Software Manual

INPUT/OUTPUT DEVICES

www.gebe.net

Thermal Printer

GeBE-PRINTER SYSTEM C32

SOFTWARE MANUAL

valid from:

firmware revision V.2097
bootloader revision V.1533
Sections coloured in red are not yet implemented!

The printer system C32 is a powerful thermal printer control using a 32 bit Cortex M3 or M4 processor. It can be configured as typical line printer (A8 syntax) or as forms or page oriented printer as well. Further printer emulations are available. Interfaces are USB full speed or RS232 up to 1 Mbaud.

GeBE Elektronik und
Feinwerktechnik GmbH
Beethovenstraße 15
82110 Germering
Germany

phone +49.89.894141-0
fax +49.89.894141-33

email info@gebe.net
Internet www.gebe.net

GeBE-Document-No. SoMAN-C32-E-V2.0-0793
date February 18, 2019

LIST OF CONTENTS

1 SYSTEM DESCRIPTION

1.1 INTERFACES

The controller contains a USB full speed interface and a RS232 interface.
The RS232 can be operated with V.24 level up to 460kbps and with TTL level even up to 3.6 mbps.
When both, USB and RS232 interfaces are equipped, the USB has priority. When USB is plugged in, an automatic reset activates the USB interface. When unplugging USB, an automatic reset reactivates RS232.

1.1.1 USB INTERFACE

The printer conforms to USB specification V1.1 for full speed units. The printer is compatible to USB V2.0 bus systems. The interface is implemented as bidirectional USB printer class. The printer will be operated in self powered mode. Each printer gets an individual USB serial number.
The module contains a USB printer device with bidirectional interface (bInterfaceClass=7, bInterfaceSubClass=1, bInterfaceProtocol=2) according to "Device Class Definition for Printing Devices Version 1.1." The device is implemented as full-speed self-powered device with GeBE-Vendor ID. idVendor = 0x19DB (VID, GeBE)

In addition to the control-endpoint the device uses 2 function-endpoints for bulk transfers from and to the USB host. Both endpoints use double buffering with wMaxPacketSize=64. The USB host requests messages and status through bulk-IN.

Bulk-IN transfers will always be confirmed by ACK-handshake. ZLP (Zero Length Packet) will be sent in case of no ready data, means the host gets data packages with 0 … wMaxPacketSize bytes.

Bulk-OUT transfers will be acknowledged by NACK-handshake, in case the bulk-OUT buffer is occupied with 2 packets. Reset (flush) is possible through class-request SOFT_RESET. Using class-request GET_PORT_STATUS sends a status byte with information of („paper empty“, „select“, „error“ = 3 bit) to the USB host. Depending on the settings of the USB module (control gate) the following response will be sent:

STANDBY:	0x18 (default value with „select“=1)
SELECTED:	„select“=1(fix), „paper empty“, and „error“ will be derived from system variables
DISABLED:	0x08, „select“=0(fix) enables the USB-host/printer driver to react with a timeout in case a printing job was sent to USB device by mistake

GeBE USB Product IDs

Cortex M3	**VID: 0x19DB**	
PID: 0x0276	Bootloader STM32F103 CLASS A	Piano (GCT-4662), FLASH (GCT-4362-3-4)
PID: 0x02FB	Bootloader STM32F103 CLASS B	testprinter, MaxiMulde, GCS-696x, GCS-436x
PPID: 0xnnnn	Bootloader STM32F103 CLASS C	USB & LAN (to be defined)
PID: 0x0EB4	Evaluation board	
PID: 0x4FDD	Piano 203 and 300 dpi	
PID: 0x543A	FLASH HPiR/BLE	
PID: 0x56EE	PrinterLab V1.0 and V1.2	
PID: 0x576E	Chipset GCS-696x, 203 and 300dpi	
PID: 0x57D9	Chipset GCS-436x	
PID: 0x587A	FLASH HPiR	
PID: 0x597E	MaxiMulde	
PID: 0x522C	FLASH USB/(HPiR future)	
Cortex M4	**VID: 0x19DB**	
PID: 0x02FF	Bootloader STM32F427 CLASS A	
PID: 0x0300	Bootloader STM32F427 CLASS B	Piano (GCT-4663), Maximulde (maximould), GCS-696x, GCS-436x
PID: 0xnnnn	Bootloader STM32F427 CLASS C	USB & LAN (to be defined)
PID: 0x57E9 (P22505)	Chip set GCS-436x	
PID: 0x56EF (P22255)	EW board to M4	
PID: 0x5845 (P22597)	Printer lab V2,0	
PID: 0x5B32	MaxiMulde	

1.1.2 SERIAL INTERFACES RS232 / TTL 3.3V

The input buffer is adjustable from minimum 1 kbyte to maximum 4 kbyte by parameter. With 1 kbyte being default (parameter 13.1). The buffer overhead is adjustable from 32 byte to 255 byte by parameter with 64 byte being default (parameter 13.2 and 13.3).
In case the memory reaches the set buffer overhead size, the controller switches the CTS-line onto stop and sends the control code <XOFF> via serial data line to the host. This stops the data stream from the host to the controller. This means, that the printer sends 64 byte until the buffer is filled-up and a <XOFF> (13h) sets the busy signal.

When the buffer decreases down to 128 characters the controller releases the CTS-line (level change) and a control code <XON> (11h) will be send to the host for further data transfer.

1.1.3 HARDWARE HANDSHAKE

The sending data source (host or data controller) recognizes whether the receiving side is able to accept data or not. This is usually indicated by the status of the potential level being present on the hardware wires.
At data receipt, the printer controller monitors the handshake line CTS (clear to send) together with the whole input buffer control. Through the data lines, the signal is simultaneously controlled together with the performed software handshake (<XON>/<XOFF> protocol). When sending data e.g. status messages to the host, parameter 10.6 can select whether the printer considers a host hardware handshake or not. Default is: not consider.

1.1.4 SOFTWARE HANDSHAKE

The control of data transfer from host to controller is not only made by hardware handshake but also per <XON>/<XOFF> protocol. The decision between both methods is made according to the options and the settings of the host software.

1.1.5 TROUBLESHOOTING

Not implemented yet
In case of a defective character, the printer prints a „?“ (at parity or framing error) and a „!“ (at overrun error). Subsequently a „X“ will be sent via serial interface. As default: printing „?“ at parity or framing error is not activated.

1.1.6 INTERFACE HPIr

See description INFO-HPIR-E-0417 (download from www.gebe.net).

1.1.7 INTERFACE GeBE-Ir

See description INFO-GeBEIR-E-0395 (download from www.gebe.net).

1.1.8 INTERFACE Bluetooth®

Description

2 OPERATION

2.1 IMPORTANT INFORMATION FOR OPERATION

- It is recommended to program 8 dot lines FEED prior to each printing job in order to assure proper start of the printer mechanism.
- When changing running direction, the printer automatically compensates the back lash. Nevertheless little positioning errors are possible.
- Changes in printing speed can lead to print quality losses (e.g. through discontinuous data stream).
- Thermal printers are not useable for continuous operation without breaks, in generally. Please inquire for existing limitations concerning your printer system.
- Not every setting combination of current consumption and print dynamics will result in good printing quality.
- Avoid printing continuous lines. Under some circumstances the printer mechanism could stick together.
- Increasing the default print density will result in reduction of printer head life time.
- Never activate the printer driver at the end of a print job. Data may be lost.

2.2 PRINTER OPERATION MODES

The printer contains of 2 basic operation modes:

1. **GeBE Line Mode (compatible to GeBE A8 syntax)**
   In line printer mode each printer action will be performed immediately. A „hello world“ followed by carriage return prints „hello world“ with a line feed.

2. **GeBE Forms Mode**
   In forms mode a command „start of ticket“ sends all forms data to the printer. In the printer memory all forms data will be collected and built up as a page. Finally a command „end of ticket“ initiates the printout.

System commands are active in all operation modes and emulations.

2.3 PRINTER OPERATION MODES AND EMULATIONS

Printer parsers are suited to easily implement other printer emulations. Emulations of older GeBE printers and customary printer series of other printer producers are already integrated.
Emulations will be determined in parameter 13.4 by emulation chain. The parser scans for valid commands according to the emulation sequence, and performs the command.

System commands serve for system configuration or system control and they are active in all operation modes and emulations.

Following operation modes and emulations are supported:
1: GeBE forms mode
2: GeBE line mode – compatible to GeBE A8 systems

3: GeBE N78 emulation – line mode
4: HPIR emulation – line mode
5: PCL3 emulation – PCL graphics mode
6: FGL (Friendly Ghost Language) emulation – FGL forms mode

By default, the emulation chain is 2, 5, 1.

2.3.1 HEX DUMP MODE

The printer can be controlled by the following manipulations in a hex dump mode.
In this mode, the data stream is no longer interpreted according to its command content, but only the hexadecimal content of the entire data stream is printed out.
This mode can be very helpful for diagnostic tasks, especially when servicing the printing system. The SELF TEST which can be called up with the FEED button when switching on the printer is also very useful.

Activation of the HexDump mode:
a) by command <RS> 'z' HEXDUMP
(The password "HEXDUMP" serves as protection against unintentional input.)
b) Pressing the FEED button for at least 8 seconds when turning on the printer.
The start in this mode is confirmed by the printout of HEXDUMP.

Deactivation of the HexDump mode:
a) by switching-off the printer
b) by a reset of the printer

The number of bytes displayed per line is determined by the width of the printing unit (10; 12; 18; 24 characters).
The system font 0 (for example, 8x16) is always selected as the font.

An expression could look like this:

HEXDUMP

0000 31 32 33 34 35 36 37 38 39 30 31 012345678901
0010 41 42 43 44 45 46 47 48 49 4A 4B ABCDEFGH IJ

No. character code (hex) ASCII characters

3 COMMAND SETS

3.1 NOMENCLATURE

Following nomenclature is valid for command tables:

- All codes and parameters of one command are compiled of single bytes (1 byte = 8 bit). They are ASCII characters, hexadecimal values or placeholders for numerical values or bit strings.
- For definite command handling the symbols of the command tables are used according to following rules.
  Example:
  General nomenclature for the command: „perform a paper feed by x lines“:
  <ESC>“F“lh, ll., lh:={0, ... ,9} ; ll := {0, ... ,255}. The parameter value range is limited. Therefore, the command <ESC> "F" 03h E8h indicates a paper feed by exactly 1000 lines (at 8 dot/mm this means 125 mm). Examples appear in a way as to be entered directly into Win Z1 or the toolbox.

symbols	description	example	meaning	information in WinZ1 or toolbox
d	decimal value	27d	1 byte	<27d>
h	hexadecimal value	0Ah	1 byte	<0Ah>
<>	ASCII control code	<LF>	command: line feed	<0Ah>
[ ]	8 flag bits	[FLAG]	001x 1111 b b indicates a bit order from MSB up to LSB	
" "	printable characters	ASCII-characters "E"	letter E	
( )	parameter	(NAME)	byte variable	
<" ">	one character	<"n">	character variable	
l, m, n	per one byte	n:= {0, .. , 100}	byte variable	
nh , nl	2 byte value	nh:= 03 nl:= E8	word variable	
{ }	value set	{VALUE-RANGE}	{$00, .. ,$FF} all hex values from 00h to FFh	

4 SYSTEM CONFIGURATION

One set of parameters is used to configure the printer system in all kind of emulations. For more transparency these parameters are grouped in modules. Some of the parameters are permanently linked to the hardware and can only be changed ex factory. Within one module, the parameter index addresses the parameter. Command parameters have to be entered in ASCII format.
Input 18,2 addresses parameter no. 2 in parameter module no. 18.
All permanent parameter settings will be stored in the printer's FLASH, EEPROM or RAM. Different storage levels will be used to save the settings.

4.1 STORAGE LEVELS

- FIRMWARE SETTINGS:
  The printer's basic setting is stored in the printer's FLASH memory within the firmware. It is not changeable and always the same for each firmware. On demand, these FACTORY SETS can be adapted ex factory.

- *FACTORY SETTINGS **(not implemented yet, actually realized in firmware settings)**:*
  The printer's basic settings are stored in the FACTORY SETS, changeable ex factory only. By command, a reset to the basic FACTORY SETS is possible. In delivery status, factory- and user sets are identical.

- USER SETTINGS:
  Ex works the factory settings are additionally saved as a copy in the internal EEPROM. With each poweron or reset, the user settings will be read for operation. With command „Set printer parameter“ the user settings may be changed and saved.

- RAM SETTINGS (temporary):
  RAM settings will be deleted after reset or power-off and reset to USER SETTINGS at next power-on.

5 SYSTEM COMMANDS

5.1 SET PRINTER PARAMETERS

<RS>[a:m,x:w]
| | | | |
| | | | |
| | | | +-------- value (1…n characters)
| | | +---------- parameter index
| | +------------ module
| +---------- task
+---------- C32-command

Figure 1: Nomenclature System Commands

INFORMATION FOR COMMAND INPUT:

- Within the start and end identifyer for system parameter commands [ ], space characters (0x20) are tolerated before and after the command parts a, m, x and w.
- Input of new system parameter values:
  - Single elements of one array have to be separated by comma
  - Logical values as single ASCII characters
    (FALSE: optionally '0', 'F', 'f', 'N', 'n' / TRUE: optionally '1', 'T', 't', 'Y', 'y', 'J', 'j')
  - Numerical values in decimal format (default) or alternatively as hexadecimal value or binary number with prefix identifier '0x' resp. '0b'.
- Command input
  - for 16-bit encodings: UTF8, UTF16 and 16Bit
  - since a 16-bit value is expected for 16-bit encodings for entering the command operator, the <ESC> or the <RS> must be supplemented with 00hex. Then the parser continues to work for this command in 8-bit mode.
  - for the encodings: UTF8, UTF16 big endian - here a 00hex must be set before the <ESC>, <RS>.
    example: set parameter 13.9 to 8 bits <0h> <RS> [2: 13.9: 0]
  - for the encodings: UTF16 little endian - here a 00hex must be set after the <ESC>, <RS>.
    example: set parameter 13.9 to 8 bit <RS> <0h> [2: 13.9: 0]
  - with coding 1: = "8-16 bit automatic switching" this is not necessary

Parameter index and task identifier

0	change value of RAM parameter	
1	*change value of EEPROM parameter*	*USER SETTINGS*
2	change value of RAM parameter and activate parameter	
3	*change value of EEPROM parameter and activate parameter*	*USER SETTINGS*
4	delete one or all EEPROM parameter sets	
5	copy RAM parameter	to EEPROM parameter
6	*copy FACTORY parameter*	*to EEPROM parameter actually replaced by firmware parameter*
7	copy FIRMWARE parameter	to EEPROM parameter
8	copy EEPROM parameter	to RAM parameter
9	*copy FACTORY parameter*	*to RAM parameter actually replaced by firmware parameter*
10	copy FIRMWARE parameter	to RAM parameter
11	send RAM parameter	to output
12	send EEPROM parameter	to output
13	*send FACTORY parameter*	*to output actually replaced by firmware parameter*
14	send FIRMWARE parameter	to output
15	send parameter help text	to output
16	send 11,12,13,14, and 15	to output, performed subsequently
17	initialize external EEPROM	

Figure 2: Overview Parameter Index

Attention:

When writing EEPROM parameters use the least possible amount of individual commands for write access. That means at first change the EEPROM parameters for a module only in RAM and copy them subsequently en bloc to the EEPROM with the command <RS>(5:<module no>,0). With the commands <RS>(1:...) and <RS>(3:...) only change individual system parameters.

5.1.1 PARAMETER ACCESS AUTHORISATION

u_a User configuration:
Changes become effective immediately.

u_w User configuration:
Changes become effective after a reset.

e_a Expert configuration:
Not changeable by user, but by expert. Changes become effective immediately.

e_w Expert configuration:
Not changeable by user, but by expert. Changes become effective after a reset.

r/o Factory configuration only

5.1.1.1 EXAMPLES

1. Output of all EEPROM parameters (user settings) via host interface:

<RS>[12:0,0:0]
<1Eh><5Bh><31h><32h><3Ah><30h><2Ch><30h><3Ah><30h><5Dh>

„12“ copy EEPROM parameter to output
„0“ 0 = all modules
„0“ parameter index (complete list)
„0“ digital output via interface

2. Delete all EEPROM parameters:

<RS>[4:0,0]
<1Eh><5Bh><34h><3Ah><30h><2Ch><30h><5Dh>

„4“ delete all EEPROM parameters
„0“ 0 = all modules
„0“ parameter index (complete list)

5.2 LIST OF PRINTER PARAMETERS

Module 3 Power Manager

3,1	u_a	Idle time [s] power off time (depending on hardware) 0 =: off 10 - 32000 in seconds to turn off after the last action

Module 9 Host I/F

9,1	e_w	number of data items sent from data host to receiving queue
9,2	e_w	*placed bit: activates the GeBE ASCII protocol*

Module 10 RS232

10,1	r/o	1: interface disabled / 2: interface enabled
10,2	u_w	baud rate **0: 115200 baud (default)** 1: 921600 baud 2: 460800 baud 3: 230400 baud 4: 115200 baud 5: 57600 baud 6: 38400 baud 7: 19200 baud 8: 9600 baud 9: 4800 baud 10: 2400 baud 11: 1200 baud
on demand:		*12: 600 baud* *255: 3,686,400 baud* *254: 1,843,200 baud*
10,4	u_w	parity — 2 = even / 1 = odd, **0 = no parity**
10,5	u_w	stop bits — 1= 2 stop bits / **0 = 1 stop bit**
10,6	u_w	flow control CTS host→ printer — 1 = CTS enable / **0 = CTS disable**
10,7	u_w	transmit to host — **1 = enabled** / 0 = disabled

Module 11 USB

11,1	r/o	set bit: interface not present /other: interface present

Module 12		Message		
12,3	u_a	1: enable / 0: disable message on every recognized syntax error		
12,4	u_a	1: enable / 0: disable message on every recognized internal system error		
12,5	e_a	1: enable / 0: disable debug status messages on every change		
12,6	e_a	cyclic cumulative debug status message period [ms] / 0: infinite period, i.e. disabled		
12,7	u_a	1: enable / 0: disable emulation specific status messages on every change		
12,8	u_a	cyclic emulation specific system status message period [ms] / 0: infinite period, i.e. disabled		
12,9	e_a	1: enable / 0: disable cumulative debug analog value messages on every change		
12,10	e_a	cyclic cumulative debug analog value message period [ms] / 0: infinite period, i.e. disabled		
12,11	e_a	1: enable / 0: disable emulation specific analogue value messages on every change		
12,12	e_a	cyclic emulation specific analogue value message period [ms] / 0: infinite period, i.e. disabled		
12,13	e_a	EoT notification		
12,14	u_a	0: emulation specific messages only / 1: emulation specific status messages and C32 simultaneously		

Module 13		Parser		
13,1	e_w	parser input buffer size [bytes]	(255 ..... 4000 Bytes)	**default 0x600: 1536 Bytes**
13,2	e_a	threshold value for initiation of a Rx stop [bytes]	(32 ..... 1000 Bytes)	**default 72 Bytes**
13,3	e_a	threshold value for annulment of a Rx stop [bytes]	(64 ..... 1000 Bytes)	**default 250 Bytes**
13,4	u_w	emulation chain 0: no emulation (hex dump mode) 1: similar to C32 system 2: similar to A8 system 3: similar to N78 system 4: similar to HPIR system 5: PCL3 emulation (subset only) 6: similar to FGL system		
13,8	e_a	TRUE: all patch array items are unpatched indexes		
13,9	u_a	'code-decode system' Determines the decoding of the string code at host interface 0: 8 bit Suitable for font sets of max. 256 characters (e.g. ASCII or GeBE standard fonts) 1: 8-16 bit automatic switch. Suitable for GeBE font SimHei_GB2312_24x224_9 2: UTF16 big endian. Suitable for character sets of max. 65536 characters. 3: UTF16 little endian. Suitable for character sets of max. 65536 characters. 4: UTF8. Suitable for character sets of max. 65536 characters.		
13,10	u_w	2: no POR batch file / 1: POR text file no. 16		

Module 14		Parser_GeBE_C32 (GeBE forms mode)	
14,1	u_w	default font index	**0: system font**
14,2	u_w	default barcode font index	**0: system font**
14,3	u_w	default frame orientation	0:= Portrait 1:= Landscape 2:= Portrait upside down 3:= Landscape uside down
14,4	u_w	redirection of font indices	[0 .....31]
14,5	u_w	redirection of logo indices	[0 .....31]
14,6	u_w	redirection of canned command indices	[0 .....31]
14,7	u_a	status message format	0:= debug messages, **1:= C32,** 2:= A8
14,8	e_a	barcode CODE39 human readable text	strip asterisks 0:= with *, 1:= without *
14,9	e_a	control characters	0: ignore all / 1: execute CR and LF
14,13	u_w	autoload batch	0: disable / 1: enable

Module 15		Parser_GeBE_A8 (GeBE line mode)	
15,1	u_w	default font index	**0: system font**
15,2	u_w	default barcode font index	**0: system font**
15,3	u_w	TRUE: data mode	**F:= Portrait 0°(text mode)** T:= Portrait 180°(data mode)
15,4	u_w	redirection of font indices	[0 .....31]
15,5	u_w	redirection of logo indices	[0 .....31]
15,6	u_w	redirection of canned command indices	[0 .....31]
15,7	u_a	status message format	0:= debug messages 1:= C32 messages **2:= A8 messages**
15,8	e_a	barcode Code39 human readable text	strip asterisks 0:= with * 1:= without *
15,13	u_w	0: disable / 1: enable autoload batch	0:= disabled 1:= enabled

Module 16		Parser_GeBE_N78		
16,1	u_w	default font index	**1: = GeBE N78 16x24 GB216x24.addon**	
16,2	u_w	default barcode font index	**1: = GeBE N78 16x24 GB216x24.addon**	
16,3	u_w	TRUE: Data mode	F:= Portrait 0°(text mode) T:= Landscape 180°(data mode)	
16,4	u_w	redirection of font indices	[0 .....31]	
16,5	u_w	redirection of logo indices	[0 .....31]	
16,6	u_w	redirection of canned command indices	[0 .....31]	
16,7	u_a	status message format	0:= debug messages	**3:= N78 messages**
16,8	e_a	barcode Code39 human readable text	strip asterisks	0:= with *, 1:= without *
16,13	u_w	0: disable / 1: enable autoload batch		
16,14	u_w	Reset type for ESC @	0: warm reset 1: delete all buffers 2: weak reset: set all attributes to default	

Module 17		Renderer		
17,1	e_w	number of job buffer elements		
17,2	e_w	number of object list elements		
17,3	e_w	size of renderer RAM [byte]		
17,4	u_a	number of leftmost dot within printing area	0: no left margin	
17,5	u_a	number of leftmost dot beyond printing area / at least printer width		0: no right margin
17,6	u_a	number of topmost dot line from top of form within printing area		0: no top margin
17,7	u_a	number of topmost dot line from top of form beyond printing area		0: uncapped
17,23	u_w	basic scaling factor X	***HINT: Restrictions on barcodes and logos***	
17,24	u_w	basic scaling divisor X	***HINT: Restrictions on barcodes and logos***	
17,25	u_w	basic scaling factor Y	***HINT: Restrictions on barcodes and logos***	
17,26	u_w	basic scaling divisor Y	***HINT: Restrictions on barcodes and logos***	

Module 18		Print	
18,1	e_w	size of image drum [dot lines]	
18,2	e_w	starting speed of printer motor [mm/s]	
18,3	u_a	maximum speed of printer motor [mm/s]	(max.=500)
18,4	u_a	maximum backwards speed [mm/s]	(max.=500)
18,5	e_w	printer motor backlash [dot lines]	
18,6	u_a	printing density	**default = 25**
18,7	u_a	bytes per soft strobe	
18,8	u_a	heated bytes per soft strobe	
18,9	e_a	latency after stall condition [ms]	**default = 40**
18,10	u_w	stall timeout [ms]	
18,11	r/o	motor follow-up time [ms]	
18,12	u_a	1: ticket length control enabled / 0: ticket length control disabled	
18,13	u_a	blackmark minimum length [dot lines] number of black lines until detection of blackmark	**default = 3** dot lines
18,14	u_a	blackmark sensor index 0: PaperEnd 1: AUX1 2: AUX2 3: AUX3 4: AUX4 5: AUX5 6: AUX6 7: AUX7	
18,16	u_a	distance from start of most recent blackmark to form feed position [dot lines]	TOF = 0 dot lines
18,17	u_a	number of segments per ticket blackmarks per ticket 0-31 0: each blackmark is used and is overwriteable 1: each blackmark is used, too long printouts will be cut 2: every other blackmark is used, too long printouts will be cut 3: every third blackmark is used, too long printouts will be cut	0: variable number default = 0
18,21	u_a	1: autoload enabled / 0: autoload disabled	
18,22	u_a	maximum autoload speed [mm/s]	**default: minimum speed** of printer mechanism [mm/s]
18,23	u_a	autoload feed length [mm]	**default = 100,** max. 255 mm
18,24	u_a	autoload stopping status 0: ToF – Attention! Parameter 18,39 > 18,23 1: AUX1 2: AUX2 3: AUX3 4: AUX4 5: AUX5 6: AUX6 7: AUX7	**default = 0**
18,25	u_a	1: reverse autoload feed	
18,26	u_a	subsequent backwards feed to autoload [dot lines], negative =: feed forwards	
18,27	u_a	extension of printable area after PaperEnd detection [dot lines]	0-500 (**default = 122**)
18,28	e_w	printer motor step mode	0: full step / dot line 1: half step / dot line
18,29	e_a	fixed strobes on time [µs]	0: variable strobes on time

18,30	e_a	minimum strobes off time [µs]	
18,31	e_w	acceleration of printer motor [mm/s²]	
18,32	e_a	fixed S.L.T. [µs]	0: variable S.L.T.
18,33	u_a	T: autoflush on PaperEnd	
18,34	u_a	autoflush timeout [s]	0: no timeout
18,35	u_a	T: EoT terminates autoflush	
18,36	e_w	micro step resolution: 1...8	
18,37	e_w	motor current: 0...31	(hardware dependent)
18,38	e_w	motor current in general 0: = low / 1: = high	(hardware dependent)
18,39	u_a	ticket segment length: e.g. distance of blackmarks [decimillimeter]	
18,40	u_a	ticket length allowance for blackmark detection [mm]	
18,41	u_a	minimum paper snippet length [mm]	for consecutive form feeds
18,42	u_a	autoflush on head up	**default: F:** none
18,43	u_a	feed at PaperEnd also	**default: F:** no feed
18,44	e_a	anti jam sensor, 0: NONE , 1: AUX1 , 2: AUX2 , 3: AUX3 , 4: AUX4 , 5: AUX5 , 6: AUX6 , 7: AUX7	
18,45	u_w	motor current settings for STSPIN220: from minimum to maximum value in %	
18,46	u_w	*motor stall detection, minimum speed in mm/s*	*(default 150, 0 = off)*
18,47	u_a	*history sceme*	
18,48	u_a	*print history strobe length percentage*	
18,49	u_a	batch no. 17 printer motor idle timeout [s], 0 : no timeout	(1–64000 seconds = 18 hours)
18,50	u_a	T. HeadDn terminates autoflush	
18,51	e_a	threading sensor, 0: NONE , 1: AUX1 , 2: AUX2 , 3: AUX3 , 4: AUX4 , 5: AUX5 , 6: AUX6 , 7: AUX7	
18,52	u_a	feed lenght ToF to threading sensor [decimillimeter]	
18,53	u_a	T. autoflush on feed jam	

