
3. Menus 70
4. Working with Windows 71
5. Dialog Boxes 76
6. File Selection Box 79
7. Help System 81
8. Project Management 83

Appendix A - Model Numbers

Delivery Contents 85

1. Delivery Contents 87
2. Additional Model Numbers 88
   - 2.1. PG2000 Individual Components 88
   - 2.1. Documentation 88

Appendix B - Glossary 89

Index 99

CHAPTER 1
INTRODUCTION / INSTALLATION

1. Information Concerning this Manual

The **PG2000 Programming System** is a high performance software package for programming the B&R SYSTEM 2000 controller generation.

PG2000 can be used to program each **PCC** – **P**rogrammable **C**omputer **C**ontroller – in the B&R SYSTEM 2000. The following components can be programmed with PG2000:

- PCC CPU (**C**entral **P**rocessing **U**nit)
- MP (**M**ulti**P**rocessor)
- IP (**I**ntelligent I/O **P**rocessor)

There are three programming languages to choose from:

- **PL2000** - Structured text oriented high level language that conforms to IEC 1131
- **LAD** - Ladder Diagram
- **STL** - Statement List

What can I find in this manual?

This manual is an introduction to the PG2000 Programming Software. It is designed to give you an overview and help you to get started programming the B&R SYSTEM 2000 controller generation.

Chapter 1 Introduction / Installation provides you with all of the important information required to install the PG2000 Programming Software.

Chapter 2 The First Steps offers you the possibility to become familiar with the functionality of PG2000 in practice using examples. If you work your way through this chapter, you will gain experience with PG2000 and gather valuable information.

Chapter 3 Operation of PG2000 contains basic information concerning the operation of PG2000. How is the screen structured? How do I use the mouse and keyboard? How do I work with windows?

2. Where do I find the answers to my questions?

2.1. Online Help

Online help is an **important information source** for questions regarding the PG2000 programming system. In order to call online help, you can either press the [F10] function key or select *Help* from the *Help* menu.

Online help is context sensitive. That means the corresponding help text will be called dependent on the context in which PG2000 is found.

2.2. This Manual

The *PROGRAMMING SYSTEM PG2000 Quick Start Manual* contains basic information concerning PG2000 and describes the most used features. In the glossary, you can find explanations for important terms that are used when dealing with a control system from the B&R 2000 family.

2.3. Information Concerning Current Software Version

This manual is valid for version 2.23 of the PG2000 programming system. Information that was not yet available when this manual was printed can be found in the README file. These files are copied to the README subdirectory when PG2000 is installed.

2.4. Additional Documentation

Additional Useful Documentation:

- ❑ Programming Languages User's Manual
- ❑ Advanced Programming Manual
- ❑ Library Reference Manual
- ❑ System Software Reference Manual
- ❑ System Configurator and Profiler User's Manual

2.5. B&R Support

B&R is ready and willing to help you with facts and advice. If you have questions that are not answered in our documentation, please contact your support partner at B&R.

Mailbox

Newest program versions, drivers, libraries, etc. can be found in the B&R Mailbox. You can also leave messages for B&R or B&R employees and inquire about problem analysis.

The B&R Mailbox can be reached with the following numbers:

1.) +43 (7748) 6586-623

2.) +43 (7748) 6586-86

3. Installation of PG2000

In this section, you will find all of the information required to install PG2000.

Read the following notes before beginning the installation.

- Before the installation, make sure your PC meets the criteria listed in section *3.1 System Requirements for PG2000*.
- Make a backup copy of the installation diskette before the installation. Additional information concerning copying diskettes can be found in your MS DOS manual.
- If you want to install PG2000 on a network server, follow the steps described in section *3.4. Installation on a Network Server*.
- The installation program can be operated either with a mouse or the keyboard. If a mouse driver is not installed, the installation program indicates this with a message.

3.1. System Requirements for PG2000

The computer where PG2000 is to be installed must meet the following minimum requirements:

- 386 SX processor, 25 MHz, at least 2 MBytes RAM
- Operating System:

  MS DOS Version 3.3

  MS DOS Version 5.0 or higher

  DR DOS Version 6.0 or higher
- 3½" Disk Drive (1.44 MByte)
- Hard Disk, Free Memory for Installation:

  Standard Installation: at least 8 MBytes

  Full Installation: at least 9 MBytes

- ❑ 640 KBytes Main Memory (at least 550 KBytes free)
- ❑ Online Interface to Controller:

  Free serial interface (COM1 or COM2) or

  PROFIBUS card

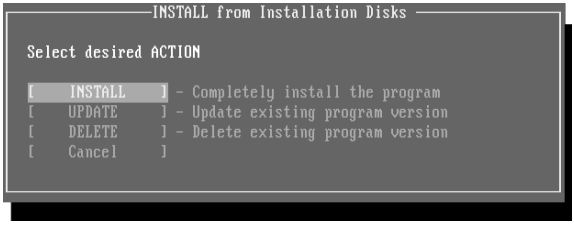
B&R also recommends:

- ❑ At least 4 MBytes RAM
- ❑ Mouse operation (optional)

3.2. PG2000 New Installation

1. Insert the first installation diskette in the 3½" diskette drive.
2. In DOS, go to the drive containing the installation diskette and start the PG2000 installation by entering **INSTALL**.
3. First, the installation program informs you of the PG2000 version that will be installed. You can cancel and exit the installation program here.
4. Select the desired installation mode:

Installation Mode:





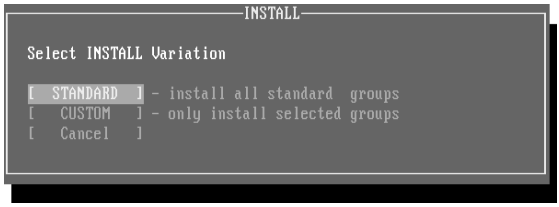
CAUTION, when you select **INSTALL**!

If you already have an older version of PG2000, select the **UPDATE** installation mode (*3.3 PG2000 Update)* so that your user specific settings are not lost. Otherwise the existing PG2000 is deleted and all user specific settings are lost!

5. After you have selected **INSTALL**, enter the desired target directory including drive specification. You can enter a path with more than one subdirectory if desired. All directories that do not exist will be created automatically.

6. Select the installation variation:

Installation Variation:





RECOMMENDED INSTALLATION: STANDARD

Use the CUSTOM variation if ...

... your controller has an old operating system software version (e.g. version 1.05),

... you want to save space on your hard drive by not installing certain parts of PG2000.

7. After selecting the installation variation, PG2000 will be installed in the given directory.

3.3. PG2000 Update

When updating, all sections of PG2000 that are already installed are compared with the new version. Only the files from the installed version that have an old version number are replaced with the new version.

Differences Compared to New Installation:

- All user specific settings from earlier PG sessions are kept. They are not overwritten.
- The installation program does not require an entry for the target path, it searches for installed PG2000 versions on your hard drive. The search can be limited to certain drives.

3.4. Installation on a Network Server

If you are installing PG2000 on a network server, pay attention to the following points:

Network Driver

In the workstation (PC that is connected to the server and calls PG2000 via the server), you should only install the network drivers required for pure workstation operation. Components such as EMAIL, MESSAGE, CHAT, REMOTE etc. should not be installed so that enough memory is available for PG2000.

File Locking

If you manage the data bank or project files on a shared drive, the server has to support file locking.

User Specific Settings

In order for several users to be able to use PG2000 installed on the server with user specific settings for each user and separately managed projects, PG2000 must be given the directory where all user specific data is saved during start-up. This directory is entered with a command line option:

```
pg /N X:\pfad
```

If e.g. all user specific data is saved on the local hard drive in directory "C:\PG2000\USER", enter the following at the DOS command line to start PG2000:

```
pg /N C:\PG2000\USER
```



Make sure that the directory given EXISTS!

Naturally, you can also enter a directory on the network server.



User specific data on shared drives:

Make sure that no other user is using the directory for this purpose.

Temporary Files

During operation, PG2000 creates temporary files that are deleted again when the program is ended. These files are saved in the directory pointed to by the environmental variable TMP.



Set the TMP environmental variable in network operation!

For network operation, make sure that the environmental variable TMP points to a directory that is not used as temporary directory by any other user.

Note:
Additional information about setting and deleting environmental variables can be found in your DOS Manual.

3.5. Editing your CONFIG.SYS

Check the settings in the CONFIG.SYS file before starting PG2000 the first time. You can open, view and edit this file with any ASCII editor (e.g. with the editor EDIT.COM that is delivered with MS DOS).

FILES and BUFFERS

The entries FILES and BUFFERS in CONFIG.SYS should be assigned at least the following values:

```
FILES=40
BUFFERS=20
```

If these entries are not available, insert them with the ASCII editor.

Country Setting, Character Set

Help text, error messages and menu text can be shown in German and English. The language is selected by PG2000 automatically depending on the country code. The country code is set with COUNTRY in CONFIG.SYS:

```
COUNTRY=xxx,yyy,COUNTRY.SYS
```

xxx .... Country code
yyy .... Code for character set

If you want the standard dialog language for PG2000 to be German, then enter one of the following country codes for xxx:

041 .... Switzerland / Liechtenstein
043 .... Austria
049 .... Germany

The dialog language for PG2000 is English for all other country codes.

Additionally, the character set for the screen output is set with yyy. Here, enter the value 437 (USA - character set), then all ASCII characters incl. the semi-graphic characters are displayed correctly.

Note:
Additional information about the COUNTRY command, country codes and character sets can be found in your DOS Manual.

Memory Management

In order to use the working memory optimally, you should install a memory manager. Additional drivers for memory management are delivered with MS DOS (e.g. HIMEM.SYS and EMM386.EXE).

With the help of the memory manager, you can load drivers and memory resident programs in the upper memory area (UMA). This technique gives you more free main memory for the operation of PG2000.

Mouse

In order to operate PG2000 with a mouse, you need ...

... a mouse,

... a free serial connection for the mouse,

... a driver for the mouse (this software is normally delivered with the mouse; additional information can be found in your mouse manual).

You can also use other pointing devices instead of a mouse. However, the term **mouse** will be used in this manual.

4. The First Program Start-Up

Check the following points again before starting PG2000:

- ❑ Was the installation program ended with the message "*The installation was completed successfully!*"?
- ❑ Did you check the settings in the CONFIG.SYS file (see section *3.5. Editing your CONFIG.SYS* in this chapter).
- ❑ If you want to use a mouse, it has to be connected to the respective interface and the mouse driver must be loaded.

PG2000 Directory

During installation, you had the possibility to enter the drive and directory where PG2000 was to be installed. The directory "C:\PG2000" is given as default.

Note:

When referring to the directory where PG2000 is installed in this manual, the term "*PG2000 Directory*" is used and in examples the directory "C:\PG2000" is used.

Executable Program

The PG2000 directory contains a subdirectory called *PG2000.EXE* where the executable program *PG.EXE* that is used to start the programming system is found.

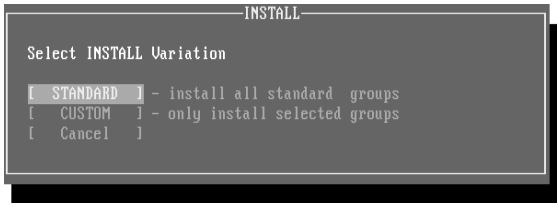
Starting PG2000

Go to the directory PG2000.EXE using the respective DOS command and start the programming system. e.g.:

```
c:                          => Change drive
cd \pg2000\pg2000.exe       => Change directory
pg                          => Start PG2000
```

6. Select the installation variation:

Installation Variation:





RECOMMENDED INSTALLATION: STANDARD

Use the CUSTOM variation if ...

... your controller has an old operating system software version (e.g. version 1.05),

... you want to save space on your hard drive by not installing certain parts of PG2000.

7. After selecting the installation variation, PG2000 will be installed in the given directory.

3.3. PG2000 Update

When updating, all sections of PG2000 that are already installed are compared with the new version. Only the files from the installed version that have an old version number are replaced with the new version.

Differences Compared to New Installation:

- All user specific settings from earlier PG sessions are kept. They are not overwritten.
- The installation program does not require an entry for the target path, it searches for installed PG2000 versions on your hard drive. The search can be limited to certain drives.

CHAPTER 2
THE FIRST STEPS

1. Preliminary Notes

Example Program

For the first steps with PG2000, we will write a small example program in the high level language PL2000.

Hardware

In order to test this example on a controller, naturally you require the respective hardware:

A controller from the B&R 2010 or 2005 family with CPU, digital output module, power supple module and base plate.

Note:
Additional information concerning the hardware in the B&R SYSTEM 2000 controller generation can be found in the *B&R 2000 Hardware User's Manual (MASYS2HW-E).*

If you work through the example in this chapter, you will get some insight as to the capabilities and structure of the PG2000 programming system.

Recommendation

The *B&R SYSTEM 2000 Programming Languages Manual* (MASYS2PLM-E) is an important manual when beginning to work with PG2000 and programming in one of the three programming languages:

- After you have worked through the example in this chapter, you can find other examples and application possibilities for practice and to help you increase you knowledge in the *Programming Languages Manual*.
- The *Programming Languages Manual* should allow you to learn the three programming languages for the B&R SYSTEM 2000 quickly and using practical examples.
- Additionally, you can find answers to questions regarding the syntax of the programming language, examples, programming techniques etc in the *Programming Languages Manual*.

