
Programming System PGS67

(c) ertec GmbH 06/2005
Am Pestalozziring 24
91058 Erlangen
Germany
Phone 49 9131 7700-0
Fax: 49 9131 7700-10
eMail: info@ertec.com
Internet: www.ertec.com

Table of Contents

Overview	1
First approach	1
Immediate operation	1
Extensions	1
File PGS67.DOC	1
Hotline	1
Internet	2
Error protocol	2
Declaration of conformity	4
Getting started	5
State of Delivery	5
Hardware Structure	5
Power supply	6
Software	6
Data channel	7
Initializing	8
Hardware	9
Components / Items delivered	9
Master module	10
Light emitting diodes (LEDs)	10
Acoustic signals	10
Firmware update	11
Programming modules	12
Power Supply Unit PGS67P	13
Repair	14
Technical data	14
Production Software Interface PSI67	15
Overview	15
Features	15
Installation	16
Operating modes	16
2 Operating modes	16
Operation mode "Application"	17
To start the program	17
Operating console	18
Concurrent mode	20
Operating mode "Supervisor"	22
Overall View	22
Configuration	22
Edit of Programming jobs	27
HEX editor	36
Logbook of programming jobs	40
Logbook of insertion cycles	41
DOS-Software	42

In general.....	42
Files.....	42
Data channels	43
Initializing the file PGS67.INI	43
Communication between PGS67 and host computer	43
Syntax	45
General invocation format.....	45
Sequential order of parameter evaluation.....	45
Abortion	45
Selecting a device type.....	45
Function parameters.....	46
Operation parameters (Options)	47
Check sums	48
Silicon signature	49
Format types of programming files	50
Binary format.....	50
Relative formats	50
Absolute formats.....	50
16/32 bit data	50
Directly readable formats	50
Automatic recognition	51
Non-automatically recognised formats.....	51
Output-File Format.....	52
Data filtering	53
Extracting data from files.....	53
Number of programming data.....	54
Data partition of devices	54
Diversity of functions	54
Screen displays	59
Operation messages	59
End message on error-free completion	59
End message with error information	60
System messages.....	63
Process message in the file PGS67.ERR	63
Error level.....	64
Batch processing (batch-, make files)	66
Auxiliary programs	67
Graphic presentation via PGS67ERR.EXE	67
Examples.....	69
Program invocations.....	69
Example of a batch file	70
Programming modules	71
Overview.....	71
Index	72

Overview

First approach

The purpose of this manual is to enable you to easily approach your new programming system PGS67 and to help you become acquainted with its features as quick as possible. You are recommended to take your time to study this manual. Thus you will learn about all capabilities and shortcuts provided as well as how to select the optimum processing methods for your individual tasks.

Immediate operation

The software handling meets the generally agreed standards. If your time is tight and you want to program your first device without delay, you are recommended to follow the steps outlined in chapter „Getting started“ , page 5. This section briefly explains how to install the system, connect it to the PC, switch it on, as well as how to try the basic functions. In case of problems or if you need more explanation about the functions offered, please check with the index listed in the appendix of this manual.

Extensions

Besides a continuous extension of the system's device data base and operating functions, we regularly take care of updating this manual's contents. To make certain that we can reach you in the future for news coming up, please, provide us with your individual data by filling in and returning the attached registration card.

File PGS67.DOC

To check the very latest developments, you ought to access the file "PGS67.DOC" which is part of the DOS software package. It may contain valuable additional pieces of information for your tasks.

Hotline

Dial +49 9131 7700-0 for questions or comments. In case that a problem needs to be solved, please provide the following data first:

- serial no. of your unit (see label on the bottom side of the master module, marked no.100),
- version no. of the software and device data base (see message on screen),
- your operating system (WINDOWS 95/98/NT)

- software used (DOS or PSI67)
- kind of file format in use (created by which Assembler/Compiler/Locator? 8/16 bit application?),
- device type to be programmed,
- program invocation you selected.

If your problem cannot be solved via phone, kindly submit the completed error protocol by via fax or eMail to: hotline@ertec.com.

Internet

On our homepage www.ertec.com you receive current information of our total offer. As a registered customer (after sending back the registration card or with a valid software-service-agreement) you are able to download or install directly the current software including the device data base at any time.

Error protocol

In order to analyse your problem clearly and to being able to provide an early solution, we need to receive the following information from you.

In the sense of our common interest in a fine customer support you are kindly requested to fill in our error protocol form and to submit it via fax to:

ertec GmbH, Germany , +49 9131 7700-10

or send the information via email to

hotline@ertec.com

ERROR PROTOCOL

to ertec GmbH, at fax +49 9131 7700-10, email: hotline@ertec.com

Serial number of master module (see bottom plate): 165... ..	Name /type of host computer:
Installed operating system:	PGS67-software used (version) (DOS , PSI67 ?)
Version of driver software at DOS (or date of file "PGS67.EXE"):	Version of data bases (or date of files "PGS67.DB? "):
Version of master firmware	Types of the programming modules in use
Device to be programmed/ device mode (printed on device package, kind of device)	Programming module used (Serial number → see bottom plate):
Format of the file to be programmed:	Do you work with serial data transmission or with Ethetnet? (IPX or TCP/IP, in system or point to point)
What is the contents of your INI-file (PGS67.INI or PSI67.INI)? Please add your listing if of assistance!	
Describe single steps of programming, listhe calls or batch-files (DOS) or send us your file JOB.DBF, PGS67.LOG (use PSI67) for the purpose of analysis.	
What problem did you find? When does it happen? If it is possible please name the precise error message you received.	
Sender´s adress / Phone no./ Email: Company´s name / Department/ Name:	

Declaration of conformity

This is to certify that the product PGS67 meets the conditions of the corresponding standards of the European Community. Violation of the instructions for appropriate usage of this product given in this manual lead to the annulment of this declaration.

This declaration is issued by:

ertec GmbH Elektronik und Automatisierung
Am Pestalozziring 24
91058 Erlangen
Germany

The related tests were carried out at:

EMV - Testhaus GmbH
Gustav-Hertz-Str. 35
94315 Straubing
Germany

ERG - Elektrotechnische Revisionsgesellschaft mbH
Reetzstr. 58
76327 Pfinztal
Germany

The evaluation was based on the following standards:

EN 50081-1 (EN 55022:1987 Class B)
EN 50082-2 (IEC 801-2, IEC 801-3, IEC 801-4)
EN 60950:1992 + A1:1993 + EN60950/A2:1993

Erlangen, December 12, 1995

ertec GmbH



Konrad Kruse

Getting started

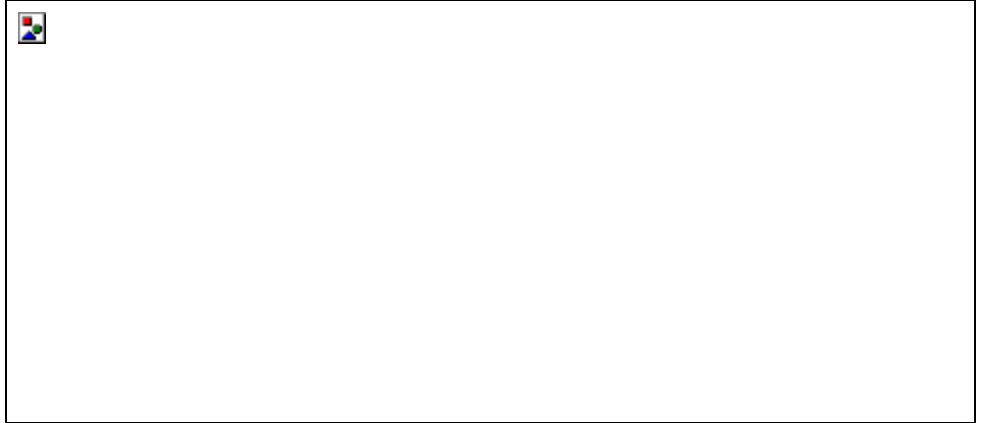
State of Delivery

As a rule, the unit comes assembled completely: The modules are inserted into the chassis. However, only in case that this is not applicable (e.g. if you have purchased one or more modules in order to complete your existing chassis), please, follow the instructions as described below:

Caution: Please use the original connectors, cables only!

Hardware Structure

- Put the chassis (eloxed black aluminum profile) onto a bench in front of yours.
- Please record the master module's serial number (black PVC-housing marked '100'). The serial number is labeled on the module's bottom plate. The last three figures of the serial number are also noted on the upper side.
- Insert the master module into the very left slot of the chassis (slot no. 1). For this purpose, lay down the rear housing edge into the rear chassis track (the nose of the rear housing bottom to fit into the chassis' notch), carefully press the module into the connector on the backplane in the chassis and fix the module using the screw on the front margin.
- According to this pattern you may insert the available programming modules into the remaining slots of the chassis. The sequence is at your choice.



PGS67 Basic Set Medium

Power supply

- A power cable is fixed to the chassis. Connect it to the delivered power supply.
- Connect the power supply to 115VAC- or 230VAC-mains.
- Turn on the programming system using the ON/OFF switch on the rear of the master module. After this, the light emitting diodes (LEDs) are lit at the master module and the programming modules. In addition, an acoustic signal is audible for some seconds.
- Now, turn the programming system OFF again.

Caution: Please, use the original power cable only!

Software

Components

PGS67 comes with 3 software packages:

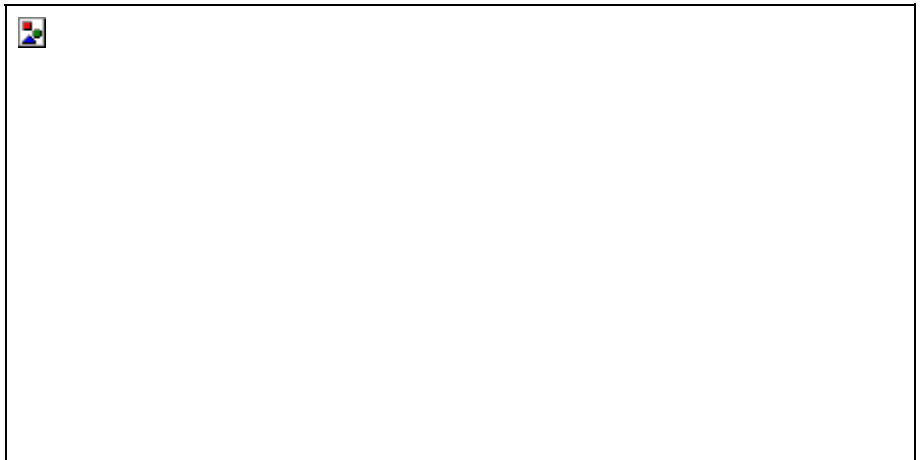
- **PSI67**: a Windows-Software designed for application at production sites
- **PGS67.EXE**: a DOS-Software designed for DOS-skilled users

Installation

Insert the delivered CDROM and invoke SETUP.EXE. Follow the instructions to install the complete software as recommended.

Data channel

The available data channels are Ethernet and RS232C.



Rear of the master module

Instructions to use Ethernet:

- On the rear of the master module, a connector is available to use Ethernet.
- Use an appropriate data cable (Ethernet 1:1) for installing the connection to your computer net. Possibly, it is necessary to interpose a HUB (conversion of 10Base2 in 10BaseT).
- If you want to work with point to point-contact (PCS67 - PC), you have to use a crossover combined cable and connect it with the networkcard.
- After switching ON two signal tones are audible for a short time, indicating the successful finishing of the autotest and the realisation of a properly functioning connection to the Ethernet data channel.
- If the DOS Software is installed enter the command:

```
PGS67 ? -xipx:nnn
```

- Configurate PSI67 like above for Ethernet-data interface with the respective configuration menu.

(nnn symbolises the serial number of your individual master module, which is labeled on the master module's bottom plate.)

Instructions to use RS232C:

- Connect the delivered RS232C-cable to the 9-pin SUB-D-connector on the rear of the master module. Connect the remaining end to your computer (COM1, COM2). If necessary, put the delivered 9/25-adaptor in between.

Caution: Please, use the original data line only!

- After switching ON two signal tones are audible for a short time, indicating the successful finishing of the autotest and the realisation of a properly functioning connection to the RS232C data channel.
- If the DOS software is installed enter the command:

```
PGS67 ? -xsio:n,b
```

- Configure PSI67 like above for serial data interface with the respective configuration menu.

(n symbolises the number of the COM-port, b stands for the Baudrate in Bit/s - 9600...115200)

Initializing

Now, the programming system is initialised and by working with DOS-software a syntax list and the configuration of the your PGS67 appears on the screen (order of programming modules within the complete system).

```
PGS67 command-line-driver V1.31 (c)ertec GmbH
usage: PGS67 command [-option] [-option] [...]
command:
?      more information i      device identifier
w      write e                devive erased?
r      read p                 print types
c      compare #<hex>[,...]   special function
d      duplicate s            calculate checksum

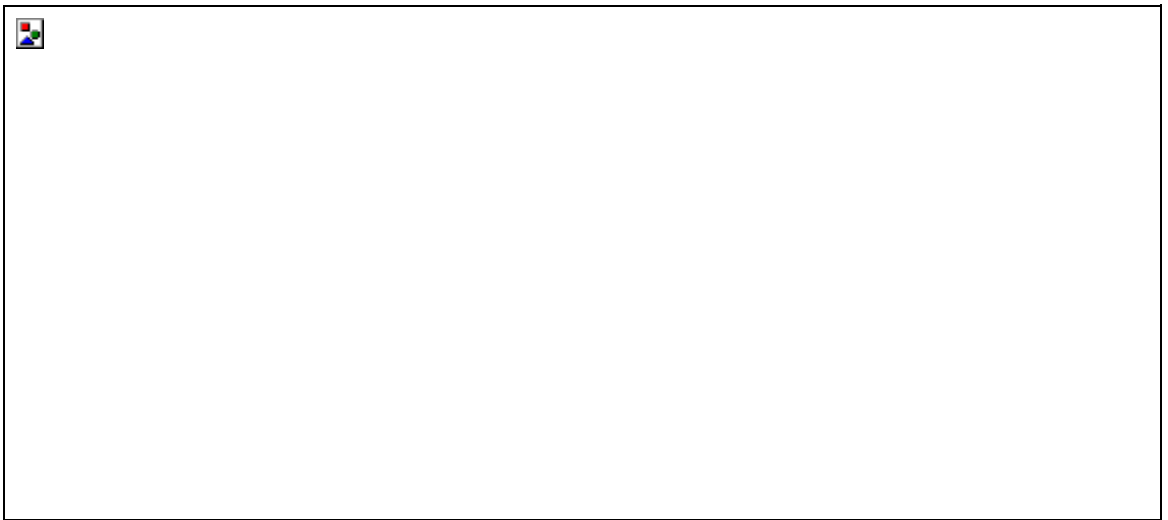
option:
e<txt> device type b(..)      data mode
f<txt> filename d(..)        device mode
t<type> file type c(<req-crc>[,...]) verify checksum
of<hex> offset in file x<protocol> protocol parameters
od<hex> offset in device s(<dec>[,...]) slot number(2.15)
l<hex> data length swap      big endian

Slot 01 master is active (V2.12/1)
Slot 02: module 200 is active
Slot 03: module 200 is active
Slot 04: module 650 is active
Slot 05: module 410 is active
```

By working with PSI67 after start of the suitable software the current module placement is shown graphically on the screen.

Hardware

Components / Items delivered



Basic System Medium with external power supply PGS67P

Power supply, chassis, master module

The basic components for the programming system PGS67 comprise a power supply, an aluminum chassis and a master module inserted therein. This module controls the data exchange between the host PC and the programming modules in use.

Programming modules

The actual programming functions are realised by the various programming modules which can be accommodated by the chassis in all kinds of sequences (beginning with slot no. 2).