Module 19		Sensors		
19,1	e_w	sensor evaluation period [ms]		(20 ms)
19,2	u_w	AUX1 sensor	polarity: 0 = low =active / 1 = high = active state for reflex sensor	
19,3	u_w	AUX2 sensor	polarity: 0 = low =active / 1 = high = active state for reflex sensor	
19,4	u_w	AUX3 sensor	polarity: 0 = low =active / 1 = high = active state for reflex sensor	
19,5	u_w	AUX4 sensor	polarity: 0 = low =active / 1 = high = active state for reflex sensor	
19,6	u_w	AUX5 sensor	polarity: 0 = low =active / 1 = high = active state for reflex sensor	
19,7	u_w	AUX6 sensor	polarity: 0 = low =active / 1 = high = active state for reflex sensor	
19,8	u_w	AUX7 sensor	polarity: 0 = low =active / 1 = high = active state for reflex sensor	
19,9	u_w	near PaperEnd sensor	polarity: 0 = low =active / 1 = high = active state for reflex sensor	
19,30	e_a	MCU Pin AD_NearPaperEnd: ADC lower limit	[raw data values below convert to low logic level]	
19,31	e_a	MCU Pin AD_NearPaperEnd: ADC upper limit	[raw data values above convert to high logic level]	
19,32	e_a	MCU Pin AD_AUX1: ADC lower limit	[raw data values below convert to low logic level]	(0x0555)
19,33	e_a	MCU Pin AD_AUX1: ADC upper limit	[raw data values above convert to low logic level]	(0x0AAA)
19,34	e_a	AUX2: ADC lower limit of analog level		(0x0745)
19,35	e_a	AUX2: ADC upper limit of analog level		(0x0AAA)
19,36	e_a	MCU Pin AD_AUX5: ADC lower limit	[raw data values below convert to low logic level]	
19,37	e_a	MCU Pin AD_AUX5: ADC upper limit	[raw data values above convert to high logic level]	
19,38	e_a	MCU Pin AD_AUX7: ADC lower limit	[raw data values below convert to high logic level]	
19,39	e_a	MCU Pin AD_7: ADC upper limit	[raw data values above convert to high logic level]	
19,40	r/o	MCU Pin AD_HeadTemp: ADC lower limit	[raw data values below convert to high logic level]	
19,61	e_w	MCU Pin AD_VP: ADC upper lower hysteresis threshold		
19,62	e_w	MCU Pin AD_VP: ADC upper limit hysteresis threshold		
19,63	e_w	MCU Pin AD_HeadTemp: ADC lower limit hysteresis threshold		
19,64	e_w	MCU Pin AD_HeadTemp: ADC upper limit hysteresis threshold		
19,65	e_w	MCU Pin AD_MotorTemp: ADC lower limit hysteresis threshold		
19,66	e_w	MCU Pin AD_MotorTemp: ADC upper limit hysteresis threshold		
19,69	e_w	MCU Pin AD_BatteryTemp: ADC upper lower hysteresis threshold		
19,70	e_w	MCU Pin AD_BatteryTemp: ADC upper limit hysteresis threshold		
19,71	e_w	MCU Pin AD_HeadAlignment: ADC upper lower hysteresis threshold		
19,72	e_w	MCU Pin AD_HeadAlignment: ADC upper limit hysteresis threshold		

Module 20		Parser_FGL	
20,1	u_w	default font index	14: = FGL font3 17x31
20,2	u_w	default barcode font index	14: = FGL font3 17x31
20,3	u_w	default frame orientation	0:= portrait, 1:= landscape, 2:= portrait upside down 3:= landscape upside down
20,4	u_w	redirection of font indices	[0 .....31]
20,5	u_w	redirection of logo indices	[0 .....31]
20,6	u_w	redirection of canned command indices	[0 .....31]
20,7	u_a	status message format	0:= debug messages **6: FGL messages** 2: C32 messages
20,8	e_a	barcode CODE39 human readable text	strip asteriks 0:= with * / 1:= without
20,13	u_w	0: disable / 1: enable autoload batch	

Module 22		Cutter
22,1	e_a	maximum cutter motor speed [pps]
22,2	e_a	half cut: number of pulses
22,3	u_w	distance from dot line to cut [dot lines]
22,4	u_a	TRUE: always move cutter blade to home position on init command

		FALSE: move blade only when out of home position	
22,5	e_a	micro step resolution: 1...8	
22,6	e_w	motor current: 0...31	(hardeware dependent)

Modul 23 Winder

23,1	u_a	feed length ToF to picking sensor [decimillimeter]	
23,2	u_a	feed length ToF to peeler [decimillimeter]	
23,3	u_a	winder motor start speed [PPS]	
23,4	u_a	winder motor final speed [PPS]	
23,5	u_a	number of winder motor speedup stages	
23,6	e_w	microstep resolution, range 1...8	
23,7	e_w	motor current: 0...31	
23,9	u_a	run-down time fwd [ms]	
23,10	u_a	forward time fwd [ms]	
23,11	u_a	forward time bwd [ms]	
23,12	u_a	continuation time bwd [ms]	
23,13	u_a	winding time fwd [ms]	

Module 24 End of Ticket (EoT)

24,1	u_w	output equipment	0: cutting edge / 1: cutter
24,2	u_a	1: enable cuts / 0: disable cuts	
24,3	u_a	distance between cuts [2 dot lines]	0: single cut
24,4	u_a	1: partial cut / 0: full cut	
24,5	u_a	1: enable feed after cut / 0: disable feed after cut	
24,6	u_w	picking timeout [seconds]	0: never wait / 255 wait forever
24,7	u_a	TRUE: 1st cut only on PaperEnd condition FALSE: inhibit cuts	
24,8	u_a	eject length [dot lines]	0: do not eject paper on PaperEnd condition
24,9	u_a	picking sensor 0: NONE, 1: AUX1, 2: AUX2, 3: AUX3, 4: AUX4, 5: AUX5, 6: AUX6, 7: AUX7	

Module 26 Parser_HPIR

26,1	u_w	default font index	
26,2	u_w	default barcode font index	
26,3	u_w	default text orientation	0:= portrait 0° (text mode), T:= landscape 180°(data mode)
26,4	u_w	font patch table	[0 .....30]
26,5	u_w	logo patch table	[0 .....31]
26,6	u_w	batch patch table	[0 .....31]
26,7	u_a	status message format	0:= debug messages **4: no messages**
26,13	u_w	autoload batch file T2	0: = disabled 1: = enabled

Module 27 Menu

27,1	u_a	menu 1: = inexistent / 0: = existent
27,2	u_a	*index of menu standard text*

Module 30 Msg_GeBE_A8

30,1	u_a	1: enable / 0: disable	1st	statusbyte
30,2	u_a	1: enable / 0: disable	2nd	statusbyte
30,3	u_a	1: enable / 0: disable	3rd	statusbyte
30,4	u_a	1: enable / 0: disable	4th	statusbyte

30,5	u_a	1: enable / 0: disable	5th statusbyte
30,6	u_a	A8 statusbyte 4 bit 0:	current number of C32 message flag e.g. 70
30,7	u_a	A8 statusbyte 4 bit 1:	
30,8	u_a	A8 statusbyte 4 bit 2:	
30,9	u_a	A8 statusbyte 4 bit 3:	
30,10	u_a	A8 statusbyte 5 bit 0:	
30,11	u_a	A8 statusbyte 5 bit 1:	
30,12	u_a	A8 statusbyte 5 bit 2:	
30,13	u_a	A8 statusbyte 5 bit 3:	

Module 37 Actuators

37,1	e_a	actuator 1	HeadLift: motor speed [pps]
37,2	e_a	actuator 2	HeadPush: motor speed [pps]
37,3	e_a	actuator 3	Opt1: motor speed [pps]
37,4	e_a	actuator 4	Opt2: motor speed [pps]
37,5	e_a	HeadLift: max. vertical lift beyond HeadUp sensor detection [steps]	
37,6	e_a	HeadLift: required number of additional down steps after sensor match	
37,7	e_a	HeadLift: allowance for HeadUp sensor detection [steps]	
37,8	e_a	HeadPush: platen force reference value [dN]	
37,9	e_w	motor current: 0...31	(hardeware dependent)

Module 38 Parser_PCL3

38,1	u_w	default font index	
38,2	u_w	default barcode font index	
38,3	u_w	TRUE: data mode	0:= portrait 0° (text mode), T:= portrait 180°(data mode)
38,4	u_w	font patch table	[0 .....31]
38,5	u_w	logo patch table	[0 .....31]
38,6	u_w	batch patch table	[0 .....31]
38,7	u_a	status message format	0:= debug messages, 1: C32 messages, 2: A8 messages
38,8	e_a	barcode CODE39 human readable text: strip asterisks	0:= with * / 1:= without
38,9	u_a	default maximum empty rows	
38,10	u_a	end of graphic with form feed	<ESC>"*rC" and <ESC>"*rB" perform a formfeed
38,11	u_a	end of graphic with end of ticket	<ESC>"*rC " and <ESC>"*rB "perform an end of ticket (EoT)
38,12	u_a	default batch index	<ESC>"*rC "and <ESC>"*rB" run batch file number 38,12 0:= no batch file will be executed
38,13	u_w	0: disable / 1: enable autoload batch	

Modul 39 MSG_LED

39,2	u_w	0 : all LEDs without function 1: LED on, flashing in case of error 2: LED off, flashing in case of error 4: GeBE_N78 mode 1 6: GeBE test printer

Module 41 Printhead specification

Module 41 contains all printhead specific data

41,1	e_w	printhead class	0: standard flat head / 1: KCE53
41,2	e_w	horizontal dot density [dots/dm]	

41,3	e_w	vertical dot density [dots/dm]	
41,4	e_w	*number of spare dots left*	
41,5	e_w	number of heatable dots per line	
41,6	e_w	*number of spare dots right*	
41,7	e_w	number of simultaneously heatable dots per line	
41,8	e_w	nominal average dots resistance [Ohm]	
41,9	e_w	contact resistance [mOhm]	
41,10	e_w	nominal thermistor resistance [kOhm]	
41,11	e_w	thermistor B value [K]	
41,12	e_w	SPI clock frequency [MHz]	1/2 divider rates of 168 Mhz at F4 systems
41,13	e_w	SPI data direction *(fix 0 not implemented yet)*	**0: leftmost dot first**, 1: rightmost dot first
41,14	e_w	SPI clock polarity	**0: CK to LO when idle**, 1: CK to HI when idle
41,15	e_w	SPI clock phase	**0: 1st clock transition is first data capture edge,** 1: 2nd clock transition is first data capture edge
41,16	e_w	0: single, 1: *multiple* data in for each dot group	*(not implemented yet)*
41,17	e_w	0: single, 1: *multiple* strobes for each dot group	*(not implemented yet)*
41,18	e_w	number of dots per group	*(must not be zero - not implemented yet)*
41,19	e_w	strobe polarity	0: negative pulse, 1: positive pulse
41,20	e_w	typical strobes ON time [µs]	working point at 25°C / speed low
41,21	e_w	minimum strobes ON time [µs]	
41,22	e_w	maximum strobes ON time [µs]	
41,23	e_w	VP [mV] belonging to 1st data point of Vp heating adjustment table	
41,24	e_w	Vp heating adjustment table:	input value difference [mV]
41,25	e_w	Vp heating adjustment table:	scaling bits
41,26	e_w	Vp heating adjustment table	
41,27	e_w	temperature ADC value belonging to 1st data point of head temperature heating adjustment table	
41,28	e_w	head temperature heating adjustment table:	input value difference [ADC value]
41,29	e_w	head temperature heating adjustment table:	scaling bits
41,30	e_w	head temperature heating adjustment table	
41,31	e_w	line time [µs]	belonging to 1st data point of line time heating adjustment table
41,32	e_w	line time heating adjustment table:	input value difference [µs]
41,33	e_w	line time heating adjustment table:	scaling bits
41,34	e_w	line time heating adjustment table	
41,35	e_w	preheat temperature [°C] [°F]	
41,36	e_w	preheat temperature tolerance [1/10°C] [33.8/50°F]	
41,37	e_w	preheat strobes on time [µs], 0: preheat strobes disabled	
41,38	e_w	preheat pattern	
41,39	e_w	preheat cycle time [µs]	
41,40	e_w	head platen force [dN]	
41,41	e_w	head platen force tolerance [dN]	
41,42	e_w	head alignment [µm]	
41,43	e_w	head alignment tolerance [µm]	
41,44	e_w	preheat stage 2 cycle time [µs]	

Module 42 Rotary cutter

42,3	e_w	distance from dot line to cut [dot lines]
42,4	u_w	TRUE: always move cutter blade to home position on init command FALSE: move blade only when out of home position

Module 44		Charge	
44,1	e_w	battery type	**1**
44,2	e_w	maximum charging time	**26**
44,3	e_w	Vp_Count	**60**
44,4	e_w	Vp_Repeat	**3**
44,5	e_w	Vp_Delta t	**10**
44,6	e_w	maximum Vp	**155**
44,7	e_w	maximum Vp_Max_Repeat	**16**
44.8	e_w	Temp_Count	**60**
44,9	e_w	Temp_Repeat	**4**
44,10	e_w	Temp_Delta	**100**
44,11	e_w	maximum temperature	**79**
44,12	e_w	maximum Temp_Repeat	**3**
44,13	e_w	minimum Vp	**139**
44,14	e_w	Trickle_Charge_OnTime	**1**
44,15	e_w	minimum Vp_Delta	**128**

Module 48		Scanner	
48,1	e_a	barcode scanner enabled	
48,2	e_a	baud rate index	8: = 9600
48,3	e_a	end of scan code	0x0A: = line feed

5.2.1 EXAMPLES OF PRINTER PARAMETER PROGRAMMING

– Writing RAM parameters of module 18 into EEPROM (user settings):
<RS>[5:18,0] <1Eh><5Bh> <35h><3Ah><31h><38h><2Ch><30h> <5Dh>
„5“ copy RAM parameter → EEPROM-Parameter
„18“ module print
„0“ parameter index (complete list)

– Changing density in RAM PrinterUserParams to value „30“:
<RS>[2:18,6:30] <1Eh><5Bh><31h><3Ah><31h><38h><2Ch><36h><3Ah><33h><30h><5Dh>
„2“ changing RAM parameters with subsequent activation
„18“ module print
„6“ parameter index Print_DENSITY
„30“ density value

– Send EEPROM parameter of module 17 (renderer) to an active interface:
<RS>[12:18,10:0] <1Eh><5Bh><31h><32h><3Ah><31h><38h><2Ch><31h><30h><3Ah><30h><5Dh>
„12“ copy EEPROM parameter output
„17“ module Msg
„10“ stall timeout
„0“ output to interface

– Example of a module 29 Msg_GeBE_C32 answer for parameter value 18,10:
[AR:0x0000002C#[0x004CE0C7ms0x001D0001:(UEEP): 18,10: 2000 #i__]#0x9624A77E:AR]
Answer: stall timeout time = 2000ms

5.3 REQUEST SYSTEM INFORMATION

[Name]	Request System Infos					
[Format]	ASCII	<RS>	“x“	“ i“	n	m
	Hex	1Eh	78h	69h	n	m
	Decimal	30d	120d	105d	n	m
[Description]	parameter x, X defines the output destination:					
	x	information will be sent to host				
	i	"i"				

n = 00d	ADDONSPARE	total available memory capacity for AddOns
n = 01d	ADDON	read AddOn
	m = ADDON number	
n = 02d	NUMBER_OF_ADDONS	number of AddOns in the system
n = 03d	MCU_SERIAL	serial number of MCU
n = 04d	GE_APPL	firmware version status (GE number) of application
n = 05d	GE_BOOT	firmware version status (GE number) of boot loader
n = 06d	DEVICE_BOOT	device name of boot loader firmware
n = 07d	REPOSREV_BOOT	repository revision of boot loader
n = 08d	REPOSREV_APPL	repository revision of application
n = 09d	HWSPEC_APPL	hardware specification of application
n = 10d	DEVICE_APPL	device name of application firmware
n = 11d	ITEMNUMBER_APPL	item number
n = 12d	ITEMDESCRIPTION_APP	item description
n = 13d	*ADDONSPARE_APPL*	*total available memory capacity for AddOn program (for application only)*
n = 14d	GeBE_Serial	serial number assigned by GeBE (tags area)
n = 15d	FONT	read font (for application only)
	m = font index	(index 0: system font)
n = 16d	NUMBER_OF_FONTS	number of fonts in system (for application only)
n = 17d	LOGO	read logo (for application only)
	m = logo index	(index 0: system logo)
n = 18d	NUMBER_OF_LOGOS	number of logos in system (for application only)
n = 19d	BATCH	read standard text (for application only)
	m = standard text index	(index =: 1 standard text from AddOn range)
n = 20d	NUMBER_OF_BATCHES	number of standard text batches in system (for application only)
n = 21d	HARDWARE-VERSION	Request command for hardware name of the application
n = 22d	FIRMWARE CRC	CRC check sum of loaded firmware
n = 23d	BOOTLOADER CRC	CRC check sum of bootloader
n = 24d	HEAPSPARE	available heap space
n = 25d	GP_APPL	request of software package no. (GP number)

5.3.1 EXAMPLE: READ NUMBER OF ADDONS

– Response example of module 6 Flash, with message ID 0x0006 and message code 0x0002:
[aR:0x00000022#[0x000E35ABms0x00060002: 0x00000014 #i__]#0x201B7970:aR]

Answer: (14Hex) = 20 AddOns are stored in Flash memory

5.4 PRINT SYSTEM DATA

[Name]	Request System Info				
[Format]	ASCII	<RS>	“y“	<output format>	<source> <type>
	Hex	1Eh	78h	<output format>	<source> <type>
	Decimal	30d	120d	<output format>	<source> <type>

[Description] Command to insert system data into the data stream. The attached system data do not have any own formatting and thus the current formatting is used for the printout.

[Output format]

0:	output in standard form (as it is also read out)	e.g. 0x06 or 0x0CF4
1:	output in decimal form	e.g. 3472 or -184
2:	output in Boolean form	e.g. „T“ or „F“
3:	output as string	
10:	output of a converted ADC voltage in decimal form without decimal place	e.g. 24
11:	output of an analog value in decimal form with a decimal place	e.g. 4.3
12:	output of an analog value in decimal form with two decimal places	e.g. 14.36
13:	output of an analog value in decimal form with three decimal places	e.g. 0.332
20:	output of a converted temperature in decimal form without decimal place	e.g. 24
21:	output of a converted temperature in decimal form with a decimal place	e.g. 4.3
22:	output of a converted temperature in decimal form with two decimal places	e.g. 14.36
23:	output of a converted temperature in decimal form with three decimal places	e.g. 0.332

If an unsupported output form is specified, the output is selected hexadecimal.

[Source]

"A": position counter of actuators

Example: Insert the position counter of actuator 1 in decimal form into the print data flow
<RS>y<1d>A<1d>

„a“ : <ADC Index> sensor data

Example: Insert the power voltage (ADC index 8) with 2 decimal places into the print data flow
<RS>y<12d>a<9d>

„i“ : System information

Example: Insert REPOS_REF of the application (index 8) into the print data flow
<RS>y<3d>i<8d>

„f“ :System parameter [task identification, module, parameter]

Task identification: 11 RAM parameter
12 EEPROM parameter
13 *factory set parameter (actually replaced by firmware parameters)*
14 firmware parameter
15 parameter help text

Example: Insert RAM parameter, vertical dot density [dots/dm] into the print data flow
<RS>y<0d>f[11:41,3]

„r“: reference data

„s“: statistic data

„t“ :tracing data

5.5 PRINTER REFERENCE VALUES

5.5.1 READ PRINTER REFERENCE VALUES

[Name]	Request reference values				
[Format]	ASCII	<RS>	“x“	“ r“	n
	Hex	1Eh	78h	72h	n
	Decimal	30d	120d	114d	n
[Description]	Read reference value(s) for index n.				

5.5.2 WRITE PRINTER REFERENCE VALUES

[Name]	Write reference values					
[Format]	ASCII	<RS>	“w“	“ r“	n	[value]
	Hex	1Eh	77h	72h	n	[value]
	Decimal	30d	119d	114d	n	[value]
[Description]	Write reference value(s) for index n with value. Input of [value] in ASCII-Format within square brackets. Only available in expert mode.					

5.5.3 DELETE PRINTER REFERENCE VALUES

[Name]	Delete reference values from ADC value				
[Format]	ASCII	<RS>	“d“	“r“	n
	Hex	1Eh	63h	72h	n
	Decimal	30d	99d	114d	n
[Description]	Delete reference value(s) for index n. Only available in expert mode.				

5.5.4 COPY PRINTER REFERENCE VALUES (ACTUAL VALUE)

[Name]	Copy reference values from ADC value				
[Format]	ASCII	<RS>	“c“	“r“	n
	Hex	1Eh	63h	72h	n
	Decimal	30d	99d	114d	n
[Description]	Copy reference value(s) for index n from actual measured value. Only available in expert mode.				

5.5.5 INCREMENTING REF VALUE 54 AND 55

[Name]	Increment ref value 54 or 55				
[Format]	ASCII	<RS>	“i“	“r“	[n]
	Hex	1Eh	69h	72h	[n]
	Dezimal	30d	105d	114d	[n]
[Description]	Increase ref value by the value n.				

Index n of reference values (in EEPROM)		kind of access (x, w, d, c)	
n = 00d	all reference values (reading only)	x	
n = 01d	head name	x, w, d	
n = 02d	serial number 1	x, w, d	
n = 03d	serial number 2	x, w, d	
n = 04d	weight of head module (g)	x, w, d	
n = 05d	head alignment	x, w, d, c	
n = 06d	actual average of dot resistance [Ohm]	x, w, d, c	(copy dot test, actually from param. 41,8)
n = 07d	head push: platen force reference value [dN]	x, w, d	
n = 08d	poscount actuator1 reference value (HEADLIFT)	x, w, d	
n = 09d	poscount actuator2 reference value (HEADPUSH)	x, w, d	
n = 10d	poscount actuator3 reference value	x, w, d	
n = 11d	poscount actuator4 reference value	x, w, d	
n = 12d	ADC value AUX1 reference value	x, w, d, c	
n = 13d	ADC value AUX2 reference value	x, w, d, c	
n = 14d	ADC value AUX3 reference value	x, w, d, c	
n = 15d	ADC value AUX4 reference value	x, w, d, c	
n = 16d	ADC value AUX5 reference value	x, w, d, c	
n = 17d	ADC value AUX6 reference value	x, w, d, c	
n = 18d	ADC value AUX7 reference value	x, w, d, c	
n = 19d	ADC value VP reference value	x, w, d, c	
n = 20d	ADC value MotorTemp reference value	x, w, d, c	
n = 21d	ADC value HeadTemp reference value	x, w, d, c	
n = 22d	ADC value PaperEnd reference value	x, w, d, c	
n = 23d	ADC value NearPaperEnd reference value	x, w, d, c	
n = 24d	ADC value Vref reference value	x, w, d, c	
n = 25d	correction value 1	x, w, d	
n = 26d	correction value 2	x, w, d	
n = 27d	correction value A	x, w, d	
n = 28d	correction value B	x, w, d	
n = 29d	ADC battery temp	x, w, d, c	
n = 30d	correction value 3	x, w, d	
n = 31d	correction value C	x, w, d	
n = 32d	head temperature value	x, w, d	
n = 33d	head attachment offest	x, w, d, c	
n = 34d	motor temp high limit	x, w, d, c	
n = 35d	user serial number	x, w, d	(can be changed without expert mode)
n = 36d	user string1	x, w, d	(can be changed without expert mode)
n = 37d	user string2	x, w, d	(can be changed without expert mode)
n = 38d	correction value U1_low	x, w, d	
n = 39d	correction value U2_low	x, w, d	
n = 40d	heater size width [dµm = 100 nm]	x, w, d	
n = 41d	heater size height [dµm = 100 nm]	x, w, d	
n = 42d	motor temp correction	x, w, d	
n = 43d	thermistor R25	x, w, d	
n = 44d	thermistor B value	x, w, d	
n = 45d	battery temp correction	x, w, d	
n = 46d	battery temp R25	x, w, d	

n = 47d	battery temp B value	x, w, d	
n = 48d	correction value U1_mid	x, w, d	
n = 49d	correction value U2_mid	x, w, d	
n = 50d	correction value U1_high	x, w, d	
n = 51d	correction value U2_high	x, w, d	
n = 52d	not used!	x, w, d	
n = 53d	description power supply	x, w, d	
n = 54d	custom counter 1	x, w, d	z.B. increment 1: <RS>ir<54d>[1]
n = 55d	custom counter 2	x, w, d	

Figure 3: Overview Index n of reference values (in EEPROM)

VALUES for:

n = 53d description power supply

Power supply	GNG-24V-2,7A	GNG-24V-6,5A
Printer mechanism		
HSP-2208		
HSP-3208		2187, 1106, 6250, 213, 497
HSP-3512	3333, 95, 4125, 45, 160	3542, 380, 5860, 150, 38
ELM4071		
ELM3071		
ELM208LV		
ELM-208HS		

5.6 STATISTICS

Statistics stores certain printer actions ("events") in order to plan optimal printer service. The events recorded and accumulated in the statistics are stored in the EEPROM in the form of statistic entries in a ring buffer. Older entries will be overwritten by current ones. Each statistic entry includes the event type, the associated accumulated count, and required administrative information. The individual events are accumulated in 40-bit counters, a single statistic entry can thus max. 1,099,511,627,775 events included. The ring buffer is capable of storing a total of 2048 such statistics entries - distributed across all event types. The counter readings are updated with each event directly in the RAM and written as statistics entries from time to time in the background into the EEPROM. The counter readings in the statistics entries are not resettable. Converted into appropriate units, they can be read out by command. It should be noted that by randomly switching the printer on and off without using the shut-down command **(ATTENTION: command exists only for systems with disabled VCC)**, it may happen that individual statistics entries are not or not completely be written before the voltage supply of the EEPROM broke-down. The next time you turn it on, the most up-to-date statistics entries in the EEPROM must be used as the basis for the further event counting. This particularly concerns data on the running performance of the printer or printhead, which can then show significant errors.