2. Project, Task, Program, ...?

In this section, a few basic terms will be explained for PG2000 and B&R SYSTEM 2000 controller generation. Detailed information concerning the individual terms can be found in both of the following manuals:

- B&R SYSTEM 2000 Programming Languages Manual (Chapter 5 The Real World)

- B&R SYSTEM 2000 System Software Reference Manual

Previously, a single program was written to create an automation solution that took care of the entire application. This program was then transferred to the controller and processed cyclically. This is also possible with a B&R 2000 controller but the advantages of multitasking are not used.

Multitasking

This new technique allows individual application tasks to be processed parallel to each other. The individual programs are not actually executed at the same time. They continually share CPU time by switching from one program to another.

Program

A program represents a section of the application and consists of the individual commands, requests and function calls. Each program can also be divided into an initialization section and a main section whereas the initialization section is only executed once when the program is started. The main section is executed cyclically in a defined time pattern.

Variable Declaration

Variables are used in each program. You have to declare each variable, however, declaration is not a part of the program. Variable declaration defines the hardware reference (e.g. memory, input and output modules, temperature measurement cards, ...). Since program and variable declaration are separate, it is possible e.g. to move a PCC input from one module to another without having to change the program code. Only the variable declaration has to be changed.

Task

The combination of program and variable declaration create a task that can be transferred to a PCC CPU. Several tasks can be running on a PCC CPU at the same time and exchange data with each other. The tasks are grouped with respect to their priority and so-called task class.

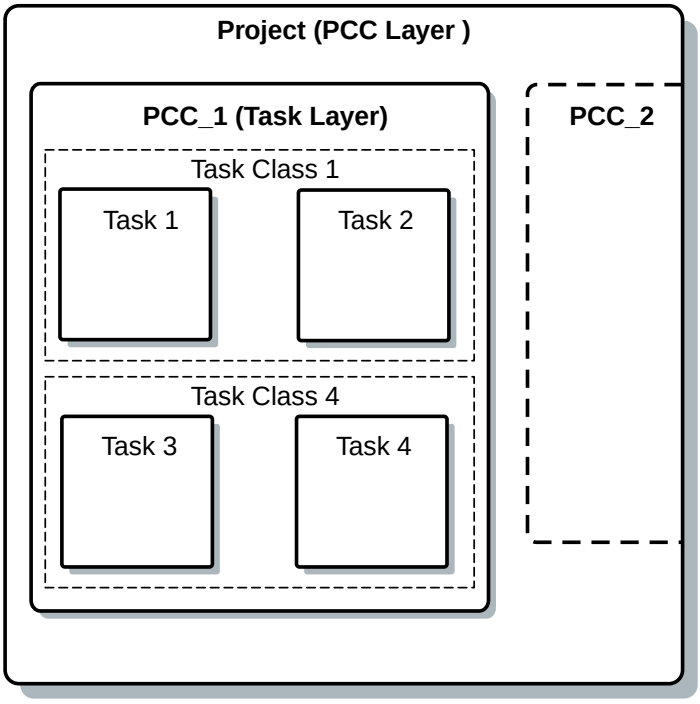
Task Class

Each task must be assigned to a certain task class. A cycle time is defined for each task class which can be set by the user. Each task is executed once per cycle.

Project

All tasks in all task classes add up to a project. The following diagram shows this in graphical form:

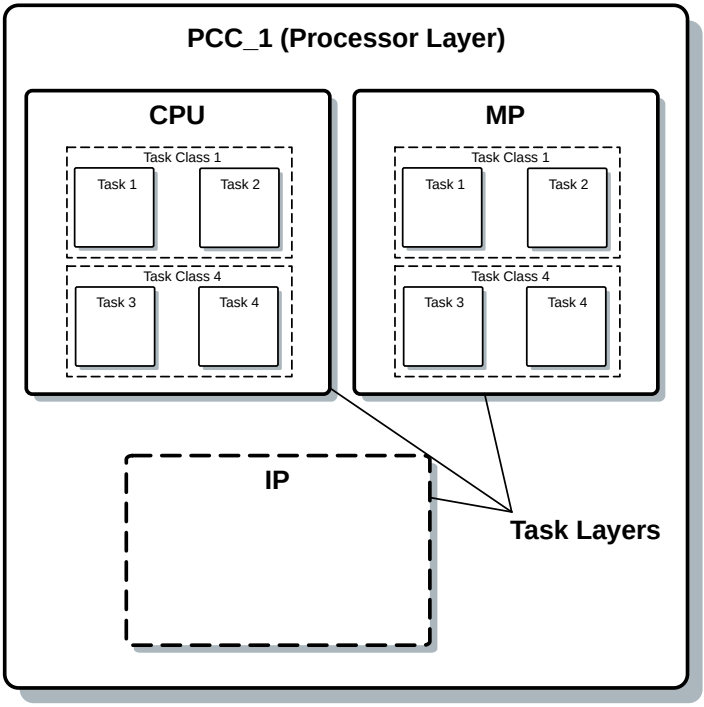
Diagram:



As shown in the diagram, a project can consist of more than one controller! The above diagram shows layer PCC_1 in the task layer. All tasks in PCC_1 are found in this layer.

For most applications, the tasks are found on a PCC processor – the PCC CPU. However, it is possible to integrate other intelligent modules (e.g.: multiprocessors or intelligent I/O processors) in a controller. In this case the individual tasks are no longer in layer PCC_1, they are in the corresponding task layer for the respective processor. The function of layer PCC_1 now corresponds to that of the processor layer and not the task layer:

Diagram:



Take a good look at this diagram because project execution with PG2000 takes place based on this structure.

3. What is my goal?

Projects in the real world are much more complex than the example in this chapter. However, the basic preliminary considerations are similar for each project and can also be shown with this example.

3.1. General Formulation of the Application

Before you begin programming, formulate the application in your own words.

Example

The weight is to be checked continually when loading a material lift. If the weight exceeds the maximum limit, a light is to be switched from green to red and a lockout is to be activated that prevents the lift from starting unintentionally. The light is switched to yellow when the weight is reduced below the maximum value. The lockout is deactivated and the light is switched to green after a 30 second delay.

3.2. Structured Formulation of the Application

The second step is to formulate the application in a more structured way, so-called **pseudo code**:

```
Initialization during start-up
    Activate lockout
    Switch light to red
    Set maximum weight (100 kg)
Compare material and maximum weight
    If too heavy:
        Switch light to red
        Activate lockout
    If not too heavy:
        If light is red:
            Switch light to yellow
            Set time counter to 30 seconds
        If light is yellow:
            If 30 seconds has not yet passed:
                Time counter counts down
            If 30 seconds has passed:
                Switch light to green
                Deactivate lockout
```

3.3. Formulating the Application in PL2000

The application looks like this in the high level language PL2000:

Initialization Section:

```
lockout      = 1
red_light    = 1
yell_light   = 0
grn_light    = 0
max_weight   = 100
```

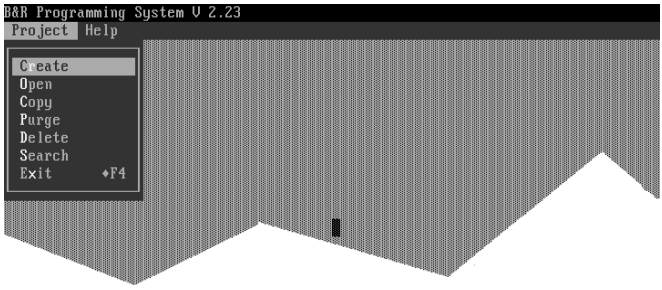
Main Section:

```
if act_weight > max_weight then
     lockout    = 1
     red_light  = 1
     yell_light = 0
     grn_light  = 0
else
     if red_light = 1 then
         red_light   = 0
         yell_light  = 1
         delay       = 3000     ; corresponds to 30 seconds
                                ; if cycle time is 10 msec
     else if yell_light = 1 then
         if delay > 0 then
             delay = delay - 1
         else
             yell_light  = 0
             grn_light   = 1
             lockout     = 0
         endif
     endif
endif
```

4. How do I create a project?

A project must be created in order to be able to enter a program. Start PG2000 (see *Chapter 1 Introduction / Installation* in section *4. The First Program Start-Up*). The first time the program is started, the project management menu appears:

Project Management



Create a new project by clicking on the **Create** entry in the **Project** menu with the mouse. A dialog box is shown:



Enter e.g. the following data:

```
Path           : d:\pg_proj
Project Name   : quickst
Description    : Quickstart - Example
PCC_Name       : PCC_1
```

A directory that doesn't exist can also be entered as **Path**. PG2000 creates the directory automatically. You can also create the project on a different drive.

Recommendations

Do not create projects in the PG2000 directory. This keeps projects (data) and the programming system (program) clearly separated from each other.

You can enter any name (max 8 characters) for the **Project Name**. All project specific data is saved in the directory that was created from *Path* and *Project Name*:

```
Path \ Project Name.PGP
```

You can enter a **Description** so that the project can be explicitly identified (max 30 characters).

You must enter a **PCC_Name** (max 8 characters). When creating a project, a PC Object is created automatically using this name.



Pay attention to syntax for **Path**, **Project Name** and **PCC_Name**!

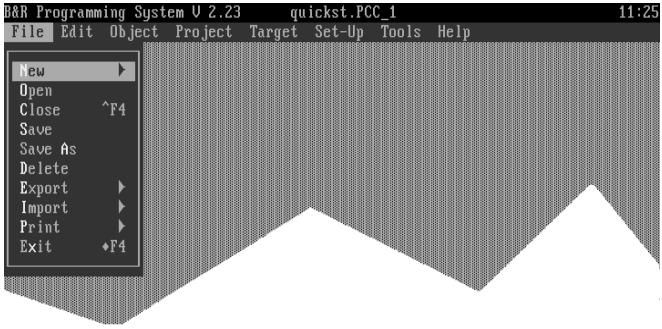
The syntax must correspond to the standard DOS conventions for directory names (further information can be found in your DOS manual).

Acknowledge you entries by ...

... clicking [OK] with the mouse, or

... selecting [OK] with the cursor key and pressing ENTER.

Respond to the next two windows with [Yes]. The project will be created and opened:



As you have noticed, the PG2000 menu has changed. All functions to required to edit you project are now available.

The project name and PCC are shown in the title line (top line).

5. My First Task (PL2000 Program)

You have created you first project, now you can edit it. We will work through the following steps in this section:

- ❑ Introduction to the GDM
- ❑ Inserting a Task Symbol
- ❑ Entering and Correcting Program Code
- ❑ Variable Declaration
- ❑ Creating / Compiling a Task

5.1. GDM

Now we will introduce the GDM so you can edit the project. The GDM is the most important tool for editing a project. It has the following features:

- ❑ It shows the entire project in a clear graphic form.
- ❑ It allows the project to be constructed in a structured manner.
- ❑ It is helpful for planning and analyzing projects for automation applications.
- ❑ The individual tasks are connected to each other with the GDM.
- ❑ Project maintenance is made easier.
- ❑ GDM – Graphic Design Method – makes PG2000 more user friendly.



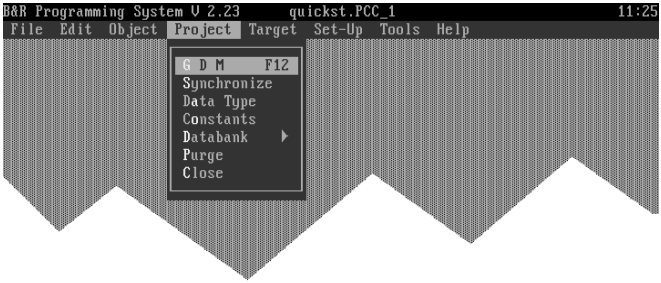
Edit all sections of the project with the GDM !

Open the GDM

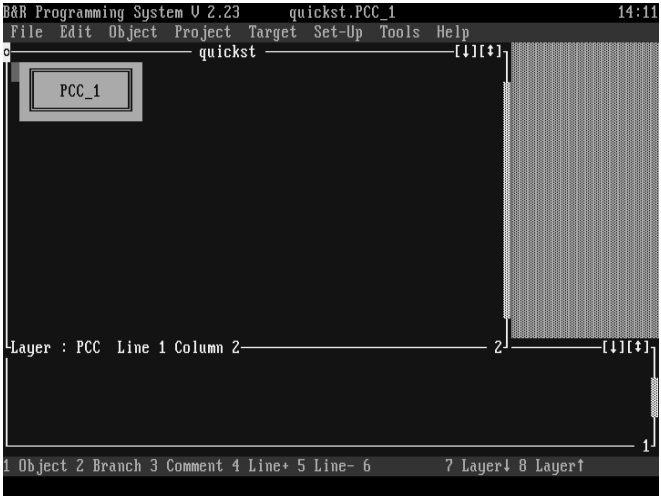
Open the GDM by ....

... pressing the F12 function key, or

... calling the menu function *Project=>GDM*



The GDM will be opened. A window appears with all PCCs in the project QUICKST graphically displayed as symbols. When creating a project, a PCC Object is automatically inserted in the PCC layer. This PCC Object has the name **PCC_1** which was entered when the function *Project=>Create* was called:



A message line is shown on the bottom line of the screen. Here, you can see the current assignments for the most important function keys at any time.