Marking of modules

On the upper right of each module, a 3-digit type-specific module number is imprinted. The master module always is named 100, programming modules bear numbers from 200 and up.

Further items delivered

Part of the items delivered are also:

- a CDROM with the complete software (PSI67, DOS)
- one RS232C-data cable including one 9/25-adaptor

- Ethernet cable
- this operating manual

Master module

Power switch

The power switch is located at the rear of the master module. This switch activates the power for the complete programming system.

Caution: This power switch is only responsible for the programming system itself. The external power supply unit is equipped with a separate power switch!

RS232C-Connection

The rear of the master module provides a 9-pin SUB-D connector for the RS232C data channel. The pin assignment is organised for the use of a 1:1 cable to be connected to a standard PC-connector (COM1, COM2). This type of cable is part of the items delivered as standard.

Ethernet-Connection

The rear of the master module provides also an Ethernet connector.

The 8-pin Western connector (similar to phone connectors) is suitable for the connection of a Twisted Pair (10BaseT)

Light emitting diodes (LEDs)

Once the internal operating voltage is reached each LED of both the master module and the connected programming modules are lit. During the stand-by- mode after turning on the programming system, the LEDs are lit dimly. An inconstant bright illumination indicates that the module(s) involved are processing data currently.

An error is indicated by a periodical steady blinking of the master module's LED, which cannot be influenced or stopped by commands to the programming system.

Acoustic signals

The master module is equipped with an acoustic signal generator:

- Two brief signals confirm the successful finishing of the autotest and the properly functioning connection of a data channel to the master module. After this signal sequence the PGS67 is prepared to accept commands from the host computer.
- One long beep warns that the communication to the host computer cannot be established, e.g. if there is no proper cable connection between the computer and the programming system, if the network connection is chattered or if the host is not switched on while communicating via the RS232C data channel.
- During operation and while executing a command from the host computer, the programming system produces short cracking noises of varying frequencies in order to keep the user informed that a command is being processed.

Firmware update

Due to an optimum flexibility the master module's firmware is stored in a reprogrammable Flash-EEPROM device. In case that an update is required in the future the manufacturer will provide a disk containing both the firmware to be programmed and all related necessary programs. Follow the steps mentioned below:

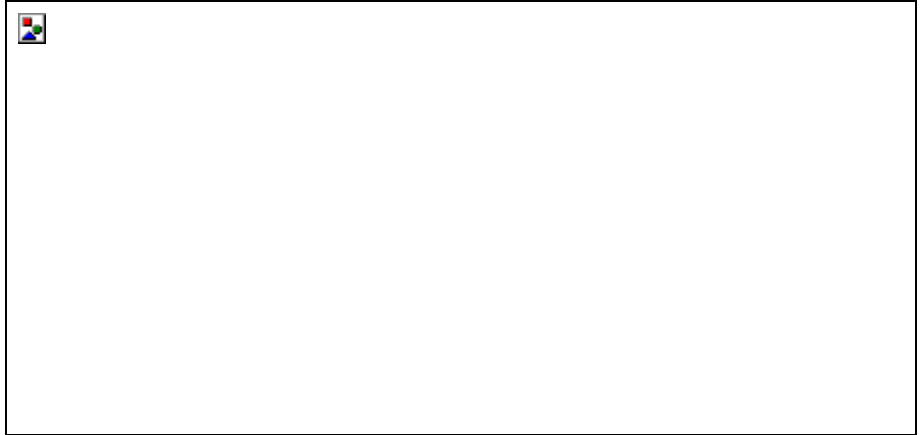
- Insert the master module into slot no. 3 of the chassis. In case this slot is already engaged by a programming module, please remove it for the time of updating.
- Connect the RS232C of your PGS67 to COM1 of your host computer. Switch on the power supply as well as the master module.
- In case COM1 of your host computer is already engaged, you can also use COM2. If so, please do not miss to substitute COM1 through COM2 by editing the file PRG67.BAT.
- Copy all files of the "master firmware disk" into a separate sub-directory on your hard disk.
- Start the file PRG67.BAT from this sub-directory via entering:

```
PGS67
```

- After being asked enter the serial number of your master module as IPX-MAC-address(see bottom metal plate of the master module). Entering the lowest ordered digits will do (e.g. 122 for 1650000122). Confirm this entry via <RETURN>.
- The input of the ITCP/IP-address is handled like above(UPD-address).
- After approx. 1 minute the processing of the firmware update via the batch file PRG67.BAT will be concluded. In case you receive an error message on screen, please contact the ertec-hotline.
- Now, please switch off the master module as well as the power supply. Reinsert the master module into slot no. 1 of your chassis. Also reinsert the programming module into slot no. 3 which if it had been inserted there before you started the firmware update.
- After connecting the data channel (RS232C or Ethernet) you can keep on working within your PGS67-directory as before. The sub-directory, which had been installed for the firmware updating only, may be deleted now.
- When entering <PGS67 ?> in DOS the current firmware version is indicated on screen (see information in brackets), e.g.:

```
Slot 01:  master is active  (V2.12/1)
```

Programming modules



Programming modules

Light emitting diodes (LEDs)

Once the internal operating voltage is reached each LED of all connected modules are lit. During the stand-by-mode after turning on the programming system, the LEDs are lit dimly. An inconstant bright illumination indicates that the module(s) involved are processing data currently.

Programming sockets

The programming modules accommodate the programming sockets on their top side. Depending on the type of programming module its programming sockets are suitable for various device packages and device types. The appendix of this manual describes the currently available programming modules and their applications.

Marking

The programming sockets are marked with letters commencing with "A". The left front socket corresponds to "A", those sockets next to A towards your right hand side are "B", "C" etc. In case one programming module accommodates two rows of sockets the second row is marked counter-clockwise (counting pattern like DIL devices).

Appendix

Individual information regarding the various programming modules are listed in the appendix of this manual.

Power Supply Unit PGS67P

Usage

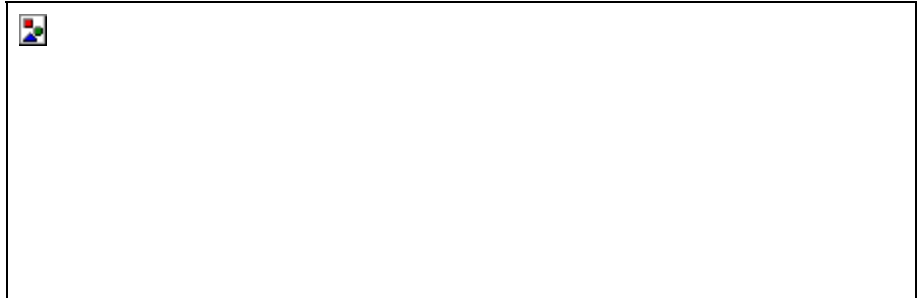
The power supply unit PGS67P is designed to provide the required power to operate the programming system PGS67.

Output voltage

The unit supplies an output voltage of 13,5VDC/10A. This voltage is internally set as a firm value and must not be varied. The unit is short circuit- and overload protected. In case of an overload the unit changes automatically from the voltage regulation mode into the current regulation mode.

Input voltage

The unit provides two possible input voltages: 230V or 115V ($\pm 10\%$), each between 50 and 60 Hz..



Front cover of the power supply PGS67P

Power cable

As far as possible stick to using the delivered original power cable. In countries with deviating power plug systems you should use an appropriate cable set procured directly from your home market. In this case, only use a 3-wire power cable with a ground wire.

Caution

The power supply unit is only permitted to be connected to a protected mains connector.. Only 3-wired protected power cables may be used.

Visual check

Before putting the unit into operation you need to visually check the unit and the power cable on possible damages. In case you discover any damage the unit must not be connected to mains!

Environment

The unit is only permitted for operation in dry places.

Mains connection

- Plug the power cable into the protected mains connector.
- Switch the unit on via the power switch at the front cover plate: the green LED is lit now. Switch the unit off again.

Connection to the programming system

- Connect the power cable of the programming system PGS67 to the 3-wire secondary connector situated at the front cover plate.
- To disconnect press the release lever marked "PUSH".

Transportation

For transportation please use the original packaging resp. an appropriate packaging which sufficiently protects the system against shock, heat and moisture. Never transport the unit with connected power supply cables.

Storage

The unit may only be stored in dry places.

Maintenance

The unit does neither require a regular maintenance nor calibration.

Repair

A repair or any other hardware access may only be carried out by authorized qualified personnel.

Always make certain to remove the power plug before opening the unit.

Technical data

Input power voltage	115 / 230VAC \pm 10%, 50/60 Hz
Nominal input current	4,8 / 2,7 A
Output current	10 A
Output jacks	Pin 1: negative potential Pin 2: positive potential Pin 3: earthing contact / shield
Operating environment	temperature: 0...40 degrees Celsius humidity: 20 up to 80 %, non-condensing
Noise emission	<70 dB(A)
Weight	3000 g
Dimensions	177 x 68 x 275 mm (width x height x depth)

Production Software Interface PSI67

Overview

The PSI67 is an interactive operating software for the programming system PGS67, designed for applications in the production area.

The PSI67 runs under the operating systems WINDOWS 9x / NT / 2000 / XP.

In particular, you can utilize the PSI67 to:

- prepare and arrange programming jobs for production-related applications
- select and start programming jobs in the production area
- define label texts and formats or generate labels on the production site when using the label printer, type , P67LP1
- protocol the processes in terms of their history and as per job
- edit files hexadecimally and prepare them for production

Features

Standard

The PSI67 is part of the items delivered as standard coming with the programming system PGS67. The present version meets the basic requirements of a quality-oriented production facility.

Extensions

The conditions of the typical data structure, work organization and production scheduling are realized on a general basis. In-depth and application-specific functions are:

- administration and selection of additional production data
- aggregation of order data
- automatic storage and updating of program data in the local area network
- integration of barcode readers and -printers for the production control and quality assurance
- check-inquiries for critical device- and job selections
- acquisition of operator inputs and work procedures for statistical evaluation

- acquisition and reporting of work processes for the logistics and procurement department
- acquisition of work processes for the controlling of maintenance measures (life times of programming sockets)
- safeguarding of certain functions (setting of programming jobs) via passwords against unauthorized actions

Upon a common specification and agreement we can extend the PSI67 towards an individual solution.

Installation

Via Internet

On our homepage www.ertec.com you can install directly with the link "download software."

CDROM

If you have received the software on a CDROM from us:

Insert the CDROM and invoke SETUP.EXE

Operating modes

2 Operating modes

The PSI67 provides 2 operating modes.

- The operating mode Application comprises the actual application in the production area. Select this operating mode if you start the PSI67 directly without any additional parameters.
- The operating mode Supervisor is responsible for the configuration of programming jobs and the hardware settings. This operating mode is selected by starting the PSI67 with the parameter "Supervisor".

Operating mode „Application“

To start the program:

Start/Programs/ertec Programming System PGS67/Production Suite

Alternative: PSI67.

Operating mode „Supervisor“

To start the program:

Start/Programs/ertec Programming System PGS67/Administration Suite

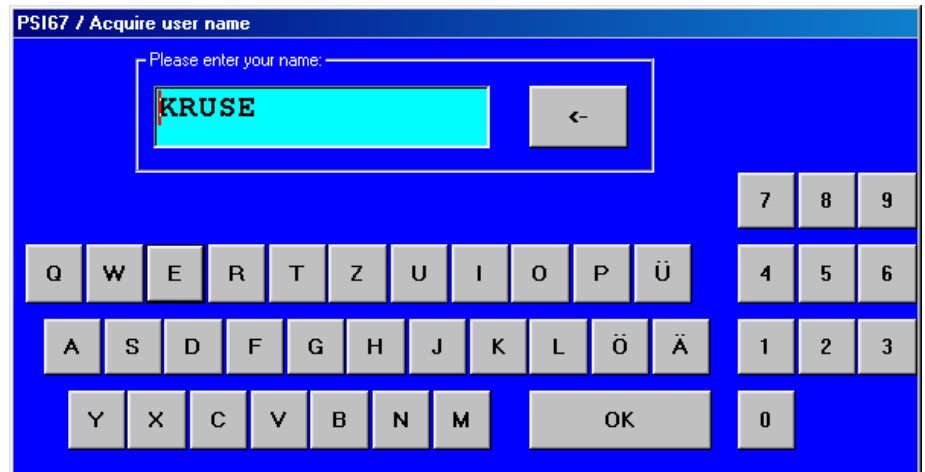
Alternative: **PSI67 supervisor**

Operation mode "Application"

To start the program

Depending on the installation on you computer, start the program through a double-click on the image **[PSI67]** on your screen respectively through pressing the allocated function key. After the program has started you will be asked for the operator name (when it is configured that way).

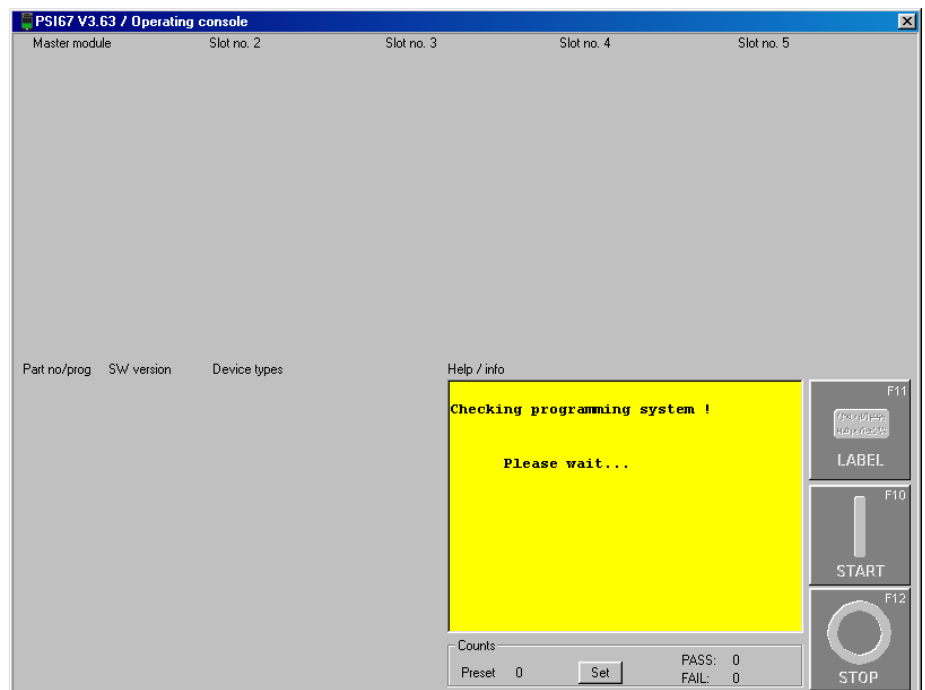
Aquire operator's name



- Enter your name via the keyboard or by clicking the letter cells.
- the cell [**<**]- allows you to make corrections
- Click on [**OK**] to accept the entered data.

Checking the Programming System PGS67

The screen shows:



The Programming System PGS67 is initialized and the module placement is inquired into.