5.6.1 READ STATISTIC VALUES

[Name]	Request statistic values				
[Format]	ASCII	<RS>	“x“	“s“	n
	Hex	1Eh	78h	73h	n
	Decimal	30d	120d	115d	n
[Description]	Read statistic value for index n				

Index n of statistic data (in EEPROM)

n = 00d	Total statistics
n = 01d	Number of power-on cycles
n = 02d	Operating time of the printer in seconds
n = 03d	Operating time of the printhead in seconds
n = 04d	Number of print starts at head temperature below -40°C
n = 05d	Number of print starts at head temperature from -40°C to -30°C
n = 06d	Number of print starts at head temperature from -30°C to -20°C
n = 07d	Number of print starts at head temperature from -20°C to -10°C
n = 08d	Number of print starts at head temperature from -10°C to 0°C
n = 09d	Number of print starts at head temperature from 0°C to +10°C
n = 10d	Number of print starts at head temperature from +10°C to +20°C
n = 11d	Number of print starts at head temperature from +20°C to +30°C
n = 12d	Number of print starts at head temperature from +30°C to +40°C
n = 13d	Number of print starts at head temperature from +40°C to +50°C
n = 14d	Number of print starts at head temperature from +50°C to +60°C
n = 15d	Number of print starts at head temperature from +60°C to +70°C
n = 16d	Number of print starts at head temperature from +70°C to +80°C
n = 17d	Number of print starts at head temperature from +80°C or more
n = 18d	Mileage of the system in meters
n = 19d	Mileage of the printhead in meters
n = 20d	Mileage of the printhead in dot lines
n = 21d	Paper length since the last paper change in 1/10 meter
n = 22d	Number of detected changes to PaperEnd (PE)
n = 23d	Number of detected head-up's
n = 24d	Number of cutter cuts
n = 25d	Number of EoT operations
n = 26d	Number of failed system schut-downs via VCC_enable

Figure 4: Overview index n statistic data (EEPROM)

5.7 LED BEZEL

The C32 System is capable to operate a LED bezel.

5.7.1 BLINK LED BEZEL

[Name]	Disable LED Blink (synchronously with printer data)					
[Format]	ASCII	<RS>	“$“	<0d>	<0d>	<0d>
	Hex	1Eh	24h	<0h>	<0h>	<0h>
	Decimal	30d	36d	<0d>	<0d>	<0d>
[Name]	Enable LED Blink (synchronously with printer data)					
[Format]	ASCII	<RS>	“$“	<0d>	<1d>	<0d>
	Hex	1Eh	24h	<0h>	<1h>	<0h>
	Decimal	30d	36d	<0d>	<1d>	<0d>
[Name]	Disable LED Blink Disable LED Blink (immediately)					
[Format]	ASCII	<RS>	“$“	<1d>	<0d>	<0d>
	Hex	1Eh	24h	<1h>	<0h>	<0h>
	Decimal	30d	36d	<1d>	<0d>	<0d>
[Name]	Enable LED Blink n (immediately)					
[Format]	ASCII	<RS>	“$“	<1d>	<1d>	<0d>
	Hex	1Eh	24h	<1h>	<1h>	<0h>
	Decimal	30d	36d	<1d>	<1d>	<0d>

The LED bezel can blink to indicate the printer status. Either a command or an end of ticket (EoT) enables blinking. To disable the blinking either use a command or "open" the exit sensor if it was activated in the parameters.

Information: The blinking after EoT is stopped one minute after EoT at the latest.

[Example] The bezel is to blink as soon as the printing starts.
This is done with the command: <RS>$<0d><1d><0d><RS>$<0d><1d><0d> before the start.
The flashing is terminated by the command: <RS> $ <0d> <0d> <0d> or after end of the EoT sequence.
If a "picking sensor" is defined in parameter 24,9, the flashing is not terminated until it becomes free.
In parameter 12, 13, an automatic flashing can also be defined during the EoT phase.

5.8 ACTUATORS

The C32 System is capable to operate up to 4 actuators.

5.8.1 MOVE ACTUATOR

[Name]	move actuator						
[Format]	ASCII	<RS>	"#"	<"n" >	<"m" >	lh	ll
	Hex	1Eh	23h	<"n" >	<"m" >	lh	ll
	Decimal	30d	35d	<"n" >	<"m" >	lh	ll

<RS> # „Actuator number" „<direction>" <quantityH><quantityL>

n:= number of actuator is a CHARACTER value

„0" head lift

„1" head contact pressure

„2" head alignment

„3" open printer mechanism

m:= direction of motion

„+" moving out (closed printer head, increase head contact pressure)

„-" moving in

"c" automatic moving out (closed printer head, increase head contact pressure)

"o" automatic moving in

lh and ll: amount of motor steps to perform

[Description] The actuator will be moved in definite amount of steps.
A standard actuator has a step range of 500 steps and a stroke length of 12,7 mm.

[Example] Increase spring contact pressure by 166 motor steps: <RS>#1+<0d><166d>
For the movements "c" and "o" the actuator is moved with the indicated number of steps until a sensor is detected.

5.8.2 MOVE ACTUATOR AUTOMATICALLY

[Name] move actuator automatically

[Format] <RS> # „Actuator number“ „<direction>“ <quantityH><quantityL>

<actuator number>	number of actuator is a CHARACTER value
„c“ or „C“	moving out (close printer head, increase head contact pressure)
„o“ or „O“	moving in („O“ is the non-synchronized command)

[Description] The actuator will be moved in the specified amount of steps given by the sensor, who controls steps and stroke length. Actuator number in lower case letters cause actuator movement, only after all commands previously given have been processed. In capital letters , the actuator is moved as soon as possible. If immediate execution of the command is not possible, the parser tries for maximum 10 seconds to reinitiate the execution of the command. This is the case, if for example a key was pressed twice during the synchronization of the previous command. During this time no further commands will be accepted. If after terminated synchronization still no acceptance of the instruction is possible, the command will be discarded and a syntax error command ' ignored ' reported.

[Example] open Piano box: <RS>„#“„3“„O“<1d><120d>

5.8.3 ACTUATOR NUMBERS

0: Head Lift *(at GeBE-PrinterLab GPT-10000)*

The head lift command automatically moves the head attachment of the PrinterLab in print or standby position. The printer head sinks until the Head_UP sensor is closed (maximum <quantityH><quantityL>).
Afterwards additional motor steps up to defined amount in parameter 37,7 will be performed in order to safely move the head attachment to its end position. The head lift command only reacts with closed Head_UP sensor and will subsequently move the preset motorsteps <quantityH><quantityL>. If necessary, the motor step amount will be limited to the system parameter value „37,5“. A completely opened head can not be closed when a too low number of motor steps is selected. In this case the command must be repeated. An issued command for automatic closing will be non-effective, when the sensor is no more detecting 'HeadUp'. Same as a command for automatical opening while the sensor already detects 'HeadUp'. In those cases first issue a reverse command in order to receive a defined 'HeadUp' status.

1: Head Contact Pressure *(at GeBE-PrinterLab GPT-10000)*

At moving-in the amount of steps will be performed until shift of sensor AUX3. Afterwards the step amount defined in parameter 37,7 will be performed in order to safely move to zero-position. Once the zero-position is reached, the actuator only moves between position 7 – 492!

2: Head Alignment *(at GeBE-PrinterLab GPT-10000)*

3: Head Open

The <RS>#3c<n><m> command moves the actuator to home position. After each reset, the actuator checks the home position. If the sensor is closed nothing will happen. If not, <RS>#3c<n><m> will be performed. A warning will be issued if the sensor does not reach the home position.
The <RS>#3c<n><m> command moves the actuator 500 steps. Subsequently it is moved back to the home position.

5.8.4 READ ACTUATOR POSITION

[Name] Request actuator position

[Format]

ASCII	<RS>	“x“	“A“	<“n“ >
Hex	1Eh	78h	41h	<“n“ >
Decimal	30d	120d	65d	<“n“ >

n:= Number of Actuator Value is a character

- “0“ Head Lift
- “1“ Head contact pressure
- “2“ Head Alignment
- “3“ Opening printer mechanism

[Description] position counter of actuator with „actuator number“
Actuator totally moved-in gets position counter value = 0
Actuator totally moved-out gets position counter value = 500

HINT: At the moment only actuator 1 is supported.

[Example] <RS>xA1
Response example of module 37 Print_Aktuator, with message ID 0x0025 and message code 0x0000:
[aR:0x00000032#[0x00001242ms0x00250000: 0x00000000, 0x0001, 0x0000 #i__]#0xA229DD6F:aR]
Answer: actuator 1 reports position 0.

5.8.5 BARCODE SCANNER

This activates or deactivates the bar code scanner in a bar code scanner comprising system.
In module 48 the bar code scanner interface is configured.

[Name] Activate/Deactivate Barcode Scanner

[Format]

ASCII	<RS>	“b“	"n"
Hex	1Eh	62h	"n"
Decimal	30d	98d	"n"

[Name] The scanner is activated/deactivated.

- n:= "0" Deactivate Barcode Scanner
- n:= "1" Activate Barcode Scanner
- n:= "2" Pulse Barcode Scanner – 10 ms

5.8.5.1 RESPONSE OF A BAR CODE SCANNER

[aR:0x00000032#[0x000295A0ms0x00300000: 0x000024D1, 00030000567414 #w_+]#0xAF64B081:aR]
The bar code scanner detected the value 00030000567414 at the position 0x000024D1.
The bar code scanning was terminated with a linefeed (0d hex).

5.8.5.2 CONFIGURE BAR CODE SCANNER

The command "configure bar code scanner" sends commands to the bar code scanner.
Bar code scanner command {MC01WT8} is send to the printer as: <ESC>b{MC01WT8}.
After a correct transfer of the command, the bar code scanner gives following feedback:
[AR:0x0000002C#[0x0000158Dms0x00300000: 0x00000000, {MC01,OK}#w_+]#0x6022771D:AR].

5.8.6 MOTION SENSOR

As soon as the printer is switched on the motion sensor (64bit) permanently checks the paper movement in both directions in printer resolution units. The "read motion position" command enters the last covered distance in forward direction into the motion position counter and displays it. Now these contents can be compared to the included printer position counter.

ATTENTION: The maximum measuring length of the sensor is 380 mm. Latest after this length a reading operation must have been occured.

[Name]	Read Motion Position				
[Format]	ASCII	<RS>	“x“	“p“	“0“
	Hex	1Eh	78h	70h	30h
	Decimal	30d	120d	112d	48d
[Name]	Read Motion Position – Reset Motion Position Counter				
[Format]	ASCII	<RS>	“x“	“p“	“1“
	Hex	1Eh	78h	70h	30h
	Decimal	30d	120d	112d	48d

[Description] The sensor contents are added to the sensor position counter and displayed as 64bit value.
If the last command parameter is = 0, both values are read. If it is „1“, the motion position counter is additionally equalized with the printer position counter.

5.8.6.1 RESPONSE OF A MOTION SENSOR

[aR:0x00000027#[0x0011FFCEms0x00310000 *0x0000000A*: *0x0000000A* #i_+]#0x96808FE5:aR]

Printer position counter value: motion position counter value
The module 49 (31h) reports that 10 dot lines have been covered since the last poweron.
The configuration of the motion sensor is carried out via module 49 (31h).

5.9 READ ANALOG VALUES

[Name] Request analog values

[Format]						
	ASCII	<RS>	“x“	“a“	1d	n
	Hex	1Eh	78h	61h	01h	n
	Decimal	30d	120d	97d	01d	n

[Description] The command issues the analog values of all 8 available analog channels Data are output as 16 bit values per AD channel.

ATTENTION: The AD converter resolution is only 12 bit.Therefore the upper 4 bits are always 0.

The output assigns indices in ADC order up to maximum available data.
Not available ADC indices will be output as 0x_ _ _ _.

n:= ACD index	Description	Typical application
ADC Index 0:	all ADC values	(message with GMID 001D 0004)
ADC Index 1:	AUX 1	blackmark
ADC Index 2:	AUX 2	2. blackmark / head alignment
ADC Index 3:	AUX 3	n.c.
ADC Index 4:	AUX 4	n.c.
ADC Index 5:	AUX 5	motor temperature / pressure sensor
ADC Index 6:	AUX 6	n.c.
ADC Index 7:	AUX 7	additional Vp measurement / battery temperature
ADC Index 8:	Vp	
ADC Index 9:	Motor Temp	
ADC Index 10:	Head Temp	
ADC Index 11:	PaperEnd	
ADC Index 12:	NearPaperEnd	
ADC Index 13:	Vref	
ADC Index 14:	Battery Temp	
ADC Index 15:	Head Alignment	

[Example] read AUX 3 <RS>xa<1d><3d>

Response example of module 19 Print_Sensor, with message ID GMID 0x0013 and message code 0x000F: [aR:0x00000032#[0x00001242ms0x00130003:0x0926, #i +]#0xA229DD6F:aR]

5.9.1 REQUEST PRINTHEAD VOLTAGE

ADC channel 8: printhead voltage

The AD value is issued as 16 bit value with 12bit resolution. The first 4 bits are always 0.

One digit = 7.253 mV - for printers with 24V nominal voltage

One digit = 2.477 mV - for printers with 5-8V nominal voltage

ADC channel 13: reference voltage

The AD value is issued as 16 bit value with 12bit resolution. The first 4 bits are always 0.

One digit = 806 mV - in general

One digit = 732 mV - for GeBE PrinterLab 10000

5.9.2 REQUEST PRINTHEAD, MOTOR, AND BATTERY TEMPERATURE

n_{12bit} is the issued AD value

$$Tx = \frac{1}{\frac{1}{B}\ln\left[\frac{R_P}{R_{25}}\left(\frac{1}{\frac{4095}{n_{12bit}}-1}\right)\right]+\frac{1}{298°}} - 273°$$

ADC channel 10: Printhead temperature

B constantly: 3950 K R25: 30 kΩ at 25°C (77°F) Rp: 27 kΩ

[Example] The value 0x086B is issued. Thus printhead temperature is 25°C/77°F.

[Example] The value 0x0375 is issued. Thus printhead temperature is 45°C/140°F

ADC channel 14: Battery temperature

B constantly: 3950 K R25: 30 kΩ at 25°C (77°F) Rp: 27 kΩ

[Example] The value 0x08C5 is issued. Thus battery temperature is 25°C/77°F.

[Example] The value 0x0586 is issued. Thus battery temperature is 45°C/113°F

Hint: Please notice that the following sensors have a more or less reliable thermal interlinking, i.e. the issued temperature value matches only approximately that of the item that is to be measured. Please, request the corresponding coupling factor from GeBE.

ADC channel 9: Motor temperature

B constantly: 3950 K R25: 30 kΩ at 25°C (77°F) Rp: 27 kΩ

[Example] Polling the motor temperature gives the value :0x0175

[Example] GeBE Compact: coupling factor: 1,0 Thus the motor temperature is 90°C/194°F.

not yet implemented: storage of motor temperature offset in RefVal: 34

[Example] GeBE Piano: temperature offset 10°C (50°F) - thus the motor temperature is: ~ 100°C/212°F

[Example] Compact Plus GPT-4673: temperature offset 5°C (41°F) - thus the motor temperature is: ~ 95°C/203°F

As soon as 95°C/203°F real motor temperature are exceeded the printer reports a motor temperature error and stops the printing.

6 PRINTER INITIALIZATION

[Name]	Initialize Printer			
[Format]	ASCII	<RS>	<DLE>	n
	Hex	1Eh	10h	n
	Decimal	30d	16d	

[Description]

n:= 00d Restarting the system (PoR) while avoiding the bootloader **Standard RESET**
n:= 01d Restarting the system (PoR) with default bootloader properties
n:= 02d Restarting the system (PoR), with enforcing the bootloader to wait for commands (using time-out)
n:= 03d Restarting the system (PoR), with enforcing the bootloader to wait for commands (using time-out)
n:= 04d Shut-down the system (deactivate Vcc_enable) = Power OFF
n:= 05d Erasing all buffer contents of the renderer and of all print instructions
n:= 06d Erasing all tickets not yet started
n:= 07d Abortion of the ticket being printed
n:= 08d Abortion of data download asynchronous – acts directly
n:= 09d Abortion of data download synchronous to print data

n:= 255d Restarting the system (PoR), with enforcing the bootloader to wait for commands (without time-out)

6.1 CANCEL LAST INCOMPLETE TICKET AFTER PAPER END

When parameter 18,33 is set and the printer detects „PaperEnd“ during a motor movement just before a ticket print is completed, all present data of this ticket are accepted and cancelled. The data will be ignored as long as either a command „End of Ticket“ (parameter 18,35) or „End of Auto Flush“ is sent or until the preset timeout (parameter 18,34) is reached.

„EoT“ abortion condition only cancels the actual ticket, however the timeout condition deletes the complete print job. Afterwards the printer gets back to „idle“ mode without paper. A possible new ticket print job will be performed as soon as new paper is inserted.

The process is indicated in the first status word bit 15. This bit will be deactivated with inserting new paper. The „CancelTicket“ process will not be started by removing paper or switching-on „PaperEnd“ condition.

6.2 PARAMETER FOR CONTROLLING CANCEL TICKET

Parameter	Description
18,33	AutoFlush on PaperEnd
18,34	AutoFlush timeout [s]
18,35	EoT terminates AutoFlush

7 BATTERY CHARGER

7.1 SOFTWARE BASED CONTROL IN GENERAL

If the controller is used with NiHM batteries a charging connection backed by the processor is available. The performance of this charging connection and its survey is controlled by an appropriate configuration via system parameter. See list of printer parameters: Charge (44)

7.1.1 STARTING THE CHARGING PROCESS WITH FORMAT CHARGING

The battery charging device consists of a harware component and the corresponding charging control software in the µ-processor. On application of the charging voltage the hardware first controls the battery charge condition. If the battery is exhaustively discharged the charging cycle starts with a formatting charge to protect the battery (the load indicator is off). As soon as the battery voltage for fast charging is reached the controller takes control and activates fast charging.

7.1.2 DISPLAYING AND REQUESTING CHARGING STATUS

Different blinking intervals of the status LED indicate the charging status. When a status request command is made the serial interface issues corresponding confirmations. See: Error and status messages in printing operation.

7.2 CHARGING MANAGEMENT

A current limitation of the charging circuit does not exist in some GeBE controllers.
In the controllers this charging management is done by the processor. Some special controller variations have a charging current regulator "on board" which allows charging from a fixed voltage (power supply 8V - 28V) or from a car battery (12V or 24V).
Some GeBE controllers (Low cost mostly) have a NiMH charging circuit **without** own charging current limitation. The limitation is made with the proper GeBE battery charger.

ATTENTION:
Use the proper GeBE battery charger only or contact us if you need one!!
To recharge NiMH batteries never use a fixed voltage power supply without a defined charging current limitation. It is absolutely necessary to use a suitable GeBE power supply.

7.2.1 ENDING CRITERIA OF NiMH BATTERY RAPID CHARGING

As soon as one of the following conditions is fulfilled rapid charging is ended and switched to trickle charging:

- End of charging by timer
- Minus Delta V identification
- Maximum V identification
- Delta T identification
- Maximum T identification

7.2.2 CHARGING LIMITATION BY TIMER

When the charge rate is very low the required end of charging can not be securely accomplished by recognizing the cell voltage falldown (Minus Delta L). Some modern NiMH batteries allow a timer controlled charging with charge currents of up to 1/3 C. Therefore in case of a battery with 1200mAh a charge time limitation of about 3 - 4 hours is reasonable. The parameter 15 determines the battery voltage that has to be accomplished before the preset charging time starts and which will limit the charging in case other charging end criteria will not work. So you have a security perimeter that should be very tight.

Attention:

The timer controlled charging limitation is a secondary battery protection, because the battery charge condition is unknown at the beginning of charging and charging end criteria might not be recognized by the software.

7.2.3 MINUS DELTA U IDENTIFICATION

(Voltage inversion at end of charge)

When a NiMH battery is full (end of charge), the battery voltage drops though it is further furnished with charge current. This voltage drop is interpreted as end of charge. To be able to identify this properly, the charge current must not drop below 300 mA. A 12 bit A/D converter integrated in the µ-processor measures the charging voltage. Several measurements are averaged and added to a 16 bit value in order to eliminate deviations in individual measurements. The measuring interval per measurement is two seconds. With the number of thus gained values configured in parameter P3, the time delta t is determined, which is used to dedicate the voltage drop..

7.2.4 MAXIMUM BATTERY VOLTAGE

This criterion is found out by measuring the cell voltage and shall protect the battery. The value specified for the battery must be entered here.

7.2.5 DELTA T / DELTA t IDENTIFICATION

(Change in temperature)

If the battery temperature T rises in time t faster than the parameter indicates (charging energy is turned to heat only) the battery is estimated full. This works only when a preset battery voltage is exceeded.

7.2.6 MAXIMUM T IDENTIFICATION

(Maximum battery temperature)

This criterion is meant to protect the battery and should be set to the value specified by the battery.

With the charging voltage applied the charging is restarted with each reset. Moreover the charging restarts when the voltage falls below a defineable minimum battery voltage (typical for 1.2V / cell of the standard battery pack).

7.2.7 RESTART CHARGING WITH CONNECTED CHARGING DEVICE

The printer may work while the charging device is connected. Printing operation does not interfere with complete charging.

If the battery voltage falls under a minimum value that can be entered with a battery charge command, charging restarts.

7.3 CONFIGURATION BY SYSTEM PARAMETERS

44,1: = 1 battery type: NiMH charging (default). Other values are not allowed or do not switch on charging.

44,2: Timer set charging: 1 = 1/10h, i.e. 1: = 6 minutes. On the other hand 250: = 25 hours default is 40: = 4h. Charging time starts with the charging voltage applied, with received command, e.g. from the TINIT in case of a reset, or when the voltage value of the battery falls below a limit fixed in 44,13.

44,3: Amount of values used to make the difference of the Delta U voltage. Default is 44,3=60. The outcome of this is a measurement period of 60 x 2 seconds: = 2 min. The lower the charge current is, the higher this value ought to be.

44,4: The amount of identifications with Delta U identified as negative, until the end of charge is determined. Default is 44,4=1.

44,5: Indicates the voltage difference, from where Delta U is identified as valid minus Delta U. 1 LSbit = 0.671 mV. Default is 44,5=18 x 0.565V = 10mV.

44,6: Maximum voltage value. 1 LSbit = 39.782 mV. Default is 44,6=169. Thus the maximum voltage value is 169 x 39.782mV = 6.72V. For 4 cells this comes up to a voltage of 1.68V/cell. Charging is ended when this maximum voltage value is exceeded successively as often as indicated in 44,7.

44,7: Repeat counter for the exceeding of the maximum voltage specified in 44.6. The maximum voltage has to exceed the value indicated in 44,6 44,7 times successively in order to identify the end of charging and to stop charging. Default is p7=1.

44,8: P8 matches with 44,3 for measurement of temperature changes. P8 is the number of values measured several times within 2 seconds to find out temperature changes. Default is 44,8 = 60. I.e. a Delta T identification measurement time of 60 x 2 seconds = 2 minutes.

44,9: corresponds with 44,4 for the measurement of temperature changes. 44,9 is the number of Delta T identifications (with 44,8=60 within 2 minutes) until Delta T is identified as valid. Default is 44,9=1.

44,10: matches with 44,5 for Delta temperature difference. a LSbit is about 0,01°C/32°F. Default is 44,10=64. That means that the recording interval is identified as a change in temperature at the end of charging, if the difference in temperature preset in 44,10, which is 0,64°C/33°F (44,10:=64) is succeeded within the amount of time set in 44,8 (8:=60, e.g.2 minutes).

44,11: Maximum temperature value. Matches 44,6 for temperature. The temperature is measured with an NTC resistor welded into the battery pack. P11 indicates the maximum temperature value. If this value is exceeded 44,12 times while charging the batteries, rapid charging is stopped. An estimation formula for a 6.8K NTC is: p11:=(60°C /140°F - Tmax°C) / 0.6°C. Default setting is 44,11:=25. This is Tmax about 45°C/113°F

Attention: high temperatures result in low and therefore inexact measured values.

44,12: Corresponds with 44,7 for temperature. 44,12 is the specification for the repeat counter for the acceptance of the temperature exceeding criteria. Default is 44,12=1.

44,13: Determines the voltage level for charging restart with decreasing voltage. In theory 1 LSbit is: 39.782 mV. Default is p13:=133. This is 133 x 39.782 mV = 5.29V. With 4 cells this means 1.2V/cell.

44,14: Trickle charge ratio. After the end of rapid charging this value determines the average value of the trickle charge in case the charger remains switched on. 1 LSbit equals 2 seconds runtime with 512 seconds off time. Default is 44,14:=10. That means that in a period of 512 seconds the charge current with supposed 300 mA is on for 10 x 2 seconds = 20 seconds. 20 seconds means 3.9% of 512 seconds. Thus charging is effected with 3.9% corresponding to a medium continuous current of abut 12 mA (trickle charge).

44,15: Indicates the minimum battery voltage below which a charge time limitation (switch off after a preset charge time) and the Minus Delta U identification are not realized. Default is 44,15=140. With 4 cells this means a value of 140 x 39.782mV = 5.57V.

8 PRINT POWER MANAGEMENT IN ECO PRINT MODE

Control print current Module 18: Print, Parameters: 18,7 and 18,8

In ECO mode parameters preset how much peak current the printer can accept. Accordingly the printer adjusts its print speed dynamically. Peak power and dynamics are adjustable. The current peaks during printing can be adjusted by command. The maximum value results from the maximum number of simultaneously energized dots. The limited = minimum value results from the simultaneous energization of at least 8 pixels. This means if many pixels are energized sequentially in a horizontal line, the printing speed reduces severely. A high power voltage Vp also results in high pixel currents. This is because the current limitation is realized through a limitation of the number of simultaneously energized pixels and not through a current measurement. The command adjusts the maximum allowable peak current indirectly through the number of pixels, the known dot resistance and the maximum possible power supply voltage. The maximum number of dots to be printed is controlled so that the predetermined peak current is not exceeded. This allows the power consumption of the printer to adapt to the possible current output of a higher ranking electrical system. In addition, a further parameter influences the pressure dynamics and thus the print quality.

Parameter 18,8 / adjust printer current in ECO print mode

The current peaks are determined during the printing by the number of simultaneously energized dots. Parameter 18.8 multiplied by 8, specifies how many dots per process may be activated simultaneously. The value range from 1- 255 permits the simultaneous activations of a minimum of 8 to a maximum of 2040 dots. See Hardware Manual. A high number of dots increases the print speed but also the print peak current.