You can move between the various layers in a project with two keys:

F7 a layer deeper
(is only possible if a symbol is selected)

F8 a layer higher

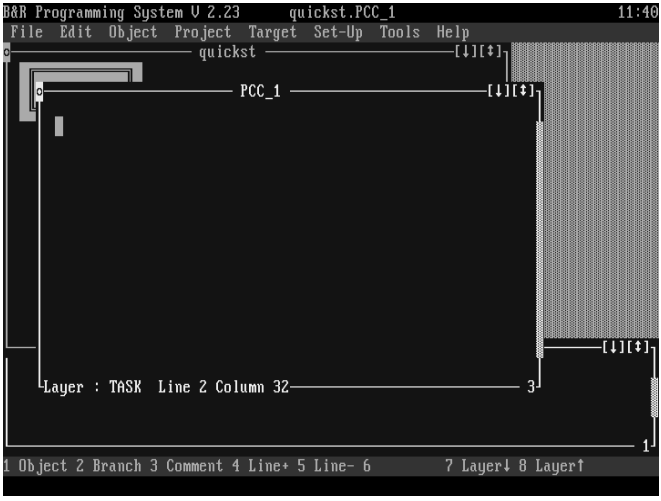
Task Level

Go to the task layer for PCC_1:

1. Select symbol **PCC_1**
   (with cursor key or mouse)

2. Press the **F7** function key

The task layer for PCC_1 will now be shown:



Practice

Switch between the layers with the **F7** and **F8** keys to get a feeling for the use of these keys.

There are also other ways to switch between windows:

- Click on a window with the mouse (not on the window frame)
- Go directly to a window using the key combination [Alt]+[Window Number]. The window number is shown in the bottom right of the window frame.
- Cycle through the windows using the key combination [Alt]+[F5].
- Select the desired window from the window list (called from *Tools - Window List*).

You will recognize that the windows are not closed when you switch between layers. If you want to close a window because the screen is becoming to crowded, you can do this in a few different ways:

1. With the key combination **[Ctrl]+[F4]** (or **[Strg]+[F4]**)

2. With the menu command *File => Close*

5.2. Insert Task Symbol

Activate/open the task layer for **PCC_1**. Insert a task symbol as follows:

1. Move the cursor to a free spot and press the **F1** key (see message line).

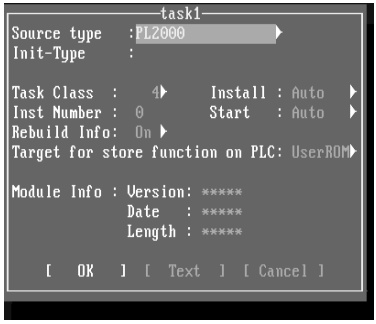
2. A mask appears in the task layer in the size of a task symbol. You can position the task symbol by moving this mask within the window using the cursor keys.

3. A help message will be shown on the last line containing the special key functions for this mode, such as e.g. Move the mask with the CURSOR keys.

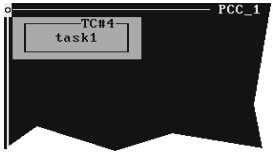
4. Press the ENTER key and select the entry *Cyclic Task* from the list.

5. Enter a name for the task; e.g.: *task1*

6. The task parameters will be set in the next dialog. Only change the default settings if the task parameter is different to the ones shown below.



7. Close the dialog with [OK]. The task symbol appears in task layer **PCC_1**:



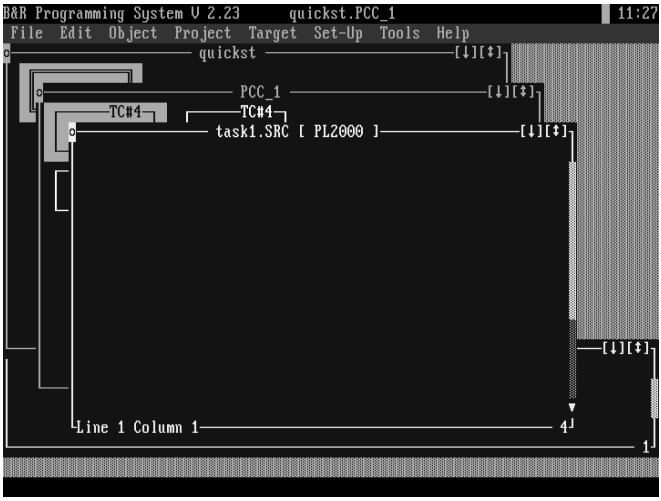
This symbol is the graphic representation for a tasks. Program code and variable declaration for this task still have to be set.

5.3. Enter Program Code

The next step is to enter the program code for this task. To enter the program code from section *3.3. Formulating the Application in PL2000*, simply go one layer deeper as you learned before for the GDM. The *PL2000 Editor* will be called automatically since you selected **PL2000** as source type in the task parameters.

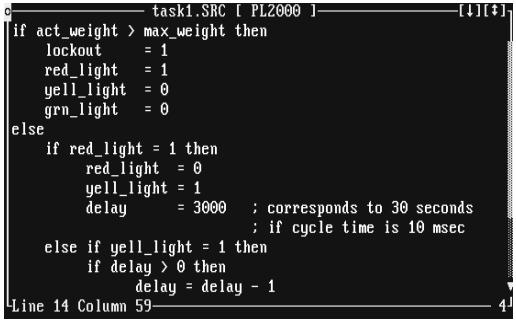
Move the cursor to the task symbol *task1* in the task layer **PCC_1**. Call the PL2000 Editor by pressing [F7]:

PL2000 Editor



Enter Main Section of the Example Program

The **PL2000-Editor** can be compared to an ASCII editor. Try to enter the main section from our example program shown on page 28 in the editor window. The PL2000 Editor functions similarly to most text editors:



Additional information concerning the operation of the editor can be found in *Chapter 3 Operation of PG2000*.

Enter Initialization Section of the Example Program

The initialization section of a program is entered separately. Select *Object=>InitSP* to open the window for the entry of this program section. An initialization section can be written either in PL2000 or STL. Select the PL2000 programming language from the dialog window and then enter the program code:



Save your entry with *File=>Save*. A syntax check is made with each save and all errors are shown in the message window.

Syntax Check when Saving

A syntax error check is made when saving the PL2000 code in the InitSP window. The errors are shown in the message window:



Error Correction

If you have entered the program code as shown in the initialization section picture on the previous page and without empty lines, you should get the same error message as shown in the syntax check picture on the previous page (Syntax error in line 1).

Correct the error by placing the cursor in the respective line. Move the cursor with ...

... the cursor keys; note the bottom window frame where the current line is shown.

... the menu command *Edit=>Go To*; you can jump to a certain line with this command.

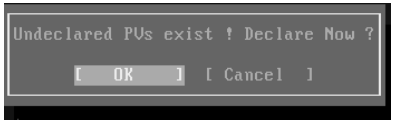
There is an initialization in line 1. The variable "`lockout`" is assigned a value. An error description is given in the message window: "`Unknown Object`".

This variable is invalid since it contains an invalid character. Change the variable to "`lockout`". Try to make the change with the search/replace function.

Then close the InitSP window. Another acknowledgment window appears before the window is closed asking if the changes should be saved. Respond with [Yes].

If there are a syntax errors in the main section. These errors can also be corrected as described.

Save corrections and changes with *File=>Save*. If no more errors are present, the following message appears:

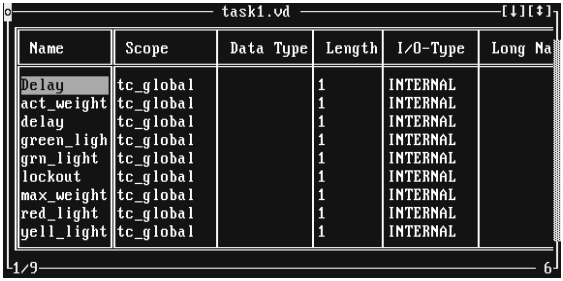


Respond with [OK]. The window for variable declaration is called.

5.4. Variable Declaration

The variable declaration can also be called directly from the PL2000 Editor: Select *Object=>Variable*.

All variables used in the program *task1* are shown in the window:


task1.vd

Name	Scope	Data Type	Length	I/O-Type	Long Na
Delay	tc_global		1	INTERNAL	
act_weight	tc_global		1	INTERNAL	
delay	tc_global		1	INTERNAL	
green_ligh	tc_global		1	INTERNAL	
grn_light	tc_global		1	INTERNAL	
lockout	tc_global		1	INTERNAL	
max_weight	tc_global		1	INTERNAL	
red_light	tc_global		1	INTERNAL	
yell_light	tc_global		1	INTERNAL	

1/9 6

Variable declaration is carried out as follows:

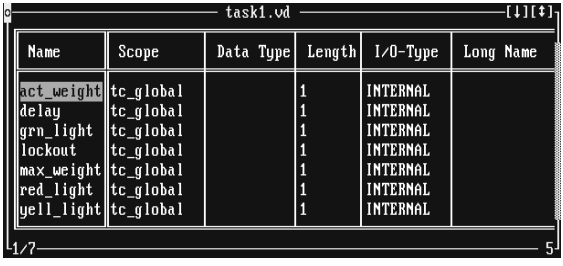
- Check all variable names! Spelling and capitalization errors can create unwanted new variables. These two variable exist in the picture above:

  `Delay` <===> **`delay`**

  This example shows the one of the most common reasons for errors which is not recognized by the syntax check since the difference between small and capital letters is taken into consideration for variables. The variables `Delay` and `delay` are therefore considered as two different variables.

  Before you continue with the declaration, correct all spelling and capitalization errors in the program code. Close the variable declaration when making corrections.

- You need some more information in order to be able to carry out the next step; the declaration of all variables. The meaning of the elements in the variable declaration are listed in the following section:


task1.vd

Name	Scope	Data Type	Length	I/O-Type	Long Name
act_weight	tc_global		1	INTERNAL	
delay	tc_global		1	INTERNAL	
grn_light	tc_global		1	INTERNAL	
lockout	tc_global		1	INTERNAL	
max_weight	tc_global		1	INTERNAL	
red_light	tc_global		1	INTERNAL	
yell_light	tc_global		1	INTERNAL	

1/7 5

Name: Name of the variables as entered in the program code.

Scope: The scope determines if the variable e.g. is only valid in a task, throughout the task class or in the entire PCC.

Data Type: The data type defines the characteristics of a variable e.g. the number range or the precision of the number saved in the variable.

Length: An array is defined if the length is larger than 1, that means a number of variables with the same data type.

I/O Type: Determines if the variable is an input/output on a module or a variable in the memory in the CPU.

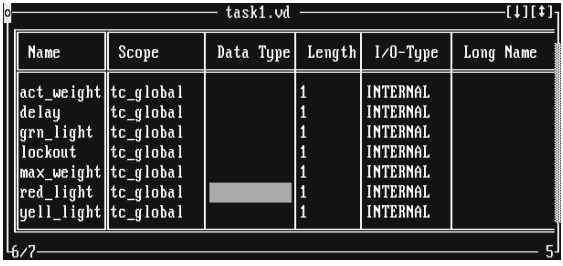
Long Name: Can be used for comments.

Additional and detailed information concerning the elements of the variable declaration can be found in the *B&R SYSTEM 2000 Programming Languages Manual*.

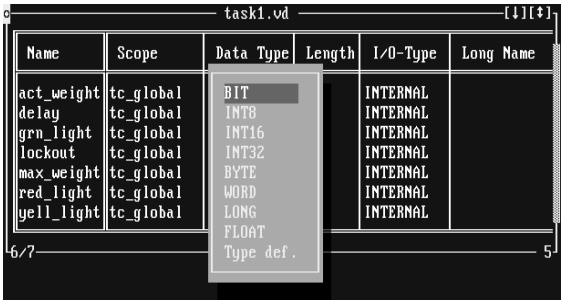
- Declaration of the variable `red_light`:

A digital output is to be assigned to this variable - channel 1 on a digital output card. Follow the steps as described:

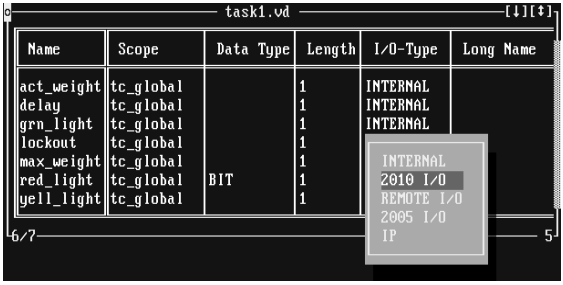
1. Move cursor to the row with *red_light* and the column with *Data Type*:



2. Press the space bar and select the data type **BIT**:

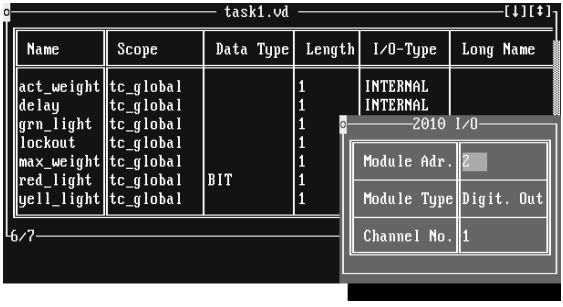


3. Move the cursor to the I/O Type column and press the space bar. Select the respective type from the list:



The variable `red_light` is to be assigned to an output on a digital output card. Selecting I/O Type **2010 I/O** or **2005 I/O** informs PG2000 that the output is found on a module inserted in a controller from the B&R 2010 or B&R 2005 family. Acknowledge the selection with the ENTER key.