Error upon initializing

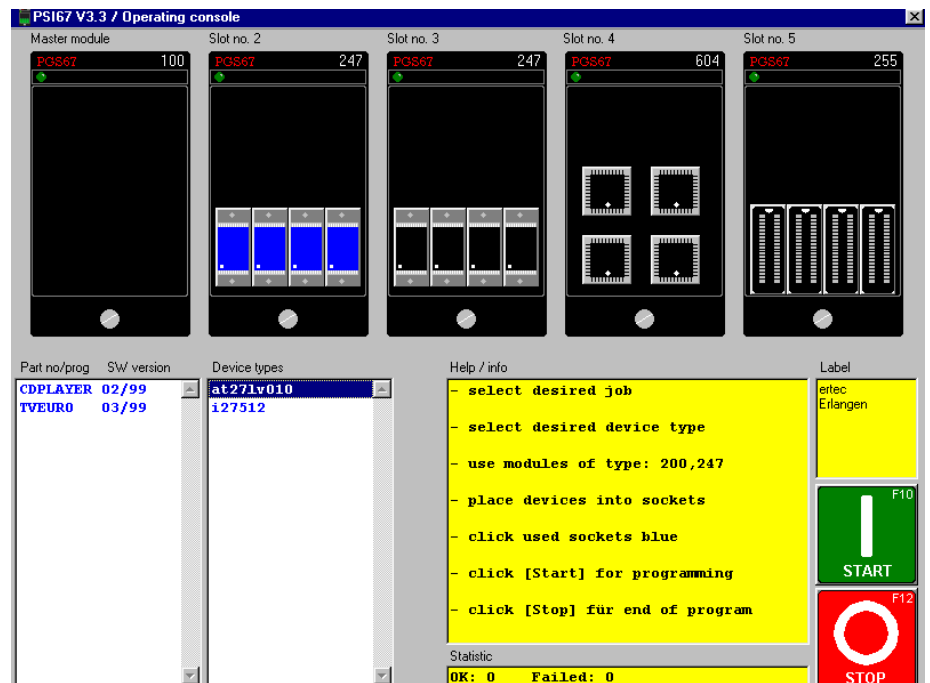
An error is reported approximately as following:



In such case, close the program down and try to recover the error as outlined!

Operating console

When the communication with the programming system PGS67 works properly, the operating console below is available on screen:



In particular, the screen provides:

- the placed module types in the upper part of the screen; the appropriate sockets for the current programming job are marked in blue.

- the list of available programming jobs on the bottom, left; the most recently selected programming job is marked with a blue background.
- next to this on your right, a list of possible device types; the most recently selected device type is marked with a blue background.
- below this, a list holding the appropriate programming modules
- next to this on your right, a current note on the operation
- next to this on your right, the operating buttons for further operations (label / start / stop)

To set the programming job

Click on the requested programming job. The windows is updated accordingly.

To set the device type

Click on the requested device type.

To set the sockets

Sockets turned on are marked with a blue background, sockets turned off are marked with a black background. You can activate the requested configuration by clicking the sockets one by one.

Within one module, you can only turn on sockets commencing at the first socket. The software provides support by correcting possible inappropriate combinations.

To place devices

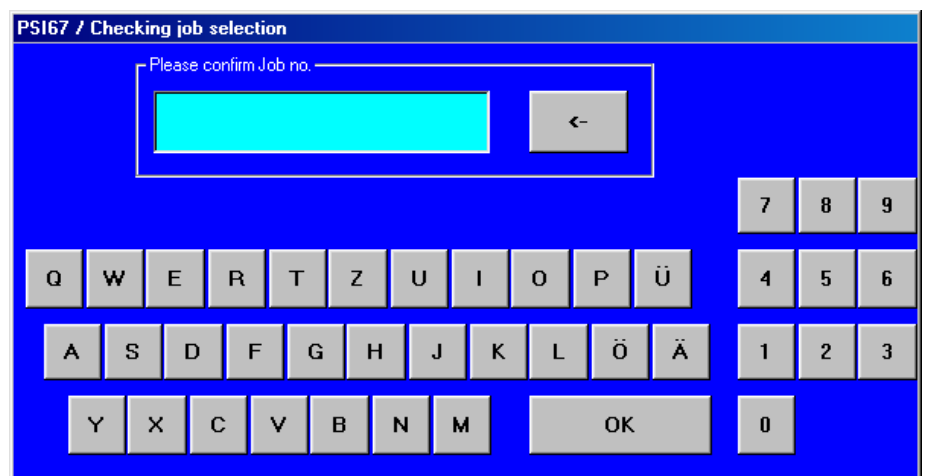
Place devices into the sockets marked in blue.

To start the programming job

Click on the button [START] or press the key [F10].

To verify the selected programming job

If configured the user has to confirm the selected programming by entering the name (partNo./program) into the window:



The colour of the modules turned on change from blue to yellow.

Progress indicator

Within the master module, a progress indicator is shown, providing information on the time remaining to finish the programming job and the currently proceeded programming step.

The progress indicator uses the time flow of the previous programming job as a reference; therefore, in terms of the initial start of a programming job, such data are not relevant.

Completion the programming job

After ending the programming job, each of the sockets are marked according to the success:

- devices in sockets marked green are properly programmed, i.e. error-free
- devices in sockets marked red are faulty; information on the error occurred is provided after clicking on this socket

To print labels

Provided that a label printer is connected, you can start a label printout by clicking the button [Label] or by pressing the key [F11] (it is possible to printout labels while programming-yellow sockets).

To remove devices

Remove the devices and store them in an appropriate manner. After this, click on the button [START] or press the key [F10]. Active sockets are presented in blue again.

To restart the programming job

A restart of a programming job is only possible after passing a time lock. This is to prevent devices in the state " green" from being placed. Thus, confusion among programmed and non-programmed devices is taken precaution of (pause error!).

To end the program

Click on the button [STOP] or press the key [F12]. The program is closed down.

Concurrent mode

Overview

The operating mode Concurrent enables each and every programming module to work on its own. After the devices are placed into a programming module, it starts its programming job immediately. During this time, the remaining programming modules may accommodate their devices. Thus, an optimum efficiency of the programming system is achieved. However, it bears a higher risk of operating errors (mixing up sockets and devices).

To start a module

After selecting the job, the sockets of the first module are shown in blue. The devices for this module can be placed. After this, press [START]. These sockets are now indicated in yellow and sockets of the next module are shown in blue (? release for device insertion).

Reporting of a module finished

Once a module is finished, the sockets are shown in green resp. red. Remove the devices therefrom and press [START]. After a waiting time, these sockets are shown in blue again to indicate the release for inserting devices anew

The end of the job

If you wish to end the selected job, click off the sockets appearing in blue. As soon as further modules become ready, you can remove devices therefrom. After this, consequently, also click off these sockets. Once all running programming procedures have been ended in the described manner, the button [STOP] is activated. Now, you can either select another programming job or end the program.

Progress indicators

Within this operating mode, the progress indicators are located in the head of the single modules. The indication of the currently proceeded sequence is available on the top of each module.

Operating mode „Supervisor“

Overall View

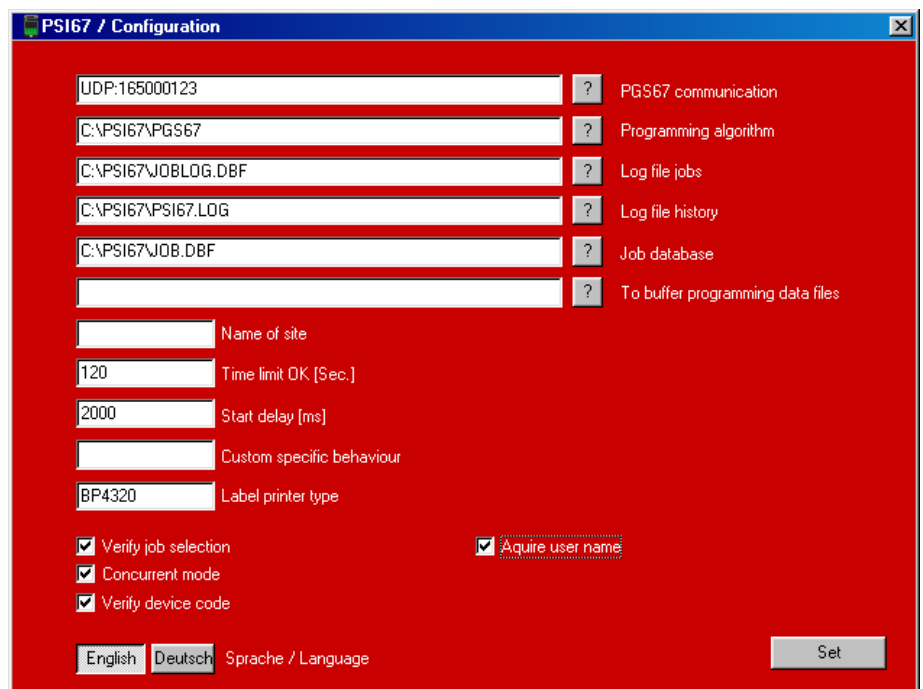
The operating mode Supervisor allows you to:

- set the hardware configuration
- set single operating functions
- define the programming jobs
- edit the different Log-files
- analyse, optimize and modify programming files and calculate their checksum

The operating mode “Supervisor” is activated when you invoke the program with the parameter “Supervisor” (PSI67 SUPERVISOR).

Configuration

Start the program in the operating mode “Supervisor” (invocation by “PSI67 supervisor”). The main menu is shown.



There, select [Config]. The window **[Configuration]** is shown.

PGS67 communication

- Enter via which channel the programming system PGS67 is connected
SIO..., if serial via COM-channel,
UDP..., if via Ethernet / TCP/IP,
IPX..., if via Ethernet / IPX
- Click [?] and you may select interactively from the available communication channels

Programming algorithm

The files 'PGS67.DB0', 'PGS67.DB1' and 'PGS67.DB2' specify how the systems performs the programming steps. You will update these files while updating and expanding the device type library.

- Click [?] near to **[Programming algorithm]** and select via explorer-dialog a path and database file. You can also edit this input field directly, e.g.:
- Please ensure the proper updating of these files by setting up appropriate rules in your administration.

Log file jobs

The results of completed programming jobs will be reported in a database file.

- Click [?] near to [Log file jobs] and select via explorer-dialog a path and a directory for this log file job. You can also this input field directly, e.g.:

Log file history

The user actions, specific system messages and exceptions will be recorded in a text file.

- Click [?] near to [Log file history] and select via explorer dialog path and a directory for this log file history. You can also edit this input field directly, e.g.:
- Please pay attention to the ever growing of this file, everytime you invoke the PSI67 software and thereby the capacity of your fixed disk reduces. Sometimes you must control the size of this file und if necessary delete it.

Job database

The defined programming jobs are held by a database file.

- Click [?] near to [Job database] and select via explorer dialog path and a directory for this job database. You can also edit this input field directly, e.g.:

To buffer programming data files

Optionally you may select a mode in which the files to be programmed will be copied and held in a local directory. This feature is dedicated for usage inside a local area network if programming data files are held at a server. So you can reduce network usage.

- To ensure correct data programmed we recommend urgently to complete each programming job by a checksum / CRC check.
- Click [?] near to [To buffer programming data files] and select via explorer dialog path and a directory for this buffering. You can also edit this input field directly, e.g.:

Name of site

You may define a name of the set up working place. This name will be used internally for different log functions, custom specific release controls and for generating datecodes to be printed on labels.

Please pay attention, that in this operation mode only those jobs are shown, for which was defined the adequate name of the programming place.

File updating

Click [?] and then you can define, whether the directories/files should be updated or not and in which periods.

Time limit OK [sec]

- By entering values in seconds you may limit the duration of displaying the results of completed jobs. By using this feature you can eliminate the known „pause failures“.

Background of this time limit is, that after the users interrupted the operation of the system and do not know the last action when they return, they can mix up programmed devices with not programmed ones.

Start delay [ms]

- By entering values in milliseconds you may disable directly switching from completing a job to start the next job. By using this feature you may eliminate the known „mix up failures“.

Background of this start delay is, that after the users interrupted the operation of the system and do not know the last action when they return, they can mix up programmed devices with not programmed ones. The start delay enforces the user to put single steps of the programming in order and to receipt them. After the job is completed (green or red sockets), the devices must be picked out of the sockets and placed in a tray. Then the start button must be pressed and when the sockets turn blue on the screen the devices must be placed in the sockets. The next programming job will be started when you press the start button a second time.

Custom specific behaviour

- If individual functions have been implemented for you, enter your company's name here.

Label printer type

- If connected, enter the type of label printer here (e.g. P67LP1).
- Before you can use your label printer you have to setup a windows printer „Universal / no text“.
- Using WINDOWS NT at first install the driver from CD, rename to P67LP1 and select under properties "TEXT"
- Using WINDOWS XP add a new printer and select manufacturer "Standard" and type "Generic/Text"
- Connect the label printer to the communication channel defined by the printer setup (e.g. LPT1).

Verify job selection

- If the switch is selected the user has to confirm the selected programming job by entering the job no. before the processing of the job.

Concurrent mode

- Here, you can activate the operation mode "Concurrent mode".

Thus, an optimum efficiency of the programming system is achieved. However, it bears a higher risk of operating errors (mixing up sockets and devices).

Please use the operation mode "concurrent" only if the users are carefully instructed and if they are in charge of a attendant.

Verify device code

- This switch allows you to set, whether or not the internal identification of the devices (ID- and manufacturer code) is automatically checked before each work process.
- Indeed, not in every case this switch is useful and does mean a higher rate of security, because many programmable devices (e.g. microcontroller) have no internal device code. In such cases, the switch is ignored by the software.

Aquire user name

By activating this feature PSI67 aquires the user name each time the programm will be started in application mode. The user´s name will be logged.

To save changes

By clicking [Set] the values are written into the file PSI67.INI and the program can be started anew.

PSI67.INI

You can also configure the behaviour by making changes in the file PSI67.INI via a text editor.

A typical contents shows as following:

```
[USAGE]
; to set English language
language=ENGLISH
; to set concurrent mode
concurrentmode=0
; to activate a window for checking the selected job:
checkprogn=1
; to limit the time [seconds] showing OK-sockets:
oklimit=10
; to force a break between pick und place of devices
StartDelay=
; to log job results in a database *.dbf:
joblogfile=C:\PSI67\joblog
; to log actions in a text file *.log:
logfile=C:\PSI67\PSI67.LOG
; ff. items are inserted automatically
startdate=27/02/99
starttime=22:58:26
stopdate=27/02/99
stoptime=22:59:37
lastselected=1
lasttype=4
lastjobtime1=9
lastjobtime2=9
lastjobtime3=9
lastjobtime4=9
lastjobtime5=9

[PGS67]
; to define the data channel the programming system PGS67 is connected:
; i.e.: channel=SIO:1, 57600 (COM1, baudrate=57600 Bit/s)
; i.e.: channel=UPD:128.0.0.043 (TCO/IP, IP-adress:1628.0.0.043)
; i.e.: channel=IPX:165000043 (IPX, Master P67100-SN: 165000043)
channel=IPX: 165000043
; to define the location of files PGS67.DB? (algorithm database)
; i.e.: Files located in directory C:\usr\tmp
Database=C:\USR\TMP\PSI67

; to activate checking of device ID codes:
CheckIdentifier=0

[PRINTER]
type=

[JOBS]
; to define the location of the file job.dbf
dir=C:\psi67\JOB.DBF

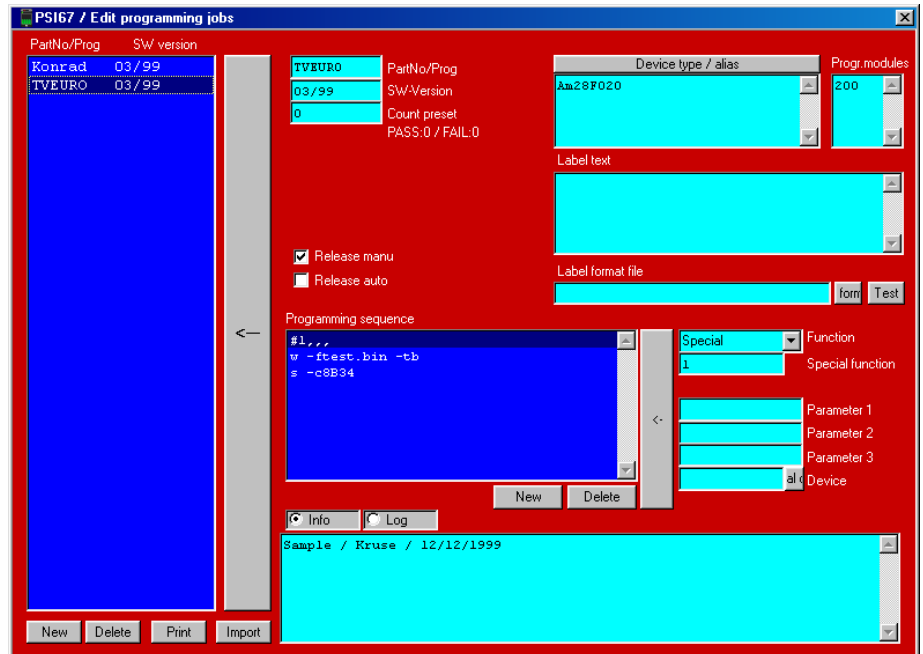
[CUSTOM]
; to activate a custom specific behaviour
name=

[ROBOTER]
; to use inside a automatic handling system
name=
```

Edit of Programming jobs

Start the program in the operating mode „Supervisor“ (invocation by „PSI67 supervisor“). The main menu is shown.