Once the number of black pixels to be printed in a line segment reaches a certain value, the next print line is filled with 0 and then the line is activated. In the next cycle , the already printed pixels are filled with 0 and then other pixels (maximum n) will be activated etc.

The maximum current I depends on the power supply V_p and dot resistance R_{dot}:

$I = (V_p \text{ x number of pixels} / R_{dot}) + I_{Vcc} + I_{motor}$

for example: $V_p = 5V$, $R_{dot} = 123$ Ohm, $I_{motor} = 0.5A$, $I_{Vcc} = 0$

with n = 8 (i.e. 64 Pixel) => $I_{peak} = 5 \text{ x } 64 / 123 + 0.5 = 3.1A$

Parameter 18,7 / adjust print dynamics (virtual segment size)

Minimum is 1, maximum is the printing width in mm, e.g. 80 for GPT 4633. This parameter determines the p rint speed dynamics: High dynamics means the printer prints each line as fast as permitted by the maximum specified current. Therefore a blank line is printed faster than a full line. With suppressed dynamics each line is printed as if it were completely black. Constant motor rotation = good print quality . The parameter sets how many bytes are printed simultaneously (the same when they contain only 0, i.e. no dots will be printed). When selecting „1“ the printer as a principle splits the line in 8 pixel wide segments , i.e. segments to be activated. When m includes all bytes being arranged in one line (if 8 pixels/mm, then the total effective print width in mm), the printing line can be activated in a single operation.

9 PRINT POWER MANAGEMENT IN CONSTANT SPEED MODE

In test mode, the printer control is carried out via the fixed specification of cycle time of a print line and the indication of the heating time .

Parameter:

18,32 fixed S.L.T. [µs], 0: variable S.L.T.
Parameter 18,32 defines the cycle time of a dotline and thus the printing speed. TSLT is active as soon as a value is > 0. In this case, the indication of the printing speed in parameter 18,3 is ineffective. Once this value is given, the printer accelerates according to its acceleration ramp up to this speed and holds it . Acceleration and reaching the set speed is displayed in the status byte 8.

ATTENTION: The printer stops immediately if necessary printing data for the dotline are not available on time!

Example: speed 250 mm/s at a resolution of 8 dot/mm (200 dpi) -> Tcyl = 1/(250*8) = 500 µs

18,29 fixed strobes on time [µs], 0: variable strobes on time
Parameter 18,29 defines the pressure energy (density) of the printout and is reported in microseconds. Typically density 50% does not exceed Tcyl . If the parameter 18,29 value is > 0, the parameter 18,6 density becomes ineffective.

18,30 minimum strobes off time [µs]
Parameter 18,30 defines the minimum switch-off time of the dots within one cycle, in order not to overload the heating elements. The parameter is specified in microseconds.

NOTE: Anti Jam

The printer can, if mechanically possible, be programmed with an anti-jam function. If a reflex sensor with parameter 18,44 is selected, the printer stops immediately as soon as the selected reflex sensor "closes". The printer will continue to operate as soon as the sensor re-opens. This function is optional available for e.g. CompactPlus printer model.

10 TICKET AND LABEL PRINT

The printer controller has several commands that generally allow control of labels and tickets. Various types of control can be distinguished:

- Control according to paper length
- Control by blackmark (tickets and labels)

10.1 CONTROL ACCORDING TO PAPER LENGTH

The paper length of the receipt is entered via parameters. The paper length is determined by the feed of the printer mechanism. The paper length control does not use a paper positioning sensor. In the narrower sense, this is not a ticket control. This means that the starting point of the tickets can be set anywhere on the paper. No positioned preprint on the paper roll is required for synchronization. If less paper than the length specified by the command is printed, the final FormFeed or "End of Ticket" command triggers the final print and the paper feed continues until the page length counter indicates that the length has been reached.
If, however, so much paper is printed that the set ticket length is exceeded, the paper length counter automatically remains at the "max." until a FormFeed occurs. Only then the given paper length of the next ticket will be measured by the counter.

Configuration:

Parameter 18,12 = true	switch on label / ticket printing
Parameter 18,13 = 0	disable blackmark detection
Parameter 18,30	paper length of the label / ticket in [mm]

10.2 CONTROL ACCORDING TO MARKS (TICKETS AND LABELS)

Marks are detected with a sensor (light barrier) on the paper. This is controlled e.g. with preprinted infrared light absorbing marks or with holes of a certain size cut into the paper. This makes it possible to use preprinted paper and to set the printer always to the correct position on the preprinted forms to print in the correct position. A parameter can be preset to determine the length or a mark, a gap or a hole to be recognized as such. Different sensors can be used to measure the position. The choice of the correct sensor for the form triggering is made with **parameter 18,14**. Parameters can also be used to combine tickets/marks in segment groups. If e.g. the **parameter 18,17** is set to 3, 3 tickets are combined in one and only every third mark is addressed.

10.3 CONTROL MARKS

The control marks must be printed with an infrared absorbing color in the range 910-950 nm with an optical density of minimum 1.0. The position of the control mark is not the printing position of the printhead.

10.4 TICKET DEFINITION

In order to define a ticket for the printer, all physical characteristics of the ticket and its desired behavior have to be communicated to the printer. These settings are identical for all emulations.

10.4.1 CURSOR-URSPRUNG DES TICKETS FESTLEGEN

Parameter 14,3

C32: define cursor origin of the ticket (in which corner the cursor is located). By default, at a glance at the paper and printer, the cursor origin is "left front".

Parameter 20,3

= 1: landscape mode. In FGL emulation the cursor origin is "right back"!

= 3: this parameters allows as well the print image or the ticket to be turned by 180°.
This parameter is not active with line modes.

0: = portrait 0° (standard A8)
1: = landscape 90° (standard C32)
2: = portrait 180° (data mode)
3: = 270° (standard FGL)

10.4.2 TICKET MODE ON/OFF

Parameter 18,12

In ticket mode a ticket is always concluded with a FORM FEED or EoT. The printer finishes the ticket by searching a black mark or completing the ticket to a specified length.
on = 1, off = 0

10.4.3 BLACKMARK DETECTION LENGTH

Parameter 18,13

Blackmark detection length in dotlines: 0 = blackmark detection off.
Length for the detection of a mark in dotlines. This parameter is used to define how many dotlines must be recognized to accept a blackmark as valid. Preset a low value, will quickly detect the blackmark. However here also other elements, such as pre-printed text may be detected as blackmark. With default setting = 10 preprinted elements in the range of <1 mm are ignored. A change of the value does not affect, for example, the ToF position. The value is automatically compensated. If the setting is 0, the whole ticket length will be imprinted.

10.4.4 DEFINE ACTIVE BLACKMARK SENSOR

Parameter 18,14

0 = Paper End Sensor

1 = AUX1 Sensor

2 = AUX2 ……

default sensor in C32 mode = AUX 1
default sensor in FGL mode = AUX 2

If 0 = Paper end sensor is selected, the sensor must not halt on a blackmark. The paper end delay of parameter 18,27 must be entered sufficiently larger than the size of the blackmark.

10.4.5 DISTANCE ToF TO BLACKMARK

Parameter 18,16

This parameter determines how far the printer continuously prints after blackmark detection before the ticket terminates after a form feed. Thus, the blackmark does not necessarily have to be located at the beginning of the ticket. With this parameter the blackmark can be "shifted".

10.4.6 NUMBER OF TICKET SEGMENTS

Parameter 18,17

This parameter specifies how many segments (blackmarks) a ticket contains and ticket groups can be generated. Normally data exceeding the physical ticket will be cut off (clipped). In the setting: 0 ticket data will not be clipped.

=0: each blackmark is used but can be overwritten.
=1: each blackmark is used, too long printouts are cut
=2: every 2nd blackmark ...

10.4.7 DETERMINE TICKET LENGTH

Parameter 18,39

Ticket length is stated in 1/10 mm and should be set to approximately 1.5 to 2 times the label length.
Ticket length can span several blackmarks (see parameter 18,17).

10.4.8 BLACKMARK SEARCH LENGTH

Parameter 18,40

Indicates how long (minutes) the printer is supposed to search for the blackmark after end of ticket.
The parameter is set to "0" when the ticket mode is used according to length without blackmark.

10.4.8.1 MINIMUM BLACKMARK SEARCH LENGTH

Note: until a blackmark is accepted

Parameter 18,41

If the printer sensor is placed on a mark after a ToF, the next ticket would immediately detect a new blackmark situation. This parameter defines the number of mm that have to be searched for a blackmark before accepting it.

10.4.9 AUTOLOAD

10.4.9.1 AUTOLOAD ON/OFF

Parameter 18,21

=1: autoload enabled
=0: disabled

10.4.9.2 MAXIMUM AUTOLOAD SPEED

Parameter 18,22

input in mm/s

10.4.9.3 AUTOLOAD FORM FEED LENGTH

Parameter 18,23

input in mm/s. Maximum 250 mm possible.

10.4.9.4 AUTOLOAD STOP CONDITION

Parameter 18,24

0: autoload ends as soon as ToF is reached
1: autoload ends as soon as sensor AUX1 is detected
2: AUX2 ..... AUX4 is typically exit sensor

10.4.9.5 RETRACTING LENGTH

Parameter 18,26 Retracting length after detection of autoload condition.
Input in mm. With this entry, the ticket will be retracted once the autoload condition has been reached. Afterwards the status ToF will be set. This is useful if the black mark can only be found on the next ticket. In this case the printer skips the first ticket at autoload and continues to the next one. Then the printer retracts the paper by the input length in mm stated in parameter 18.26.

10.4.10 DELAY OF PAPER END DETECTION

Parameter 18,27

This parameter is used to define how many paper end lines must be recognized to initiate a paper end condition. A paper end error will be cleared after paper is detected. For this purpose, no line feed is necessary. When using blackmark control by paper end sensor, too high a value leads to a delay in paper end message.

10.4.11 DEFINE ToF CUTTER DISTANCE

Parameter 22,3

This parameter is used to define how many dotlines ex ToF (Top of Form) position will be transported forward and back before an “End of Ticket Cut". The backlash is automatically compensated. When setting the parameter to the distance value printhead – cutter, the ticket will be cut off at the printing position and transported back to the printing position.

When using other cutters, another parameter "x,3" may be appropriate.
Driving back at EoT must be separately permitted via system parameter [24,5].

Finally, the printer must be reset to activate the parameters.

Example:

<1Eh>[4:0,0] *(delete user settings. HINT: modules can be deleted seperately)*
<1Eh>[1:20,3:1]
<1Eh>[1:18,12:1]
<1Eh>[1:18,13:16]
<1Eh>[1:18,14:1]
<1Eh>[1:18,16:32]
<1Eh>[1:18,17:1]
<1Eh>[1:18,18:1200]
<1Eh>[1:18,19:40]
<1Eh>[1:18,24:0]
<1Eh>[1:18,26:0]
<1Eh>[1:18,27:122]
<1Eh>[1:22,3:94]
<1Eh><10h><00h> *(restart printer)*

10.5 LABEL PRINTING WITH THE OPERATION OF A WINDER

Depending on the version, a paper winder can be connected to the GCT-4663 controller.
The winder serves to wind up the backing paper and to maintain the tension needed to peel off the labels.
If a printout is to be triggered, the winder must first tension the backing paper, so that a following label can be detached in any case. This lead time is set with **parameter 23,10** in [ms]. If the paper is still taut as a result of a previous print, since the winder is still in operation, this time is not executed. During printing, the winder always has to run a little "faster" than the printer so that the tension remains. The speed differences are compensated by a friction clutch, aet by **parameters 23,3 and 23,4**. The **parameters 23,5 – 23,7** need not be changed.
Parameter 23,2 defines the distance from the peeler edge to the start of the next label "ToF".
The peeling process can be monitored by the "picking" sensor. To do this, the sensor must be selected via **parameter 24,9**.
If the **parameter 24,5** is activated ("T"), the printer places the label after the peeler edge and immediately pulls the following label back to the ToF position.
If **parameter 24,5** is deactivated ("F"), the printer does not pull back the paper. The locked backfeed must be set at the end of the ticket data. This is then executed as soon as the "picking" sensor is released.
Since the printer can not pull the backing paper back against the winder, the winder must run backwards at the right time to allow this. The winder first runs backwards the set time in [ms] (**parameter 23,11**) to run ahead the printer. Afterwards, the winder runs backwards in parallel mode to the printer for the set time in **parameter 23,12** in [ms] in order not to "block" the printer.
After the printer is back to the ToF position, the winder runs forward in [ms] by the time set in **parameter 23,13** to wind up the backing paper. Thereafter, the tension is maintained for the time in [ms] set in **parameter 23,9**.

10.5.1 PARAMETER SETTINGS

Parameter 23,1 feed length ToF to picking sensor [decimillimeter]
This parameter sets the mechanical distance from the first print position (ToF) to the „picking“ sensor. If the sensor does not detect the label within this length, an error condition is generated. The value should therefore be chosen slightly larger than the physical distance (default: 500). The „picking“ sensor is selected with parameter 24,9.

Parameter 23,2 feed length ToF to peeler edge [decimillimeter]
This parameter sets the mechanical distance from the first print position (ToF) to the peeler edge. The value should correspond to the mechanical distance. If the value is too small, the printer will not cast off the label completely behind the peeler edge. If the value is too large, the follow-up label can "retrieve" the current once again.
If this value is 0, then no reversing operation is carried out. Otherwise, the reversing operation is performed by this value.

Parameter 23,3 winder motor start speed [PPS]
Start speed of the winder motor (default 300)

Parameter 23,4 winder motor final speed [PPS]
Final speed of the winder motor (default 800)

Parameter 23,5 number of winder motor speedup stages
Speedup stages of the winder motor (default 4)

Parameter 23,6 microstep resolution of the winder motor
Microstep resolution of the winder motor, range 1...8, (default 8)

Parameter 23,7 setting of winder motor current
Setting the winder motor current RMS (0=0,4A …… 31=1,5A) (default 31)

10.5.1.1 SETTING PARAMETERS FOR TIMING SEQUENCE

Parameter 23,9 follow-up time fwd [ms]
Halt of the printer motor after moving forward

Parameter 23,10 lead time fwd [ms]
Start of the printer motor resp. change of direction (new direction of the printer motor: forwards)

Parameter 23,11 lead time bwd [ms]
Start of the printer motor resp. change of direction (new direction of the printer motor: backwards)

Parameter 23,12 continuation time bwd [ms]
Stop the printer motor after backward movement

Parameter 23,13 winding time fwd [ms]
End of the lead time bwd

Parameter 23,14 winding direction
= 0: forwards
= 1: backwards

printer mechanism		fwd		bwd			
winder motor	fwd		bwd	bwd		fwd	fwd
SysP	[23,10] lead time fwd		[23,11] lead time bwd	[23,12] Fortsetzungszeit bwd		[23,13] winding time fwd	[23,9] follow-up time fwd

10.5.1.2 STATUS MESSAGES

The following status flags are used primarily to analyze the processing status of a ticket, (note: bit indexes count from 0, bit numbers from 1):

Name	Flag position GeBE_C32	Flag position GeBE_A8
PICKING_POS	current bit no. 18	3. byte, bit index 1
EOT_JAM	current bit no. 19	3. byte, bit index 2
TICKET_PICKED	current bit no. 20	-
PICKING_SENSOR	current bit no. 21	-

The flag PICKING_SENSOR indicates whether the picking sensor detects paper at the current position (bit set if paper is present). Firmware-internally, this status is evaluated in order to detect whether, on the one hand, the peeling process was successful and, on the other, whether the ticket was taken.
As soon as the status PICKING_SENSOR is set within the feed length specified in system parameter [23,1], the peeling process is considered successful; otherwise the error status EOT_JAM is set when the feed length is exceeded.
In the case of a successful peeling process, the flag PICKING_POS indicates that the ticket is located at the removal position specified in system parameter [23,2] and that it can be removed. While waiting for the removal, the follow-up ticket is pulled back to the top-of-form position in parallel and the winder tightens the paper.
As soon as the removal sensor is released again while waiting for the removal, the flag PICKING_SENSOR is withdrawn and the flag TICKET_PICKED is set. As soon as the ticket print is terminated, the set flag is recanted again.
The printing of interim possibly received data for the follow-up ticket begins only after the current ticket is taken-off and the paper is tautened.
If the peeling process was unsuccessful, the feeding back and forth between top-of-form position and removal position is eliminated and the winder does not have to re-tauten the paper. The flag EOT_JAM is recanted as soon as the ticket print is terminated and thus the printing of the follow-up ticket is immediately released.

10.5.2 MONITORING OF TICKET OUTPUT OR JAM DETECTION

The exit or theading sensor are used to monitor the ticket printout or a paper jam in the printer.

If the exit sensor is not detected from Top of Form (ToF) within a specified length, the printout is stopped and a threading error is issued.

10.5.2.1 PARAMETER SETTINGS

Parameter 18,50 headDn terminates AutoFlush
Cancel the AutoFlush threading error by closing the head. If the system parameter is enabled, AutoFlush will stop when the printhead down is detected.

Parameter 18,51 threading sensor index
= 0: NONE
= 1: AUX1
= 2: AUX2
= 3: AUX3
= 4: AUX4
= 5: AUX5
= 6: AUX6
= 7: AUX7
Activate monitoring of ticket output via specified sensor. No monitoring at value 0.

Parameter 18,52 feed length ToF to threading sensor [decimillimeter]
The correct ticket issue is implied, when the specified sensor has been reported occupied within the specified feed length (status = 0).

Parameters 18,53 autoflush on feed jam
If the parameter is activated, an autoflush is triggered when an error status is set.

10.5.2.2 STATUS MESSAGES

Setting the error status Print_STATUS_FEEDJAM (display via GeBE_A8 status mask 0xD4 or GeBE_C32 status flag no. 19) with theading error (feed length exceeded without interim sensor detection). Undoing Print_STATUS_FEEDJAM when opening the printhead.

New command for asynchronous resetting of FeedJam status flags: Syntax: <RS> dk <D4h>

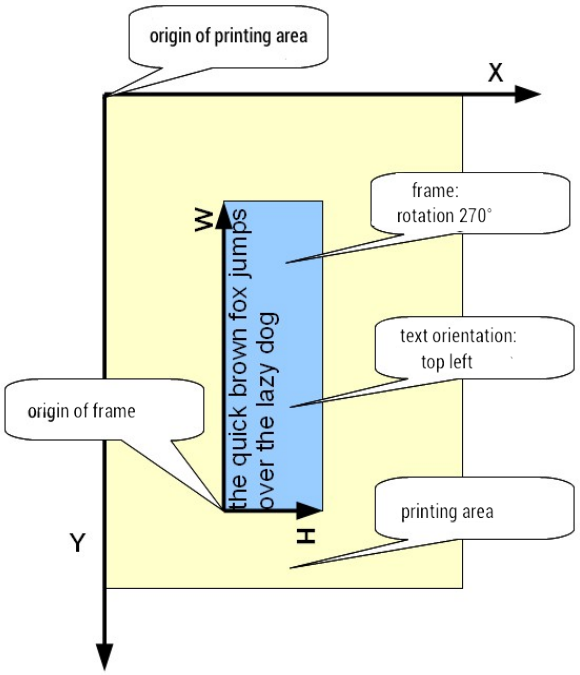
11 FORMS MODE GeBE C32 STANDARD COMMAND SET

11.1 GeBE_C32 FORMS AND FRAMES

In GeBE_C32 forms mode, the print image is first completely defined before the print is started with the command [GeBE # EoT]. The print data are divided in rectangular areas, called frames, and they are placed in the printing area and rotated, if required.

Position and length data are named in this document as follows:

- X or Y when they relate to the coordinate system of the printing area
- W and H if they relate to the coordinate system of the frame
  - Note for interested: Both cases are “left” systems.



When working in portrait mode, the front edge of the paper forms the X-axis of the printing area (see system parameter 14,3). Frames are swapped in the order of their definition. They can be opaque or transparent. An opaque frame which is defined after another overlapping frame, will partly or even completely hide this.

In addition, you can specify that a frame should be displayed in negative print (white to black).

Depending on the type of contents other attributes may be specified, e.g. in case of text frames the alignment of the text within the frame.

Each frame can be turned in 90° steps compared to the basic orientation of the ticket. The rotation is always performed around the current cursor position, which determines the so called “origin of the frame” defined as left upper corner (referring to the contents of the frame, not referring to the printing area!).

In many cases, the dimensions of the frames can automatically be determined. Thus the explicit specification of frame width and height or alternatively the determination of its lower right corner is not necessary. When setting frame dimensions, the data display exceeding the frame area will optionally be suppressed (clipping). The same applies to those areas of a frame which exceed the printing area.

11.2 FRAME CLASSES

Frames are classified according to the nature of their contents. In a frame may be respectively presented:

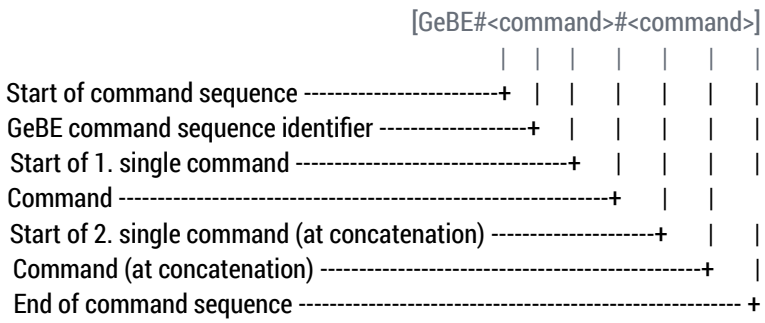
- graphics
- geometric primitive
- text (also multi-line)
- the value of a variable (i.e. ticket counter)
- Barcode

Mixture of different content types (text and graphics) in a frame is not possible. Barcodes with plain text are displayed in separate frames that are automatically opened and positioned in a suitable location.

11.3 GeBE_C32 COMMAND SEQUENCES

The GeBE_C32 standard command set embedds so-called command sequences, which consist of one or more individual commands. The first command in a command sequence is the main command of the sequence and determines the context for the interpretation of the (optional) sub-commands. Subcommands are only effective for the duration of the main command. For example, when defining the “font format” as a subcommand to the “font selection” command, it will only be valid until the next font selection. Subcommands should only be applied by advanced users.
It is generally possible to define a ticket by sequential command sequences with just one (main) command.
Selected applications of subcommands are described in the command overview. A GeBE_C32 command sequence is enclosed in square brackets and consists entirely of printable ASCII characters. The opening bracket is followed by the GeBE-command sequence identifier (text 'GeBE') and at least 1 command.

11.3.1 STRUCTURE OF A COMMAND SEQUENCE



Example:
Font 8 in 16-dot font with double width [GeBE # F8:16 # ScT 2:1]

To increase the readability for the user, space characters can be inserted but will be ignored. Within identifiers or trailing parameters no space characters are allowed.

Space characters outside the square brackets are interpreted as text characters to be printed!

The beginning of a command is indicated by a preceding command separator '#'. The command itself consists of a command identifier consisting of one or more letters and one or more trailing parameters depending on the type of the command sequence. A colon ':' is used to separate two trailing parameters belonging to the same command.

Command sequences have to be completely imported into the parser input buffer, therefore the following system parameters may be adjusted:

[Parser NEWDATABUFFERSIZE]
[Parser NEWDATABUFFERTHRESHOLDXOFF]

However, it is not necessary that ticket command sequences are placed together in the parser input buffer. The amount of sequences belonging to a ticket is limited by the size of the renderers buffer.

11.3.2 DATA FORMAT OF TRAILING PARAMETERS

Depending on the command, trailing parameters are defined as logical values (booleans), integers, rational numbers (fractions) or as text:

Kind of trailing parameter	Value format
Logical value (boolean)	Single ASCII character: 'N', 'n', 'Y', 'y', 'J', 't', 'T', 'j', 'f', 'F', '0', or '1'
Text	Alphanumeric character (character set and number of characters depending on command)
Integer	optionally as • decimal notation (default), e.g. 255 • hexa-decimal notation, format: 0x..., i.e. 0xFF • binary notation, format 0b..., i.e. 0b11111111
Rational number (fraction)	optionally as • common fraction of two integers [1, 255] • with divider symbol '/' (<numerator>/<denominator>), i.e. 7/4 • decimal fraction with decimal divider symbol '.' or ',' • (decimal notation only), e.g. 1.75 or 1,75 The indication of the numerator is sufficed for integer values, means the input can be performed as an integer value, e.g. 2,0 = 2/1 = 2
ASCII hex byte	ASCII character pair, e.g. 00 or 7F

Integer and rational numbers can have algebraic signs for some commands, which will be prefixed to the command value as '-' or '+'. The permitted value range depends on command type.

12 COMMAND OVERVIEW

If there is no explicit indication in the commands, the parameters are non-negative integers. Examples are generally indicated as command identifier specified with trailing parameters. A complete sequence is only specified if the effect of sub-commands is to be explained in the context.

12.1 SUPERIOR COMMANDS FOR FRAME SPECIFICATION

To open a new frame, the cursor position is set to the origin of the new frame (upper left corner). Frames have to be defined in one piece. Therefore opening a new frame concludes the previous one. Subsequent appending of data is not provided.
Once the print data are no longer compatible with the opened frame (frame class changes), a new frame will open automatically. In this case, the cursor position is automatically updated. However GeBE reserves the right to change the algorithm for cursor position update without notice. Therefore, it is recommended to specifically set the cursor position with each frame change. This is also documented for further commands closing the actual frame.
In addition to the original coordinates further details are required for the positioning of a frame in the printing area: Orientation as rotation against the printing area and if necessary, additionally the width and the height of the frame (based on the contents) will be specified. Alternatively, the postion of the origin's opposite corner can be indicated. The system then calculates therefrom the frame orientation as a multiple rotation 90°, as well as its width and height.

12.1.1 SET CURSOR POSITION (open frame)

The position specifications refer to the coordinate system of the printing area. The cursor position is the origin of the frame, i.e. its upper left corner, which is the center for possible rotations.

[Command identifier]	FPosTL (short therms Fpos and FP)	
[1st parameter]	X coordinate (algebraic signed integer out of [-32767, 32767])	
[2nd parameter]	Y coordinate (algebraic signed integer out of [-32767, 32767])	
[Example]	Positioning upper left corner of the printing area:	[GeBE#FPosTL 0:0]

12.1.2 FRAME ATTRIBUTE OPAQUE

[Command identifier]	FAttrO (short term FAO)
[Parameter]	Boolean: T = opaque frame, F = transparent frame

12.1.3 FRAME ATTRIBUTE NEGATIVE PRINT

[Command identifier]	FAttrNeg (short terms FAttrN and FAN)
[Parameter]	Boolean: T = negative print (white in black), F = black on white

12.1.4 FRAME ATTRIBUTE ORIENTATION (Rotation)

The rotation will be performed around the current cursor position and relates to the basic orientation of the printing area.