4. The next dialog defines ...
   ... where the module is inserted,
   ... the module type,
   ... which channel (output) is to be accessed.



Module Adr.: On 2010 modules, the module address is shown on the status display. On 2005 modules, the module address is dependent on the slot. Additional information can be found in the *B&R 2000 Hardware User's Manual*.

Module Type: Select "Digit. Out" for a digital output card.

Channel No.: Enter the number of the output the variable is to access.

The variable declaration for `red_light` is complete after this step.

- Variable declaration for `max_weight`:

  This variable contains the value that represents the maximum allowed weight. According to our example, it is to contain the value 100. The data type **BYTE** (number range 0 - 255) is sufficient for this value.

  This value is found in the internal memory on the CPU. Select the I/O Type **INTERNAL**.

- Declaration for the rest of the variables:

  All INTERNALs:

Name	Scope	Data Type	Length	I/O Type
act_weight	tc_global	BYTE	1	INTERNAL
max_weight	tc_global	BYTE	1	INTERNAL
delay	tc_global	BYTE	1	INTERNAL

  All Digital Outputs:

Name	Scope	Data Type	Length	I/O Type
yell_light	tc_global	BIT	1	2010 I/O
grn_light	tc_global	BIT	1	2010 I/O
lockout	tc_global	BIT	1	2010 I/O

Place all outputs on the same module. That means the module address and module type are identical to the declaration for red_light. Enter e.g. the following channel numbers for the variables:

Name	Channel Number
yell_light	2
grn_light	3
lockout	8

- ❑ Save the variable declaration with *File=>Save* and close the window.

5.5. Creating a Task

After you have entered the program code and declared all variables, you can create a task. This task can be transferred to the PCC and executed.

Creating a tasks will be called **COMPILING**. Select *Object=>Compile* to create the task. The current object will be compiled:

Current Object	The following tasks are compiled:
Object in the PCC layer	All tasks on the PCC
Object in the processor layer	All tasks on the processor (PCC CPU, multiprocessor or intelligent I/O module)
Object in the task layer	The current task
Open editor (PL2000, LAD or STL)	The current task

When compiling, the program code (main and initialization section) and the variable declaration are made into a B&R module. A B&R module is a file that can be transferred to the PCC with another command and which will also be *understood* by the processor in the PCC.

For the user, nothing changes in the GDM. A task object consists of the program code, variable declaration, task parameters and the B&R module. Dependent on the function called, either the program code will be called (with [F7 layer↓]) or the B&R module will be transferred to the controller (with *Object=>RUN*). This added advantage of the GDM simplifies the operation of PG2000 and you can concentrate on the important matters without having to worry about managing various files.

In order to create a task from the program code and variable declaration, either the PL2000 Editor with example program *task1* must be open or the symbol *task1* must be selected in the task layer.

Select *Object=>Compile* or press the [F9] key. The task is compiled. A report for this procedure is given in the message window and error messages are also given.

If you declared the variables as described previously, the following error message will be given:



Check the line in the program code that is shown in the message window. You will find the following assignment:

```
delay = 3000
```

According to the error message, an "`Incorrect Data Type`" was given here. The variable `delay` was given the data type **BYTE** in the variable declaration. However, the value 3000 lies outside the number range for a BYTE (0 ... 255). Therefore, the data type must be changed:

- ❑ Call the variable declaration.
- ❑ Change the data type for `delay` to **WORD**. The number range (0 ... 65535) is sufficient in this case.
- ❑ Save and close the variable declaration.
- ❑ Compile *task1* again. The message window should contain the following or a similar output:

```
*** Project Processing PCC > PCC_1 < Processor > CPU < ***
Processing file task1.src
      :
      :
**** TASK BUILDER   Ok.
```

If the *Compile* function is completed with the message "`TASK BUILDER Ok.`", a task was created that can be transferred to the PCC.

6. How do I Connect PC and PCC?

You've created a task that can be transferred to a PCC. However, before you transfer the example program, you have to make a connection between your programming device (PG2000 and PC) and the PCC.

6.1. General Information

The programming software for B&R 2000 controller generation is delivered in a set (including documentation and **Online Cable**). The cable for the connection between the PC and PCC can also be ordered from B&R separately:

Cable PC Controller 2000 RS232 — Model No.: 0G0001.01-090
(Online Cable for the connection between PCC and programming device)

Further Considerations:

- Either COM1 or COM2 on your PC must be available. If both of these interfaces are already being used, you will have to do without one of the two devices (modem, ...) while you are using the interface for the Online Connection to the PCC. Otherwise you have to expand your system with additional interfaces and a connect device (from COM1 or COM2) to one of the new interfaces. Additional information can be found in the user's manual for your PC, the user's manual for the device connected or from your PC handler.

- There are two 9 pin D-type connectors (F) on the Online Cable. Since some PC interfaces are equipped with a 25 pin D-type connector (M), you may require an adapter. These adapters are often contained in the delivery of a mouse. Otherwise you can get them from your PC handler.

- Any interface on the PCC CPU (starting with PCC Operating System Version 1.10) can be used for the Online Connection to PG2000.

! ATTENTION ! ATTENTION ! ATTENTION ! ATTENTION !

The cable delivered is only suited for a connection between two RS232 interfaces. Do not use an adapter to connect the PC interface COMx with a RS485/RS422 interface. You need a special interface converter for this.

! ATTENTION ! ATTENTION ! ATTENTION ! ATTENTION !

6.2. Making the Connection

Before you connect the PCC and PC to each other, check the following points:

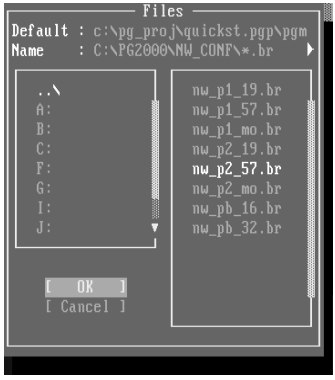
- ❑ Is the PCC CPU turned on?
- ❑ Are both the LEDs READY and RUN lit on the status display of the PCC CPU?

  If required, trigger a TOTALINIT. See the *B&R 2000 Hardware User's Manual* for your CPU.

Now you can make the connection between PCC and PC with the Online Cable e.g. IF1 on the PCC CPU with the serial interface on the PC.

Now you can establish the connection. To do this, select the interface and baudrate in PG:

- ❑ Select *Set-Up=>Connection*
- ❑ Select the [Config] dialog box
- ❑ Select the interface driver from the list shown:



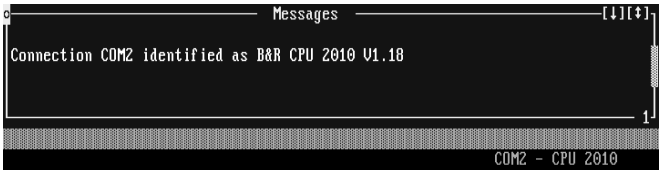
The interface drivers are found in the NW_CONF sub-directory of the PG2000 directory. Select the driver that corresponds to the interface and baudrate used.

Normally, you can use a fast connection with 57600 baud for a serial interface. Select either NW_P1_57.BR for COM1 or NW_P2_57.BR for COM2 according to the interface used for the Online Connection. If the interface cannot handle this transfer rate, select 19200 baud.

- Select [OK]. After a short time, a dialog box appears that shows the selected interface again:



- Select [OK] again. The connection is established. A successful connection is indicated in the message window and in the message line:



After the connection is successfully established, you will see information concerning the interface and the PCC CPU in the message window. In the example shown above:

Interface — **COM2** on the PC is being used

CPU — a **B&R CPU 2010** is identified as PCC CPU

PCC Operating System — the PCC CPU returns the version number => **V1.18**

- Now the connection between PG2000 and PCC CPU is established. Tasks can now be transferred.

6.3. Interface Driver

The following can be used for interface between PC and PCC:

- COM1/COM2 for direct serial connection
- COM1/COM2 for a modem connection
- PROFIBUS PC Card (AT bus) for a PROFIBUS connection

In order to be able to establish a connection to the PCC, a certain driver must be selected depending on the interface used:

COM1/COM2 — Direct Serial Connection

Driver	Interface	Baudrate
NW_P1_19.BR	COM1	19.200 Baud
NW_P1_57.BR	COM1	57.600 Baud
NW_P2_19.BR	COM2	19.200 Baud
NW_P2_57.BR	COM2	57.600 Baud

COM1/COM2 — Modem Connection

Driver	Interface	Baudrate
MO_P1_24.BR	COM1	2400
MO_P1_96.BR	COM1	9600
MO_P1_19.BR	COM1	19200
MO_P2_24.BR	COM2	2400
MO_P2_96.BR	COM2	9600
MO_P2_19.BR	COM2	19200

PROFIBUS PC Card — PROFIBUS Connection

Driver	PROFIBUS Network Configuration
NW_PB_32.BR	Configuration for 32 connections to B&R PCC controllers
NW_PB_16.BR	Configuration for ... ... 16 connections to B&R PCC controllers ... 16 connections to PANELWARE C300 operator panels

Additional information concerning PROFIBUS can be found in the *B&R SYSTEM 2000 PROFIBUS User's Manual*.

7. How do I transfer a task to the PCC?

If the connection between PG2000 and the PCC is established successfully, you can transfer the task to the PCC. Select *Object=>RUN*. The object that is currently selected will be transferred to the PCC:

Current Object	Transfer to the PCC
Object in the PCC layer	All tasks on the PCC
Object in the processor layer	All tasks on the processor (PCC CPU, multiprocessor or intelligent I/O module)
Object in the task layer	The current task
Open editor (PL2000, LAD or STL)	The current task

If a task was not yet created (B&R module doesn't exist yet) or the program code was changed after the last compile, the task must be compiled AGAIN and then transferred to the PCC.

Progress and completion will be reported in the message window:

```
*** Project Processing PCC > PCC_1 < Processor > CPU < ***
Processing file task1.src
File task1.src unchanged. Compilation skipped
Download task1
Download complete (M=$0712, T=$0580)
```

If you have followed this example as described, the task will be started automatically when the transfer is complete.

The LEDs on the output module are to be lit as follows:

1. The LEDs for output `yell_light` (channel 2) and `lockout` (channel 8) are lit.

2. After 30 seconds, these two outputs go out and the output `grn_light` (channel 3) is lit.

In order to check the function of the program, the value for the variable `act_weight` must be changed. However, since this variable is not connected to a weight sensor (e.g. analog input), you have to set this variable with the help of PG2000. You can used the so-called PV Monitor to do this (see section *8. How do I influence tasks on the PCC?*).

8. How do I influence tasks in the PCC?

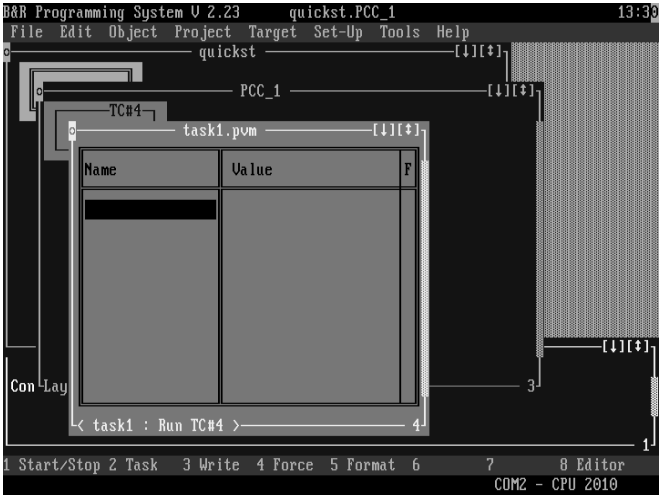
PG2000 offers you various possibilities to influence tasks on the PCC:

- **Read/Write** individual variables
- **Stop** individual tasks, task classes or the entire PCC
- **Start** individual tasks, task classes or the entire PCC
- **Remove/Delete** individual tasks from the PCC
- **Reset** the PCC in four different ways (Init, Total-Init, Reset or Diagnose)

8.1. PV Monitor

With the PV Monitor, you can read from or write to individual variables on the PCC. Call the PV Monitor:

- Select the symbol *task1* in the task layer or open the program code for *task1*.
- Select *Object=>PV Monitor* or press the [F11] function key. The PV Monitor will be opened:



The PV Monitor is only started if ...

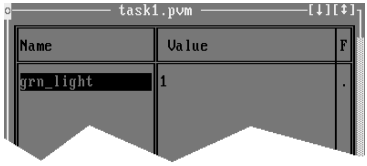
... an active connection exists between PG2000 <=> PCC, and

... the task exists on the PCC.

- Variables are not yet shown in the PV Monitor. They must be entered. Pressing the space bar displays a list containing all variables in the current task.



- Select the variable `grn_light` and acknowledge with [OK]. The selection list is closed and the variable selected is entered in the PV Monitor. Its value is shown in the middle column.



  Pressing the space bar in the PV Monitor when a variable is selected (as shown above) causes a box containing information about the variable to be displayed.