There, select [Jobs]. The window „Edit Programming Jobs“ is shown:



Job list

Click on a job on the left side of the list. The job's contents is indicated in the cells on the right side.

Now, you can change the single entries and, if requested, save them in the job list via the cell [B].

Via the cell [NEW] you can enter a new job. The values of the currently selected job are used as a preassignment.

Via the cell [DELETE] you can delete selected jobs out of the list.

Via the cell [PRINT] you can print all listed jobs and the accompanying informations. When you click you firstly get a printing preview.

Via the cell [IMPORT] you can import inactively single jobs from the job database into another directory. That means adding to the current job list.

PartNo / Prog

In this cell you must input the name or the term of the programming job.

This name/term appears in operation mode "Application" in the shown job list.

If you have configured the option [Verify job selection], the user will be asked for this name/term in operation mode "Application" the first time at each change of the selected job.

SW Version

In this cell you must input a number/term (e.g. software version) for the further description of the job.

This term appears in the operation mode „Application“ in the shown job list as SW-Version.

Count present

In this cell you can define as instruction, how much devices (exclusiv the fail-devices) from this job should be programmed.

PASS / FAIL –RESET

The number of Pass- and fail-devices (correct and missprogrammed devices), which appear correspond to the current number of the job, which is a model when you define something anew.

Click **[RESET]** to put numbers to 0.

Release manu

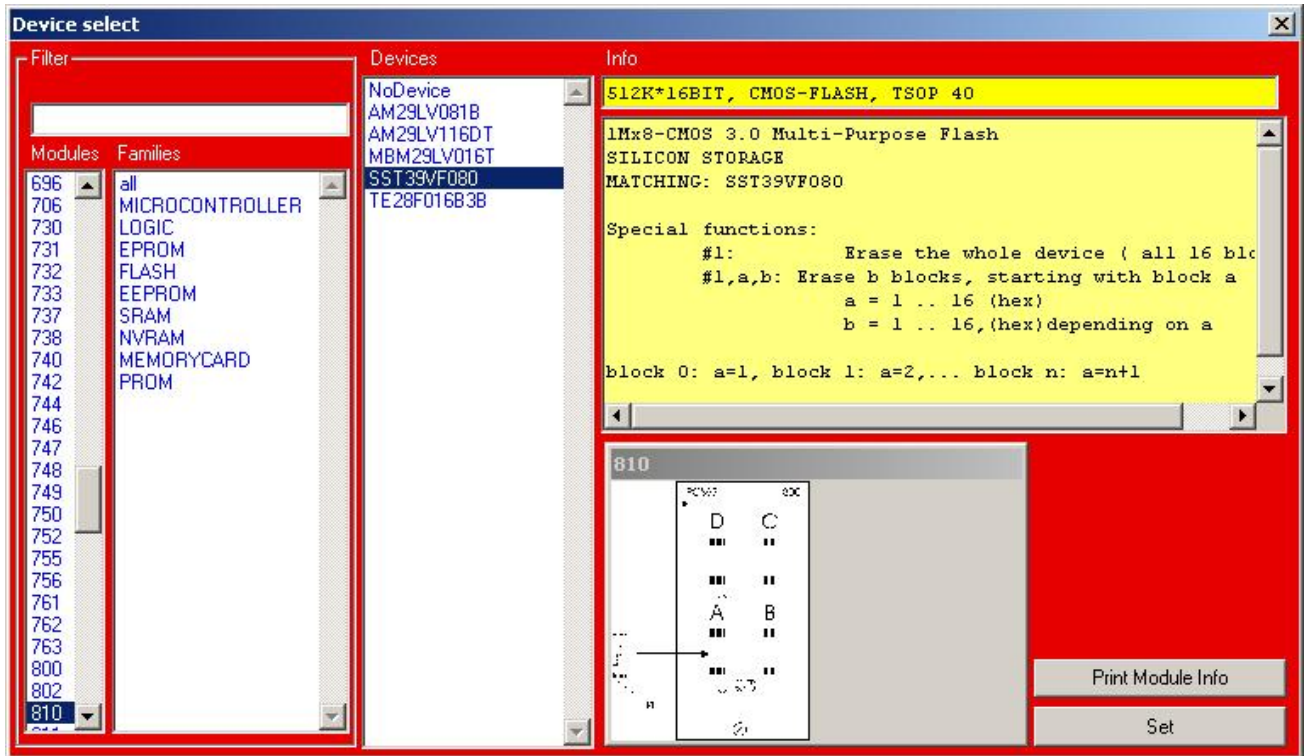
If the job should be processed (also) on a PGS67-programming-system, you have to activate this option.

Release auto

If the job should be processed (also) on a programming roboter (BEAVER or MH240E), you have to activate this option.

To explore and to set device type

Click at [Device type / Alias] . The window " Device select" is shown:



Here you can search purposeful for device types. In the yellow cell appear further information of the current device. You can filter optionally via the both lists in the left of the screen window for programming modul-no. or device families (EPROM, Flash, Microcontrollers,...). Additionally you can use filter terms.

At bottom of the window is shown a basic graphic of the selected Programming Module. You may reposition or zoom the picture via simple mouse interaction.

Click [SET] and the current selected device will be accepted.

If you want later to list for the user another term, you can select marked with "/" an alias-term, e.g. at27lv010/at27lv010a.

Programming modules

Enter a list of possible PGS67-programming modules in the cell „Programming modules“.

Format file for label (.lab)/Laser (*.mkh)*

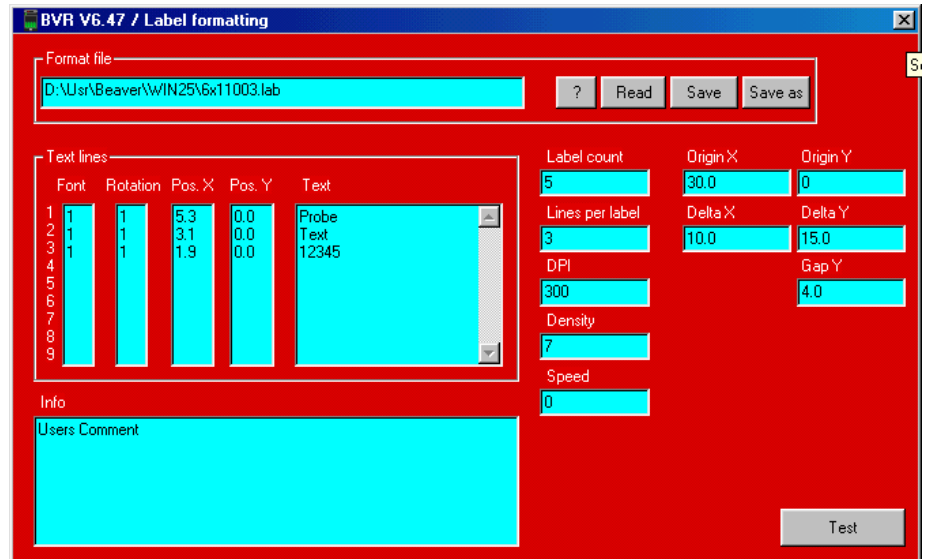
Under [Label format], enter the name of a file with label formats. There, details are defined as to which texts in what size, position, orientation, number, ... are printed.

Click [?] and select via explorer-dialog a defined and tested label format file (file expansion LAB).

To setup and check label formats

If you do not have a tested label format file for the label you want to print or if you want to change the parameters concerning the printing of the label (font size, posture, arrangement, density, ...), you can create such files here and test it online.

Click [TEST] and the following window appears:



- To print one row of labels click [Test]
- The printed label will be formatted due to the actual settings of the selected label format file. If parameters are changed you have to save the settings into the label format file by clicking [Save]
- Select for density values 2 ... 14
- Select for speed: 0; speed values 1..3 will speed up the printer but will decrease quality of printout
- Delta-X is measured from label to label in printing direction
- Delta-Y is measured form label to label abeam of printing direction
- Gap-Y is measured between labels in printing direction
- Caution: the first 1.5 mm of labels cannot be printed
- Font selection:

Standard:	0 ... 4
horizontal enlarged:	5 ... 9
vertical enlarged:	10 ... 14

Label text

Under [Label text] enter which texts are to printed on the labels. Texts with multiple lines are accepted.

Datecode

You may generate automatically generated datecodes by inserting a line:

%datecode%

The generated datecode is i.g.:

070300AB for 07/03/2000, first and last letter of site name

Grafic

You may insert grafics by inserting a line:

%pcxfile%filename.pcx

Filename.pcx has to be a file with pcx-coded grafics.

Special small fonts

Via

www.zebra.com/sd/utilities.htm

you may download a software tool , Font Downloader" to download selectable additional fonts to the label printer P67LP1.

For example the font , Arial, 10pt, Width=3, Cell size right=2" is best to print up to 15 characters / 10 mm.

To activate a downloaded font set the value "FONT" inside the window "Label formatting" to 97 for Font "a", 98 for font "b" and so on.

Please note: Due to selected rotation you have to download a proper rotated font (see setup of the Font downloader)

Programming sequence

Under " Programming sequence" , enter the sequence of steps to be carried out one after another. The syntax is in accordance with the DOS-software for the PGS67. However, here the syntax is created interactively via the cell group that is positioned on the right of the cell.

Via the cells [New] and [Delete] you can insert new sequence steps or delete existing ones.

Once you click on a sequence step, its contents is expanded into the cells right next to it. There, you can make changes which then are saved via the cell [<-].

To program date, time, week of year

Invoking PSI67-Software in application mode a binary file named etime.bin will be created in the PSI67 path. This file contains following system data in different formats (HEX, BIN,..):

year, month, day, time, week of year

Using this file it is possible to program devices with these data.

Struktur of file etime.bin e.g. for 11/06/2002 14:33 (week 45):

addr:	data in HEX:	ASCII:
000000	32 30 30 32 31 31 30 36 31 34 33 33 34 35 02 00	20021106143345..
000010	00 02 01 01 00 06 01 04 03 03 04 05 02 2D FF FF-

To program serial numbers

The programming of serial numbers may be part of a programming sequence. In this case, each device is allocated to a successive number.

Select the function **[SerNo]**. In the field below, give a file name whereto the program PSI67.EXE shall automatically deliver the current serial number. Further, you can determine, from which number the allocation shall be activated, and, from which address the numbers are to be programmed into the devices.

You can choose to give the serial numbers as byte, word or double-word (32 bit). After reaching the end-value, counting starts at 0 again.

Default: little endianness, activating SWAP switch to big endianness

Activating the switch VERIFY the programming step "Verify serial numbers" will be defined (normally after "Programming serial numbers").

Examples:

- address: 0xA00: start-serial number: FE
result: socket A: address 0xA00: FE
socket B: address 0xA00: FF
socket C: address 0xA00: 00
socket D: address 0xA00: 01
- address: 0xB00: start-serial number: 0102 ,
SWAP deactivated (little endianness)
result: socket A: address 0xB00: 02 address B01: 01
socket B: address 0xB00: 03 address B01: 01
socket C: address 0xB00: 04 address B01: 01
socket D: address 0xB00: 05 address B01: 01
- address: 0xB00: start-serial number: 0102 ,
SWAP activated (big endianness)
result: socket A: address 0xB00: 01 address B01: 02
socket B: address 0xB00: 01 address B01: 03
socket C: address 0xB00: 01 address B01: 04
socket D: address 0xB00: 01 address B01: 05
- address: 0x800: start-serial number: 01020304 ,
SWAP deactivated (little endianness)
result: socket A: address 0x800: 04 801: 03 802: 02 803: 01
socket B: address 0x800: 05 801: 03 802: 02 803: 01
socket C: address 0x800: 06 801: 03 802: 02 803: 01
socket D: address 0x800: 07 801: 03 802: 02 803: 01
- address: 0x800: start-serial number: 01020304
SWAP activated (big endianness)
result: socket A: address 0x800: 01 801: 02 802: 03 803: 04
socket B: address 0x800: 01 801: 02 802: 03 803: 05
socket C: address 0x800: 01 801: 02 802: 03 803: 06
socket D: address 0x800: 01 801: 02 802: 03 803: 07

Please note, that after programming a serial number, the devices have different check sums. Therefore, either insert a calculation of the check sum (CRC-check) before the programming of the serial number, or, leave the section with the serial number away, when calculating the check sum.

If you want to use a special algorithm for programming of serial numbers, it is possible, that we can conform it for you. Entry the name of such a special serial number method into the field [Method]. The latest implemended methods are:

Special SN methods

Serial number method	Description
SN2416LE	<ul style="list-style-type: none"> • runs 24 Bit (from 00.00.00 upto FF.FF.FF) • Each Byte goes to 16 Bit (one word) of the device to be programmed (Bit no. 8... 15 = 0) • Disposition: little endianess (LSB -> MSB)
SN2416LE811	<ul style="list-style-type: none"> • runs 24 Bit (from 00.00.00 upto FF.FF.FF) • Each Byte goes to 16 Bit (one word) of the device to be programmed (Bit no. 8... 15 = 0F, i.e. for 12 Bit words of PIC devices) • Disposition: little endianess (LSB -> MSB)
SN2416LE813	<ul style="list-style-type: none"> • runs 24 Bit (from 00.00.00 upto FF.FF.FF) • Each Byte goes to 16 Bit (one word) of the device to be programmed (Bit no. 8... 15 = 3F, i.e. for 14 Bit words of PIC devices) • Disposition: little endianess (LSB -> MSB)
SN2416BE	<ul style="list-style-type: none"> • runs 24 Bit (from 00.00.00 upto FF.FF.FF) • Each Byte goes to 16 Bit (one word) of the device to be programmed (Bit no. 8... 15 = 0) • Disposition: bit endianess (MSB -> LSB)
SN2416BE811	<ul style="list-style-type: none"> • runs 24 Bit (from 00.00.00 upto FF.FF.FF) • Each Byte goes to 16 Bit (one word) of the device to be programmed (Bit no. 8... 15 = 0F, i.e. for 12 Bit words of PIC devices) • Disposition: bit endianess (MSB -> LSB)
SN2416BE813	<ul style="list-style-type: none"> • runs 24 Bit (from 00.00.00 upto FF.FF.FF) • Each Byte goes to 16 Bit (one word) of the device to be programmed (Bit no. 8... 15 = 3F, i.e. for 12 Bit words of PIC devices) • Disposition: bit endianess (MSB -> LSB)

Device

Under the list with the programming sequences you can instruct purposefully differing devices for single steps of sequence via this box. This is useful e.g., when you use different device types for different physical clusters (e.g. Microcontroller with EPROM and EEPROM-memory).