In case the cursor position equals the origin of the printing area, after the rotation the frame will be located almost entirely outside of the printing area and will therefore be clipped!

[Command identifier]	FAttrRot (short term FAR)	
[Parameter]	0 or multiple of 90 (negative algebraic sign allowed)	
[Examples]	270° rotation clockwise of a frame FAttrRot270	
	equal to a 90° rotation counter-clockwise	[GeBE#FAR -90]

12.1.5 SPECIFY TEXT FRAME WIDTH AND HEIGHT

The length data refer to the coordinate system of the text frame. This means, that the position of the second corner depends on the cursor position and also on the frame orientation (rotation against printing area).

[Command identifier]	FWH	
[1. Parameter]	Width: integer max. 32767	
[2. Parameter]	Height: integer max. 32767	
[Example]	Frame size 70 dots wide and 30 dots high.	[GeBE#FWH 70:30]

12.2 FORMATTING COMMANDS FOR TEXT CHARACTERS

Commands for formatting text characters are effective for all subsequent text characters. Within the frames or even in a row formatting may be changed several times.

12.2.1 FONT SELECTION

In case the selected font does not exist in the system, the actual selected font remains active.

[Command identifier]	F	
[Parameter]	Sequential font number	
[Example]	Select font number 2:	[GeBE#F2]
	Select font number 8 with size 16 dots:	[GeBE#F8:16]

12.2.2 MANIPULATE THE FONT SIZE

Width and height of each character is defined in the font. If a different character size is required, without changing the font, 2 different methods are available:

- Setting the font size, that is, specifying the desired font height:
- this method ensures that all characters regardless of the font have the same height, the width is automatically scaled by the system so that the sign is not distorted. This setting is effective for each specified font.
- Scaling of the character, that is, specifying a factor for width and height:
- This method makes it possible to resize width and height independent of each other. The character will be distorted. This setting works for all fonts globally.

Both methods affect all subsequent text characters. With both methods it is ensured that the base lines of all characters of a text line are positioned on a uniform height. The position of the baseline in the total height of a text character is also stored in the font. If both methods are used in combination, the scales act multiplicatively. In this way it is e.g. possible to print a text with double-width characters, and to ensure uniform font height regardless of the font. For reasons of clarity, the scaling factor for the height should be set to 1, so that the height is equal to the font size.

An isolated ScT command globally changes the scaling for the printer. A chained command is revoked after EoT.

Technical note:

It is necessary that the resulting scaling can be displayed internally as a fraction with numerator and denominator of [1, 255]. Scaling factors can thus be specified in 255th part increments of 1/255 up to maximum 255. If necessary, the system rounds suitably or limits the value, in order to meet the threshold.

12.2.2.1 SCALE TEXT CHARACTERS

[Command identifier] ScalT (short form ScT)
[1. Parameter] Scale width: rational number
2. Parameter: Scale height (optional): rational number
If no value is indicated, width and heigt value will be the same.

[Examples]	Text in double width and half height	[GeBE#ScT2:1/2]
	Font 8 in size Pt16 with double width	[GeBE #F8:16 #ScT 2:1]
	Text in 1.5-fold width and height	[GeBE#ScT 1.5]

12.2.2.2 OUTPUT OF SPECIAL CHARACTERS

[Command identifier] ChrF
[Parameter] Font related character code
The code to choose depends on each active font!

[Examples]	Euro Zeichen in Codepage 858	[GeBE#ChrF:0xD5]
	Euro Zeichen in Standard GeBE Font ausgeben	[GeBE#ChrF:0x80] or [GeBE#ChrF:128]
	Euro Zeichen in Unicode ausgeben	*[GeBE#ChrF:0x20AC]*

12.2.2.3 LINE FEED

[Command identifier] CRLF
[Parameter]

12.2.2.4 CHANGING TRACKING (letter spacing) AND LEADING (interline spacing)

Contrary to the requirements of the font, tracking and leading may be increased (positive values) or decreased (negative values), e.g. for ligatures. The number of dots is stated which indicate by how much the spacing has to be changed based on the unscaled font. This means, that scaling a text also changes the spacing accordingly. When enlarging a text, additional white space is inserted towards the right neighbor character or the succession line. In case of a reduction characters or rows are moved closer together. This may lead to an overlap of characters. Depending on the setting of the frame attributes the left character sign or the top line are partly obscured.

Changing the tracking
[Command identifier] SpacingW (short form SpW)
[Parameter] Changing the tracking: integer

Changing the leading
[Command identifier] SpacingH (short form SpH)
[Parameter] Changing the leading: integer

12.2.3 FONT STYLE SELECTION

The default font style can be preset via the system parameters. For the formatting of characters within a ticket, the following commands are available.

12.2.3.1 SUPPRESSING THE INFLUENCE OF FONT STYLE TO CHARACTER THICKNESS

[Command identifier] CharstyleC (short form CsC)
[Parameter] Boolean: T = thickness constant, F = thickness depending on font style

If the thickness remains unaffected, the tracking perceived by the eye may be influenced.

12.2.3.2 UNDERLINING

[Command identifier] CharstyleU (short form CsU)
[Parameter] Boolean: T = underlined, F = not underlined

12.2.3.3 VERTICAL ARRANGEMENT (SUBSCRIPT/SUPERSCRIPT)

[Command identifier] CharstyleS (short form CsS)
[Parameter] Code letter L, l, N, n or H resp. h:

- L or l: subscript (e.g. for indices)
- N or n: in line
- H or h: superscript (e.g. for exponential notation)

A subscript or superscript text character will be scaled with the factor ½ relative to the default text. If it is determined that the set width shall remain unaffected, the width of a standard text character will be reserved regardless of this.

12.2.3.4 INVERSE PRINT FOR INDIVIDUAL CHARACTERS

[Command identifier] CharstyleN (short form CsN)
[Parameter] Boolean: T = reverse frame setting, F = according to frame setting
[Example] Subsequent text will be printed white on black within an inverse print frame [GeBE#CsN T] or [GeBE#CsNT]

12.2.3.5 ITALIC PRINT

Using oblique print in order to imitate italic print no. 9.
[Command identifier] CharstyleO (short form CsO)
[Parameter] Boolean: T = italic, F = non italic
The degree of inclination is defined through the system parameters. This command may affect the character thickness.

12.2.3.6 BOLD PRINT

Bold print in order to imitate the font style.
[Command identifier] CharstyleB (short form CsB)
[Parameter] Boolean: T = bold, F = non bold
This command may affect the character thickness.

12.2.3.7 GREY LEVEL PRINT

[Command identifier] CharstyleG (short form CsG)
[Parameter] Boolean: T = grey, F = non grey

12.3 COMMANDS FOR TEXT FRAMES

These commands need to be issued before the text is entered. An open text frame may be closed.
Alignment of text frame contents (alignment)
This command is available for text frames only and defines the text alignment (in relation to the coordinate system of the frame). It is not possible to change the alignment for individual lines or paragraphs of the frame. In this case the text has to be devided into several frames.

Command identifier: Align (short form A)
[Parameter] series of two characters

- TL > top left
- CL > centered left
- BL > bottom left
- TC > top centered
- CC > centered
- BC > bottom centered
- TR > top right
- CR > centered right
- BR > bottom right

[Examples] place centered text vertically in the centre of the frame

- ACC

place right aligned text top right of the frame

- Align TR

12.3.1 AUTOMATIC WORD WRAPPING

(to the letter)
This command is available only for text frames.

[Command identifier] WrapC
[Parameter] Boolean: T = enabled, F = disabled

When automatic word wrapping is activated the text is automatically wrapped to the letter when the preset width of the frame is reached. Otherwise parts of the line may be clipped.

12.4 COMMANDS FOR GRAPHICS_FRAMES

Small graphics can be sent directly as parts of the data stream (graphics data as part of command sequences). This needs relatively much storage capacity. Therefore it is advisable to file often needed graphics as a logo in the AddOn of the flash memory and to open them if needed.
Graphics sent directly as part of the data stream can be scaled if necessary. Scaling logos is not provided. These must be filed in the AddOn area in all necessary sizes.

12.4.1 INSERT LOGOS

[Command Identifier]	**GrLogo**	
[Parameter]	sequential number of the Logo	
[Example]	display Logo number 3	[GeBE#GrLogo3]

Width and height of the logo are saved. So the necessary frame size can be automatically configured.

12.4.2 DEFINE GRAPHICS DATA

To reduce the size of the necessary storage capacity graphics data ought to be compressed. The following table shows the supported compressions and their names, which are part of the graphics command.

Name of compression	Description
U	uncompressed graphics bit 7 of the first byte contains the dot which is furthest left
R	Run-Length-Encoded (RLE) graphics series of 2byte sequences ("runs") consisting of • 1 byte number of repetitions of the following graphics byte • 1 byte uncompressed graphics data (bit 7 contains the dot which is furthest left)
T	Tagged Image File Format (TIFF) revision 4.0 Series of packets, the first byte of which is a control byte (int8_t) with its prefix and its value being interpreted as follows: • The value of the control byte incremented by 1 serves as numerator (<number>) • The prefix gives the quantity and interpretation of the trailing bytes: • positive prefix: <Number> tailing bytes: uncompressed graphics bytes • negative prefix: 1 trailing byte: <number> plus trailing byte = RLE sequence *If control byte == 0, both possible interpretations lead to the same result.*
D	Delta-Row-Graphics Series of packets consisting of: • 1 or 2 control bytes with the following information: • Number of following backup bytes (bit 5-7 of the first by shifted to 0-2, then 1 added: Range [1, 8]) • relative offset within the line from last relative offset (last relative offset == 0 with first packet) • Offset < 0x1F: • Coding in 1 byte (bit 0-4 of the first byte)

- Offset ≥ 0x1F:
- Coding in 2 bytes (bit 0-4 of the first byte + bit 0-7 of the second control byte)

- 1 to 8 (see control byte) backup bytes with the data information to be replaced from actual relative offset.

Particular case 0 packets: Repetition of the previous line

A new graphics frame cannot start with this compression because a previous line is not defined. Instead use e.g. Delta-Row-Graphics with erased previous line. (see below)

DB Delta-Row-Graphics with erased previous line

The graphics within a graphics frame is built of several graphics lines transferred one after another via the command described below. The graphics height is determined by the number of transferred graphics lines.
The graphics width is set as a part of the command. Within a graphics frame the set width of all graphics lines has to be identical. Otherwise a new graphics frame is opened with each change.

[Command identifier]	**GrData**	
[1. parameter]	Graphics line width in dots If the set graphics width and the number of transferred bits are not identical, white dots are added or left out on the right side.	
[2. parameter]	Name of compression	
[3. -n. parameter]	Bytes of graphics data in ASCII-Hex format as continuous data stream	
[Examples]	White dotline of a 205 dots wide graphics:	[GeBE#GrData 205:DB] [GeBE#GrData 205 : U]
	Graphics data are automatically added in white	
	Black dotline of a 205 dots wide graphics: ***<Number> = 0x1A = 26 bytes (208 dots)***	[GeBE#GrData 205:R:1A FF]

12.4.3 SCALE GRAPHICS DATA

[Command identifier]	**ScalingG** (short form ScG)	
[1. parameter]	Scaling of width: rational number	
[2. parameter]	Scaling of height (optional): rational number *If not, width and height are scaled with the same factor.*	
[Examples]	Graphics of double width and half height	[GeBE#ScG2:1/2]
	Graphics of one and a half width and height	[GeBE#ScG 1.5]

The command has to be issued previous to the data to be scaled. If not, the frame is closed.
Scaling parameters follow the coordinate system of the frame. The resulting scaling needs to be displayed as a fraction with numerator and denominator of [1, 255]. Scaling factors thus can be provided in steps of a fraction of 1 over 255 from $^1/_{255}$ up to maximum 255. If necessary the system rounds adequately or limits the value so that limit values can be met.

12.5 BARCODE COMMANDS

12.5.1 DEFINE A BARCODE

[Command identifier] BarcodeD (short form BcD)
[1. Parameter] Kind of barcode (..._T: plain text)

- UPCA or UPCA_T
- EAN8 or EEAN8_T
- EAN13 or EAN13_T
- I2of5 or I2of5_T
- Code39 or Code39_T or 3of9 or 3of9_T
- Code128 or Code128_T
- QR

[2. Parameter] Barcode bytes as continuous data stream in packets like

- <delimiter>data<delimiter>
  Neither space character nor closing bracket nor colon are accepted as delimiter.

[3. Parameter] Input format ASCII or binary (optional)

- ASCII or A
  (Default for all barcodes, with a character range limited to ASCII characters)
- HEX or H
  (Default for all barcodes, with a character range **not** limited to ASCII characters)

[Example] Barcode code 128 with text to represent "GeBE":

- BarcodeD Code128_T : *GeBE*:A
- BarcodeD Code128_T : * 47 65 4247*
- BarcodeD Code128_T : *G*!e!eBeyEy

[Example] Ticket printing with QR code: (see: QR code description chapter 17.4.4.1)

- [GeBE#SoT90] // Start of Ticket 90° alignment
- [GeBE#ScB 8] // Barcode enlargement 8x8-fold
- [GeBE#BcD QR_T:5:M:*QR_GeBE1234567890*:A]
  T: content text
  7: version
  M: ECC level
  A: ASCII data
- [GeBE#EoT] // End of Ticket

12.5.2 SELECTING BARCODE HEIGHT

[Command identifier] **BarcodeH** (short form **BcH**)
[1. Parameter] Line width: positive integer, default line height = 80 dots
[Example] Barcode with line height 120 dots [GeBE#BarcodeH 120]
This command effects 1 D barcodes only. It is meant to adjust the height.

12.5.3 SCALING BARCODE

[Command identifier] **ScalB** (short form **ScB**)
[1. Parameter] Scaling factor: positive interger
[Examples] 1D barcode with double element width: [GeBE#ScB 2]
This command scales a barcode. With scaling factor 1 the minimal possible element size is printed. With 2-D barcodes width and height are scaled proportionally. With 1-D barcodes the scaling factor refers to element width only.

13 COMMANDS FOR THE OUTPUT OF GRAPHIC PRIMITIVE TYPES

Graphic primitives are simple vector graphics like lines or rectangles. Fattened lines always near growing coordinates.

13.1 SET ATTRIBUTES

[Command identifier]	**GrPrimAT**	
[Parameter]	Line width in dots	
[Example]	Line width is 4 dots	[GeBE#GrPrimAT4]

13.2 PRINTING A LINE

Starting from the actual cursor position a line with the actual line attributes is printed. Actually only paraxial lines are supported. That means the x coordinate or the y coordinate must equal 0.

[Command identifier]	**GrPrimL**	
[1. Parameter]	x coordinate of the vector (signed integer out of [-32767, 32767])	
[2. Parameter]	y coordinate of the vector (signed integer out of [-32767, 32767])	
[Example]	Right shift line by 320 dots from the actual cursor position	[GeBE#GrPrimL 320:0]

13.3 PRINTING PARAXIAL RECTANGLE

Starting from the actual cursor position a rectangle with the actual line attributes is printed.

[Command identifier]	**GrPrimR**	
[1. Parameter]	x coordinate of the diagonal vector (signed integer out of [-32767, 32767])	
[2. Parameter]	y coordinate of the diagonal vector (signed integer out of [-32767, 32767])	
[Example]	Right shift rectangle by 320 dots and down shift by 46 dots from the actual cursor position	[GeBE#GrPrimR 320:46]

14 SETTING AND OUTPUT OF SYSTEM VARIABLES

14.1 IDENTIFIER FOR SYSTEM VARIABLES

The following variable identifiers are recognized by the commands for printing or setting of system variables.

[Identifier] **TCount** (short form **TC**)
[Note] Setting a ticket number is only exceptionally necessary because the system counts them automatically. The value range for ticket numbers reaches from 0 to 4294967295 (= 0xFFFFFFFF), i.e. it covers the complete 32 bit value range. Once the maximum value is reached the ticket counter will be reset to 0 for the following ticket (wrapping).

14.2 PRINTING A SYSTEM VARIABLE

[Command identifier] **Vp**
[Parameter] System variable identifier
[Example] Printing the ticket counter [GeBE#Vp TC]

This command prints the actual variable value at the beginning of the ticket print.


Note: A variable frame can only describe the value of one variable. So this command automatically opens a new frame.

14.3 SETTING A SYSTEM VARIABLE

[Command identifier] **Vs**
[1. Parameter] System variable identifier
[2. Parameter] System variable value

14.4 RETRIEVAL OF SYSTEM INFO

14.4.1 STATUS MESSAGE

[Command identifier] **GetStatus**
This command sends a status message.

14.4.2 SYSTEM INFORMATION RETRIEVAL

[Command identifier] **GetSysInfo**
[1. Parameter] System information identifier

SERIAL	controller serial number
GE_APPL	firmware version (GE number) of the application
GE_BOOT	firmware version (GE number) of the boot loader
DEV_BOOT	device name of boot loader firmware with repository revision
REV_BOOT	repository revision of boot loader
REV_APPL	repository revision of application
HW_APPL	hardware specification of the application with repository revision
DEV_APPL	device name of the applications firmware with repository revision
ITEM_NR	item number
ITEM_DESCR	article description
ADDONSPARE	free memory for ADDONs
NR_OF_ADDONS	number of AddOns in the system

14.5 HIGHER LEVEL COMMANDS FOR TICKET CONTROL

14.5.1 STARTING A TICKET

[Command identifier] **SoT**

This command initializes the internal data structures for the start of a new ticket. The frame orientation can be specified as an optional parameter. Without setting the basic orientation in the SoT command, the basic orientation preset via system parameter 14.3 is selected for the ticket, e.g. [GeBE # SoT90]

14.5.2 REPEATING A TICKET

[Command identifier] **TRepeat**
[Parameter] Number of repetitions

This commands indicates the number of ticket repetitions (Parameter = 0: print ticket once, no repetition). With each print the ticket number increments by 1.

14.5.3 COPYING TICKET

[Command identifier] **TCopy**
[Parameter] Number of copies

This command indicates the number of ticket copies (Parameter = 0: print ticket once, no copy). Ticket number remains the same for all copies and is incremented after the last copy.

14.5.4 END OF TICKET

[Command identifier] **EoT**

This command starts the ticket print.

14.6 REQUESTING STORED COMMAND SEQUENCES

It is advisable to file often needed series of command sequences in the AddOn of the flash memory and to open them if needed.

[Command identifier] **CannedC** (short form **CC**)
[Parameter] Serial number of the stored sequence.

14.7 SYSTEM PARAMETER COMMANDS

System parameters are used for system configuration. When switching on the printer or after a reset the values filed in FLASH (default values ex works) or EEPROM (modifications designed to user specifications) are stored in copy in the RAM and are activated. According to system parameters the value may be changed by the user during operation. Please pay attention to the access rights for the respective system parameter.

14.7.1 READING HOST INTERFACE INFORMATION

[Command identifier] **SyspRead**

[1. Parameter] System parameter identifier
accepted wild cards:
- 0,0: all system parameters of all modules
- x,0: all system parameters of the module x

[2. Parameter] series of identifiers for required information
- FIRM: firmware parameter value
- FACT: factory parameter value
- UEEP: EEPROM parameter value
- URAM: RAM parameter value
- HINT: access rights and help texts

If several identifiers are noted, these are separated with empty spaces. Output are single messages in preset order. If wildcards are used for the system parameter identification an end identification is sent after the message string of the last system parameter.

Abbreviation	User modifications
r/o	not tolerated
e_w	in expert mode only, activation only after RESET
e_a	in expert mode only, activation possible without RESET
u_w	in user mode, activation only after RESET
u_a	in user mode, activation possible without RESET

14.7.2 VALUE MODIFICATION DESIGNED TO USER SPECIFICATIONS

[Command identifier] **SyspModify**
[1. Parameter] System parameter identification (wildcards not tolerated)
[2. Parameter] New value
[3. Parameter] Storage area:
- UEEP: EEPROM (permanent storage)
- URAM: RAM (volatile storage, non permanent value)

[4. Parameter] This parameter is needed in special cases only. If not needed the new value is activated, if permitted by access rights (see above).
Boolean, optional:
- T: with activation
- F: without activation

This command changes the value of a system parameter. If the access rights are not met an error message is released and the command will not be executed.
System parameters are organised in module groups. If a system parameter value designed to user specifications is filed in the EEPROM, a complete set for this module is created in the EEPROM on the basis of ex works default values.


Normally filing in RAM without activation or storage in the EEPROM is not reasonable.

14.7.3 COPYING SYSTEM PARAMETERS

[Command identifyer] **SyspCopy**
[1. Parameter] System parameter identification
tolerated wildcards:
- 0,0: all system parameters of all modules
- x,0 all system parameters of module x

[2. Parameter] Source
- FIRM: firmware parameter
- FACT: factory parameter
- UEEP: parameters designed to user specifications in the EEPROM
- URAM: RAM

[3. Parameter] Target
- UEEP: parameters designed to user specifications in the EEPROM
- URAM: RAM

[4. Parameter] This parameter is needed in special cases only. If not needed the new value is activated, if permitted by access rights (see above).
Boolean, optional:
- T: with activation


This should be avoided if wildcards are used for system parameter identification, because not all module system parameters suffer activation during operation. (Error message and suppression of command).

- F: without activation

This command copies a system parameter value from source to target storage area.

14.7.4 CLEARING SYSTEM PARAMETERS IN THE EEPROM

[Command identification] **SyspClearUEEP**
[1. Parameter] Module number x of the system parameter identification
tolerated wildcards: 0: all modules
Alternatively a specification in the format x,0 or 0,0 is tolerated.

This command clears one or all system parameter sets in the EEPROM.
TCancel A (for all tickets) or TCancel L (for last ticket)
At the moment this clears all received data: corresponds to <ESC>"A" of <RS><DLS><05d>

Example of a ticket with a paper width of 58 mm:
[GeBE#SoT] (we recommend to start each ticket with a command "Start of Ticket")
[GeBE#FAttrO F]
[GeBE#FP 10:235][GeBE#GrPrimAT3][GeBE#GrPrimR130:50]
[GeBE#FP 25:195][GeBE#F1][GeBE#ScT2:2]PRICE
[GeBE#FP 25:215][GeBE#F1][GeBE#ScT2:2]PRICE
[GeBE#FP 25:245][GeBE#F2][GeBE#ScT2:2]5.50
[GeBE#FP 145: 10][GeBE#GrPrimAT3][GeBE#GrPrimR580:170]
[GeBE#FP 145:190][GeBE#GrPrimAT3][GeBE#GrPrimR580:105]
[GeBE#FP 430:254][GeBE#F2][GeBE#ScT1:3]Fassbinder
[GeBE#FP 170:254][GeBE#F2][GeBE#ScT2:3]28.12
[GeBE#FP 310:254][GeBE#F2][GeBE#ScT2:3]17:30
[GeBE#FP 160: 20][GeBE#F2][GeBE#ScT2:4]Atelier/Bollwerk
[GeBE#FP 390: 77][GeBE#F2][GeBE#ScT1:1]PRESENTS
[GeBE#FP 160:100][GeBE#F2][GeBE#ScT2:5]BRIGHT STAR - MEINE LIEBE.EW
[GeBE#FP 185:195][GeBE#F1][GeBE#ScT2:3]DATE
[GeBE#FP 330:195][GeBE#F1][GeBE#ScT2:3]TIME
[GeBE#FP 440:195][GeBE#F1][GeBE#ScT2:3]THEATRE
[GeBE#FP 550:195][GeBE#F1][GeBE#ScT2:3]ROW
[GeBE#FP 640:195][GeBE#F1][GeBE#ScT2:3]SEAT
[GeBE#FP 185:220][GeBE#F1][GeBE#ScT2:3]DATE
[GeBE#FP 330:220][GeBE#F1][GeBE#ScT2:3]TIME
[GeBE#FP 420:220][GeBE#F1][GeBE#ScT2:3]THEATRE
[GeBE#FP 550:220][GeBE#F1][GeBE#ScT2:3]ROW
[GeBE#FP 640:220][GeBE#F1][GeBE#ScT2:3]SEAT
[GeBE#FP 860: 15][GeBE#F2][GeBE#ScT1:3]Atelier/Bollwerk
[GeBE#FP 830: 60][GeBE#F2][GeBE#ScT1:3]BRIGHT STAR - MEINE LIEBE.EW
[GeBE#FP 830:110][GeBE#F2][GeBE#ScT2:3]28.12 17:30
[GeBE#FP 830:150][GeBE#F2][GeBE#ScT2:3]Fassbinder
[GeBE#FP 830:190][GeBE#F2][GeBE#ScT2:3] 5.50 1
[GeBE#FP 830:230][GeBE#F2][GeBE#ScT2:3]DISCOUNT DAY
[GeBE#FP1050:250][GeBE#F2][GeBE#ScT3:8]K

[GeBE#FP 065:055][GeBE#F2][GeBE#ScT3:8]K
[GeBE#FP 000:315][GeBE#F1][GeBE#ScT1:2] 1176489-28.1216:0501
[GeBE#FP 810:300][GeBE#F1][GeBE#ScT1:2] 1176489-28.1216:0501
[GeBE#EoT]

15 C32 PRINTER STATUS

Two kinds of status transmission are available for the default print modes A8 and C32.
1. Default status message via flags called A8 status messages
2. Extended status message with protocol called C32 status messages

A8 messages are based upon the flag principle. Each flag of a status byte corresponds to a message. To make a distinction between the 5 status bytes and other data, status bytes are in always situated above 80hex. Moreover the upper flags are used as identifiers to make it possible to identify the individual status bytes.
C32 messages are always transferred in an ASCII format with a secured data format. A message contains an identifier and a checksum to recognize defective transmissions.

In the parameters "module 12 message" and "module 30 message_A8", A8 as well as C32 messages can be configured to decide whether they shall be sent automatically or on demand. Also which kind of messages are sent or if they are sent at all.

Parameter:

12,14: 1: emulation specific status messages and C32 simultaneously
0: emulation specific messages only

Settings for additional messages (not status messages)

12,3 1: enable / **0: disable** releasing syntax error messages in the C32 format

12,4 1: enable / **0: disable** releasing internal error messages in the C32 format

Emulation specific status messages:

12,7 emulation specific automatic status messages enable/disable **1:on**/ 0:off

12,8 recurrent emulation specific status message in [ms] / **0: off**

15.1 REQUESTING PRINTER STATUS

[Name]	Request System Status			
[Format]	ASCII	<RS>	“k“	n
	Hex	1Eh	6Bh	n
	Decimal	30d	107d	n

[Description] With parameter n the output format can be selected irrespective of the actually set emulation.

n:= 0	status message in debug format
n:= 1	status message in GeBE-C32-format
n:= 2	status message in GeBE-A8-format
n:= 3	status message in GeBE-N78-format
n:= 6	status message in FGL-format
n:= 254	status message in the actually set format + status message in GeBE-C32 format
n:= 255	status message in the actually set format

16 MESSAGE FORMAT C32

The message format GeBE-C32 ensures that the message cannot be misunderstood as A8 status messages or control command. Therefore messages are always a series of ASCII characters. Alternatively you can switch to a format containing binary information also.
Messages are saved via a printer hardware CRC. It is up to the user whether he wants to utilize this CRC. Each message must be identifiable with a unique index, so that clear text, e.g. for the output of native language information, can be exchanged at any time.