- Move the cursor to a free line and call the selection list again (space bar). You can also select several variables at one time. Mark the desired variable with the space bar. Marked variables are shown with check marks:



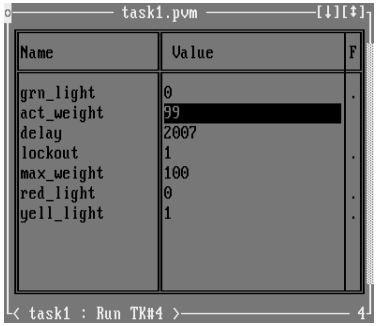
- Select [OK] to enter the variables in the PV Monitor:



The values of the variables are shown in the middle column.

- Move the cursor to the value for the variable `act_weight`. Enter a value larger than 100 (max_weight) with the keyboard and press ENTER. The `lockout` will be activated immediately and the variable `red_light` will be set to 1.

- Change the value of for `act_weight` again - this time to a value smaller or equal to 100. The variable `yell_light` is set to 1. The counter variable `delay` changes continuously - from 3000 to 0:



- When the counter variable `delay` reaches 0, the variable `grn_light` is set and the lockout is `deactivated`.

- You can also save the variable list for the PV Monitors (*File=>Save*). Therefore the variable list for *task1* will be shown immediately the next time the PV Monitor is called.

In this way, you can read and write variables on the PCC. You can see that this method can also be used to test a program.

8.2. Stopping, Starting, Removing Tasks

Various functions can be carried out for each individual task found on the PCC with the *Target=>Tasks* function:



A list of all task found on the PCC is shown by clicking the task list section field shown above.

Select the desired task. You can ...

... start a task found on the PCC with **Start**

... stop a task found on the PCC with **Stop**

... remove/delete a task found on the PCC with **Remove**

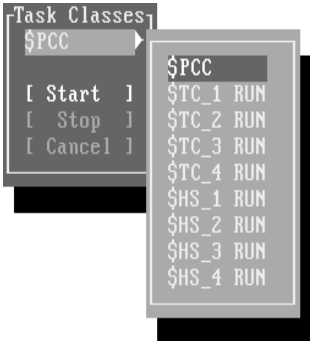
... restart an installed task found on the PCC with **Restart** (this is only possible for tasks if the option *Start* is set to manual in the task parameters).

... replace a B&R module found on the PCC with **Replace**, that means the module on the PCC is deleted and then transferred from the PC to the PCC.

... transfer a task from the PC to the PCC with **Download**. The task in the PCC does not exist on the PCC (otherwise the **Replace** function must be used).

8.3. Starting/ Stopping Task Classes

All tasks in an individual task class can be started or stopped with *Target=>Task Classes*. It is also possible to start/stop the entire PCC with this function. The task class is selected similarly to the function *Target=>Tasks*:



Either RUN (task class is not stopped) or STP (task class is stopped) is written next to each task class.

If the entire PCC is stopped ($PCC stopped), STP is written next to all task classes. The RUN and READY LEDs go out on the PCC CPU and the ERROR LED is lit.

Take note of the following:

- With *Stop=>$PCC,* not only the cyclic task classes are stopped. The idle time, interrupt and exception tasks (see glossary) are also stopped.
- If individual task classes are stopped with *Stop=>$TC_x* or *Stop=>$HS_x*, they cannot be started again with *Start=>$PCC*.
- Individual tasks can be started even when the PCC is stopped with *Stop=>$PCC*. The RUN and READY LEDs do not become lit in this case. The ERROR LED also remains lit.
- If the PCC CPU is started again with *Start=>$PCC,* the ERROR LED goes out and the RUN and READY LEDs are lit again. Task classes that where stopped explicitly with *Stop=>$TC_x* or *Stop=>$HS_x* remain stopped.

9. Ladder Diagram - LAD

In this section, we will create a small program using Ladder Diagram programming. You will learn how to use the LAD Editor and get an introduction to the LAD Debugger. In addition, you will find out how to import a function library into the project.

9.1. Example

As soon as the light turns red (see PL2000 example), a warning light is to be turned on in another room. This warning light is to blink once every second. The blinking will be programmed in LAD. This example shows how data can be exchanged between tasks. The LAD program reads an output (`red_light`) that is written by *task1*.

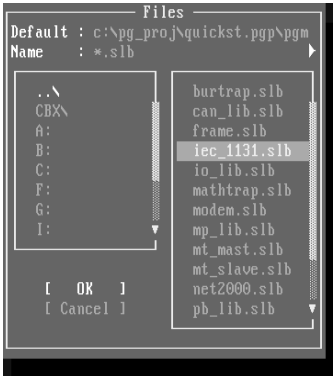
9.2. Library Import

For this example, you will require the **TON** function block. The **TON** FBK is used to create the blink delay.

This FBK is contained in the IEC 1131 Library. The library was copied to your PC along with other function libraries during the PG2000 installation. Additional information concerning the libraries can be found in the *B&R SYSTEM 2000 Library Reference Manual*.

In order for library functions to be used in a project, they have to be imported into the project:

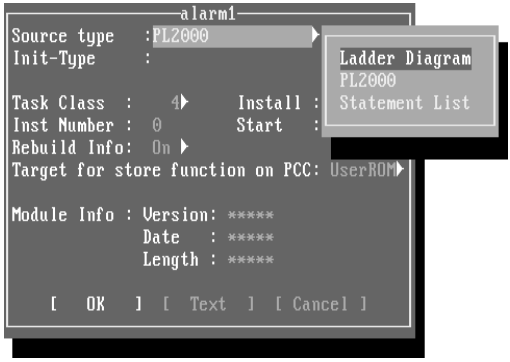
- ❑ Open the project where the library is to be imported.
- ❑ Select *File=>Import=>Library*. You can select the IEC 1131 Library form the dialog box:



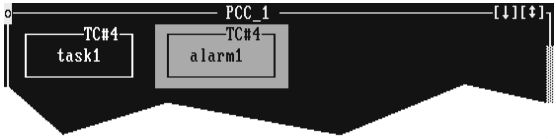
The selected library is imported after you press [OK]. All functions in this library are now available for your use.

9.3. Entering a Program

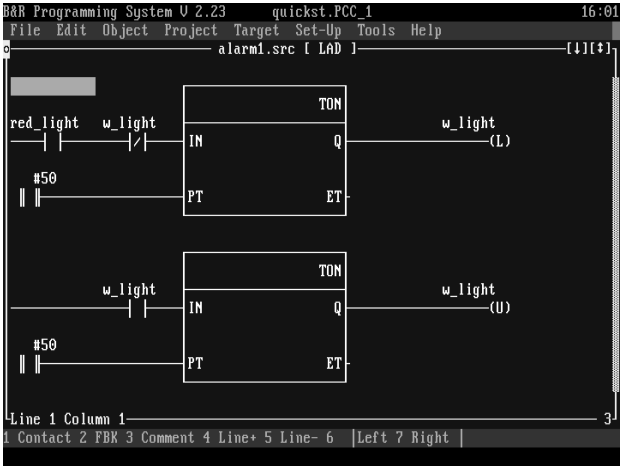
In order to create a LAD program, insert another **Task Symbol with the Name *alarm1*** in the task layer of **PCC_1**. This is done exactly as described earlier in *5.2. Insert Task Symbol*. However, select **Ladder Diagram** for the source type in the task parameters instead of **PL2000**:



The task layer should look like this:



If the task symbol *alarm1* is marked, you can now go into the LAD Editor by pressing [F7]. You can place various symbols such as inputs and outputs here. If you already have experience with LAD, simply try entering the following LAD program:



Practice a little with the function keys in the message line.

Online Help - **LAD**

Or call Online Help with the LAD Editor open. In this way, you can find detailed information concerning the current topic quickly. Simply click on the desired topic (highlighted with light print) to get additional information.

Online Help - **Library**

You can call Online help for each library. In this way, you can find the individual functions with syntax and description. Select *Help=>Functions*. Select the desired library from the FBK list and call context sensitive Online Help with [Help].

Operating the LAD Editor

Here are a few notes to help you enter the LAD example program:

Inserting Contacts:

A contact is simply inserted with a menu. Call the menu with [F1 Contact]. All contacts that can be entered at the current cursor position are highlighted. Contacts that are not allowed are deactivated and cannot be selected.

Enter either the variable name or a constant (decimal, hexadecimal or binary value, floating point, character string) as **Contact Name**.

Changing the Name / Constant:

The contact name or a constant can simply be changed by placing the cursor on the contact and select *Name* from the context menu. The context menu can be called with ...

... [Ctrl]+[F3] or [F1]

... two mouse clicks on the contact

Changing the Contact Type:

The contact type (negated input, positive or negative one shot input, latching or unlatching, ...) can be changed by placing the cursor on the contact and pressing the minus key. You will cycle through the contact types.

You can negate a contact with the [N] key. However, this only functions for simple contacts and not for one shot or latching/unlatching contacts.

Inserting a FBK:

With [F2 FBK], you can call the FBK list containing the libraries and function blocks that you can select. All libraries that are imported in the project are shown in the FBK list. The FBK is only inserted if enough space is available in the LAD Editor.

You can give an ALIAS name when inserting a FBK. If you give an ALIAS name in this example, make sure that you do not give the same name twice. If you give the same ALIAS name twice for FBKs, the same memory will be used for both FBKs. In other words, the output Q for both FBKs in the LAD will always be identical. The LAD example will not function properly!

Drawing / Deleting Lines:

You can draw a line at the current cursor position with the space bar. If a line or contact is already at the current cursor position, it will be deleted.

With the function keys [F6 left] / [F7 right]], you can draw a vertical link line left/right of the cursor position up to the next symbol (line or contact). If a line already exists in this position, it will be deleted.

Inserting / Deleting LAD Rungs/Lines:

[F4 Line+]	Insert a LAD rung Line will be made longer
[F5 Line-]	LAD rung without contacts/FBKs will be deleted Line will be deleted

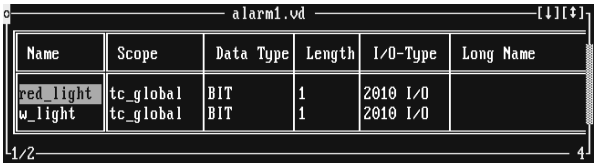
Inserting / Deleting Lines:

If the cursor is on a line or contact, ...

... a line can be inserted with [Ins]. Contacts, FBKs and lines are moved right.

... a line or contact can be deleted with [Del]. Contacts, FBKs and lines are moved left.

Save your entries in the LAD Editor and declare the variables as follows (see *5.4. Variable Declaration*):



- The variable `red_light` is already defined. Both tasks *task1* and *alarm1* must be in the same task class TC4 (see task parameters). Only then is this variable available in both tasks. The variable was assigned the scope *tc_global* (valid globally for all tasks in a task class).

- Place the output `w_light` on the same output module as the outputs from the PL2000 example (e.g. channel 4).

Note

Either all variables (declared and undeclared) for the active object or only the undeclared variables will be shown depending on when and where the variable declaration was called.

9.4. Testing a Program

After you have entered the LAD program and declared the variables, you can compile it and transfer it to the PCC:

- Select the task symbol *alarm1*.

- With *Object=>RUN,* the LAD program will be compiled and immediately transferred to the PCC.

- See the following sections in this chapter:

  5.5. Creating a Task

  6. How do I connect PC and PCC?

  7. How do I transfer a task to the PCC?

- Select the task symbol *task1*

- Call the PV Monitor for *task1* with [F11]

- If you saved the PV list the last time the PV Monitor was called, the variables for *task1* appear in the PV Monitor. Change the value of the variable act_weight to a value larger than 100.

- The LED for the output `w_light` (channel 4) begins to blink.

10. Statement List - STL

In this section, we will create a small program with using Statement List programming. You will get an introduction on how to use the STL Editor.

10.1. Example

The PL2000 example from "*3. What is my goal?*" will be the basis for the STL example. Here is the formulation of the example in STL:

Initialization Section:

```
INIT:       LD      1
            ST      lockout
            ST      red_light
            LD      0
            ST      yell_light
            ST      grn_light
            LD      100
            ST      max_weight
```

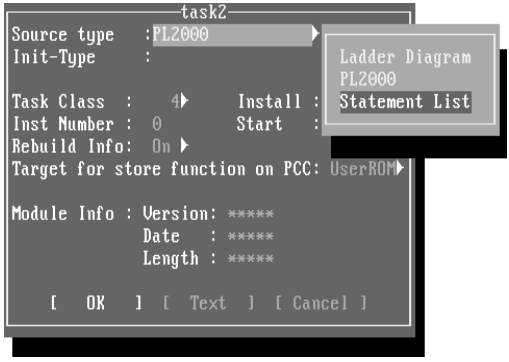
Main Section:

```
START:      LD      act_weight
            LE      max_weight
            JMPC    WEIGHT_OK

            LD      1
            ST      lockout
            ST      red_light
            LD      0
            ST      yell_light
            ST      grn_light
            JMP     END

WEIGHT_OK:  LD      red_light
            JMPCN   NOT_RED

RED:        LD      0
            ST      red_light
            LD      1
            ST      yell_light
            LD      3000
            ST      delay
            JMP     END

NOT_RED:    LD      yell_light
            JMPCN   END

YELLOW:     LD      delay
            GT      0
            JMPC    ST_DELAY



DELAY_END:  LD      0
```

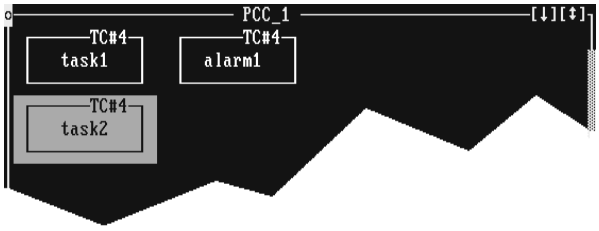
```
                ST      yell_light
                ST      lockout
                LD      1
                ST      grn_light
                JMP     END

ST_DELAY:       LD      delay
                SUB     1
                ST      delay

END:
```

10.2. Entering a Program

In order to create an STL program, insert another **Task Symbol with the Name *task2*** in the task layer for **PCC_1**. This is done in the same way as described earlier in *5.2. Inserted a Task Symbol*. However, select **Statement List** for the source type in the task parameters instead of **Ladder Diagram**:



The task layer should look like this:



If the task symbol *task2* is marked, you can go into the STL Editor with [F7]. You can now enter the STL code here. The **STL Editor** can be compared to an ASCII Editor. Since text editors are operated similarly, simply try entering the main section of our STL example program in the editor window. Open the InitSP window with *Object=>InitSP* and enter the initialization section.