Recommendation

To make certain to generate actually safe and plain work procedures, we urgently recommend you:

- to use the „HEX“ file format only
- not to use offsets, neither to use the switch **[SWAP]**. Instead, prepare an appropriate HEX-file in its place.

Info and history log

In the cell [Info] you may notice comments to the selected job. To see and edit the comments click [Info].

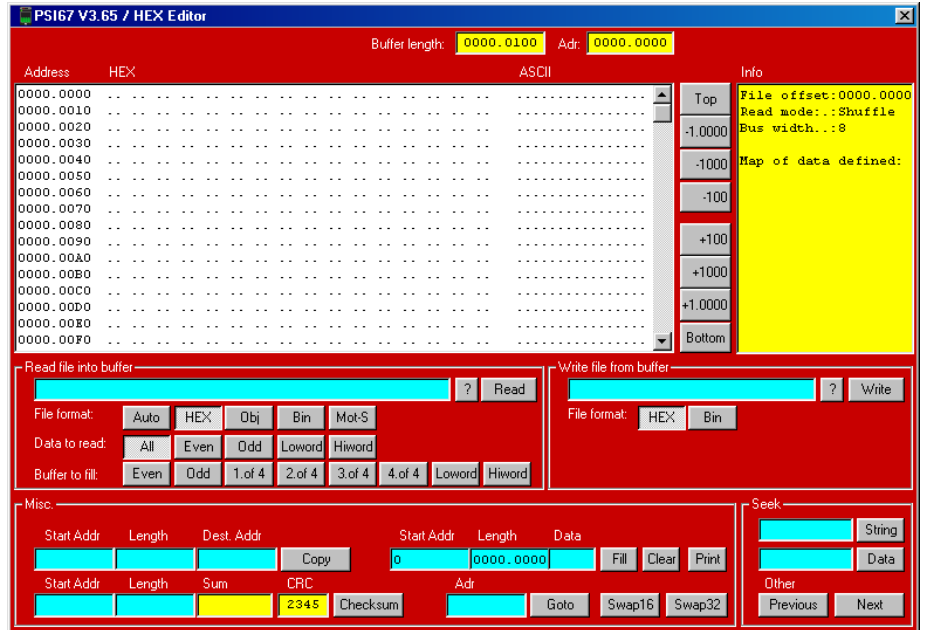
The following text is shown in the yellow window in the operation mode "Application" after the selection of the job, by placing the word "user info" in front of the actual recommendation.

By clicking on [Log] you may analyze the reported log of any changes made to the selected job.

HEX editor

Start the program in the operating mode "Supervisor" (invocation via "PSI67 supervisor"). The main menu appears on the screen.

There, select [EditFile]. The window "HEX editor" is shown.



Data window

The white data window shows the currently loaded data as hexadecimal data and ASCII-code. Undefined sections are marked by ".". You can:

- change single data after clicking on them
- position or scroll via the cursor-keys
- scroll with the slider on the very right; the current address is faded in a yellow cell
- position blockwise resp. Jump directly to the beginning or the end by using the buttons right next to it

Info

The yellow window on the right hand side edge presents:

- the current file offset (for PSI67 the file offset is always set to 0)
- a list with information of the current memory area (memory map)

Read a file into buffer

Select a file format:

Auto	the programm autodetects the file format
HEX	to read Intel 8/16 Bit Hex files
OBJ	to read Intel 8/16 Bit Object files
BIN	to read binary files
MOT-S	to read Motorola S2/S3 files

Select the way how to filter data from the file to be read / Data to read:

ALL	all data are processed
EVEN	only Bytes at even addresses are processed
ODD	only Bytes at odd addresses are processed
LOWORD	only Words at even word addresses are processed
HIWORD	only Words at odd word addresses are processed

Alternatively you may specify the way how the editors buffer will be filled up / Buffer to fill:

EVEN	only Bytes at even addresses will be written
ODD	only Byte at odd addresses will be written
1.OF 4	only the first Byte of double words will be written
2.OF 4	only the second Byte of double words will be written
3.OF 4	only the third Byte of double words will be written
4.OF 4	only the fourth Byte of double words will be written
LOWORD	only the first two Bytes of double words will be written
HIWORD	only the last two Bytes of double words will be written

Via the button [?] you can select the file to be read; or you can enter the file name directly into the blue cell.

Write a file with buffer contents

Select a file format:

HEX	to write Intel 8/16 Bit Hex files
BIN	to write binary files

Via the button [?] you can select the file to be written; or you can enter the file name directly into the blue cell.

Using the button [WRITE], the preselected file is written or created new.

Copy

In order to copy data blocks to different addresses, enter the following information into the blue cells: start address, length and the target address. You can also define the start address and length directly in the data window: Use the left mouse key and click on the date of the start address, then use the right mouse key and click on the final date you want to copy. You can start the copying procedure with the button [Copy].

Fill

In order to pre-assign data blocks, enter the start address, the length and a data value into the corresponding blue cells. You can also define the start address and length directly in the data window: Use the left mouse key and click on the date of the start address, then use the right mouse key and click on the final date you want to copy. You can start the assigning procedure with the button [Fill].

Clear

You can erase the complete editor contents with the button [Clear].

Calculation of checksum

In order to calculate the checksum, enter the following information into the blue cells: start address, length and the target address. You can also define the start address and length directly in the data window: Use the left mouse key and click on the date of the start address, then use the right mouse key and click on the final date you want to have calculated. You can start the calculating procedure with the button [Checksum]. Then, the result appears in the associated yellow cell; on the left as a 32-bit value for the arithmetical sum, on the right as 16-bit CRC-sum.

Goto

In order to jump to a certain address, enter the address into the associated blue cell and click the button [Goto].

Swap

In order to manipulate the current editor contents according to the swapping technique, click on the buttons [Swap 16] resp. [Swap 32].

With Swap 16 the contents are swapped bitwise; data at odd addresses are swapped with the data at even addresses.

With Swap 32 the swapping is handled via 4 byte.

Search

You can search for strings and for data sequences. Enter the sequence to be searched for into the associated blue cells and click the button [String] resp. [Data]. The search is started at the current position.

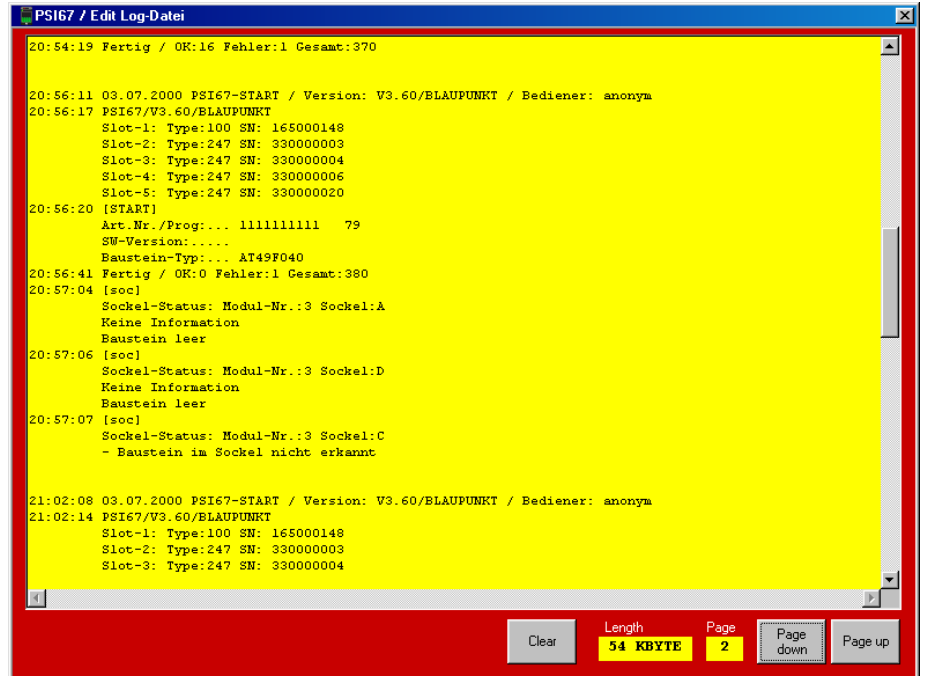
In order to search for the next or previous date, which deviates from the one at the current position, click the button [Previous other] or [Next other].

Click [Previous other] resp. [Next other] to search for the next or the previous date, with is different from the current position.

Analyse the log file history

Start the program in the operating mode "Supervisor". (invocation via "PSI67 supervisor"). The main menu appears.

There, select [Log+Statistics] and [HistoryLog]. The window "edit log file" is shown on the screen:



```
PSI67 / Edit Log-Datei
20:54:19 Fertig / OK:16 Fehler:1 Gesamt:370

20:56:11 03.07.2000 PSI67-START / Version: V3.60/BLAUPUNKT / Bediener: anonym
20:56:17 PSI67/V3.60/BLAUPUNKT
Slot-1: Type:100 SN: 165000148
Slot-2: Type:247 SN: 330000003
Slot-3: Type:247 SN: 330000004
Slot-4: Type:247 SN: 330000006
Slot-5: Type:247 SN: 330000020

20:56:20 [START]
Art.Nr./Prog:... 1111111111 79
SW-Version:.....
Baustein-Typ:... AT49F040

20:56:41 Fertig / OK:0 Fehler:1 Gesamt:380
20:57:04 [soc]
Sockel-Status: Modul-Nr.:3 Sockel:A
Keine Information
Baustein leer

20:57:06 [soc]
Sockel-Status: Modul-Nr.:3 Sockel:D
Keine Information
Baustein leer

20:57:07 [soc]
Sockel-Status: Modul-Nr.:3 Sockel:C
- Baustein im Sockel nicht erkannt

21:02:08 03.07.2000 PSI67-START / Version: V3.60/BLAUPUNKT / Bediener: anonym
21:02:14 PSI67/V3.60/BLAUPUNKT
Slot-1: Type:100 SN: 165000148
Slot-2: Type:247 SN: 330000003
Slot-3: Type:247 SN: 330000004

Clear Length Page Page down Page up
54 KBYTE 2
```

The yellow cell shows the contents of the current log file including a listing of the relevant actions in historical order.

You can also take a look at the log file by using a simple text editor. The name is defined in the file PSI67.INI.

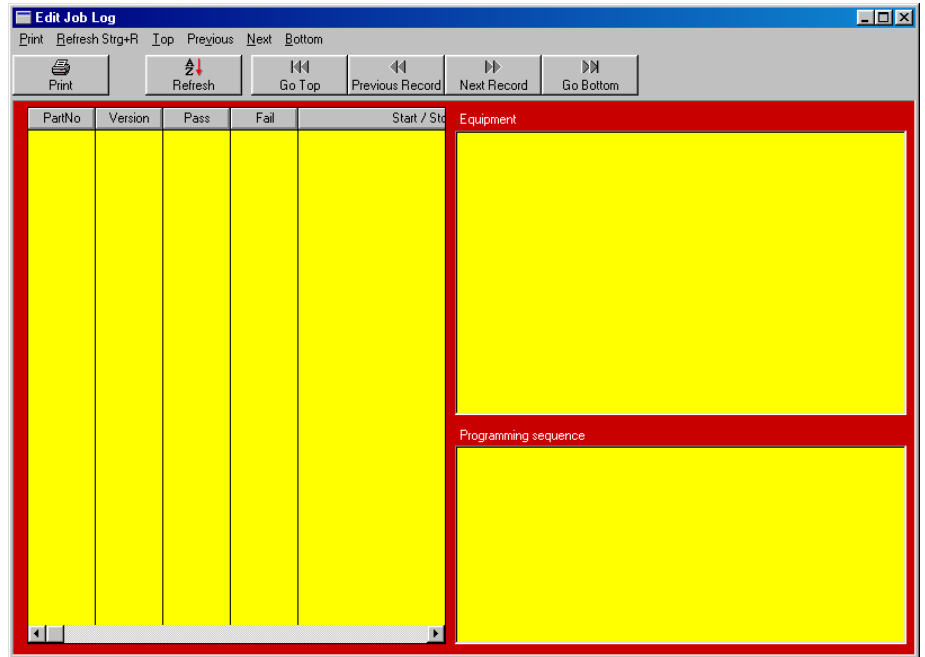
You can clear the log-file via the button [Clear].

Logbook of programming jobs

In the logbook of the programming jobs the results of the finished programming jobs are archived in a database using a packed form. This is important for, e.g., the tracability of events.

Start the program in the operating mode "Supervisor" . (invocation via "PSI67 supervisor"). The main menu appears.

There, select [Log+Statistics], [JobsLog] and [EDIT]. The window "Log of Jobs" is shown on the screen:



You can see the following bits of information:

The programming jobs carried out, the name of operator, the duration of programming jobs as well as their results (number of errors, quantity of OKed devices).

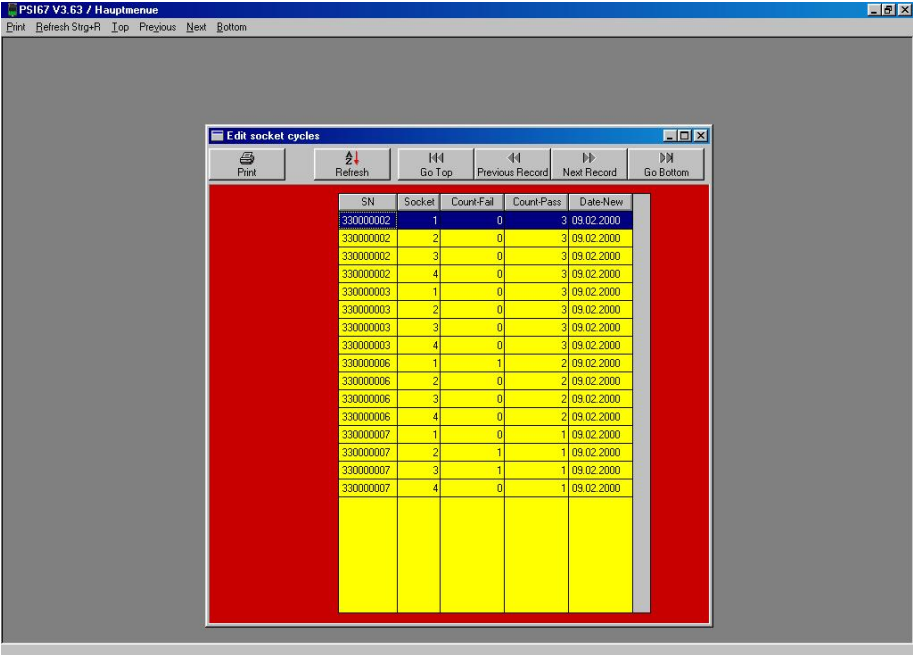
By clicking [Print] you may preview and print a list of the joblog database.

Logbook of insertion cycles

In the logbook of insertion cycles the count of passed and failed insertion cycles of each programming module and each socket is reported. You may use this feature to decide, if a module or special sockets have to be refurbished due to end of their life cycle.

Start the program in the operating mode " Supervisor" . The main menu appears.