16.1 COMPOSITION OF A MESSAGE

Messages contain use data embedded into a standardized layout. All of them are printable ASCII characters of a defined length. To make the message easily readable for the user separators are inserted. Moreover these help ensure the 4 byte alignment of the use data which is necessary to form the 32 bit CRC. Each message consists of:

- the opening brackets as message start identifier (length 1 byte)
- the original identifier (length 1 byte)
- type classification (length 1 byte)
- a colon as separator (length 1 byte)
- length (length 10 bytes)
- starting identifier of use data (length 2 bytes)
- use data (variable length divisible by 4)
- a possible position information in the use data range
- the printer status #i_ (length 4 bytes)
- the end identifier of the use data (length 2 bytes)
- the STM 32 bit hardware CRC (length 10 bytes)
- a colon as a separator (length 1 byte)
- a repetition of the original identifier (length 1 byte)
- a repetition of the type classification (length 1 byte)
- the closing bracket as message end identifier (length 1 byte)

16.2 PRINTER OPERATION STATUS

The printer status indicates the general state of the system. Several states can be active at the same time. The state which is actually the most important is displayed in each C32 message at the end of the use data range.

According to configuration different status indications via LED.

 # <Status> <Level> <further messages>

Info levels are just information in the system, no user action needed.

INFO	0: OK everything is OK no messages
INFO	1: fast loading
INFO	2: trickle charging
INFO	3: completely charged
INFO	5. preheating head
INFO	6: head preheating temperature reached

Warning levels represent a system error, but nevertheless allow further printing. According to level user action is urgent.

WARNING	1: There is no problem yet, but there will be one in the future. e.g. near paper end.
WARNING	3: A problem is expected very soon.
WARNING	5: Firmware detected an error which nevertheless allows printing with limitations (emergency operation). Service should be carried out as soon as possible. If for example the EEprom is not found the printer works uses firmware parameters.

Error levels represent serious system errors, making further printing impossible. According to the error level, it will be easy or difficult to correct that error.

ERROR	1: Firmware has detected an error that might be corrected via remote maintenance. (e.g. service or on/offline)
ERROR	3: Firmware has detected a simple error that prevents the printer from working. Service can restore the printing: e.g. paper end, paper jam....
ERROR	5: Firmware has detected a fatal error. System repair is needed and the error can not be corrected by simple service action.

16.3 EXAMPLE

Example: Answer to requesting the parameter value 18,10, Value = 2000

[AR:0x0000002C#[0x004CE0C7ms0x001D0000:(FIRM): 18,10: 2000 #i__]#0x9624A77E:AR]

Message start and end identifier

The first and the last character of a message are the opening and closing brackets.

Original identifier

The second character of a message indicates the origin of the message.

- "A" or "a" application
- "B" or "b" bootloader

In order to recognize missing messages, the original identifier comes with alternating capital letters and small letters. Bootloader and application send their first message with capital letter as original identifier.

Type classification

The third character of a message is meant to identify the kind of message. Capital letters inform you that the use data contain printable ASCII characters only. Small letters mean there is no such limitation.

- "I" or "i" information (e.g. spontaneous status message)
- "C" or "c" cyclic message
- "R" or "r" response message
- "P" or "p" protocol or syntax error

Printer operation status #i_ (see system status)

indicates general system status
divider
e most serious system status.. "i" = information, "w" = warning, "e" = error
2. status level (not yet implemented, to be replaced by _)
+ further lower levels are planned (not yet implemented, to be replaced by _)

Length specification and CRC

The length specification relates to mere use data (without general system status and without Modulo 4 fill-in-bytes). The CRC relates to the complete use data range between start and end identifier of the data. Like all other embedded use data they are always output as 32 bit values ASCII Hex Coding format 0x12345678. This is also true for messages containing binary data.

Use data start and end identifiers

Hash key and brackets "#["bzw."]#" enclose the data. This makes them easily visible for the user and they start at a fixed offset divisible by 4 (important for the CRC) within the message.

Use data

Use data begin with a 32 bit time stamp followed by a distinct 32 bit message ID plus a ":". If the use data do not solely consist of printable ASCII characters, timestamp and message ID are output as Little Endian Unsigned Integers. Otherwise the timestamp is output as ASCII Hex Coding format 0x12345678ms and the ID is output as ASCII Hex Coding format 0x12345678. Depending on the message ID further data may follow.
To form a 32 bit CRC the use data must be 4 byte aligned. If necessary the data are completed with fill in bytes (in case of ASCII Hex Coding these are spaces).

The use data length is not fixed and can be customized at any time!
Therefore it is absolutely necessary to evaluate the indicated length.

16.4 MESSAGE ID

The message ID states the message source. It is built of two parts. The GMID which is the message releasing module and the individual message code.
Example: 0x001D0000
GMID is 001Dh = 29 and thus coming from module message _C32
Message Code is 0000h which is a status messages

0:	**0000h**		**GMID_NAUGHT**	
		0000	MsgCode_NAUGHT__FATAL_ERROR	Fatal error, system reset necessary
		0001	MsgCode_NAUGHT__SYSTEM_ERROR	System error, system reset **not** necessary
		0002	MsgCode_NAUGHT__SYNTAX_ERROR	User error
1:	*0001h*		*GMID_Debug*	
		0000	*MsgCode_Flash__ADDONSPARE*	*Storage space still available for AddOns*
		0001	*MsgCode_Flash__ADDON*	*read AddOn*
		0002	*MsgCode_Flash__NUMBER_OF_ADDONS*	*Read number of AddOns in the system*
2:	**0002h**		**GMID_Boot (only sent by the bootloader)**	
3:	**0003h**		**GMID_PM**	
4:	**0004h**		**GMID_SysParams**	
		0000	SYSPUEEP_FAULT	EEPROM parameter damaged
		0001	SYSPUEEP_FAULT_MISMATCH	EEPROM parameter damaged or EEPROM-Parameter of a module is incomplete ((SysParams_FAULT_Data1Type) data word with GMID of the concerned module and EEPROM index.) Restored data are integrated into the RAM.
		0002	STARTUP_SYSP_OUT_OF_RANGE	At startup at least one system parameter was out of the allowed range of values. ((SysParams_FAULT_Data1Type) Data word with GMID of the concerned module). Value was limited.
		0003	SYSPFACT_FAULT_MISMATCH	Factory parameter damaged or incomplete. ((SysParams_FAULT_Data1Type) Data word with GMID of the concerned module). Restored data

are integrated into the RAM.

0004 REQUEST_DENIED — System parameter request was denied.

0005 SYSPUEEP_PROG_DONE — Programming command for EEPROM parameter was ended.
Identifier:
0x00000000: EEPROM initialization+bit mask of concerned EEPROMs: Bit#0 = Main / Bit#1 = Head)
0x01000000: Erasing of system parameters (subsequently system parameter identifier)
0x02000000: Writing of system parameters (subsequently system parameter identifier)

6: 0006h GMID_Flash

0000	ADDONSPARE	
0001	*ADDON*	
0002	NUMBER_OF_ADDONS	
0003	MCU_SERIAL	
0004	GE_APPL	
0005	GE_BOOT	
0006	DEVICE_BOOT	
0007	REPOSREV_BOOT	
0008	REPOSREV_APPL	
0009	HWSPEC_APPL	
000A	DEVICE_APPL	
000B	ITEMNUMBER_APPL	
000C	ITEMDESCRIPTION_APP	
000D	ADDONSPARE_APPL	available storage place for programming with the application
000E	GeBE_SERIAL	serial number assigned by GeBE (tags range)
000F	FONT	read font
0010	NUMBER_OF_FONTS	read number of fonts in the system
0011	LOGO	read logo
0012	NUMBER_OF_LOGOS	read number of logos in the system
0013	BATCH	read batch
0014	NUMBER_OF_BATCHES	read number of batches in the system
0015	HARDWARE_CODE_APPL	hardware code of the application
0016	CRC_APPL	CRC check sum of loaded firmware
0017	CRC_BOOT	CRC check sum of bootloader
0018	HEAPSPARE	available heap space

7: 0007h GMID_I2C

8: 0008h GMID_I2C_EEPROM

9: 0009h GMID_HostIF

10: 000Ah GMID_HostIF_RS232

11: 000Bh GMID_HostIF_USB

12: 000Ch GMID_Msg

0000	STATUSREQUEST_DEBUGGING	accumulated debug status messages
0001	ANALOGUEVALSDEBUGGING	accumulated debugging ADC value message: Array, sorted according to ascending ADC numbers and within an ADC according to ascending channel number (only according to $ (HARDWARE_CODE).make existing channels) @warning. Initiating the message has to be done with MsgCode_Print_Sensors__ADCVALS.

13:	**000Dh**	**GMID_Parser**	
	0000	INITPRIMARYEMULATION	Changing primary emulation
	0001	SYSCOMMAND_UNKNOWN	Syntax error with system command <RS> ...: unknown trailing byte
	0002	SYSCOMMAND_DLE_UNKNOWN	Syntax error with system command <RS> <DLE> ...: unknown trailing byte
	0003	SYSCOMMAND_SYSP_SYNTAX	Syntax error with system command <RS> ... for system paramter: general format error
	0004	SYSCOMMAND_SYSP_UNKNOWN	Syntax error with system command <RS> … for system parameter: invalid command index
	0005	SYSCOMMAND_SYSP_ID	Syntax error with system command<RS> ... for system parameter: invalid parameter identifier
	0006	SYSCOMMAND_SYSP_ADDRESSEE	Syntax error with system command <RS> ... for system parameter: invalid output target
	0007	SYSCOMMAND_SYSP_VAL_SYNTAX	Syntax error with system command <RS> ... for system parameter: invalid string for value
	0008	SYSCOMMAND_SYSP_VAL_ARRAYIDX,	Syntax error with system command <RS> ... for system parameter: invalid array index
	0009	SYSCOMMAND_INFO_UNKNOWN	Syntax error with system command <RS> ... for system information request: invalid command index
	000A	SYSCOMMAND_EMUL_UNKNOWN	Syntax error with system command <RS> ... for system status request: invalid command index
	000B	SYSCOMMAND_IGNORED,	command not (yet) supported was caught and ignored
14:	**0Eh**	**GMID_Parser_GeBE_C32**	
	0000	MsgCode_Parser_GeBE_C32__COMMAND_IGNORED	command not (yet) supported was caught and ignored . Answer is 8 byte with the echo of wrong command.
	0001	MsgCode_Parser_GeBE_C32__SYNTAX_ERROR	user error
15:	**0Fh**	**GMID_Parser_GeBE_A8**	
	0000	MsgCode_Parser_GeBE_A8__COMMAND_IGNORED	command not (yet) supported was caught and ignored . Answer is 8 byte with the echo of wrong command.
	0001	MsgCode_Parser_GeBE_A8__SYNTAX_ERROR	user error
16:	**10h**	**GMID_Parser_GeBE_N78**	
	0000	MsgCode_Parser_GeBE_N78__COMMAND_IGNORED	command not (yet) supported was caught and ignored . Answer is 8 byte with the echo of wrong command.
	0001	MsgCode_Parser_GeBE_N78__SYNTAX_ERROR	user error

17:	**11h**		**GMID_Renderer**	
		0000	MsgCode_Renderer__JOBMEM	job buffer too small
		0001	MsgCode_Renderer__OBJECTMEM	object list too small
		0002	MsgCode_Renderer__GRAPHICENCODING	required graphics coding not supported
		0003	MsgCode_Renderer__IDX_LOGO,	invalid logo index
		0004	MsgCode_Renderer__IDX_FONT,	invalid font index
		0005	MsgCode_Renderer__BARCODE_DATA	barcode data string invalid or not supported
		0006	MsgCode_Renderer__BARCODE_HMI	barcode HMI string error
		0007	MsgCode_Renderer__SCALING	required scaling not supported
		0008	MsgCode_Renderer__GRAPHIC_DATALENGTH	odd number of data bytes
		0009	MsgCode_Renderer__GRAPHIC_DELTAROW_WITDH	delta row graphics: width not stated
18:	**12h**		**GMID_Print**	
		0000	SYNCREPORT	Sync message (position counter + sync word)
		0001	TICKETCOUNT	TICKETCOUNT
19:	**13h**		**GMID_Print_Sensors**	
		0001	MsgCode_Print_Sensors__ADCVAL_AUX1	ADC value AUX1
		0002	MsgCode_Print_Sensors__ADCVAL_AUX2	ADC value AUX2
		0003	MsgCode_Print_Sensors__ADCVAL_AUX3	ADC value AUX3
		0004	MsgCode_Print_Sensors__ADCVAL_AUX4	ADC value AUX4
		0005	MsgCode_Print_Sensors__ADCVAL_AUX5	ADC value AUX5
		0006	MsgCode_Print_Sensors__ADCVAL_AUX6	ADC value AUX6
		0007	MsgCode_Print_Sensors__ADCVAL_AUX7	ADC value AUX7
		0008	MsgCode_Print_Sensors__ADCVAL_VP	ADC value VP
		0009	MsgCode_Print_Sensors__ADCVAL_MotorTemp	ADC value MotorTemp
		000A	MsgCode_Print_Sensors__ADCVAL_HeadTemp	ADC value HeadTemp
		000B	MsgCode_Print_Sensors__ADCVAL_PaperEnd	ADC value PaperEnd
		000C	MsgCode_Print_Sensors__ADCVAL_NearPaperEnd	ADC value NearPaperEnd
		000D	MsgCode_Print_Sensors__ADCVAL_Vref	ADC value Vref
		000E	MsgCode_Print_Sensors__ADCVAL_BatteryTemp	ADC value BatteryTemp#
		000F	MsgCode_Print_Sensors__ADCVAL_HeadAlignment	ADC value HeadAlignment
20:	**14h**		**GMID_Parser_FGL**	
		0000	MsgCode_Parser_GeBE_C32__COMMAND_IGNORED	command not (yet) supported was caught and ignored
		0001	MsgCode_Parser_GeBE_C32__SYNTAX_ERROR	user error
21:	**15h**		**GMID_Print_Scanner**	
22:	**16h**		**GMID_Print_Cutter**	
23:	**17h**		**GMID_Print_Winder**	
24:	**18h**		**GMID_Print_EoT**	
25:	**19h**		**GMID_HostIF_HPIr**	

26:	**1Ah**		**GMID_Parser_GeBE_HPIr**	
		0000	MsgCode_Parser_HPIr__COMMAND_IGNORED	command not (yet) supported was caught and ignored
		0001	MsgCode_Parser_HPIr__SYNTAX_ERROR	user error
27:	**1Bh**		**GMID_Menu**	
28:	**1Ch**		**GMID_Key**	
29:	**1Dh**		**GMID_Msg_GeBE_C32**	
		0000	STATUSREQUEST	Status message in format GeBE_C32
		0001	SYSPREQUESTVAL	Message of system parameter values
		0002	SYSPREQUESTHINT	Message of system parameter hint texts
		0003	SYSPREQUEST	Message of system parameter information
		0004	ANALOGUEVALSCUM	Cumulative analog value message: position counter followed by ADC values sorted by ascending index, invalid entries are indicated by blanks
30:	**1Eh**		**GMID_Msg_GeBE_A8**	
		0000	STATUSREQUEST	Status message in format GeBE_A8
		0001	STATS	Requesting a statistics message in format GeBE_A8
		0002	SYNCREQUEST	Requesting a synchronous message in format GeBE_A8
31:	**1Fh**		**GMID_Msg_GeBE_N78**	
		0000	STATUSREQUEST	Status message in format GeBE_N78
		0001	STATS	Requesting a statistics message in format GeBE_N78
		0002	SYNCREQUEST	Requesting a synchronous message in format GeBE_A8
32:	**20h**		**GMID_Msg_FGL**	
		0000	Msg_FGL__STATUSREQUEST	Status message in FGL format
		0001	MsgCode_Msg_FGL__PROM_TYPE_AND_TICKET_COUNT, PROM TYPE AND TICKET COUNT STATUS REQUEST	
33:	**21h**		**GMID_Print_DebugPWM**	
34:	**22h**		**GMID_PM_Status**	
35:	**23h**		**GMID_SysParams_Config**	
36:	**24h**		**GMID_Print_HeadTest**	
37:	**25h**		**GMID_Print_Actuators**	
		0000	MsgCode_Print_Actuators__POSCOUNT	Position counter
38:	**26h**		**GMID_Parser_PCL3**	
		0000	MsgCode_Parser_PCL3__COMMAND_IGNORED	command not (yet) supported was caught and ignored
		0001	MsgCode_Parser_PCL3__SYNTAX_ERROR	user error
39:	**27h**		**GMID_Msg_LED**	
40:	**28h**		**GMID_Msg_Status**	
41:	**29h**		**GMID_Print_HeadSpec**	
42	**2Ah**		**GMID_Print_DC_Cutter**	
43:	**2Bh**		**GMID_Utils**	

44:	**2Ch**		**GMID_Charge**	
45:	**2Dh**		**GMID_Stats_RefVals_Head**	
46:	**2Eh**		**GMID_Stats_RefVals_Main**	
47:	**30h**		**GMID_HostIF_IrDA**	
48:	**30h**		**GMID_Scanner**	
		0000	SCANNER_STRING	Message scanner string
49:	**31h**		**GMID_Motion**	
		0000	POSITION	Message position of Motion_Sensor
50:	**32h**		**GMID_HostIF_**BLE,	
		0000	BLE_STATUS_STRING	Message Bluetooth-Modul-String
		0001	Msg BLE_TIMER	Message BLE-Timer
53:	**35h**		**GMID_Stats.Collect**	
		0000	GROUP	end detection for group of all statics value messages
		0001	POR	number of power-on cycles
		0002	RUNTIME_PRINTER	operation time of the printer in minutes
		0003	RUNTIME_HEAD	operation time printhead in minutes (hardware dependent)
		0004	OPERATING_TEMP_MXX_M40	number of print starts at head temperature under -40°C
		0005	OPERATING_TEMP_M40_M30	number of print starts at heat temperature -40°C to -30°C
		0006	OPERATING_TEMP_M30_M20	number of print starts at heat temperature -30°C to -20°C
		0007	OPERATING_TEMP_M20_M10	number of print starts at heat temperature -20°C to -10°C
		0008	OPERATING_TEMP_M10_000	number of print starts at heat temperature -10°C to 0°C
		0009	OPERATING_TEMP_000_P10	number of print starts at heat temperature 0°C to +10°C
		000A	OPERATING_TEMP_P10_P20	number of print starts at heat temperature +10°C to +20°C
		000B	OPERATING_TEMP_P20_P30	number of print starts at heat temperature +20°C to +30°C
		000C	OPERATING_TEMP_P30_P40	number of print starts at heat temperature +30°C to +40°C
		000D	OPERATING_TEMP_P40_P50	number of print starts at heat temperature +40°C to +50°C
		000E	OPERATING_TEMP_P50_P60	number of print starts at heat temperature +50°C to +60°C
		000F	OPERATING_TEMP_P60_P70	number of print starts at heat temperature +60°C to +70°C
		0010	OPERATING_TEMP_P70_P80	number of print starts at heat temperature +70°C to +80°C
		0011	OPERATING_TEMP_P80_PXX	number of print starts at heat temperature +80°C or more
		0012	FEEDLENGTH_PRINTER_METER	operation performance of the system [m]
		0013	FEEDLENGTH_HEAD_METER	operation performance of the printhead [m]
		0014	FEEDLENGTH_HEAD_LINES	operation performance of the printhead [dotlines]
		0015	PAPER_LENGTH	paper length since the last paper exchange [dm]
		0016	PE	number of detected exchange on PE
		0017	HEADUPS	number of detected HeadUps
		0018	CUTS	number of cutter cuts
		0019	EOT	number of EoT processes
		002A	SHUT DOWN	number of failed shut downs via deactivation of VCC-enable

54: 36h GMID_Stats.Tracing

0000	GROUP	end detection for group of all tracing data messages
0001	PRINTMOTOR1TIME	motor1-S.L.T. [us]
0002	PRINTISRLAGTIME	printer call delay [us]
0003	PRINTISRRUNTIME_IDLE	printer running time PRINTISRSTATE_IDLE [us]
0004	PRINTISRRUNTIME_RESUME	printer running time PRINTISRSTATE_RESUME [us]
0005	PRINTISRRUNTIME_PRELINE	printer running time PRINTISRSTATE_PRELINE [us]
0006	PRINTISRRUNTIME_STARTLINE	printer running time PRINTISRSTATE_STARTLINE [us]
0007	PRINTISRRUNTIME_PREHEAT	printer running timePRINTISRSTATE_PREHEAT [us]
0008	PRINTISRRUNTIME_DOSTROBES	printer running time PRINTISRSTATE_DOSTROBES [us]
0009	PRINTISRRUNTIME_CLOSELINE	printer running time PRINTISRSTATE_CLOSELINE [us]
000A	PRINTISRRUNTIME_AWAIT_STATUS	printer running time PRINTISRSTATE_AWAIT_STATUS [us]
000B	RACING_PRINTISRRUNTIME_FIRSTLINE	printer running time PRINTISRSTATE_FIRSTLINE [us]
000C	PRINTISRRUNTIME_NOLINE	printer running time PRINTISRSTATE_NOLINE [us]
000D	PRINTISRRUNTIME_SUSPEND	printer running time PRINTISRSTATE_SUSPEND [us]
000E	PRINTSTROBESTON	strobe T_ON [us]
000F	PRINTISRRUNTIME_RETARDATION	printIsr running time PRINTISRSTATE_RETARDATIONS [us]
0010	DOTR_AVERAGE_OHM	measured average dot resistance value [Ohm]

55: 37h GMID_Stats.Ref.Vals

0000	MsgCode_Stats__REFVALS_GROUP	end detection for all groups of all reference value messages
0001	MsgCode_Stats__REFVALS_HEAD_NAME	head name
0002	MsgCode_Stats__REFVALS_SERIAL_NUMBER1	head serial number 1
0003	MsgCode_Stats__REFVALS_SERIAL_NUMBER2	head serial number 2
0004	MsgCode_Stats__REFVALS_WEIGHT_OF_HEAD_MODULE	weight of head module (g)
0005	MsgCode_Stats__REFVALS_ADC_HeadAlignment	ADC value head alignment
0006	MsgCode_Stats__REFVALS_DOTR_AVERAGE_OHM	actual average value of dot-resistance [Ohm]
0007	MsgCode_Stats__REFVALS_HEADPUSH_PLATENFORCE	HeadPush: platen pressure reference value [dN]
0008	MsgCode_Stats__REFVALS_POSCOUNT_ACTUATOR1	position counter actuator1reference value (HEADLIFT)
0009	MsgCode_Stats__REFVALS_POSCOUNT_ACTUATOR2	position counter actuator2 reference value (HEADPUSH)
000A	MsgCode_Stats__REFVALS_POSCOUNT_ACTUATOR3	position counter actuator3 reference value
000B	MsgCode_Stats__REFVALS_POSCOUNT_ACTUATOR4	position counter actuator4 reference value
000C	MsgCode_Stats__REFVALS_ADC_AUX1	ADC-Wert AUX1 reference value
000D	MsgCode_Stats__REFVALS_ADC_AUX2	ADC-Wert AUX2 reference value
000E	MsgCode_Stats__REFVALS_ADC_AUX3	ADC-Wert AUX3 reference value
000F	MsgCode_Stats__REFVALS_ADC_AUX4	ADC-Wert AUX4 reference value
0010	MsgCode_Stats__REFVALS_ADC_AUX5	ADC-Wert AUX5 reference value
0011	MsgCode_Stats__REFVALS_ADC_AUX6	ADC-Wert AUX6 reference value
0012	MsgCode_Stats__REFVALS_ADC_AUX7	ADC-Wert AUX7 reference value
0013	ADC_VP	ADC-Wert VP reference value
0014	MADC_MOTORTEMP	ADC-Wert MotorTemp reference value
0015	ADC_HEADTEMP	ADC-Wert HeadTemp reference value
0016	ADC_PAPEREND	ADC-Wert PaperEnd reference value

0017	ADC_NEARPAPEREND	ADC-Wert NearPaperEnd reference value */
0018	ADC_VREF	ADC-Wert Vref reference value
0019	CORRECTION_VALUE_1	correction value 1
001A	CORRECTION_VALUE_2	correction value 2
001B	CORRECTION_VALUE_A	correction value A
001C	CORRECTION_VALUE_B	correction value B
001D	ADC_BatteryTemp	ADC value of battery temperature
001E	CORRECTION_VALUE_3	correction value 3
001F	CORRECTION_VALUE_C	correction value C
0020	HEADTEMPCORRECTION	= difference between a measured temperature and the temperature detected by the printer in °dC (=1/10°C)
0021	HEADATTACHMENTOFFSET	
0022	REFVALS_ADC_MOTORTEMP_LIMIT_L	ADC value of motor temperature, minimum value (upper temperature limit)
0023	MsgCode_Stats__REFVALS_USER_SNR	user serial number
0024	MsgCode_Stats__REFVALS_USER_STRING1	user string1
0025	MsgCode_Stats__REFVALS_USER_STRING2	user string2
0026	CORRECTION_VALUE_U1	correction value U1
0027	CORRECTION_VALUE_U2	correction value U2
0028	HEATER_SIZE_WIDTH	heater width [dµm = 100 nm]
0029	HEATER_SIZE_HEIGHT	heater height [dµm = 100 nm]

16.5 ORGANIZATION OF USE DATA RANGE

Use data range organization of a message answer when requesting a system parameter value

Example: Answering the request of parameter value 18,10, value = 2000

[AR:0x0000002C#[0x004CE0C7ms0x001D0001: (FIRM): 18,10: 2000]#0x9624A77E:AR]

```
u : m , x : w
| | | | | | |
| | | | | | |
| | | | | | +----- Result text (variable amount of characters depending on system parameter ID
| | | | | +------- Separator
| | | | +--------- System parameter ID: module specific parameter index
| | | +----------- Character linking the two components
| | +------------- System parameter ID: module identifier
| +--------------- Separator
+----------------- Original system parameter identifier
```

Original identifier	Origin of the system parameter
(URAM)	User RAM parameter
(UEEP)	User EEPROM parameter
(FACT)	Factory parameter
(FIRM)	Firmware parameter

When reading a system parameter value a default value will be read if the requested value does not exist. Output in this case of original identifier format "(FIRM) replaces (FACT):
In case of not implemented system parameters (access right dep = "parameter is deprecated") the system parameter value and the prefixed separator are omitted.