Online Help - **STL**

If you need additional information concerning the STL Editor, call Online Help while the STL Editor is open. In this way, you can quickly find detailed information concerning the current topic. Simply click on the desired topic (highlighted with light text) to get additional information.

Operating the STL Editor

Here are a few short notes for entering the STL example program:

Tabulator - distance:

The TAB distance can be set and saved specifically for each project. Select the entry *Editor=>PL2000/STL/Text* in the *Set-Up* menu.

Comment Fields:

Select the entry *Editor=>PL2000/STL/Text* in the *Set-Up* menu. In a dialog box, you can set the option *Comment Fields*. If **YES** is selected, the cursor is placed in the first TAB position on the next line after pressing ENTER .

Variable Declaration

Save your entries in the STL Editor and declare the variables. The variable declaration is identical to the one for the PL2000 example program.

10.3. Testing a Program

After you have entered the STL program and declared the variables, you can compile the program and transfer it to the PCC:

- Delete *task1* from the PCC first (see *8.2. Stopping, Starting and Removing Tasks*). This step is necessary since the STL example uses the same variables and outputs.
- Select the task symbol *task2*.
- With *Object=>RUN,* the STL program will be compiled and then immediately transferred to the PCC.

- See the following sections in this chapter:

  5.5. Creating a Task

  6. How do I connect PC and PCC?

  7. How do I transfer a task to the PCC?

- Select the task symbol *task2*

- Call the PV Monitor for *task 2* with [F11]. Test the STL program in the same way as the PL2000 example (see *8.1. PV Monitor*).

CHAPTER 3
OPERATION OF PG2000

1. The PG2000 Screen

The PG2000 programming system consists of a project management and a programming environment section:

- **Project Management** serves to create, open, delete, copy, projects etc.
- The **Programming Environment** serves to edit a project (construct, compile, transfer to the controller, ...).

1.1. Project Management

The project management functions support working with several projects. For reasons of clarity, you should create a separate project for each application.

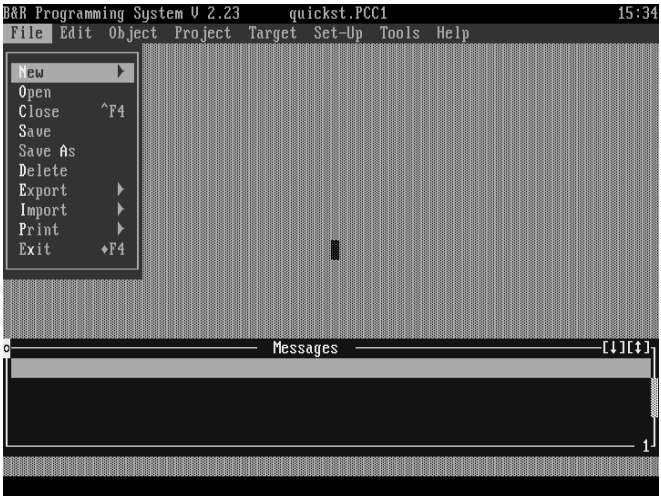
The **first** time the PG2000 programming system is called, project management will be called automatically:



- The version number of the programming system and the current time is shown in the **Title Line** for project management.
- The **Main Menu Bar** offers project management functions in the *Project* menu.
- The **Message Window** cannot be closed by the user. Closing this window causes its contents to be deleted.
- Additional information is given in the **Message Line** (last line).

1.2. Programming Environment

After opening a project, the following menus will be offered in the main menu bar for the **programming environment** functions:



- The following information will be show in the **Title Line** for the programming environment:
  - Version number of PG2000
  - Project, PCC and processor name if available in the form: "<project name>.<PCC name>.<processor name>"
  - Current time

- The **Main menu Bar** offers various menus containing the PG2000 functions.

- **Mouse Cursor:** The current mouse position is indicated by an inverted character.

- The **Message Window** cannot be closed by the user. Closing the message window causes its contents to be deleted.

- Additional information is shown in the **Message Line** (last line) (e.g. status of the connection between PCC and programming device).

- A menu (**foot line menu**) with additional functions is shown for some windows (LAD Editor, GDM, Debugger, PV Monitor).

2. Mouse / Keyboard Operation

PG2000 can be operated with both the mouse and keyboard. You can choose which type of operation best suits your purposes.

2.1. Mouse

The following terms will be used in this manual when referring to mouse actions:

Clicking

1. Position the mouse cursors on the respective item.
2. Press and release the left mouse key.

Dragging

1. Position the mouse cursors on the respective item.
2. Press and hold the left mouse key.
3. Move the mouse to the desired position.
4. Release the mouse key.

2.2. Keyboard

The following **syntax** will be used in this manual to describe keys and key combinations:

[key] ..... Keys are shown in square brackets.

[key1] + [key2] ..... Keys that are to be pressed at the same time are shown together with a "+" between them.

[key1] [key2] ..... Keys that are to be pressed one after the other are separated from each other with an empty space.

The following keys are used, among others:

English	German	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 character in the current cursor position
[Home]	[Pos1]	Move cursor to beginning of line
[End]	[Ende]	Move cursor to end of line
[Esc]	[Esc]	Escape (e.g. ends dialog boxes)
[Tab]	[Tab]	Move cursor to the next tabulator position
[Space]	[Space]	Space bar
[Ctrl]	[Strg]	Control key
[⇑]	[⇑]	Shift key
[Alt]	[Alt]	Alternate key
[⇐]	[⇐]	Back space

Additionally, some special keys are shown symbolically in the menus and in the Help System of the PG2000 programming system:

Symbol	Key
↑	[⇑]
^	[Ctrl]
◆	[Alt]

General Key Assignments

The main PG2000 keys and their functions are shown in the following list:

Key	Description
[F9]	Compile the selected object.
[F10]	Call context sensitive help text concerning the active window or dialog box. This key can be pressed at any time.
[F11]	Call the PV Monitor for the selected object.
[F12]	Call the Graphic Design Method (GDM).
[Esc]	Call the system menu, escape the current function or get out of the active pull down menu or open dialog box.
[Ctrl] + [F1]	Make the active window full screen. Pressing this key combination again returns the window to its original size.
[Ctrl] + [F3]	Call a pop-up menu (inside of a window or in an entry field in a dialog box).
[Ctrl] + [F4]	Close the active window.
[Ctrl] + [F5]	Turn on mode to move a window or change the size of a window.
[Alt] + [F4]	Exit PG2000.
[Alt] + [F6]	Switch to the next window.
[Alt] + [Window No]	Activate the window with the given number.
[⇧] + [F5]	Search for given string in the selected direction.
[⇧] + [F9]	Compile the selected object and transfer it to the PCC.
[⇧] + [F11]	Either the PL2000 Debugger, STL Debugger or LAD Debugger will be called according to the object selected.
[Ctrl] + [Ins]	The marked area is inserted into the clipboard but not deleted from the source text.
[⇧] + [Ins]	Insert the contents of the clipboard in the editor.
[⇧] + [Del]	The marked area is deleted from the source text and inserted into the clipboard.
[Esc]	Call the system menu or exit the current function.

3. Menus

You will find three types of menus in PG2000:

- The **Main Menu Bar** allows you to find the most important PG2000 functions at a glance. Clicking on a topic with the mouse opens a so-called pull-down menu where you can select the desired function. These menus can be selected using the keyboard by pressing [Alt]+[First Letter].

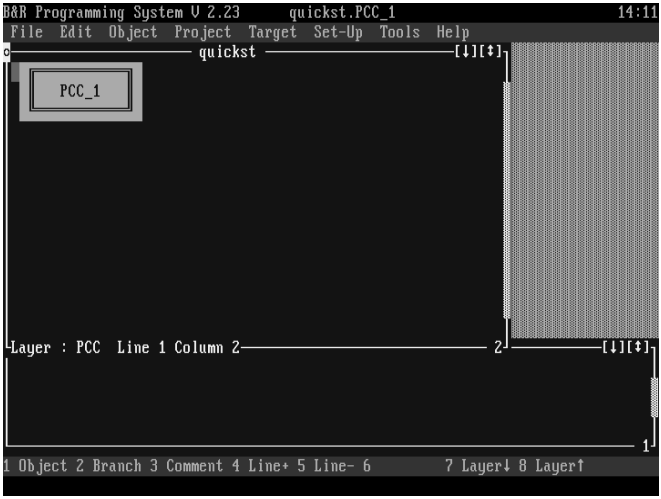
  Unlike the context menus, the functions in the main menu bar do not change while working with PG.

- The **Context Menus** offer different functions that change depending on the context. Context menus are called by double clinking them (two mouse clicks in the same cursor position) or by pressing [Ctrl]+[F3] on the keyboard.

- The **Foot Line Menus** are also context sensitive. They offer functions that correspond to the active window. The functions in the foot line menu are activated either with a mouse click or by pressing the respective function key.

4. Working with Windows

This section handles working with windows using the GDM. Call the GDM by selecting *Project=>GDM*:

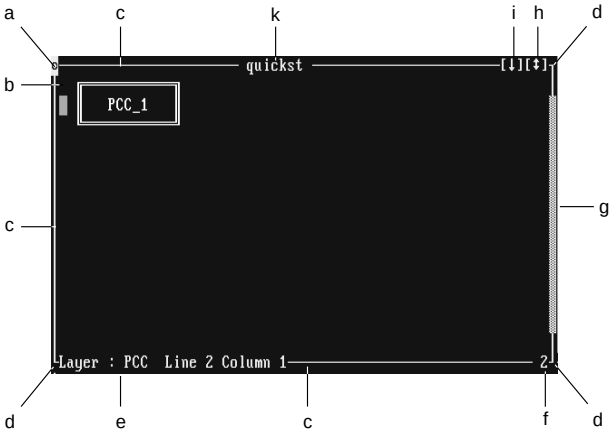


A **max. 10 windows** can be open at once in PG2000. Only one window can be active. All other windows are inactive.

- The **active window** is indicated by the fact that it is completely in the front of the screen and the frame is lighter than the inactive windows.

- The assignments of the function keys for the active window are shown in the **foot line menu**. Selecting another window causes the foot line menu to be changed or not shown.

Elements of a Window



a) System Menu Field
b) GDM Cursor
c) Window Frame
d) Corner of Window
e) Current Cursor Position or other Information
f) Window Number
g) Scroll Bar (consisting of scroll bar cursor, field and arrows)
h) Zoom Box (switches between full screen and normal size
i) Minimize Box
k) Window or File Name

System Menu

The system menu is a pop-up menu that can be activated in any window. Various functions are available in the system menu (close window, change window, etc.).

Activate the system menu for an active window:

- Clink on the *"System Menu Field (a)",* or
- Press the [Esc] key.

Close the system menu without calling a function:

- Click outside of the system menu, or
- Press the [Esc] key again.

Closing a Window

A window can be closed with:

- *File=>Close*
- System Menu: *Close*
- [Ctrl] + [F4]

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

If changes were made in an open window, you will be asked if you want to save the changes when you close the window:

Making the Active Window Larger or Smaller

The size of a window can be changed by ...

... dragging the *"Corner of a Window (d)"*

... selecting *Resize* from the system menu or pressing [Ctrl] + [F5]. The following will be shown in the message line:

```
Cursor : move; ↑Cursor : resize; ENTER : end
```

Pressing the key combination [⇑]+[Cursor Key] changes the size of the window in steps. This mode can be exited when the desired size is reached by with the [↵] key or the key combination [Ctrl]+[F5].

Moving the Active Window

A window can be moved by ...

... dragging the *"Window Frame (c)"*

... selecting *Resize* from the system menu or pressing [Ctrl] + [F5]. The following will be shown in the message line:

```
Cursor : move; ↑Cursor : resize; ENTER : end
```

Pressing the cursor keys moves the window in steps. This mode can be exited when the desired size is reached by with the [↵] key or the key combination [Ctrl]+[F5].

Making the Active Window Smaller

This function reduces the active window to a minimum size and places it on the right edge of the screen. The window is placed in the background (inactive).