There, select [Log+Statistics] and [Insertion cycles]. The window " Edit socket cycles" is shown on the screen:



The screenshot shows a software window titled "PSI67 V3.63 / Hauptmenue" with a menu bar containing "Print", "Refresh Stg+R", "Top", "Previous", "Next", and "Bottom". Inside the window, a smaller window titled "Edit socket cycles" is open. This window has a toolbar with "Print", "Refresh", "Go Top", "Previous Record", "Next Record", and "Go Bottom". The main area of the "Edit socket cycles" window contains a table with the following data:

SN	Socket	Count-Fail	Count-Pass	Date-New
330000002	1	0	3	09.02.2000
330000002	2	0	3	09.02.2000
330000002	3	0	3	09.02.2000
330000002	4	0	3	09.02.2000
330000003	1	0	3	09.02.2000
330000003	2	0	3	09.02.2000
330000003	3	0	3	09.02.2000
330000003	4	0	3	09.02.2000
330000006	1	1	2	09.02.2000
330000006	2	0	2	09.02.2000
330000006	3	0	2	09.02.2000
330000006	4	0	2	09.02.2000
330000007	1	0	1	09.02.2000
330000007	2	1	1	09.02.2000
330000007	3	1	1	09.02.2000
330000007	4	0	1	09.02.2000

By clicking [Print] you may preview and print a list of the socket cycles.

DOS-Software

In general

The program PGS67.EXE allows you to operate and control the functions of the programming system via easy commands. Such commands may be integrated into batch-processing (batch-files) or make-definitions. By combining a number of commands and by analysing the "error codes" you receive reliable, clear and reproducible working procedures, both ideal to automatise complex tasks and to plainly instruct unspecialised personnel. Production tailored programming procedures may be realised through integrating operation graphics and statistic programs.

Files

If you have selected at the installation (CD or Internet) the DOS software, too, a subdirectory "cmd" was created, which contains the following files:

File	Contents
PGS67.DOC	Text file with most recent information which could no more become included to this manual due to timing reasons. You can take a look at this file with a text editor or using the DOS-command <PRINT> for a printout.
PGS67.EXE	This file represents the actual driver software.
PGS67.DB0 PGS67.DB1 PGS67.DB2	Data bases with the programming specifications of the implemented device types. The program "PGS67.EXE" analyses these data bases and normally expects these on the same directory as the one where it is located itself.
PGS67.INI	File with individually settable configuration information. You may edit this file with a text editor and enter parameters which do not need to be indicated again when invoking the program PGS67.EXE. It is strongly recommended to make a record regarding the definition of the data channel (RS232C resp. Ethernet) used. Detailed information on the syntax of the parameter definitions and on the general structure of this file are available in chapter "Initialisation file PGS67.INI".
PGS67ERR.EXE	File for graphic presentation of socket states before, while and after the programming procedure.
PGS67.DVL	ASCII-Text file with a list of those devices which have been implemented up to the date of delivery.
VERSION	Version number of the driver software and the data base files.

Data channels

The host computer can communicate with the programming system PGS67 via a built-in serial channel (COM1 or COM2). Baudrates of up to 115200 bit/s can be achieved by direct access to the serial communication device of the PC.

The communication via Ethernet, however, allows considerably faster reaction times than the communication via RS232C. The programming system may be integrated into established Cheapernet- or Yellow-Cable-Networks. A connection to Twisted-Pair cables is also accepted.

Initializing the file PGS67.INI

This file demands the following basic structure:

```
[INIT]
- definition of data channel
- parameter 1
- ...
- parameter n
[END]
```

This file may be modified via a text editor. We strongly recommend to make a record on the definition of the data channel used:

```
-xipx:nnn           ;Ethernet / serial-no. nnn
```

or

```
-xsio:n,b          ;RS232 / COM n, Baudrate b
```

Further, you can also set defaults regarding all possible function- and operating-parameters by recording in this file. For instance, it is useful to record the file format used as well as the device type and the device mode.

Example of a file-contents PGS67.INI:

```
[INIT]
-xipx:165000087    ;Ethernet / serial-no.
;-xsio:2,57600     ;RS232 deactivated
-tb               ;binary file format
-eAM27C010        ;device type AM27C010
-s2              ;module slot no.2
-d(a,b,c,d)       ;4 sockets parallel
[END]
```

Presetting

Without giving any further parameters while invoking resp. without any further parameters given in the file PGS67.INI the driver software automatically selects a baudrate of 19200 and communicates via COM1.

Communication between PGS67 and host computer

RS232C

The selectable serial channels in terms of RS232C (COM1..COM2) are driven interrupt controlled. Please make certain that the related interrupts are not engaged by any other units (e.g. network cards). The interrupts are assigned as following:

- COM1 - IRQ4
- COM2 - IRQ3

Basically, for communication via RS232C the following Baudrates are settable:

- 9600, 19200, 38400, 57600, 115200 Baud

Ethernet

The protocol used when communicating with the programming system is based on Novell's IPX. In order to take advantage of the fast communication procedures you both need a network card with the appropriate connector and an IPX.COM driver. This driver must be installed prior to start PGS67.EXE..

Every master module bears a unique network address ("Universal LAN MAC Address" in accordance with ANSI/IEEE Std 802). This address consists of an OUI ("Organizationally Unique Identifier", specific and authorised for ertec-products: 00-20-31) and the serial number of the master module. The operation parameter "-XIPX:nnn" calls for entering the network address; nnn stands for the serial number of the master module used. The software converts this value into the MAC-address. This parameter should be saved to simplify the invocation of the file PGS67.INI (state at delivery).

Syntax

General invocation format

```
PGS67 [function] [-option1] [-option2] [...]
```

By **[function]** you determine what is to be done. The function parameter consists of one or two digits.

By **[-option]** you determine how to execute the function. Like familiar from UNIX-systems, and more and more often used in the DOS-environment, a operation parameter is marked by a foremost "-"

You do not need to enter all parameters when invoking. You can also enter the parameters in the file "PGS67.INI". In case that essential parameters are missing the program responds with a message.

Sequential order of parameter evaluation

The parameters are interpreted as follows:

- First to come are the parameters from the file "PGS67.INI", which are listed in the directory of the file "PGS67.EXE" (global option file);
- followed by the parameters from the file "PGS67.INI" which is listed in the current working directory (local option file);
- followed by the parameters entered with invocation;
- data of a response file are integrated into the sequential order when invoking.

A parameter of the same kind, read later, overwrites the one that had been read earlier.

Abortion

You can abort a running operation by using the following key combination

```
CTRL+C (STRG+C)
```

However, due to the complex internal telegram structure, this might cause a reaction time of a couple of seconds.

A sudden abortion can be achieved by using the following key combination

```
Ctrl+Break (Strg+Untbr)
```

This method of aborting, however, requires to switch the programming system off and on again, i.e. the system must be rebooted to ensure a continuous correct functioning.

Selecting a device type

Device list

All implemented devices are listed by entering the following invocation:

```
pgs67 p (>devicelist.txt)
```

Since this list is rather extensive we recommend to use the DOS redirect operator (>) for writing into a file or to the printer.

The device type you wish to program must either be entered when invoking or given in the file PGS67.INI:

```
-eDeviceType
```

As a rule, this nomenclature corresponds to the information printed on the device itself. However, access times, mask versions or further code information are not entered.

To make certain that you select the appropriate nomenclature for your device you ought to take advantage of the device filter function. Please enter:

```
pgs67 p -e"String"
```

String defines an alphanumeric pattern according to which all suitable device types are filtered. The closer the entered string comes to the actually needed device type the preciser the filtering becomes. When entering "27C256" all device types are filtered that show this very string. This practical example covers all CMOS-EPROMs of type 27C256.

However, when you enter: "M5M27C101K", only this special device type is indicated. Once entered such a complete device name, in addition, information on the manufacturer, capacity, second sources and possible special functions are indicated on screen.

The following handling is recommended:

- Enter a rough partial string of the requested device type, e.g.

```
pgs67 p -e28f
```

All device types are listed on screen that match with the string "28F" , including the one to be programmed: AM28F020.

- Enter the complete name of the device

```
pgs67 p -eam28f020
```

The screen shows: type FLASH-EPROM, manufacturer AMD, device capacity and organisation 256k x 8bit and the description of the special function #1 to erase the complete device contents.

Module-dependent filtering of device types

Some device types are available in different packages, yet using the same nomenclature. These types are programmable on different modules, depending on their package execution and the socket hardware required. That is why the filtering of device type can also be handled as per the programming module defined:

```
pgs67 p -(etype) -mndec; dec = programming module no.
```

Example:

```
pgs67 p -e68hc -m682
```

This example shows all device types programmable with the programming module no. P67682, which contain the string "68HC".

Device list

The delivered file PGS67.DVL contains a list in ASCII-format with all currently (state of delivery) implemented devices.

Function parameters

The following function parameters (commands) are invocable using the pattern:

Parameter	Memo	Function
?	help	To list syntax and configuration of the programming system
w	write	To program data from a file
d	duplicate	To duplicate a master device in socket A to devices in sockets B,...,D.
r	read	To read and save in a file
c	compare	To compare with the data of a file
s	sum	To calculate check sums
i	identifier	To get the device- and manufacturer codes (silicon signature)
e	erased ?	To test if device is erased; (blank check)
p	print devices	To print/list the implemented device types; <ul style="list-style-type: none"> • if no type is pre-selected, all types are printed/listed • if a partial string of the type is given, all fitting types are printed/listed • if a complete type name is given, an additional information text is provided
#[no,val]	special	To start special function number no (with the value val)

Operation parameters (Options)

Together with a function parameter you can also give a number of operation parameters (options). These operation parameters always must be entered with a foremost hyphen each:

Parameter	Memo	Function
-etype	EPROM	To give the device type to be programmed; type stands for the device's nomenclature
-ffile	file	To give the file to be gotten or put. File stands for a DOS-file name (possibly incl. information on drive and path).
-tformat	file format	File format. format = a automatic (default) format = b binary format = h Intel-HEX format = o Intel-Obj, AOMF format = m Motorola-S format = j JEDEC
-ofhex	offset file	Start address, wherefrom the data of a file shall be processed; hex stands for a hexadecimal value of figures; counting method per byte. Default: 0
-odhex	offset device	Offset address, wherefrom data are stored in or gotten from a device; hex stands for a hexadecimal value of figures; counting method per byte. Default: 0

-lhex	data length	Length of the data to be processed; hex stands for a hexadecimal value of figures; counting method per byte. Default: capacity of the device type set.
-bw(n0,n1,..)	data mode	To enter the data that are to be selected from a file; w determines the bus width; n0,n1.. determine the order numbers of bytes to be selected. Default: all
-d(soc op...)	device mode	To enter the engaged programming sockets soc and the data partition among them: op= , = parallel op= + = gang op= : = split Default: -d(A)
-c(s0,s1...)	crc-sum	To enter the nominal value for the calculation of the check sum; if the calculated check sum is not equal to s0, s1.... an error message is to follow; s0, s1... are hexadecimal values. The number corresponds to the number of devices which had been set via the data mode and the device mode.
-xsio:n,b	serial port	To enter the serial ports for the data channels and to give the data transmission speed; n = 1..2 = COM1..COM2 b = 9600, 19200, 38400, 57600, 115200 Default: 19200 Baud
-xIPX:sn	IPX address	Network address of the programming system; sn stands for the serial number of the master module
-sno	slot no	Number no of the module slot(s) which accommodate modules for standing by for operation; counting direction from left to right; the master module's slot has number 1. Default: 2 (= the slot for the first programming module next to the master module).
-swap	endianess	big endian; the valence of bytes within words is swapped
-z(ignore)	ignore	Ignores addresses which do not take programmable memory cells in the device to be programmed
-pdpath\pgs67.db*	path / data base	Gives the path within which the PGS67- data bases to be used are located Default: same directory like PGS67.EXE

Check sums

The function parameter "s" determines two different check sums. Especially in the application of mass programming on your production site this method ascertains a considerable increase of programming security regarding error-

free device programming. For this purpose we recommend to give the CRC-nominal values by means of the option "check sum-verify -c(..)". The calculation of both check sums is executed at the lower and upper value of the Vcc as per the device manufacturer's instructions (as a rule at 4.75 V and 5.25 V). If the results deviate from each other this is an indicator that the data contents cannot be guaranteed for a longer period of time. In this case the error message "checksum error" is shown on screen. The two following algorithms come to effect:

SUM

All data are added and the overflows (from bit D₁₆) get lost. This sum, in comparison with sums calculated by other programmers, is usable, yet this method can neither exclude systematic- nor double-errors.

CRC

In accordance with the international standard "CRC-CCITT" this complex calculation begins with shifting the highest valued data bit and the generator polynom A142H is used. The expectable probability that two devices with the same CRC-sum do have the same data contents is specified with 99.997 %.

Due to the complex calculation procedures for the CRC-sum the final handling of this calculation process may take some time. This period of time depends on the device type's capacity and the number of devices inserted per module. Example:

approx. 20s for a device with 1Mbit memory capacity (e.g. 27010)

Silicon signature

A range of devices, especially EPROMs and Flash-EPROMs, contain an internally readable manufacturer's- and device-code (silicon signature). For a good reason, the PGS67 does not support an automatic device recognition.

- There is a large number of devices, above all microcontrollers, which do not have a readable device-code.
- The assignment of this code is not even entirely reliable regarding EPROMs.
- Once trying to read the internal code of such devices they may even become damaged.

Reading this so-called silicon signature is yet supported by the function "I". If you access a device that does not bear a silicon signature an error message is provided on screen (see below).

```
S02: socket A: Device has no ID !
```

In case the device does have a silicon signature the screen shows the manufacturer's- and type-code, e.g.:

```
S03: socket A: ident: 0000012A
S03: socket B: ident: 0000012A
```

Format types of programming files

As a rule, data for devices to be programmed are generated in a computer and saved on external data carriers (floppy disk, hard disk). Frequently, an assembler / compiler converts a program and a locator generates a file with the machine code for the target system. Unfortunately, up to date there is no standardised method regarding how to format the data in this file.

Binary format

The simplest presentation gives the data without additional information about the addresses in a straight ascending order, binary listed only. In this case the user has to carry out the so-called locating of the memory programming him-/herself, i.e. the start address from where the data are to be put into the device memory must be named explicitly.

Relative formats

The EXE-format used under DOS is structured as a relative format. Also in this case the start address from where the data are to be put into the device memory must be named explicitly. From these data and from the relative information in the file, the software calculates the absolute addresses for programming.

Absolute formats

In these formats the addresses of data are specified additionally and in an absolute format. This means for the procedure of memory programming that basically the data can be stored definitely in the device memory without user parameters. Further, such formats often provide extra debug-information which have to be overread while memory programming. Precise specifications of the various file formats can be achieved from the relevant publications of the soft- and hardware manufacturers (e.g. INTEL, Texas Instruments, Motorola).

16/32 bit data

The data are always formatted as 8-bit data. Therefore, regarding applications for target systems of greater bus widths, when programming EPROMs additional procedures for the resolution of the data structure are necessary (e.g. splitting). The software PGS67.EXE provides the required functions for

- setting the procedure to select the data from a file via the "data mode",
- setting the procedure to allocate the data in the device to be programmed via the "device mode".

Directly readable formats

PGS67.EXE supports the aforementioned formats directly, i.e. the single data are automatically extracted at the defined addresses from the various format structures. Thus, conversions into the INTEL-HEX-format are not necessary. Normally, the various types of formats are automatically recognised by the software.

Automatic recognition

Through the default setting "auto mode" the software PGS67.EXE evaluates the format of the file to be read. In case that none of the formats listed below can be recognised as valid, the file is evaluated strictly binary after having given a warning note. The first byte is located at address 0.

You can switch off the automatic recognition by parameters at invocation or by entries in the file "PGS67.INI"; the file will then be evaluated in the format as given.

Non-automatically recognised formats

The software cannot recognise all file formats doubtlessly. Firstly, not all file formats contain reliable hints to define their format. Secondly, ambiguous characters may occur in different formats either. Some compiler manufacturers use standard formats (e.g. INTEL-HEX or Motorola-S), yet extend the files partially by own information (e.g. debug symbols).