Use data range organization of a message answer when requesting a system parameter hint text

Example: Answering the request of parameter value 18,10, value = stall timeout in ms
[aR:0x00000038#[0x000B28A4ms0x001D0002: u_a: 18,10: Stall timeout [ms]]#0x371C23F3:aR]

p : m , x : h
| | | | | | |
| | | | | | |
| | | | | | +----- Resulting text (variable amount of characters according to system parameter ID)
| | | | | +------- Separator
| | | | +--------- System parameter ID: module specific parameter index
| | | +----------- Character connecting both elements
| | +------------- System parameter ID: module identifier
| +--------------- Separator
+----------------- System parameter access right (Code)

Code	System parameter access right
dep	Parameter is deprecated
r/o	Factory configuration only
e_w	Expert configuration, changes get active only after system reset
e_a	Expert configuration, changes may be activated during operation
u_w	User configuration, changes get active only after system reset
u_a	User configuration, changes may be activated during operation

If the system parameters are not implemented (access right dep = "Parameter is deprecated") the hint text and the prefixed separator will be omitted.

16.6 C32 STATUS MESSAGE: REQUESTING PRINTER STATUS

[Name]	Request System Status			
[Format]	ASCII	<RS>	“k“	n
	Hex	1Eh	6Bh	n
	Decimal	30d	107d	n
[Description]	With parameter n the output format can be selcted independent of the actually set emulation.			
	n:= 1 Status message in the format GeBE-C32			
	n:= 255 Status message in the actually set format			

Message organization:

[AI:length#[Timestampms0x001D0000:Poscount,W1,W2,W3,W4#i__]#CRC:AI]

Message ID - W1 … W4: 32 Bit status words 1 – 4 - Printer operation status

Example: Output in ASCII format:

[AI:0x00000052#[0x0000000Ems0x001D0000:0x00000022,
0x00000001,0x00000001,0x00000000,0x0000BE40#i__]#0x2D500C85:AI]

The bits are defined as follows:

HINT: The current number is required to patch the corresponding bit for the A8 status messages.

1. Status word

Bit	Current No.	Level	Status	0	1
0	1	w2	Near paper end (NPE) -	-	paper OK (programmable, 19,9)
1	2	e2	paper sensor paper available	-	no paper
2	3	e2	temperature error	-	too hot or cold! (see below)
3	4	e2	head closed	-	open
4	5	e2	no cutter jam	-	cutter jam
5	6	w3	no Rx error	-	parity, overrun or Rx error
6	7	e2	Vp OK	-	Vp too low
7	8	e2	Vp OK	-	Vp too high
8	9	w2	AUX1 / Label occupied	-	empty (programmable 19,2)
9	10	w2	AUX2 occupied	-	empty (programmable 19,3)10
	11	w2	AUX3 occupied	-	empty (programmable 19,4)
11	12	w2	AUX4 occupied	-	empty (programmable 19,5)
12	13	w2	AUX5 occupied	-	empty (programmable 19,6)
13	14	w2	AUX6 occupied	-	empty (programmable 19,7)
14	15	w2	AUX7 occupied	-	empty (programmable 19,8)
15	16		physical paper present	-	physical paper out just info, no print stop
16	17		ticket removal timeout		
17	18		ticket in picking position		
18	19		ticket jam (removal error)		
19	20		ticket taken		
20	21		removal sensor active		
21	22			TBD: actually 0	
22	23			TBD: actually 0	
23	24			TBD: actually 0	
24	25		cancel ticket in progress		
25	26			TBD: actually 0	
26	27			TBD: actually 0	
27	28			TBD: actually 0	
28	29			TBD: actually 0	
29	30			TBD: actually 0	
30	31			TBD: actually 0	
31	32			TBD: actually 0	

2. Status word

Bit	Current No.	LED	Status	0	1
0	33		black mark detected		
1	34		Top of Form reached		
2	35		End of ticket command in progress		
3	36		Error treatment in progress		
4	37		ticket truncated		
5	38		start of a ticket segment		
6	39		autoload in progress		
7	40		autoload fail		
8	41		syntax error occured (event)		
9	42		internal system error (event)		
10	43	e2	power fail OK	-	Power Fail (OPTION GCS 6900)
11	44		power on reset		
12	45		failed system shutdown via deactivation of VCC_enable		
13	46			TBD: actually 0	
14	47			TBD: actually 0	
15	48			TBD: actually 0	
16	49		system	system OK	system in emergency operation
17	50		power-Off imminent		
18	51		fast charging		
19	52		concervation charging		
20	*53*		*suspend (intended)*		
21	*54*		*resume (intended)*		
22	55		battery temperature	-	battery too hot > +50°C
23	56		battery temperature	-	battery too cold < -10°C (not activated)
24	57		TEST Key	not pressed	pressed
25	58		printer speed	accelerating	constant printer speed
26	59		head preheating		in progress = 1
27	60		head preheat temperature		reached = 1
28	61	e2	printhead temperature	-	printhead too cold
29	62	e2	printhead temperature	-	printhead too hot
30	63	e2	motor temperature	-	motor too cold
31	64	e2	motor temperature	-	motor too hot

3. Status word

Bit	Current No.	LED	Status	0	1
0	65			TBD: actually 0	
1	66			TBD: actually 0	
2	67			TBD: actually 0	
3	68			TBD: actually 0	
4	69			TBD: actually 0	
5	70			TBD: actually 0	
6	71			TBD: actually 0	
7	72			TBD: actually 0	
8	73			TBD: actually 0	
9	74			TBD: actually 0	
10	75			TBD: actually 0	
11	76			TBD: actually 0	
12	77			TBD: actually 0	
13	78			TBD: actually 0	
14	79			TBD: actually 0	
15	80			TBD: actually 0	
16	81			TBD: actually 0	
17	82			TBD: actually 0	
18	83			TBD: actually 0	
19	84			TBD: actually 0	
20	85			TBD: actually 0	
21	86			TBD: actually 0	
22	87			TBD: actually 0	
23	88			TBD: actually 0	
24	89			TBD: actually 0	
25	90			TBD: actually 0	
26	91			TBD: actually 0	
27	92			TBD: actually 0	
28	93			TBD: actually 0	
29	94			TBD: actually 0	
30	95			TBD: actually 0	
31	96			TBD: actually 0	

4. Status word

Bit	Current No.	LED	Status	0	1
0	97			TBD: actually 0	
1	98			TBD: actually 0	
2	99			TBD: actually 0	
3	100			TBD: actually 0	
4	101			TBD: actually 0	
5	102			TBD: actually 0	
6	103			TBD: actually 0	
7	104			TBD: actually 0	
8	105			TBD: actually 0	
9	106			TBD: actually 0	
10	107			TBD: actually 0	
11	108			TBD: actually 0	
12	109			TBD: actually 0	
13	110			TBD: actually 0	
14	111			TBD: actually 0	
15	112			TBD: actually 0	
16	113			TBD: actually 0	
17	114			TBD: actually 0	
18	115			TBD: actually 0	
19	116			TBD: actually 0	
20	117			TBD: actually 0	
21	118			TBD: actually 0	
22	119			TBD: actually 0	
23	120			TBD: actually 0	
24	121			TBD: actually 0	
25	122			TBD: actually 0	
26	123			TBD: actually 0	
27	124			TBD: actually 0	
28	125			TBD: actually 0	
29	126			TBD: actually 0	
30	127			TBD: actually 0	
31	128			TBD: actually 0	

17 LINE MODE GeBE A8 STANDARD COMMAND SET

All A8 commands explained in this manual are summed up as follows:

Command ASCII	Function	Synchron	Cancel
<CR>	Start printing with line feed	Syn/Asyn	no
<FF>	Feed to the next label mark		
<LF>	Start printing with line feed		
<ESC> "A"	Erasing data in the printer buffer		
<ESC> "a"	Automatic status transfer		
<ESC> "b"	Stop automatic status transfer/ reset printer parameters to default		
<ESC> "C" n	Cut off paper		
<ESC> "c" n	Print bar code		
<ESC> "D" n	Print in text mode / data mode		
<ESC> "d" n	Initialize printer		
<ESC> “e” n m	End of Ticket		
<ESC> "F" nh nl	Paper feed 1-2400 lines (300 mm max.)		
<ESC> "g" n g1....gn	Pixel graphics PCL5, print graphics line with n byte length		
<ESC> "H" n	Text zoom vertical, character height of		
<ESC> "I" n	Print in black and white/white in black		
<ESC> "i"	Positioning in one line (left / right, centered)		
<ESC> "J" n	Bold on /off		
<ESC> "k"	Return actual status		
<ESC> "L" n	Print with and without underlining		
<ESC> "m" n	Set graphics mode		
<ESC> "N" n	Character tab Character superposition		
<ESC> "n" n, m, ...	Character superposition		
<ESC> "P" n	Select character set no. n		
<ESC> "Q"	<Parameter> <Value> Setup printer, parameter addresses **(realized are 3,13,14,16,20,22)**		
<ESC> "q"	Set Top of Form		
<ESC> "R" n	Relative character tab		
<ESC> "S" n	Set character spacing		
<ESC> "s" n	Set line spacing (leading)		
<ESC> "T“ n	Call up batch data		
<ESC> "v" n	Sync command		
<ESC> "W" n	Text zoom horizontal		
<ESC> "@"	Initializing printer, reset		
<ESC> "\" nh nl	Paper backfeed for 1-2400 lines (300 mm max.)		
<ESC> "/" nh nl	Locked paper backfeed for 1-2400 lines		

New commands:

<ESC> “#” n	Printout of the number n logo
<ESC>“o“	Set position counter
<ESC> "V"	Save printer setup
<ESC> "f" n,m,...,x	Feed zum Sensor

The following ESC sequences are actually not emulated:

<ESC> "B3GO_BOOT“ <ESC> “BS“ <ESC> “B9“		
<ESC> "x" n	Requesting the system parameters	
<ESC> "x" <...d>	Reading position counter	1d, 12d, 0d

17.1 FORMATTING COMMANDS

Word wrap, start printing. Characters which do not fit in a line, trigger the printing of the line. The selected font and the real printing width determine the number of characters per line.

17.1.1 WORD WRAP

[Name]	Carriage Return	
[Format]	ASCII	<CR>
	Hex	0Dh
	Decimal	13d
[Description]	Line feed (if last character <> LF)	
	Printing action with line feed. An immediately following <LF> is ignored.	

17.1.2 LINE FEED

[Name]	Line Feed	
[Format]	ASCII	<LF>
	Hex	0Ah
	Decimal	10d
[Description]	When the printer controller receives this byte, the text data in the buffer are printed.	
	Line feed (if last character <> CR)	
	Printing action with line feed. An immediately following <CR> is ignored.	

17.1.3 FORM FEED

[Name]	Form Feed	
[Format]	ASCII	<FF>
	Hex	0Ch
	Decimal	12d
[Description]	Feed to the next label mark, if label mode is active. Without label mode <FF> produces a line feed by the set number of dotlines according to parameter x,y.	

17.1.4 PAPER FEED FORWARDS

[Name]	Feed Paper				
[Format]	ASCII	<ESC>	"F"	nh	nl
	Hex	1Bh	46h	nh	nl
	Decimal	27d	70d	nh	nl

[Description] Paper feed by l lines: = (l_h x 256 + l_l) lines.
This command is only possible at the beginning of a line. Elsewhere it is ignored.
Transport is limited to 2400 dotlines per command (with 8 dots./mm = 300 mm).

[Name]	Feed Paper to Sensor			
[Format]	ASCII	<ESC>	"f"	<lH> <lL> <bDir> <sSensorsH> <sSensorsL> <sConditionsH> <sConditionsL>
	Hex	1Bh	66h	<lH> <lL> <bDir> <sSensorsH> <sSensorsL> <sConditionsH> <sConditionsL>
	Dezimal	27d	102d	<lH> <lL> <bDir> <sSensorsH> <sSensorsL> <sConditionsH> <sConditionsL>

[Description] Paper feed by l lines: = (l_h x 256 + l_l) lines.
This command feeds until a sensor condition is reached but max. (l_h x 256 + l_l) lines.

lH	feed length high byte
lL	feed length low byte
bDir	backward = ffhex / forward = 01hex
sSensorsH	
sSensorsL	Bit 12 – 15 = 0 Bit 11 : AUX7 Bit 10 : AUX6 Bit 9 : AUX5 Bit 8 : ----- Bit 7 : HEAD UP Bit 6 : ----- Bit 5 : PE Bit 4 : AUX4 Bit 3 : AUX3 Bit 2 : AUX2 Bit 1 : AUX1 Bit 0 : NPE
sConditionsH	corresponding bits to „sSensors"
sConditionsL	0 := Stop if sensor does not detect paper 1 := Stop when sensor detects paper

17.1.5 PAPER FEED BACKWARDS

[Name]	Feed Paper Backwards				
[Format]	ASCII	<ESC>	"\"	nh	nl
	Hex	1Bh	5Ch	nh	nl
	Decimal	27d	92d	nh	nl

[Description] Paper feed backwards by $l = l_h \times 256 + l_l$ lines. This command is only possible at the beginning of a line. Elsewhere it is ignored. Transport is limited to 2400 dotlines per command (with 8 dot/mm = 300 mm).

The paper must not be moved too far back. As soon as the paper loses contact with the feed roller it cannot be moved forward again.

17.1.6 LOCKED BACKFEED COMMAND

[Name]	Backfeed to TOF				
[Format]	ASCII	<ESC>	"/"	nh	nl
	Hex	1Bh	2Fh	nh	nl
	Decimal	27d	47d	nh	nl

[Description] Return to Top of Form position.

17.2 FORMATTING FONT

The printer allows two kinds of font size settings.

1. **Conventional setting: Zooming height and width**
   This kind of setting doubles width and/or height (compatible with A8 system). The zooming can be more exactly determined with the enlarged command "<FFh>". Here the width or height of the actual font can be enlarged or reduced almost open end. The zooming factor is described as a fraction with a multiplier and a divisor.
   Example: with a multiplier 7 and a divisor 4 the zoom can be set to 1.75. According to the printer resolution this leads to the next smaller pixel setting. For a 15 pixel high font this means 15 x 1.75 = 26.25, which is a height of 26 pixels.

2. **Setting via font size**
   The printer can indicate the font size. Opposite to the conventional setting this setting is independent of the printer resolution. The spezified font size applies from the first setting globally for all fonts until it is changed. The font size is given in number of dots in height. One dot means 1/72 dpi. So one dot is 25.4 mm / 72 = 0.352777 mm high.

17.2.1 SELECTING FONT

[Name]	Selecting font			
[Format]	ASCII	<ESC>	"P"	n
	Hex	1Bh	50h	n
	Decimal	27d	80d	n

[Description] The font number n is a binary value and starts with zero (first font value in the system is font zero) The command change font is effective from the beginning of the line where it is issued. It is possible to mix several fonts within one line. Ex works the controller has a preset 8x16 pixel system font with index 0 and several fonts in the AddOn range.

17.2.2 SELECTING FONT WITH SIZE SETTING

[Name]	Selecting font					
[Format]	ASCII	<ESC>	"P"	<ffh>	m	n
	Hex	1Bh	70h	ffh	m	n
	Decimal	27d	112d	255d	m	n

[Description] The font number m is a binary value and starts with zero (first font value in the system is font zero).
The command change font is effective from the beginning of the line where it is issued. It is possible to mix several fonts within one line. Ex works the controller has a preset 8x16 pixel system font with index 0 and several fonts in the AddOn range.
n : is the information for the font size.
When selecting n : =0 no font is set. The command acts like <ESC>"P"n. The font size is only effective for the selected font until it is replaced by another one or cancelled with n: = 0.
The commands <ESC> "H" or <ESC> "W" have a cumulative effect on the font size command
m = 0 .... 255 Font selection
n = 1 .... 255 Font size (limited by additional basic scaling)

[Example] select font 5 with font size 16 dots: <ESC>P<255d><5d><16d>

17.2.3 TEXT ZOOM HORIZONTALLY FIX

[Name]	Text Zoom horizontally			
[Format]	ASCII	<ESC>	"W"	(Zoom)
	Hex	1Bh	57h	n
	Decimal	27d	87d	n

[Description] This command is effective from the letter after which it is given. Possible width:
n:= 0: single 1: double 2: fourfold 3: eightfold 4: 16fold

17.2.4 TEXT ZOOM HORIZONTALLY VARIABLE

[Name]	Text Zoom horizontally					
[Format]	ASCII	<ESC>	"W"	<ffh>	Multiplier	Divisor
	Hex	1Bh	57h	ffh	m	n
	Decimal	27d	87d	255d	m	n

[Description] With this command the width or height of the actual font can be enlarged or reduced almost open end. The zooming factor is described as a fraction: This command is effective from the letter after which it was given. Multiplier : m = 1 ... 255, divisor : n = 1 ... 255.

[Example] Reduction to factor 0,66: (= 2/3) <ESC>W<255d><2d><3d>

[Example] Print threefold width <ESC>W<255d><3d><1d>

17.2.5 TEXT ZOOM VERTICALLY FIX

[Name]	Text Zoom vertically			
[Format]	ASCII	<ESC>	"H"	(Zoom)
	Hex	1Bh	48h	n
	Decimal	27d	72d	n

[Description] This command is effective from the line where it was given and is valid for the whole line. It is not possible to mix different hights within one line.
Available heights: n:= 0: single 1: double 2: Threefold up to 7: eightfold

17.2.6 TEXT ZOOM VERTICALLY VARIABLE

[Name]	Text Zoom horizontally					
[Format]	ASCII	<ESC>	"H"	<ffh>	Multiplier	Divisor
	Hex	1Bh	48h	ffh	m	n
	Decimal	27d	72d	255d	m	n
[Description]	With this command the width or height of the actual font can be enlarged or reduced almost open end. The zooming factor is described as a fraction: This command is effective from the letter after which it was given. Multiplier : m = 1 ... 255, divisor : n = 1 ... 255.					
[Example]	Enlarging to factor 2.5: <ESC>H<255d><5d><2d>					

17.2.7 CHANGING TRACKING (spacing) AND LEADING (interline spacing)

Contrary to the requirements of the font, tracking may be increased (positive values) or decreased (negative values), e.g. for ligatures. However the leading may only be increased (positive values). The number of dots is stated which indicate by how much the spacing has to be changed based on the unscaled font. This means, that scaling a text also changes the spacing accordingly. When enlarging a text, additional white space is inserted towards the right neighbor character or the succession line. In case of a reduction characters are moved closer together. This may lead to an overlap of characters. Depending on the setting of the frame attributes the left character sign is partly obscured.

[Name]	Changing tracking (spacing)			
[Format]	ASCII	<ESC>	"S"	n
	Hex	1Bh	53h	n
	Decimal	27d	83d	n
[Description]	This command is effective from the point where it was given. Possible tracking values: n:= 0 ... 127 positive values n:= 255 ... negative values until max. width of the previous character			

[Name]	Changing leading			
[Format]	ASCII	<ESC>	"s"	n
	Hex	1Bh	73h	n
	Decimal	27d	115d	n
[Description]	This command is effective from the point where it was given. Possible leading values: n:= 0 ... 127 positive values			

17.2.8 SETTING OPTICAL DENSITY

[Name]	Density on/off			
[Format]	ASCII	<ESC>	"Y"	n
	Hex	1Bh	59h	n
	Decimal	27d	89d	n
[Description]	This parameter can be set between 0 and 50. High values imply high density. Typical default value is: 25.			

17.2.9 BOLD ON/OFF

[Name]	Bold on/off			
[Format]	ASCII	<ESC>	“J“	n
	Hex	1Bh	4Ah	n
	Decimal	27d	74d	n
[Description]	This command is effective from the letter after which it is given. The character's graphics data are shifted to the right and “OR” linked with the original. n := 0 bold off, n := 1 bold on			

17.2.10 UNDERLINE ON/OFF

[Name]	Underline on/off			
[Format]	ASCII	<ESC>	“L“	n
	Hex	1Bh	4Ch	n
	Decimal	27h	76d	n
[Description]	This command is effective from the letter, after which it is given until it is cancelled.			

17.2.11 REVERSE

[Name]	Reverse on/off			
[Format]	ASCII	<ESC>	“I“	n
	Hex	1Bh	49h	n
	Decimal	27d	73d	n
[Description]	Printing black/grey on white. This command is effective from the letter, after which it is given until it is cancelled. Action: 0:= normal printing, 1:= white on black			

17.2.12 TEXT ORIENTATION

[Name]	Text orientation			
[Format]	ASCII	<ESC>	“D“	n
	Hex	1Bh	44h	n
	Decimal	27d	68d	n
[Description]	Reverse printing in text mode (n = "0") or in data mode (n = "1"). In data mode the text is turned in an angle of 180° in order to make it readable when the paper strip is hanging down. Thus the chronology of the print lines appears upside down. This command has no effect on graphics. It can be given at any position within a line, as long as the line is not yet terminated and it affects the whole line. This command is effective until it is cancelled.			

17.2.13 TEXT POSITIONING

[Name]	Text positioning (left/right or centered)			
[Format]	ASCII	<ESC>	“i“	n
	Hex	1Bh	69	n
	Decimal	27d	105	n
[Description]	Print text left-aligned (n = 0 or '0', 'l' or 'L'), centered (n = 1 or '1', 'c' or 'C') or right-aligned (n = 2 or '2', 'r' or 'R'). This command applies until a new parameter is called or a command is issued.			

17.2.14 ABSOLUTE TAB

[Name]	Absolute tab				
[Format]	ASCII	<ESC>	"N"	n	
	Hex	1B	48	n	
	Decimal	12	78	n	
[Description]	The cursor is offset by n mm. Absolutely moving forward from the left border to the horizontal millimeter position; (at 200dpi := 8 pixel, at 300 dpi 12 pixel). This command allows millimeter-accurate positioning to a print start position within a line. If the desired positioning exceeds the available range of a line, the command is ignored. The print attributes are not changed.				
[Example]	A second text column should start after 20 mm. Command: Hello<ESC>N<20d>World.				

[Format]	ASCII	<ESC>	"N"	<255d>	n1	n2
	Hex	1B	48	<255d>	n1	n2
	Decimal	12	78	<255d>	n1	n2
[Description]	Absolutely moving forward from left to the horizontal pixel position n = 256 x n_1 + n_2 This command allows pixel-accurate positioning to a print start position within a line. If the desired positioning exceeds the available range of a line, the command is ignored. The print attributes are not changed.					
[Example]	A second text column should start after 135 pixel. Command: Hello<ESC>N<255d><0d><135d>World.					

17.2.15 RELATIVE TAB

[Name]	Relative tab				
[Format]	ASCII	<ESC>	"R"	nh	nl
	Hex	1B	48	nh	nl
	Decimal	12	78	nh	nl
[Description]	Relatively moving forwards/backwards by pixel amount p = 256 x n + m n/m is determined as an integer number with key signature, as follows: nhl:= .... FFFD FFFE FFFF 0000 0001 0002 0003 ... p := -3 -2 -1 0 +1 +2 +3 ... If the desired positioning exceeds the available range of a line, the command is ignored. When moving backwards, the print attributes are not changed.				

17.2.16 CHARACTER SUPERPOSITIONING

[Name]	Tab back				
[Format]	ASCII	<ESC>	"n"	<number>	character
	Hex	1B	6E	<number>	character
	Decimal	12	110	<number>	character
[Description]	Multiple characters of the font are printed on top of each other, at the same position. The first byte specifies the number of characters to be printed and the second the characters.				
[Example]	Printing the character ‚Ø'. Command: <ESC>n<2d>O/				

This command requires a non-proportional character set

17.3 BATCH FILES/MACROS

Print data may be filed in the memory of the AddOns. A request with the command ESC T <Batch no> works as if the data of this batch file were sent to the printer via interface. Therefore batch files are able to contain texts, graphics and commands.

[Name]	call batch file			
[Format]	ASCII	<ESC>	"T"	n
	Hex	1Bh	54h	n
	Decimal	27d	84d	n

[Description] Issues the contents <Batch no> to the printer. Batch no is a binary value and starts with zero. Batch files may be "stacked", that means, if one batch file is requested within another batch file, operation goes back to where it was, after this batch file is ended.

The following batch files are linked to actions and are executed:

T0	if the FEED key remains pressed during a hardware reset.
T1	if the TEST key is pressed for minimum 30 msec and is released afterwards.
T15	if the TEST key is pressed for minimum 2 sec and is released afterwards (at GeBE-FLASH = Power OFF).
RESET	if the TEST key is pressed for minimum 12 sec and is released afterwards.
T2	*at the end of an autoload, but only if an autoload batch file was configured.*
T4	*if the FEED and T1 TEST key is pressed simultaneously, but only with an implemented mudule menu.*
T16	on power on reset, but only if a text conserve is activated.

17.4 GRAPHICS COMMANDS

17.4.1 PRINTING FILED LOGO

[Name]	Print Logo number n			
[Format]	ASCII	<ESC>	"#"	n
	Hex	1Bh	23h	n
	Decimal	27d	35d	n
[Description]	The filed AddOn Logo with the number n will be printed at actual cursor position. The left offset can be set with the command <ESC>m<4d><Offset in mm>. The reference number of the logo can be taken from the A8 logo patch chart.			

17.4.2 GEBE GRAPHICS MODE

Concerning their data organization the graphics data of these modes correspond with the commands of the PCL specification from version 3. They are compatible with the MS Windows compression procedure. Executing the compressed data is almost as fast as mere bitmap printing. Because of the smaller amount of data to be transferred, RS232 operation is much faster than the method without compression (about 1:3).

[Name]	Graphics mode - compressed	
[Format]	ASCII <ESC> "m" (Mode) (Parm)	
[Description]	Mode = 0: Graphics mode uncompressed	(without Parm)
	Mode = 1: Graphics mode Run length	(without Parm)
	Mode = 2: Graphics mode Tiff	(without Parm)
	Mode = 3: Graphics mode Delta-Row	(without Parm)
	Mode = 4: set graphics offset left:	
	Parm = 0: no left offset	
	Parm = 1: left offset	(x8 in pixels)
	Mode = 5: Erasing the Delta-Row-Line	(Seed Row/predecessor line) (without Parm)
	Mode = 6: Set graphics height	
	Parm = 0: Height = single	
	Parm = 1: Height = double …	

17.4.3 GEBE GRAPHICS DATA COMPRESSED

[Name] Graphics data – compressed
[Format] ASCII <ESC> “g“ n {graphics data}
[Description] The graphics data are processed according to the kind of compression set to <ESC> “m“.