This function can only be selected if the window is active. It is called by:

... clicking on the *"Minimize Box (i)"*
Calling this function again returns the window to the original size.

... selecting *minimize* from the system menu.
The *Restore* function from the system menu returns the window to its original size.

Zooming the Active Window

The zoom function can be used to enlarge the active window to the maximum size. It can be called by:

... clicking on the *"Zoom Box (h)"*.
Calling this function again returns the window to its original size.

... pressing the key combination [Ctrl]+[F1].
Calling this function again returns the window to its original size.

... Select the function *maximize* from the system menu.
The *Restore* function from the system menu returns the window to its original size.

Changing Windows

If several windows are open at the same time, they can be switched between or a certain window can be made active:

- Clicking on a window makes it active. The mouse cursor must be found in the window you wish to activate. The window that was active becomes inactive and is placed in the background.

  Clicking on the frame of an inactive window can be used to move it or change its size, however, it will not be activated.

- Changing windows with the keyboard:

  - **[Alt]+[Window No.]** - The window with the given number will be activated. (0 to 9; 0 Þ window no. 10)

  - **[Alt]+[F6]** - The window with the next higher number will be activated. If a window with a higher number does not exist, window 1 will be activated.

- Select the function **System Menu: *Next***. The window with the next higher number will be activated. If a window with a higher number does not exist, window 1 will be activated.

❑ Another possibility to switch to a different window is the function *Tools=>Window List*. A list of all open windows will be shown. You can select the desired window from the list.

Scrolling through Window Contents

Since the contents of a window cannot always be shown completely, only a section in the size of the window will be shown. Moving through the sections in a window will be called scrolling. You can scroll through the window in various ways (mouse or keyboard) so that you can edit or make entries in a certain position.

The visible section can be moved up or down with the **scroll bar** and the **mouse**:

❑ Clicking on the **scroll bar arrow** scrolls up or down through the window.

❑ Dragging the **scroll bar cursor** scrolls up or down through the window. The size of the scroll bar cursors is dependent on the size of the section compared to the size of the entire window content. That means the smaller the cursor, the larger the section that is not shown in the window.

❑ The following keys or key combinations can be used to scroll through the window:

[PgUp] ..... One page up

[PgDn] ..... one page down

[Ctrl]+[Pos1], [Pos1] [Pos1] ..... Beginning of window contents

[Ctrl]+[End], [End] [End] ..... End of window contents

5. Dialog Boxes

Dialog boxes are used to allow the user to make required entries. A dialog box can consist of the following types of fields:

- Menu block consists of several selection fields (buttons)
- Text entry fields
- Selection list (with scroll bar if needed)
- Entry field with pop-up selection window

Example of a Dialog Box:



User entries have different effects according to the dialog box. The following entries are valid for **all** dialog boxes:

[Ctrl]+[↵]	Exit dialog box with the active selection field in the menu block. If e.g. the selection field [Cancel] is active, [Ctrl]+[↵] has the same effect as [Esc].
[Esc]	Exit the dialog box without accepting data.
[Tab], [⇑] + [Tab], [↑], [↓]	Switch between fields in the dialog box.

Selecting a Selection Field from a Menu Block

- Clinking on a selection field e.g. [OK], [Cancel] causes the dialog box to be closed or the user's entries to be accepted.
- The cursor keys can be used to move the cursor to the desired selection field. [↵] closes the dialog box accepts the user's entries.
- [Esc] has the same effect as the [Cancel] selection field.

Text Entry Fields

If the cursor is on a text entry field, you can enter data with the keyboard. The text can be edited with the following keys inside of the field:

[Pos1] ..... Move the text cursor to the beginning of the text

[End] ..... Move the text cursor to the end of the text

[←], [→] ..... Move the text cursor to the left/right

[↑], [↓], [↵] ..... Exit the text entry field; cursor is moved to a different field in the dialog box.

[Ins] ..... Switch between insert and overwrite mode

[Del] ..... Delete the character in the current text cursor position

[⇐] ..... Delete the character to the left of the text cursor

If the text is larger than the field, only a section of the text will be shown. However, the entire text can be edited.

Move the cursor to a text entry field to edit it:

- Click on the text entry field.
- Pressing the [Tab] key multiple times cycles the cursor to the desired field.

Selection Lists

Entries are listed in a selection list that can be selected by the user (e.g. file selection list in the file selection box). If more entries than can be displayed are available, a scroll bar appears to the right of the list.

Selecting an entry:

- ❑ Click on the entry; if necessary, find the entry first using the scroll bar.
- ❑ Pressing the [Tab] key multiple times cycles the cursor to the desired field. If the cursor is in the selection list, you can select the entry with the following keys:

  [Pos1] ..... Move cursor to the first entry

  [End] ..... Move cursor to the last entry

  [↑], [↓] ..... Move through the selection list

  [↵] ..... Select the entry

  [x] ..... Pressing a letter moves the cursor to the next entry that starts with that letter.

Making Entries with a Pop-Up Selection Window

This type of entry field is marked on the right side with an arrow (" ▸ "). If the cursor is in an entry field, an entry can be selected from the selection window.

Selecting an entry:

- ❑ The pop-up window is opened by clicking on an entry field. An entry can be selected from this window.
- ❑ Pressing the [Tab] key multiple times moves the cursor to the entry field. The keys [Ctrl]+[F3], [Space], [←] or [→] open a pop-up window. An entry can be selected with the [↑], [↓] and [↵] keys.

Closing the pop-up selection window without calling a function:

- ❑ Select [Cancel].
- ❑ Click outside of the selection window.
- ❑ Press the [Esc] key.

6. File Selection Box

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



Default

When opening a file selection box, the contents of the program directory for the current project is shown in the file list. This default path cannot be changed while working with PG2000.

Name

A mask can be entered in this text entry field that influences the files displayed in the file list (can be compared with wildcards in DOS). If e.g. only the files that begin with "D" are to be shown, "d*.*" will be entered as mask.

A complete file name (with path) can also be entered. Entering a new mask or switching to another directory causes the complete path including the current mask to be shown in the *Name* text entry field.



When a file activity is executed during a PG session (Open, ...), clicking on the text entry field or pressing the [Space] bar causes a pop-up selection window to be shown in the text entry field. This pop-up selection window contains a list of all masks used for previous file activities.

File List

The files in the current directory are shown in this list (using the current mask). The desired entry can be selected from this list.

Directory List

All sub-directories for the current directory ("..\" stands for the current directory itself) and all drives are shown in this list.

The respective list entry is to be selected in order to switch to another drive or another directory and show its contents (using the current mask).

[OK]

The selected **file** (either marked in the list or entered directly for *Name*) will be opened, copied, ... (according to the function) and the file selection box will be closed.

However, if a **directory** (or **drive**) is marked in the directory list, this directory (or drive) will be made current.

If a new **mask** was entered in the text entry field, the corresponding file list will be shown.

[Cancel]

The dialog box will be closed without executing the function (Open, Copy, ...).

7. Help System

The PG2000 programming system has a Help System where the user can call information and help text concerning the function and operation of PG2000 at any time. Additionally, you can also insert your own files into the help system.

The help system is divided into three areas:

1) Information about the PG2000 programming system

2) Index of additional help topics

3) Help for the function libraries

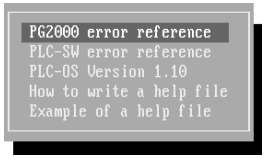
Calling Help System from the Main Menu

The following functions are available in the **Help** menu:

Help - Help system for the PG2000 programming system.

Index - A pop-up menu containing a selection of additional help topics is shown. You also have the possibility to create help files and insert them in the index.

The following topics exist in the index after installing the PG2000 software.



The respective help text will be shown on the screen after selecting one of these topics.

Functions - A dialog box with a selection list containing all function libraries available in the current project data bank is shown.

[OK] ..... Information concerning the selected function will be shown in a window (function block or inline function).

[Cancel] ..... The dialog box will be closed.

[Lib Info] ..... A dialog box with information concerning the selected function library will be shown.

[Help] ..... The help text for the selected function library will be shown.

Calling Help System with Function Key

The PG2000 Help System can be called directly at any time using the [F10] function key. Context sensitive help text will be called for the active PG2000 function.

Examples:

- If the PL2000 Editor is active, information concerning the PL2000 Editor will be shown.
- If a dialog box is active, either hints for general use of dialog boxes will be shown or specific information concerning text.
- If the variable declaration is active, information concerning the declaration of variables will be shown.

Using the Help System

When using the help system, subtopics can be selected or you can return to the previous page. If the menu item *Help* was selected, the help system main menu is shown on the screen.

Selecting a Subtopic

The highlighted text refers either to subtopics or related topics that can be called. The highlighted text can be selected with the mouse or keyboard:

- If the mouse cursor is moved onto highlighted text, it is selected and the text will be shown darker. Clicking on this position calls the help text for this topic.
- A topic can be selected with the cursor keys or by entering the first letter of the highlighted text. The corresponding help text is called by pressing the [↵] key. The [⇐] key returns you to the help text where you called the current topic.

Exiting the Help System

- If the highlighted text "EXIT HELP" is available, clicking on this text will exit the help system.
- The [Esc] key can also be used to exit the help system.

8. Project Management

PG2000 has built-in project management. Entire projects can be copied, deleted and purged (delete all backup files).

As stated earlier, PG2000 consists basically of two components: the project management and the programming environment. You can switch between the two by:

- *Project=>Close* closes an open project, exits the programming environment and switches to project management.
- *Project=>Open* or *Project=>Create* opens an existing project / creates a new project and switches to the programming environment.

Creating a Project

This function was described relatively thoroughly in *Chapter 2 The First Steps* in section *4. How do I create a project*. New projects are automatically entered into project management.

Opening a Project

This function is used to open a project that is entered in project management. If this function is selected, you can select the project you want to open from a list:



Copying a Project

You can copy an entire project including all project data with this function. It is mainly used to create backup copies and to duplicate projects.

Deleting a Project

This function deletes the selected project from the hard disk.

Purging a Project

All backup files in a project are deleted with this function. Valuable memory on the hard disk is freed in this way.

Searching for a Project

You can search through your entire hard disk and enter projects that are not being managed into project management.

APPENDIX A
MODEL NUMBERS
DELIVERY CONTENTS

1. Delivery Contents

Programming software for B&R controller generation is delivered as a set (including documentation and online cable). The PG2000 software-package is available in two different forms:

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

The PG2000 software package consists of the following components:

RS232 Cable PC <=> System 2000 Controller
(online cable to connect the PCC and the programming device)

PG2000 Programming Software
(installation diskettes for the PG2000 programming system)

Documentation

The software is delivered on 31/2" HD diskettes.

2. Additional Model Numbers

2.1. PG2000 Individual Components

Here are the individual model numbers since the PG2000 programming software is normally delivered as a set:

Product Name	Model Number
RS232 Cable PC <=> System 2000 Controller (cable for online connection)	0G0001.01-090
PG2000 Programming Software (three 3 1/2" HD diskettes)	1A2000.01

2.1. Documentation

The model numbers of the documentation referred to in this manual:

Product Name	Model Number
B&R SYSTEM 2000 PG2000 Quick Start	MASYS2PGQS-E
B&R SYSTEM 2000 Library Reference Manual	MASYS2LRM-E
B&R SYSTEM 2000 Programming Languages Manual	MASYS2PLM-E
B&R SYSTEM 2000 Hardware User's Manual	MASYS2HW-E
B&R SYSTEM 2000 System Software Reference Manual	MASYS2SYS-E
B&R SYSTEM 2000 System Configurator and Profiler User's Manual	MASYS2CFG-E
B&R SYSTEM 2000 Advanced Programming Manual	MASYS2APM-E

APPENDIX B
GLOSSARY

Application RAM

Application RAM is freely available for application software. It can be used for cyclic tasks, non-cyclic tasks and data modules among other things. With the system configurator or the PG2000 programming system, (Version 2.10 or higher) you can configure a section of application RAM so that it behaves like application ROM after a TOTALINIT. This section of memory is known as FIX RAM. Modules which are stored in FIX RAM do are not deleted by a TOTALINIT

Application ROM

Application programs (cyclic and non-cyclic tasks, data modules...) can be stored in nonvolatile application ROM (EPROM or Flash Prom)

Boot Modes

The procedure which occurs when the PCCs operating system is started up is known as "booting". A boot procedure is caused either by the user himself (e.g.: with the PG2000 programming system or with the boot mode buttons on the CPU) or by a fatal error. The CPUs in the B&R controller generation offers various boot modes:

- ❑ TOTALINIT
- ❑ INIT
- ❑ RESET
- ❑ RECONFIG
- ❑ ERROR
- ❑ DIAGNOSE

Compile

Compiling means translating the combination of program code (PL2000, LAD or STL) and variable declarations into a language which can be understood by the PCC. The result is a B&R module that can be transferred to the controller.

Cycle Time
Cyclic Tasks

Cyclic tasks are executed regularly according to a defined time plan. All tasks in a cyclic task class are executed exactly once per cycle. The cycle time for a task class can be configured.

Cycle Time Monitoring

During PCC operation, the system manager monitors if the defined cycle times are actually being held. A cycle time violation occurs if the tasks in a task class cannot be executed within the cycle time. A cycle time violation can occur for various reasons:

- ❑ The total execution time of the tasks is longer than the defined cycle time.
- ❑ A task class with a higher priority is loaded too heavily. This cuts down the processing time available for task classes with lower priority.