For this reason the following formats need to be specified clearly by invocation parameters or must be selected in the file PGS67.INI:

Format	Parameter
Binary format	-tb
INTEL-HEX formats, that do not correspond to the original Intel-HEX format	-th
Motorola-S-Formats, that do not correspond to the original Motorola-S-format	-tm
JEDEC-Format for programming of logic devices, if the data extension deviates from ".JED"	-tj

Binary format

The data are listed strictly binary and sequentially. The software localises the start address at 0. If your application requires a different start address you need to enter it via offset options (-od resp.-of) while invocation of the program "PGS67". Binary files cannot be looked at with a text editor since they do not have an ASCII-code. An advantage of the binary files is their high packing density.

HEX Formats

The most frequently used format is the HEX format (8/16/32 bit), defined by INTEL. The majority of software tool manufacturers supply at least the conversion routines to convert the produced data by compilers. The HEX format is ASCII-encoded.

You have an opportunity to read or even change the data via a text editor (Caution: The check sums need to be adapted).

There is no special HEX format for 16/32 bit applications. The allocation of data for greater bus widths must be carried out with device programming via the device mode.

Simple 8 bit applications only have two record types: data- and end records. 16/32 bit applications additionally have the segment record. This defines the current status of the CS register regarding 8086-systems.

OBJ / AOMF Formats

Another creation of INTEL is the so-called INTEL-OBJ format (8/16/32 bit) which is being supported by software manufacturers under the name AOMF format. Its structure is strictly binary i.e. you do not have an opportunity to use

a text editor for reading nor changing data. The binary format makes it very compact. By various types the records are marked to define data, segment addresses and debug information (symbols). Not all types are published. Thus, e.g. files whose data were generated with the INTEL locator RL51 for 8051 applications, contain a number of record types that are not specified for the public.

Absolute addresses are defined for all data. Further, files for 8086 applications contain the so-called "iterated-data-records", where several, likewise engaged data sections are iteratively defined in one record. Such records must be expanded for device programming via pre-processing.

Motorola-S Format

This format (S1/S2/S3) is ASCII-encoded (i.e. directly readable via editor) and its structure is similar to the INTEL-HEX format.

JEDEC Format

This format is ASCII-encoded (i.e. directly readable via editor) for logic devices.

Output-File Format

While programming or verifying the above mentioned format types can be automatically recognised resp. defined (input-file format), the writing of a file which is linked to getting device contents, however, is only achievable in INTEL-HEX format (default setting) or in the binary format.

Data filtering

Extracting data from files

The "data mode" describes according to which structure the data from a file are to be evaluated.

Setting defaults

The simplest structure is the default setting.
All data are selected as 8-bit data.

Selection

If not all but only certain parts of a bus width need to be selected this can be defined by the operation parameter "-bw(n0,n1,..)". w stands for the bus width, n0,n1.. determine the bytes to be selected.

Bus width

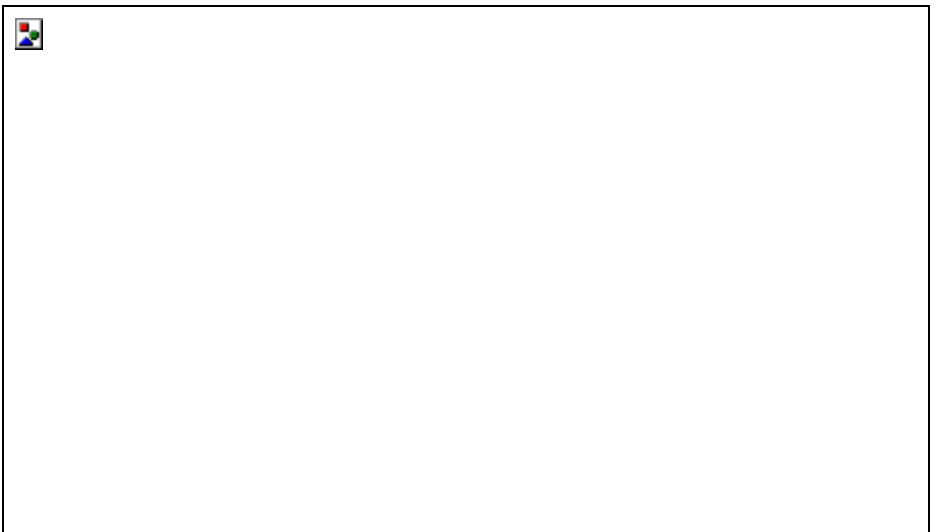
To select the data, first the bus width of the target system must be known.
Accepted values are 16, 32, 64 and 128 bit.

Bytes to be selected

The bytes to be selected are determined by their numbers. The numbering commences with 0.

Examples

Parameter	Function
-b16(0)	Only the even bytes of a 16 bit system are to be selected
-b32(2,3)	Only the odd 16 bit words for a 32 bit system are to be selected



Data mode, e.g. -b32(2,3)

File offset

The file offset (-of= offset file) determines from which position data are to be selected from a file. It refers to the byte addresses of a file.

When programming bus widths greater than 8 bit please note that the offset address always must be entered as a byte address (not e.g. as a word address).



File Offset

Example

```
PGS67 w -fTEST.OUT -of10000
```

Data from the file TEST.OUT which are localised beginning at address 10000H are stored in the device beginning at address 0.

Number of programming data

Through the option -l<hex> the number of data to be programmed / verified / read can be entered in bytes.

Thus, not all data to fit into a device are selected from a file, but only the given length.

Example:

```
pgs67 c -ftest.bin -l1800 -tb
```

Only 1800H bytes (referring to start address 0) of the binary file TEST.BIN are verified with the bytes of the device.

Data partition of devices

Diversity of functions

The programming system PGS67 offers a large variety of opportunities to separate and allocate data from files into devices. The functions seen one by one might appear simple, however, their combinations build complex functions as there are:

- Data in files are always organised as 8 bit data.

- Device may be programmed parallel (with identical contents).
- Devices may be programmed successively (gang mode).
- Devices may be programmed side-by-side (split mode).
- There are different bus widths for splitting (16 / 32 / 64... bit).
- There are different bus orientations (little- / big endian).
- There are shiftings between file- and device addresses (offset).
- If cells are not defined in full, before programming the device's contents must be reread. (Programming of a 16 bit device beginning at an odd byte address).

Function

The "device mode" describes how the data selected from a file are allocated into a device. Partitions are e.g.: parallel and/or successively into a number of devices. At the same time via the "device mode" you determine how many devices resp. programming sockets are involved in the procedure.

Syntax

To enter the "device mode" you use the following term:

```
-d(A[*B[*...]])
```

In brackets you give the identification letters of the programming sockets involved. These letters are separated via operators which determine the allocation of data towards the sockets.

Comma= parallel

This operator puts the same data to different sockets. The programming results are devices or groups of devices with identical contents (= parallel programming).

While reading (Function parameter "r") only socket A is accessed.

Colon= split

This operator splits the data towards a number of devices each depending on their bus width. The total of devices must always be dividable by two. The data are split in a way that directly following data elements are allocated into directly following devices(= Split programming).

Plus= gang

This operator separates the data towards devices or groups of devices depending on their capacities. The data are separated in a way that one device after the other is filled up with ascending addresses (= Gang-programming).

Hierarchy of operators

The three operators may be combined with each other. The evaluation is done from left to right with the rank order:

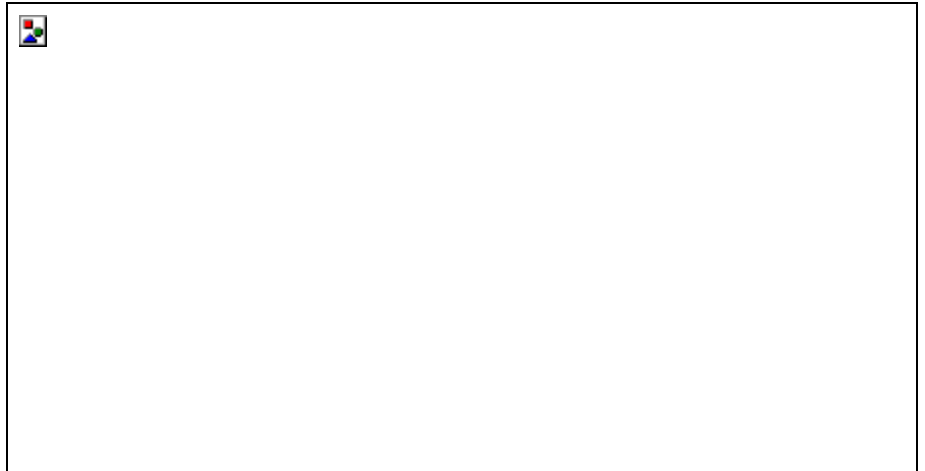
1. : split
2. + gang
3. , parallel

Default setting

Without a default of a "device mode" it is assumed that one device is inserted into the left (=A) socket. This situation corresponds to: "-d(a)"

Examples

Parameter	Function
-d(a:b)	Programming of e.g. two 8-bit devices that shall be used in a 16 bit system (or 2 x 16-bit for a 32-bit system).
-d(a,b,c)	Three devices with identical contents.
-d(a:b+c:d)	Combination of split and gang for one set of 4 devices. The data of the low address range are allocated to the sockets "A" and "B" (low- order bits to "A"), the data of the high address range are allocated to the sockets "C" and "D".



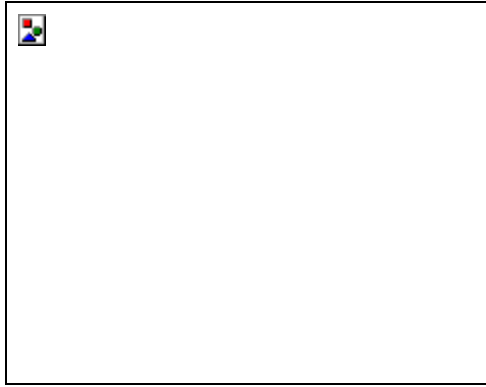
Device mode, e.g. -d(a:b+c:d)

Device offset

By giving a device offset address (-od= offset device) data from a file may be put to any address range of the device. This offset value refers to byte addresses in the device. Especially when programming devices with greater bus widths than 8 bit, this fact must be kept in mind.

A typical application for this parameter is the programming of so-called "BOOT-EPROMs" since their address location keeps changing while their internally stored program is running.

File offset, device offset and number of data may be used optionally in all imaginable combinations.



Device offset

Example

```
PGS67 w -fTEST.OUT -od8000
```

Data from the file TEST.OUT are allocated in the device commencing with device address 8000H.

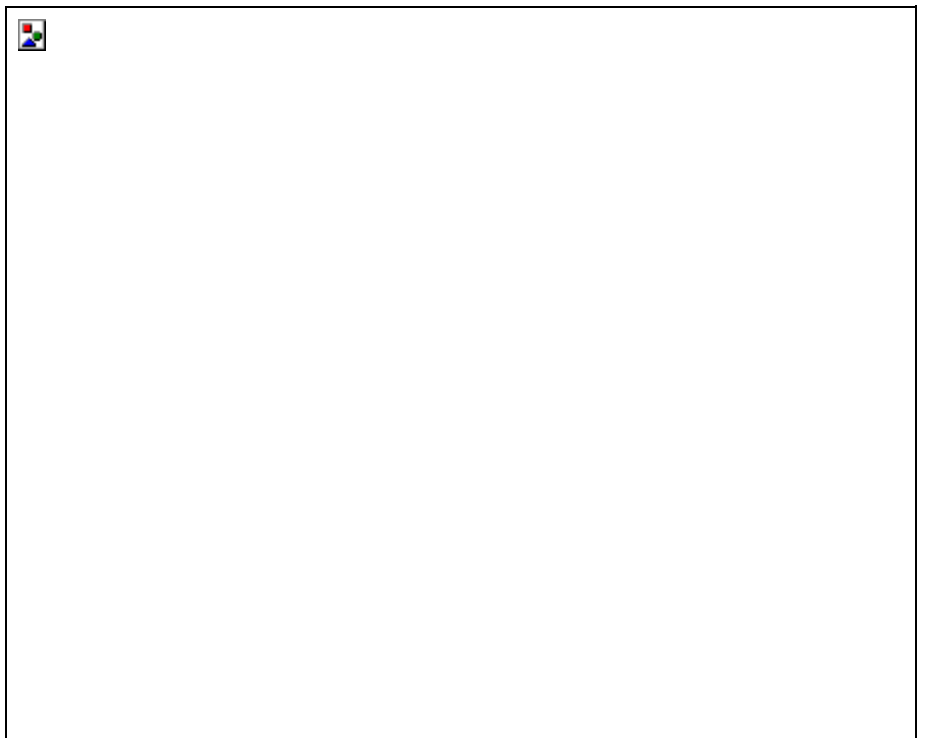
Endianess

Some types of hardware memory organisations and a number of software development tools require a swapping of the low order and high order word sections in systems with a greater bus width than 8 bit. You can set to "big endian" via the operation parameter "swap". The default setting is "little endian".

If e.g. 16 bit devices are used in hardware memory organisations where the low order 8 bit are swapped with the high order 8 bit (e.g. Motorola 68000), you can set the allocation when programming the devices via "-swap". This organisation is also referred to as "big endian".



Little Endian



Big Endian

Screen displays

Operation messages

While processing

To control the activities the screen provides a status line. This line is faded into in the currently sent data telegrams with each corresponding telegram start address during read-, program- and compare functions.

During the blankcheck, the calculation of check sums, the reading of the identifier codes, the duplication procedure and the processing of type-specific special functions, the status line displays the appropriate message about the type of function being handled.

Message	Current procedure
writing device `type` adr: 00001000 .. 0000112a	Programming of data at the device addresses 1000h thru 112ah.
comparing device `type` adr: 00000000 .. 0000017fh	Verifying the data from the device addresses 0000 thru 017fh
writing file `filename` adr: 0200 .. 03ffh	Reading of the data from device address 0200h thru 03ffh and storing in the file which had been given during invocation
duplicating ...	Duplicating
calculating check sum ...	Calculating the check sum
testing ...	Blankcheck
reading identifier	Reading of the identifier code
special function ...	Executing a special function

After processing

When a procedure is finished a message is provided on screen indicating the successful handling resp., if applicable, indicating the errors occurred.

When concluding a procedure the last status messages remain preserved. In addition, further information on special events during the processing, e.g. addresses and data where programming errors have occurred, are displayed.

End message on error-free completion

Once the programming job has been concluded error-free a confirming end message is indicated on screen as per the defined number of sockets (device mode) and per activated sockets. If the programming job was started parallel for a number of modules (different slots), the end message is provided for

each slot.

The error response value which is helpful for evaluating batch files, in this case, is stored with the value 0 in the DOS-environmental variable "error level".

After conclusion of a programming-, duplicating-, blankcheck- or compare-job as well as after error-free reading devices, an end message is provided per each active socket:

```
S02: socket A:ok.
```

After error-free processing of the other functions the end messages listed below are indicated on screen:

Function	End message on error-free completion
Programming	ok.
Comparing	ok.
Duplicating	ok.
Reading	ok.
Blankcheck	ok.
Reading of identifier code (i)	ok. ident: <hex>
Calculation of check sum (s)	ok. sum: <hex> crc: <hex>
Special function (#no.)	ok. (possibly with specific echoing values)

Example:

Command line:

```
pgs67 w -eam27c010 -ftest.hex -s(2,3) -d(a,b,c,d)
```

End message:

```
S02: socket A: ok.
S02: socket B: ok.
S02: socket C: ok.
S02: socket D: ok.
S03: socket A: ok.
S03: socket B: ok.
S03: socket C: ok.
S03: socket D: ok.
```

End message with error information

Error without abortion

Errors that occur at single sockets, yet do not affect the procedures at the remaining sockets, are provided as operation messages. Such messages do not end up in aborting the processing of the started programming job. The "error level" is set to the value 1, once there has occurred such an error at least at one socket. The error message is only provided for the socket involved:

Function parameter	Error message
w	verify error. <hex> expected:<hex>, found:<hex>
d	verify error. <hex> expected:<hex>, found:<hex>
c	compare error. <hex> expected:<hex>, found:<hex>
r	--
e	not erased <hex> expected:<hex>, found:<hex>
s	checksum error
s-c(..)	sum: <hex>, crc: <hex> *** expected <hex>.
#no.	special error

Note

The addresses indicated together with the single error messages correspond to word- resp. double-word addresses if the bus widths are greater than 8 bit.