In order to avoid cycle time violations, pay attention to the total system load when planning and developing a project.

Data Type

All variables used in a task are assigned a certain data type. There are many different data types. The type determines the characteristics of a variable, e.g. the range or precision for the number saved in the variable or which operations are possible with the variable. There are the following simple data types: BIT, BYTE, WORD, LONG, INT8, INT16, INT32, FLOAT.

There is also the possibility for you to define a variable type yourself. These variable types are created from multiple data types and are called structures.

Debugger

The term DEBUG refers to finding and removing errors. A debugger is a program that can monitor, step through, stop and start individual tasks. PG2000 offers a debugger for all three programming languages. The PV Monitor can be seen as a simple debugger. It can monitor, write and “force” variables and also start tasks, stop tasks and execute tasks for a defined number of cycles.

DIAGNOSE

In DIAGNOSE boot mode, the PCC only starts with the operating system. That means all application programs are ignored and are therefore inactive. After a DIAGNOSE boot procedure, the PCC goes into SERVICE mode (see SERVICE Mode). The PCC can only be booted with a TOTALINIT or again in DIAGNOSE mode.

DPR

DUAL PORTED RAM contains the data for all variables used in the application programs (as well as flags and I/O data). Data exchange between CPU and I/O processor takes place via this memory. They both can access the DPR.

Exception Task Class

Exceptions are fatal errors that occur while the PCCSW is running and cannot be corrected by the operating system itself. Standardly, a system E-Stop is executed if an exception occurs (the PCC goes into SERVICE mode). In addition to the error report in the error module, CPUs for B&R 2010 controllers show the error number on the status display.

Unlike other errors that can occur, exceptions give the user the possibility to not only recognize the error or exception, but also to react to it. The operating system provides a so-called exception handler for this purpose that can be used to handle the most frequent exceptions.

Exception tasks can be created by the user with the PG2000 programming system just like cyclic tasks. These tasks react to certain exception. In this way, you can program application specific exception handling.

Exceptions are e.g.: bus errors, address errors, illegal instructions, division by zero, cycle time violations, ...

FIX RAM

See *Application RAM.*

Force

Forcing variables is used to find errors. It "forces" a variable to contain a defined value. If a task writes to this variable, the changed value only remains for the current task cycle. Then the variable is automatically overwritten with the forced value. This function remains active until it is turned off again with PG2000. An LED is lit on the PCC indicating that the force function is active for one or more variables.

Functions

There are two different types of functions; inline functions and function blocks.

Inline functions always return ONE data element as result. The result of an inline function can e.g. be used as operand in the PL2000 programming language.

Function blocks are functions that return one or more values as result. Function blocks cannot be used as operand in programming statements. The result (one or more values) remain in tact from one program cycle to the next since function blocks are assigned internal memory.

Graphic Design Method (GDM)

The GDM is a graphic representation of a project (see Chapter 2 Section 2. Project, Task, Program, ...?).

INIT

When an INIT is executed, all battery buffered RAM data remains in tact (all tasks in application RAM, input/output values, flags, ...). Therefore, all tasks are in the same state after the INIT (active or inactive) as they were before the INIT. Newly inserted modules (remote master, network module, ...) are not recognized with an INIT, a TOTALINIT is required.

All initialization subprograms (InitUPs) are executed with an INIT.

Initialization Subprogram

A so-called initialization subprogram can be created for each task. It will be carried out once after an INIT, TOTALINIT and each time the task is started.

Interrupt Task Class

Interrupts are asynchronous (triggered by the hardware) events that interrupt the cyclic operation of the program. Triggering interrupts is only possible with certain hardware. You can create an interrupt task for each interrupt capable module which is executed immediately after the interrupt occurs. They can be used if fast reaction times are required for certain events.

Remark: Interrupt tasks should only be used for events that do not occur often. Events that occur often load the PCC too heavily.

Ladder Diagram

Ladder diagram is a programming method that is very similar to an electrical switching plan. Ladder diagram is especially suited for logical circuity.

Library

A library is a collection of functions or function blocks. Libraries can be imported into a project which makes additional functions immediately available to you (see Chapter 2 Section 9. Ladder Diagram - LAD).

Multitasking

Multitasking refers to several programs sharing the processing time on a CPU. The programs seem to run at the same time. Switching quickly between tasks gives the impression that the tasks are running at the same time.

Non-Cyclic Tasks

Non-cyclic tasks are not assigned to a task class. They use the processor idle time - the time the processor has “free”. Non-cyclic tasks CANNOT be created with PG2000; programming takes place in the C programming language (you can get additional information from your sales and service partner).

Operating System

The operating system of the B&R 2000 controller generation is a configurable, deterministic real time multitasking system. It is also called PCCSW (PCC Software). Primarily, the PCCSW takes charge of application resource management (memory, processor time,...) and multitasking.

Items which can be configured are e.g. individual cycle times of the task classes and memory assignments.

PCC

See *Programmable Computer Controller*.

PCCSW

See *Operating System*.

PL2000

PL2000 is a text or command oriented high level language from the so-called “third generation” of programming languages. It supports structured programming of applications.

Programmable Computer Controller

The major difference to a standard PLC is the integration of computer performance such as being able to be programmed with high level languages, multitasking of several PCC programs in a CPU and the availability of memory in the MByte range. There is also the possibility to guarantee the fastest cycle times for simple I/O processing using ladder diagram.

PV Monitor

The PV Monitor is used to monitor variables in the PCC. You can monitor (read cyclically), write (influence the program) and force (see Force) variables with the PV Monitor. In addition, tasks can be started and stopped or run for a certain number of cycles.

RESET

All battery buffered RAM data remains in tact (all tasks in application RAM, input/output values, flags, ...) in RESET boot mode (analog to INIT boot mode). However, the PCC goes into SERVICE mode (see SERVICE mode) after a RESET.

Scope

Variables have a so-called scope that determines the area and the tasks for which they are valid. The scope determines if the variable is valid e.g. in a task, in a task class or throughout the entire PCC.

SERVICE Mode

If the PCC is in SERVICE mode, all B&R modules available in application ROM and FIX RAM can be shown and deleted individually with the system configurator. This allows e.g. a faulty module to be deleted in order to avoid an ERROR boot procedure without totally deleting application ROM or FIX RAM.

Statement Lists

Statement list (abbr STL) is an instruction orientated, assembler type language of the so-called second generation programming languages. STL is based on the IEC 1131-3 standard for PLC programming languages. In addition, B&R STL has several useful extensions for B&R PCCs.

Symbolic Name

Variables are memory areas for values. These values can be in various forms, e.g. the status of a switch (ON/OFF), a work piece counter, or text (“ABxyzCD”). Variables receive a name so they can be identified. These symbolic names are mostly given so that they refer to the variable in a way that makes sense. Tasks are programmed using these symbolic names. You don't have to enter the output and module that is to be accessed in the program code, just the symbolic name. The hardware assignment takes place in the variable declaration.

System RAM

System RAM is mainly used by the operating system. It is used to backup all system parameters.

With the system configurator, free memory in System RAM can be divided into a module area and a temporary area. The temporary area is available for the user to quickly allocate non-buffered (volatile) memory within tasks (volatile memory must be reallocated after each boot procedure).

System ROM

System ROM cannot be programmed or deleted by the user. The CPU operating system is found in this memory area. The entire CPU does not have to be exchanged when updating the operating system since System ROM is in application memory.

Task

A task is created from the combination of program and variable declaration that can be transferred to the PCC CPU. Several tasks can run on the PCC CPU at the same time and exchange data with each other. Tasks are assigned to a so-called task class corresponding to their priority.

Task Class

Each task must be assigned to a certain task class. A certain cycle time which can be set by the user is defined for each task class. Each task is executed once in each cycle.

Task Parameters

The task parameters determine the characteristics of a task, e.g.:

- ❑ Programming language
- ❑ Programming language of the InitSP
- ❑ Task class
- ❑ Automatically installed and started after transfer to the PCC

TOTALINIT

The TOTALINIT boot mode corresponds to the FIRST initialization of the CPU. The operating system is fully restarted, allocated memory is freed and the system structures (all operating system information for orderly operation of the CPU) are reassigned. During a TOTALINIT, all system modules (Remote Master, Network Modules, ...) are recognized and initialized. Additionally, all modules saved in application RAM are deleted (except for the modules in FIX RAM) and all process variables (inputs, outputs and flags) are initialized.

Variable Declaration

The variable declaration creates the connection between symbolic names and hardware. Here, you can define if a variable is to access an internal memory location in the CPU, a digital output or an analog input.

Since variable declaration and program code are handled separately, the hardware can be changed easier because only the variable declaration needs to be adjusted.

INDEX

A

Application RAM 91
Application ROM 91
Application Specific Settings 15
  CONFIG.SYS Settings 16

B

Boot Modes 91
  DIAGNOSE 92
  INIT 94
  RESET 96
  TOTALINIT 97
BUFFERS 16

C

Cable: PC - Sys 2000 Controller, RS232 .... 44, 87
Channel Number 41
Character Set 16
COM1 ..... 44, 46, 47
COM2 ..... 44, 46, 47
Compiler 42, 92
CONFIG.SYS Settings 16
Connection
  Direct Serial Connection 47
  Modem Connection 47
  PROFIBUS Connection 47
Connection - PC and PCC 44
Contact Names 56
Contact Type 56
Contacts, Inserting 56
Context Menu 70
COUNTRY 16
Country Setting 16
Cycle Time 25, 98
Cycle Time Monitoring 98
Cyclic Tasks 98

D

Data Type 39, 92
Debugger 92
Delivery Contents 87
Dialog Boxes 76
Documentation 10
Download, Task => PCC 48
DPR 92

E

Example
  LAD 54
  PL2000 23
  STL 59
Exception Task Class 93

F

FBK, Inserting 57
File Selection Box 79
FILES 16
FIX RAM 93
Foot Line Menu 66, 70
Force 93
Function Blocks 93
Functions 93

G

GDM 31, 94

H

Help System 81

I

I/O Type 39
Information about this Manual 9
Initialization Subprogram 94
Installation of PG2000 12
  Character Set 16
  Country Setting 16
  Memory Management 17
  Network Driver 15
  Network Server Installation 15
  New Installation 13
  Updating an Old Version 14
  User Specific Settings 15
Installation Procedure 14
Interface Driver 45, 46
Interrupt Task Class 94

K

Key Assignments, General 69
Keyboard 67
Keyboard Operation 67

L

Label Field, Skipping 61
Ladder Diagram 54, 95
  Contact Names 56
  Contact Type 56
  Contacts, Inserting 56
  Editor 56
  Example 54
  FBK, Inserting 57
  Lines, Drawing/Deleting 57
  Program Lines, Inserting/Deleting 57
Libraries, Importing 54
Library 95
Library Import 54
Line Draw/Delete 57
Long Name 39

M

Mailbox, B&R 11
Main Menu Bar 65, 66, 70
memory Management, DOS 17
Menus 70
  Context Menu 70
  Foot Line menu 70
  Main Menu Bar 70
Message Line 65, 66
Message Window 46, 65, 66
Model Numbers 88
Module Address 41
Module Type 41
Mouse 17, 67
Mouse Operation 67
Multitasking 24, 95

N

Network Server, Installation 15
Non-cyclic Tasks 95

O

Online Cable 44
Online Help 10, 56, 81
  Library 56
Operating System 91

P

PCC 95
PCCSW 91, 96
PG2000 Directory 18
PG2000 Screen 65
PL2000 95
  Editor 35
  Error Correction 37
  Example Program 23
  Program Code 35
Preliminary Notes 23
Processor Layer 26
Program 24
Program Start, First 18
Programmable Computer Controller 95
Programming Environment 66
Project 24, 25
  Create 29, 83
Project Management 29, 65, 83
Project Name 30
Project Planning 27
PV Monitor 49, 95

R

README File 10
Reset the PCC 49

S

Scope 39, 94
Screen
  PG2000 65
  Programming Environment 66
  Project Management 65
SERVICE Mode 96
Settings, Application Specific 15
Statement List 59, 91
  Example 59
STL Editor 61

Symbolic Name 96
Syntax Check 36
System Menu 72
System RAM 96
System Requirements 12
System ROM 97

T

Tabulator, Size 61
Task 24, 97
  Compile 42
  Create 42
  Cyclic Task 98
  Delete 49
  Download 52
  Influence with PG 49
  Non-cyclic Task 95
  Overload 52
  Remove 49, 52
  Restart 52
  Start 49, 52
  Stop 49, 52
  Transfer to the PCC 48
Task Class 25, 97
Task Classes
  Start 53
  Stop 53
Task Layer 25, 33
Task Parameters 97
Task Symbol
  Insert 34, 55
Temporary Files 16
Title Line 65, 66
Transfer, Task => PCC 48

U

Updating an Old Version 14

V

Variable Declaration 24, 38, 58, 61, 97
Variables
  Reading 49
  Writing 49

W

Windows
  Changing Windows 74
  Close 73
  Maximize 73
  Minimize 73, 74
  Move 73
  Scrolling Contents 75
  Window Elements 72
  Working with Windows 71
  Zoom 74

Z

Zooming a Window 74