Error with abortion

The errors mentioned below lead to an immediate abortion of the programming job and set the "error level" to 1, too:

- Socket not loaded with the correct devices as per the selection given in the "device mode" ::

```
-d(a,b): Device missing in socket B:
S02:socket A: aborted
S02:socket B: no device
-d(a): Devices in sockets A, B and C:
      S04:socket A: aborted
      S04:socket B: not empty - remove device
      S04:socket C: not empty - remove device
```

- Device does not have a programmable memory within the device addresses to be processed:
(The switch -z(ignore) allows to make the system ignore these inappropriate device addresses.)

```
S02:socket A: device address error
S02:socket B: device address error
S03:socket C: device address error
S03:socket D: device address error
```

- Trying to read the identifier code although the device does not have such code:

```
S05:socket A: device has no ID
```

The following module errors also lead to an abortion with error level =1, however, are not indicated for each single socket but only once per module.

- Recognition of overcurrent:

```
DEV_OVERCURRENT
```

- Illegal device mode (e.g. splitting for microcontrollers - d(a:b):

```
DEV_ILL_MODE
```

- Illegal number of special function:

```
DEV_ILL_SPEC_FNC
```

Device type is not supported by the module slot selected:

Those errors leading to an abortion of the complete system at error level=2 are defined as follows:

Message	Meaning
IE_n	INTERNAL ERROR possible reasons are a hardware problem within the programming system or a software incompatibility; the indicated error number n allows to identify the error's nature for long-distance diagnosis.
TE_s	TRANSMISSION ERROR the character sequence s specifies in-depth as following:
TE_IPX_NOT_LOADED	IPX-Driver for Ethernet-data channel is not loaded
TE_TIMEOUT	Exceeding of a monitoring time; incorrect data transmission
TE_CTS_NOT_ACT	CTS-Signal at RS232C data channel not active; possible problems within cable / connector
RE_s	RUNTIME ERROR Error has been detected in the host while running; the character sequence s specifies in-depth as following:
RE_FILE_NOT_FOUND: s	File s could not be opened
RE_ILLEGAL_FILE: s	File s is illegal; e.g. by false input of a file format
RE_OUT_OF_MEMORY	Memory overflow
RE_SYNTAX_ERROR	Syntax error at invocation
RE_NO_DATA	Data error, e.g. due to illegal or inappropriate inputs of offset values resp. data length values, there are no data contained within the range
PTC_ILL_MODULE_ADR	Illegal module address

Type-specific error messages

With some certain device types there also exist type- specific error messages which either lead to aborting the procedure with error level=1 or which are only indicated as per the socket(s) involved after having concluded the job:

Examples:

```
reprogrammable logic devices (GALs, MACHs):
To check if device is erased does not make any sense!
MOTOROLA-microcontrollers , which enter into the programming mode via
their Bootstrap Loader:
No answer from Bootstrap-Loader of device
Memory cards:
Error while erasing card
```

System messages

Process message in the file PGS67.ERR

For detailed analysis and evaluation of processes by controlling programs, a file PGS67.ERR is generated after every processed invocation of the program PGS67.EXE. This file contains information like e.g.:

```
slots [module no.1 module no.2 ... module no.14] ↵  
result [slot sockstat [sockstat...]] ↵
```

Term	Meaning
module no. n	Number of module in slot n
result	Result of the foregoing invocation = 0, if no capital error has occurred that would have lead to an abortion of the job. Otherwise: Number of the capital error that did occur leading to an abortion of the job. The error specific numbering may be taken from the following list as well as possible additional information from the file PGS67.DOC.
slot	Number of module slot
sockstat	State of active sockets of the module slot. The number of sockstat corresponds to the number of active sockets of module at slot. The first value of sockstat refers to the first active socket (as a rule, socket A, followed by B, ...). = 0, if no error has occurred. Otherwise: Number of error that did occur (see the following list with error numbers).

Error number	Meaning
000	No error
010	Syntax error
011	Memory error
012	File error
014	I/O-driver error
020	The process at one socket has been aborted due to a module error.
021	Wrong data or false addresses regarding the addressed socket or device.
022	Device is inserted into the socket, however, deviating from the device mode selected
023	No special function implemented
024	Overcurrent Ipp
025	Overcurrent Icc
026	No device inserted into the activated socket
027	Compare error
028	Verify error
029	Device not erased
030	Error of calculation of check sum
040	Device not ready (e.g.: wrong type inserted, device inserted wrongly, device defective)
041	Device overcurrent
042	Wrong device address
043	Signature cannot be read
044	Illegal device mode selected
045	Duplicating not possible
046	Special function number # not implemented
099	Type specific error

Examples

Message	Meaning
010	Syntax error -> abortion since PGS67 cannot be accessed
slots 100 200 918 650 510 000 002 000 029 000 000	Device in socket B of module no. 2 is not erased
slots 100 220 681 655 230 000 002 020 020 022 000 000 004 028 000 000 000	A device is inserted in socket C of module no. 2, however, the selected device mode deviates therefrom (e.g. -d(a,b). Therefore, the programming job has been aborted at the sockets A and B also (module error). At socket A of module no. 4, a verify error has occurred in addition.

Error level

To program effective DOS-batch processing data (batch files) the evaluation of echoing values is available via IF-commands. Such an echoing value presents information on the method a program has been processed. The echoing value is saved in the DOS-environmental variable "error level".

The program PGS67.EXE provides the following echoing values:

Error level	Meaning
0	No error resp. no special event occurred
1	Error occurred at least at one socket, however, this error alone did not lead to an abortion of the complete procedure. Basically, you may keep on working.
2	A capital error has lead to an abortion of the complete procedure. Before you can keep on working you must correct the error.

Batch processing (batch-, make files)

In general

All program invocations can be integrated into batch processing files (batch files) and make definitions.

Thus you can take advantage of the opportunity of reproducing the complete programming processes at any time.

Due to the opportunity of conditioned branching in the batch file, programming abortions may happen (after occurred errors) through evaluation of the generated "error levels".

This is a necessity with regard to production oriented programming.

Comment

The command line for invoking PGS67.EXE may be concluded by a semicolon. Texts located to the right hand side of the semicolon are ignored. Thus a compact commenting of the batch processing files is made possible.

Auxiliary programs

Graphic presentation via PGS67ERR.EXE

File

For a graphic presentation of the socket states of all in the PGS67 active modules (for each programming job), a file PGS67ERR.EXE is delivered as standard. This file evaluates the file PGS67.ERR which is generated by the driver software after each programming job and presents socket errors blinking red, and active sockets where no error has occurred continuously lit green. Additionally, when invoking the program for graphic display for state of events, two parameters may be given as a text as per the pattern indicated below:

```
PGS67ERR "Text1" "Text2"
```

Text1 is faded into the state graphic in as far no error has occurred. Otherwise (error level not equal to 0) text2 is shown in the display.



Graphic representation using PGS67ERR.EXE

Request for action

On production sites this program may also be used as a visual request for action regarding insertion of devices into certain sockets. However, since at this point in time, there is no system configuration for the corresponding file PGS67.ERR, this file must be created by yourself (file1.err) and needs to be copied into the file PGS67.ERR prior to processing the programming job.

Example of a batch file with graphic display

```
copy file1.err pgs67err
pgs67err "Please insert 4 pcs. of device D27C512 into module no. 2 !"
pgs67 e -ed27c512 -m2 -d(a,b,c,d)
if error level = 1 goto error
pgs67 w -ed27c512 -m2 -ftest.hex -d(a,b,c,d)
if error level =1 goto error
pgs67 s -ed27c512 -m2 -ftest.hex -d(a,b,c,d) -
c(0a13e,0a13e,0a13e,0a13e)
if error level = 1 goto error
goto OK
:error
pgs67err " " "An error has occurred !!!"
goto end
:OK
pgs67err "Procedure OK"
:end
```

File1.err

```
slots 100 200 220 660 682
000 002 000 000 000 000
```

Examples

Program invocations

Please note that the following examples of commands are not always complete resp. sufficient. They shall only describe the functions listed below. The commands get completed e.g. through additional information in the file PGS67.INI.

```
PGS67 ↵
```

Listing of syntax

```
PGS67 ? ↵
```

Listing of syntax and system configuration

```
PGS67 ? -xipx:165000123 ↵
```

The programming system bearing the serial number 16500123 is accessed via Ethernet.

```
PGS67 ? -xsio:2,57600 ↵
```

The data channel COM2 is selected to run at 57600 Baud.

```
PGS67 p -e256 ↵
```

Listing of all implemented devices whose nomenclature contain the string "256".

```
PGS67 w -ftest.bin -tb -s2 -eam27c512 -d(a,b) ↵
```

Programming of two devices of type "am27C512"; for this purpose the file test.bin is evaluated as a binary file; the devices are programmed in the first two sockets of the programming module (=slot next to the master module, i.e. right hand side of the master module).

```
PGS67 r -ftest.hex -th -d(a+b) ↵
```

Two devices are read serially (gang); the data are saved in the file test.hex in INTEL-HEX format.

```
PGS67 d -ehn27c4096 -d(a,b,c,d) ↵
```

The contents of the device of type "HN27C4096" in socket A is fully duplicated into the devices inserted in the sockets B, C and D.

```
PGS67 c -ed2732a -ftest.obj -od400 -of400 ↵
```

The device "D2732A" in socket A is compared, beginning at device address 400h, with the data from the file TEST.OBJ (automatic format recognition), beginning at file address 400h.

```
PGS67 w -eam27c040 -fx.hex -d(a,b,c,d) -s(2,3,4,5) ↵
```

16 devices of type "AM27C040" in the sockets A through D of the modules in the slots no. 2 through 5 are programmed with the data of the file X.HEX in a parallel mode. (Caution: The device mode -d(..) needs to be equal for all modules!).

```
PGS67 #2 -emc68hc711e9_ee↓
```

The contents of the config-register of the device type "MC68HC711E9" (Motorola microcontroller) is read and indicated on screen.

```
PGS67 s -c(1234, 9abc, 2324) -d(a,b,c)↓
```

The CRC-check sum is calculated for 3 devices; it is verified if the sockets A, B, C show the values 1234, 9abc, 2324; if not true: error message and error level=1

Example of a batch file

```
rem example
:start
copy file1.err pgs67.err
pgs67err "Please insert 2 devices type MC68HC705B5 each into slot 3,
4!"
pgs67 e -emc68HC705B5 -s(3,4) -d(a,b,c,d); Test erased
pgs67log; statistic data acquisition
if error level = 1 goto error
rem programming
pgs67 w -emc68hc705b5 -s(3,4) -ftest.hex -d(a,b,c,d) -od800 -l1000
pgs67log
if error level = 1 goto error
rem compare
pgs67 c -emc68hc705b5 -s(3,4) -ftest.hex -d(a,b,c,d) -od800 -l1000
pgs67log
if error level = 1 goto error
pgs67 s -emc68hc705b5 -s(3,4) -ftest.hex -d(a,b,c,d)
-c(0a13e,0a13e,0a13e,0a13e); CRC-checksum verify
pgs67log
if error level = 1 goto error
goto OK
:error
pgs67err "An error has occurred !!!"
goto end
:OK
pgs67err "Procedure OK" " "
goto start
:end
```

Programming modules

Overview

Due to the flexible conception of the programming system PGS67 you may expect a continuous extension of the range of available programming modules as well as the permanent care and extension of the device list. A current overview is given after entering the command <pgs67 p>. Additionally you find current information in the file PGS67.DOC.

If you use PSI67, you get informed via the menu/window "Device type" about all current available modules and programmable devices at any time.

Our internet service www.ertec.com enables you to get the very latest state of developments in terms of programming modules from ertec.

Index

1

10Base2 7
10BaseT 7, 10

8

8086 applications 50
8-Bit data 48

9

9/25-adaptor 7, 9

A

Absolute format 48
Acoustic signal 6, 10
Acoustic signals 10
ANSI/IEEE Std 802 42
AOMF 45
AOMF format 49
Application 2, 6, 15, 25, 46
ASCII 31, 40
Assembler 2, 48
Auto mode 49
Auto test 7, 10

B

Batch 3, 11, 40
Batch processing 62
Baud 42
Baudrate 8, 26, 41
Big endian 8, 46
Binary 31, 41
Binary format 48
Bit/s 8, 26, 41
Blankcheck 57

C

Channel 7, 10, 22, 40
Cheapernet 41
Check sum 31, 46
Checksum 8, 22, 47
COM2 7, 10, 41

Compare 8, 45
Compiler 2, 48
Concurrent mode 20, 24
Configuration 7, 16, 22, 40
Connector 5, 10, 42
Conversion 7, 49
Crc 8, 23, 46
CS register 49
Custom specific 23

D

Data length 8, 46
Data mode 8, 46
Data transmission speed 46
Database 23, 46
Debug 48
Debug information 50
Default setting 49
Device code 24
Device mode 3, 8, 41
Device type 2, 8, 19, 23, 41, 69
Dial 1
DOS 1, 6, 9, 31, 40
DOS-file name 45
Duplicate 8, 45

E

Endianess 32, 46
EPROM 11, 29, 44
Erased 8, 45
Ethernet 7, 10, 22, 40
Ethernet-connection 10
Example 41
EXE-format 48
Extensions 1, 15

F

File format 2, 33, 41
Function parameter 43

H

HEX format 49
history 15, 23

I

Identifier 8, 42
INTEL 34, 45
INTEL-HEX 45
Interrupts 41
Invocation format 43
IPX 3, 11, 22, 42

J

JEDEC 45

L

label 1, 15, 24
label format 29
LED 10
Little endian 55
Locator 2, 48
Log file 23

M

MAC adress 11, 42
Manufacturer code 24
Module 1, 5, 9, 18, 29, 41
Motorola 34, 45

N

Network 10, 15, 23, 41

O

Offset 8, 34, 45
Operation parameter 42
Operation parameters 8, 34, 45
operator's name 17
OUI 42

P

P67LP1 15, 24
PGS67.DB1 40
PGS67.DB2 40
PGS67.DBO 40
PGS67.DVL 40
PGS67.EXE 3, 6, 40
PGS67.INI 3, 40
Print 8, 20, 27, 40
Problems 1, 60
Program 1, 15, 22, 40
programming job 18, 23, 57
Progress indicator 20
PSI67 2, 6, 9, 15, 22, 69
PSI67.INI 3, 25

R

Read 8, 34, 43
Rear 5, 10
Registration card 1
Relative format 48
RS232C 7, 9, 40

S

Segment record 49
Serial number 3, 5, 11, 31, 42
Serial port 3, 5, 11, 31, 42, 46
SIO 22
Special 8, 32, 44
Start address 35, 45
Start delay 24

SUB-D-connector 35, 45
Sum 31, 45
Supervisor 16, 22
Swap 8, 33, 46
Syntax 8, 31, 40

T

TCP/IP 3, 22
Test 30, 45
Time limit 24
Twisted pair 10

U

UDP 22
UNIX 43

V

Version 1, 11, 15, 28, 40

W

Write 8, 35, 45