0 : Unencoded
n: = graphics length in byte, g1 ...gn = graphics bytes to be printed
In text mode, starting from left to right, the dot 0 is the MSB in the first byte. The dot furthest to the right is the LSB in the nth byte. A 1 in the respective bit position means a black dot on the line. The printer automatically returns to the character mode after the nth byte. During these n bytes the printer ignores all commands.
[Example] (for a printer mechanism with a paper width of 80 mm = 640 dots).
Printing a dotted line: <ESC>m<0d> (necessary only once)
<ESC>g<80d><176d><176d>…….<176> (sending the Byte <176d>)

1 : Run length Encoded
n : = Number of following bytes
Run length interprets graphics information in pairs of bytes. Each first byte is the Repetition Count Byte for each second byte. A 0 in the Repetition Count Byte means, the following graphics byte will be printed once and not repeated. A 1 means, the graphics byte will be printed twice. The Repetition Count Byte has a range of values from 0 to 255, i.e. a printing parameter of 1 to 256. The second byte contains the graphics information to be printed. In text mode from left to right the dot furthest to the right is the LSbit. A 1 in the respective bit position means a black dot on the line. After terminating the line the printer automatically returns to character mode.
[Example] (for a printer mechanism with a paper width of 80 mm = 640 dots)
Printing a dotted line: <ESC>m<1d> (necessary only once) <ESC>g<2d><79d><170d>

2 : TIFF (4.0) Encoded
n : = Length of the following bytes
TIFF interprets graphics information as TIFF "packbits". It combines features of Unencoded and Run length Encoding. The graphics information is preceded by a control byte. The control byte indicates (prefix byte), whether the following byte is a graphics byte, which ought to be repeated (up to 127 times), or whether a number of bytes follow (up to 127), that ought to be printed as bitmap. A positive control byte requires bitmap information, a negative control byte (two's complement) requires a repeat byte.
[Example] (for a printer mechanism with a paper width of 80 mm = 640 dots)
Printing a dotted line: <ESC>m<2d> (necessary only once) <ESC>g<2d><176d><170d>

3 : Delta Row
n : = Length of the following graphics bytes
Delta Row selects those bytes of a line that differ from the preceding line, and it transfers only these differences. If only one bit is different, only the respective byte has to be transferred. The Delta data consist of one command byte and one to eight replacement bytes. The command byte contains two pieces of information. The number of replacement bytes (bit 7, 6, and 5) and the relative left offset of the last byte changed (bit 4, 3, 2, 1, and 0). If the offset is 31 an additional offset byte has to follow. The value 255 of this additional offset byte needs a further offset byte. The offset valuess are added. In text mode from left to right the dot the farthest to the right is the LSB. A 1 in the respective bit position of a replacement byte represents a black dot on the line. After termination of the line the printer automatically returns to character mode. During this line the printer ignores all commands. With Delta Row it is impossible to mix text and graphics.

17.4.4 PRINTING BARCODE

[Name] Print barcode
[Format] ASCII ESC c <barcode type> <height> <width> <leftmost position> <barcode data>
[Description] Barcode printout

<barcode type>

'B' : Code39 with clear text
'b' : Code39 without clear text
'C' : Code128 with clear text (Subset A,B,C automatic)
'c' : Code128 without clear text (Subset A,B,C automatic)
'D' : EAN-13 with clear text
'd' : EAN-13 without clear text
'I' : 2 aus 5 interleaved
'i' : 2 aus 5 interleaved without clear text
'U' : UPC-A with clear text
'u' : UPC-A without clear text
'v' : EAN-8 with clear text
'V' : EAN-8 without clear text
'q' : QR code with clear text
'Q' : QR code without clear text

<Height> Barcode height in dotlines (1 - 255 dotlines)
Extension of the barcode height:
If the previous parameter has the value 1, then 2 subsequent additional values, HighByte and LowByte determine the barcode height.

<Width> Width of an element (smallest line width) in dots = 0: default width

<Leftmost position> left barcode offset from the leftmost printing position in mm

<Barcode data> *A white surface is "printed" instead of a barcode, if the rightmost position of the line is exceeded. Nevertheless an inserted clear text will be printed*
The barcode will be ignored:
• if a wrong type or an unknown size are indicated
• if characters that do not match the character set of the code have been entered

Code39: Data start with a * and end with a *
Character set: 1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ$/-.+% <Space>
[Command example] Code39 with clear text; 123ABC print: ESC>cB<80d><1d><10d>*123ABC*

2 of 5 Interleaved: Data are closed with FFHex. The number of characters has to be even.
Character set: 0123456789
After indication of the leftmost position an optional byte follows
<option>: Bit#0 == 1: with check digit
Bit#0 == 0: without check digit
[Command example] 2 of 5 Interleaved with clear text; print 123456 and check digit:
<ESC>cI<80d><1d><10d><1d>123456<FFh>

EAN13: Data always consist of 12 characters. The checksum, which is the 13th character, is calculated and attached by the printer itself.
Character set: 0123456789
[Command example] EAN13 with clear text; print 123456789012: <ESC>cD<80d><1d><10d>123456789012

Code128: Data are completed with FFHex. The number of characters has to be even.
Character set: 0123456789
[Command example] Code128c with clear text; print 123456: <ESC>cC<80d><1d><10d>123456<FFh>

UPC-A: Data always consist of 11 characters with an "0" as prefix. The checksum, which is the 13th character, is calculated and attached by the printer itself.
Character set: 0123456789
[Command example] UPC-A with clear text; print 012345678901: <ESC>cU<80d><1d><10d>12345678901
Output in clear text : 0123456789012 2 = checksum

17.4.4.1 SELECT THE VERTICAL ALIGNMENTS

BARCODE COMMAND SEQUENCE

[Syntax] <ESC> 'c' 0xFE + alignment preset (1 byte)
alignment preset, according to GeBE_A8 command ESC 'i':
0x00, '0', 'L', 'l': left justified
0x01, '1', 'C', 'c': centered
0x02, '2', 'R', 'r': right justified
L (left justified): the object is positioned on the leftmost print area and possibly shifted to the right by the indentation specified for the frame
C (centered): the object is positioned in the middle of the print area
the indentation specified for the frame is not relevant
R (right-justified): the object is positioned on the rightmost print area and, if necessary, shifted to the left by the indentation specified for the frame

[System parameter]

A8:	[15,20]	RW_ASAP: default barcode alignment
N78:	[16,20]	RW_ASAP: default barcode alignment
GA_46:	[56,20]	RW_ASAP: default barcode alignment

17.4.4.2 SELECT ROTATION

BARCODE COMMAND SEQUENCE

[Syntax] <ESC> 'c' 0xFD + rotation specification (1 byte)
0x00, '0', 'L', 'l': left -> no rotation to the right
0x01, '1', 'T', 't': top -> rotate bottom by 90° (landscape)
0x02, '2', 'R', 'r': right -> 180° rotation to the left *not yet implemented*
0x03, '3', 'B', 'b': down -> rotate top by 270° (landscape + rotation by 180°) *not yet implemented*

[System parameter]

A8:	[15,19]	RW_ASAP: default barcode rotation
N78:	[16,19]	RW_ASAP: default barcode rotation
GA_46:	[56,19]	RW_ASAP: default barcode rotation

[Rotation value range]
0x00 to 0x01 according to GeBE_A8 (prepared for later extention 0x00 to 0x03)

17.4.5 PRINTING QR CODE

[Name] Print QR code

[Format] <ESC> 'c' 'Q' <version> <ECC> <width> <borderleft> <lenghtH> <lenghtL>: with clear text

<ESC> 'c' 'q' <version> <ECC> <width> <borderleft> <lenghtH> <lenghtL>: without clear text

[Description]

<version>	= version number (<0d>: automatically selection of minimum size , <1d>: fix version 1 , ...)
<ECC>	= Error Correction Level: label letter 'L' (7%), 'M' (15%), 'Q' (25%) and 'H' (30%)
<width>	= width of an element in dots
<borderleft>	= barcode offset left in mm from the left print line border
<lenghtH>	= number of barcode data bytes (high)
<lenghtL>	= number of barcode data bytes (low)

[Example] <ESC>cq<0d>M<10d><0d><0d><20d>12345678901234567890

In principle, all QR code versions, ie up to version 40 (177x177) are supported. The expression of large QR codes is suppressed if the size of the renderer workspace parameter 17,3 is too small. A reduction of parameter 17,3 can reduce the maximum print speed. With the current firmware parameters, only QR codes up to version 7 (45x45) can be printed.

17.4.6 CUTTING PAPER

[Name]	Cut Paper				
[Format]	ASCII	<ESC>	"C"	n	m (only with n=4)
	Hex	1Bh	43h	n	m (only with n=4)
	Decimal	27d	67d	n	m (only with n=4)

[Description] **n = 0 : Full Cut**

The paper is totally cut.

[Example] <ESC>C<0d>

n = 1 : Half cut or segment cut (depending on hardware)

The cut is effected in a way to leave a small remainder of paper.

[Example] <ESC>C<1d>

n = 2: Initialize Cutter

After each reset the system uses this command when the cutter flag bit 0 or 3 is set. The controller then makes sure that a cutter is situated in home position. If this is not the case the cutter is moved there. In case not cutter is equipped the cutter flags 0 and 3 must be set to 0. If a cutter exists, but is not at the home position, not even after about 2 seconds, the "paper jam" error bit is set and the printing is stopped. The cutter error can be reset by pressing the feed key or by sending a cut command (ESC C0/C1/C2/C3/C4) to the printer. For this second choice the input buffer of the printer has to be empty. That means that no further data are allowed to be sent to the printer after the cut error occurred.

n = 4: Retaining paper while ripping off

! This command needs an additional parameter (m) !

To avoid pulling the perforated paper off the printer after cutting, the motor can be powered for a set time after cutting (without feeding / stepping). If a new command is given during this time, it will be executed immediately and the remaining power on time will be cleared. The command is not executed if the motor temperature is above 75°C (167°F). If the maximum allowed motor temperature of about 90°C (194°F) is reached, the powering is stopped. The power on time is specified in seconds with additional parameters and varies from 1 to 25 seconds. Parameter 0 ignores the command.

Please note: The motor driver heats up. Therefore 25 seconds are the maximum allowed time.
For longer times at temperature please allow for cooling off time. Please request our advise.

[Example] Hold paper for 20 seconds after cutting: <ESC>C<4d><20d>

17.4.7 MODULE 22: CUTTER

Cutter Speed (depending on hardware)

22,1: e_a Maximum cutter motor speed [pps]

Cutting speed of stepper motor cutters.

The torque of stepper motors is highly depending on the revolution speed. Such a cutter is able to cut thicker paper when the cutting speed is lower. In this case the cutting speed is preset to the lowest speed. With parameter 22,1 the cutter increases in a uniform acceleration up to this maximum speed. When pulling back the speed always goes up to maximum. Depending on the paper thickness the following values for parameter 22,1 are recommended by the manufacturer:

Paper thickness	Printer mechanism HSP 2208	Printer mechanism HSP3208/3512	Printer mechanism FTP63A
>180 µm	300	400	--
>160 µm	300	400	--
>140 µm	300	400	800
>120 µm	500	600	800
>100 µm	700	800	800
> 80 µm	900	1000	1500
> 60 µm	1100	1200	2000

Setting width of segment cut (depending on hardware)

22,2: e_a Half cut: number of pulses

Adjusting cutter end position for stepper cutter

The width of the partial cut can be set with parameter 22,2. This indicates how far the cutter cuts into the paper. Caused by tolerances in paper width and paper guide the remaining segment may vary up to 2 mm.

Printer mechanism HSP 2500	Printer mechanism HSP3500	Printer mechanism FTP63A
?	130	217

22,3: u_w distance from dotline to cut [dotlines]

22,4: u_a TRUE: always move cutter blade to home position on init command

FALSE: move blade only when out of home position

Search for home position:

If the value is TRUE (=1), the home position is searched for actively during a reset or ESC C2. That means, during initialization the cutter blade moves back and forth rapidly.

If the value is FALSE (=0) the cutter automatically looks for the status of the home switch after a reset or ESC C2. When it is correct, the cutter does not move. When it is not correct a cut is effected.

17.5 ENDING A TICKET WITH THE "END OF TICKET" COMMAND

First a ticket is ended with a form-feed command. Considering parameter 19 the printer is positioned to the reference mark of the new ticket. The command "End of ticket" can indicate the method of ticket separation:

1. Feed to a tear-off edge
2. Feed - cut - back positioning
3 Feed - double cut (remove perforation) - back positioning
4 Eject single ticket

17.5.1 END OF TICKET (EoT)

[Name]	EoT				
[Format]	ASCII	<ESC>	"e"	(Param a)	(Param b)
	Hex	1Bh	65h	(Param a)	(Param b)
	Decimal	27d	101d	(Param a)	(Param b)

[Description] With the parameter flags a specific „End of ticket" action can directly be requested.
In parameter module 24 these actions can be permanently set.

Param a bit 0:	EoT1 identifier =0: for EEPROM param/=1: for parameter transfer in command
Param a bit 1:	(24,2) EoT_NoCut: 0 = cut / 1 = no cut
Param a bit 2:	(24,4) EoT_NoFullCut = 0: full cut = 1: half cut or segment cut
Param a bit 3:	(24,3) EoT_DoubleCut = 0: no double cut = 1: double cut with parameter 24,3
Param a bit 4:	= 0: allocated
Param a bit 5:	(24,5) EoT_Cut_NoBackfeed = 0: backfeed after cut with parameter 22,3
Param a bit 6:	= 0: allocated
Param a bit 7:	= 0: allocated
Param b bit 0-7:	= 0: allocated

Modul 24 EoT

24,1:	u_w	Output equipment 0: Cutting edge / 1: Cutter
24,2:	u_a	1: enable cuts / 0: disable cuts
24,3:	u_a	distance between cuts [2 dotlines] for double cut
24,4:	u_a	1: partial cut / 0: full cut
24,5:	u_a	1: enable feed after cut / 0: disable feed after cut
24,6:	u_w	Picking timeout [seconds] / 0: no timeout
24,7:	u_a	TRUE: 1st cut only on Paper End condition / FALSE: Inhibit cuts
24,8:	u_a	eject length [dmm] / 0: do not eject paper on PaperEnd condition

17.6 INITIALIZE PRINTER

Initialize printer buffer

[Name]	Initialize Printer Buffer		
[Format]	ASCII	<ESC>	“A“
	Hex	1Bh	41h
	Decimal	27d	65d
[Description]	When the printer controller receives this command the printer buffer is emptied. Erasing of (not yet printed) data in the line buffer. **Corresponds to command <RS><DLE><05d>**		

Initialize Printer

[Name]	Initialize Printer		
[Format]	ASCII	<ESC>	“@“
	Hex	1Bh	40h
	Decimal	27h	64h

[Name]	Initialize Printer			
[Format]	ASCII	<ESC>	“d“	0d
	Hex	1Bh	40h	0h
	Decimal	27h	64h	0d
[Description]	When the printer controller receives this command, a hard reset of the printer is initialized. (without initial application of the bootloader) The printer is initialized the same way as after power on. Between reception and processing of this command the data already received in the input buffer have to be processed. No further print data are allowed as long as the end of the reset is not accomplished. This is signalled via the serial interface. Otherwise these data sent during reset would be lost, because the reset sequence erases the input buffer. **Corresponds to command <RS><DLE><00d>**			

17.7 SYNC ORDER

[Name]	Sync Command			
[Format]	ASCII	<ESC>	“v“	n
	Hex	1Bh	76h	n
	Decimal	27d	118d	n

[Description] Print and report synchronized character via the serial interface. The sync character is returned as soon as the printer has completely finished the previous print. Any 8 bit value can be the sync character. It is sent separately.

[Example] <ESC>v<2d>Hello World<ESC>v<3d>
If the printer returns 2 decimal it has received the print job. If it returns 3 decimal the printing of "hello world" is completely finished.

Do not use sync characters above 127 decimal, or 11hex, and 13hex.
These could not be distinguished from A8 status bytes.
Hint: Message in format C32:
[AR:0x0000002E#[0x0002BCF2ms0x00120000: 0x000002A0, 0x00000002 #i_+]#0xC90C9E0C:AR]
Returned: The module 0012: print reports the sync character 02hex

17.8 DEFAULT STATUS MESSAGE VIA FLAGS CALLED A8 STATUS MESSAGE

17.8.1 STATUS MESSAGE REQUEST IN A8 MODE

[Name]	Request status		
[Format]	ASCII	<ESC >	“k“
	Hex	1Bh	6Bh
	Decimal	27d	107d
[Description]	On receiving this command the printer controller sends the status bytes.		

17.8.2 STATUS MESSAGE GENERAL REQUEST

[Name]	Request actual active Status			
[Format]	ASCII	<RS>	“k“	255d
	Hex	1Eh	6Bh	FFh
	Decimal	30d	107d	255d
[Description]	On receiving this command the printer controller sends the set default status.			

[Name]	Request A8 Status			
[Format]	ASCII	<RS>	“k“	02d
	Hex	1Eh	6Bh	02h
	Decimal	30d	107d	02d
[Description]	On receiving this command the printer controller sends the A8 status bytes.			

[Name]	Request C32 Status			
[Format]	ASCII	<RS>	“k“	01d
	Hex	1Eh	6Bh	01h
	Decimal	30d	107d	01d
[Description]	On receiving this command the printer controller sends the C32 status bytes.			

In module 30 Msg_GeBE_A8 the individual status bytes can be enabled and disabled.

30,1: T: enable / F: disable 1st status byte

30,2: T: enable / F: disable 2nd status byte

30,3: T: enable / F: disable 3rd status byte

30,4: T: enable / F: disable 4th status byte

30,5: T: enable / F: disable 5th status byte

17.8.3 ASSIGNING STATUS FLAGS

Bit 7 6 5 4

1 0 x x Status byte 1
1 1 0 0 Status byte 2
1 1 0 1 Status byte 3
1 1 1 0 Status byte 4
1 1 1 1 Statusbyte 5

The bits are defined as follows:

1. Status byte

Bit	LED	Status	0	1
0	on	Near paper end (NPE)	little paper left	paper OK
1	1:1	Paper sensor	paper available	no paper
2	1:1	Temperature	temperature OK	printhead too hot/cold
3	1:1	Head	closed	open
4	1:1	Cutter jam	no error	error
5	on	Rx error	no error	Rx error (possible buffer overflow)
6			*always 0 (identifier)*	
7			*always 1 (identifier)*	

2. Status byte (AUX sensors)

Bit	LED	Status	0	1
0	on	AUX1 or. label mark	occupied	not occupied
1	on	AUX2	occupied	not occupied
2	on	AUX3	occupied	not occupied
3	on	AUX4	occupied	not occupied
4			*always 0 (identifier)*	
5			*always 0 (identifier)*	
6				*always 1 (identifier)*
7				*always 1 (identifier)*

3. Status byte (Presenter)

Bit	LED	Status	0	1	C32 No.
0	on	Timeout	ticket take-off error	ticket drawn back	17
1	on	Ticket in take-off position	error	ticket waiting in take-off position	18
2	1:1	Ticket take-off error	error	error on ticket take-off	19
3			TBD: actually 0		
4				*always 1 (identifier)*	
5			*always 0 (identifier)*		
6				*always 1 (identifier)*	
7				*always 1 (identifier)*	

4. Status byte (Custom1)

Bit	LED	Status	0	1
0	*Via parameter 30,6 any flag of the C32 messages (current no.) can be displayed.*			
1	*Via parameter 30,7 any flag of the C32 messages (current no.) can be displayed.*			
2	*Via parameter 30,8 any flag of the C32 messages (current no.) can be displayed.*			
3	*Via parameter 30,9 any flag of the C32 messages (current no.) can be displayed.*			
4			*always 0 (identifier)*	
5				*always 1 (identifier)*
6				*always 1 (identifier)*
7				*always 1 (identifier)*

5. Status byte (Custom2)

Bit	LED	Status	0	1
0	*Via parameter 30,10 any flag of the C32 messages (current no.) can be displayed.*			
1	*Via parameter 30,11 any flag of the C32 messages (current no.) can be displayed.*			
2	*Via parameter 30,12 any flag of the C32 messages (current no.) can be displayed.*			
3	*Via parameter 30,13 any flag of the C32 messages (current no.) can be displayed.*			
4				*always 1 (identifier)*
5				*always 1 (identifier)*
6				*always 1 (identifier)*
7				*always 1 (identifier)*

17.8.4 READ OUT SYSTEM DATA WITH A8 COMMAND SYNTAX

Compatible with the A8 system, the following system data can be read out with the A8 command syntax:

<ESC> x <1d> <01d> <00d>	returns status byte 1	
<ESC> x <1d> <01d> <01d>	returns status byte 2	
<ESC> x <1d> <02d> <00d>	operation voltage	
<ESC> x <1d> <02d> <01d>	print head temperature	
<ESC> x <1d> <02d> <02d>	motor temperature	
<ESC> x <1d> <02d> <04d>	distance sensor	
<ESC> x <1d> <03d> <Param>		// factory settings
<ESC> x <1d> <04d> <Param>		// EEPROM parameter
<ESC> x <1d> <05d> <01d>	firmware string	
<ESC> x <1d> <05d> <02d>	bootloader string	
<ESC> x <1d> <05d> <03d>	serial number string	
<ESC> x <1d> <06d> <Dummy>	number of existing fonts	
<ESC> x <1d> <07d> <Font-Nr>		// font name
<ESC> x <1d> <08d> <Font-Nr>		// font hight
<ESC> x <1d> <09d> <Font-Nr>		// font footer
<ESC> x <1d> <10d> <Font-Nr>		// font underlined underline
<ESC> x <1d> <11d> <Font-Nr>		// font width (must be 1)
<ESC> x <1d> <12d> <Dummy>		// position counter (in 2 bytes)
<ESC> x <1d> <13d> <Dummy>		// mark length (1 byte)
<ESC> x <1d> <14d> <Param>		// output RAM settings
<ESC> x <1d> <15d> <Wert>	output statistic values	
<ESC> x <1d> <16d> <00d>	output saved CRC e.g.: „36601“	
<ESC> x <1d> <16d> <01d>		//calculation of CRC e.g.: „36601“

Remarks:

- Printing with <ESC> x <2d> n m is currently still stapled with BAUSTELLE_MSG_GEBE_A8_OUTPUT_PRINTER.
- Parameters with comment "//" are not yet implemented or not possible.

17.8.4.1 STATISTICS

In statistics, certain actions "events" of the printer are stored, e.g. optimally planning the printer service. The measured values recorded in the statistics are stored in the EEPROM in a ring buffer. The specified maximum values are therefore only assumed values, which were used to calculate the storage requirements. The counter readings are updated with each event directly in the RAM and are written in the background in the EEPROM from time to time.
The statistics counter readings are not resettable.
It should be noted that data operation time and counter reading may have significant errors by turning the printer on and off. The operating time counter reading is increased after one minute. If a printer is switched off before that time, the counter remains unchanged. In this case, the number of "PowerOn" can be included in the evaluation of the operating time.

Measurement value and maximum value

identifier	*measurement value*	*granularity*
0	operation time	1 minute
1	mileage	3 cm
2	power on	1
3	cuts	1
4	head up	1
5	paper end	1
6	EoT command	1

Output of the statitic values via the message command
The message output of the measurement values always takes place via the message command in 8 digits in ASCII hex format.

[Example] Message command for query of cuts:
<ESC>x<1d><15d><3d>
[Output] Number of cuts: 2DB4h = 11700

1Bh 78h 30h 46h 30h 33h 30h 38h 00h 00h 00h 00h 32h 44h 42h 34h
<ESC> x 0 F 0 3 0 8 0 0 0 0 2 D B 4

18 N78 EMULATION

18.1 CONFIGURATION OF PRINTER IN N78 EMULATION

N78 printer are set with the batch file Tinit. When starting the system commands for the printer configuration are sent to the system. In the C32 system this is not designed. All system parameters of the C32 system are available.

18.2 EMULATED COMMANDS

The following commands of the N78 system are emulated:

For command description see manual SoMAN-N78-E-0485 resp. SoMAN-N78-D-0484.

In the following only differences to the original N78 system are explained.

Command	description	value range
<FF>	Form FEED to the set length or to mark (TOF)	
<ESC> "A"	Erases the data in the printer buffer	
<ESC> "b" p1 ....p8	Prints barcode (EAN8, EAN13, CODE 39, 2 of 5 interleaved)	
<ESC> "C" n	0: full cut / 1: half cut / 2: cutter init	
<ESC> "D" n	Print text mode / data mode	n:={0,1}
<ESC> "e" n,m	Sleep mode (power down only)	
<ESC> "F" lh ll	Paper feed (Feed) by l_h x 256 + l_l lines.	
<ESC> "g" n g1....gn	Pixel graphics PCL5 , print graphics line with n byte length	
<ESC> "H" n	Change character height 0: single height up to 7: eight fold height	n:={0,1, ...,7}
<ESC> "I" n	Print black on white / white in black	n:= {0,1}
<ESC> "k"	Return actual status	
<ESC> "L" n	Print with / without underlining	n:= {0,1}
<ESC> "l"p_h p_l	Set side lenght	p_h:={1, .,4}; p_l:={1, .. ,4}
<ESC> "m" n	Set graphics mode (Select coding)	n:={$00 , .. , $05}
<ESC> "m" n o	Offset to the right for graphics	
<ESC> "N" p_h p_l	Go to the absolute dot position p = 256 x p_h + p_l.	
<ESC> "o"	Set start of page	
<ESC> "P" n	Select character set number. n	n:={1, .. ,n}
<ESC>“Q“ n	Old N34 quality command (see note at the end of the chapter)	
<ESC> "R" p_h p_l	Relatively move (back or forth) to the dot position	p = 256 x p_h + p_l.
<ESC> "S" n	Increase character spacing	
<ESC> "T" x	Print batch file no. x	x:= { 0 ...255)
<ESC> "V" X	Send synchronized character "X" to host	
<ESC> “v“ n	Read statistics	n:= {1,2,3,4}
	n := 0 Reading cuts	
	n := 1 Reading the total operational performance of the printer mechanism in 1/10 meters. Example: 000001A9 = 425, 42,5m paper length	
	n := 2 Reading operation time in abt. 1/10 hours Example: 000001A9 = 425, 42,5 operation hours	
	n := 3 Reading paper length since last change of paper roll in 1/10 meters Paper end sets the counter to zero.	
	n := 4 Reading the last 10 error messages. Not realized. Output is always "XXXXXXXXXX".	

Command	description	value range
<ESC> "v" "8" "Q" [DUMMY]	Read Ge number	
<ESC> "W" n	Print normal width / double width	n:= {0,1}
<ESC> "Y" n	Set density of paper individually	(n:= 10 ...75)
<ESC> "[" n m	Set power consumption and print quality	
<ESC> "\" l_h l_l	Paper backdraw by l_h x 256 + l_l lines.	
<ESC> "@"	Initializes the printer with a RESET pulse	
<ESC>“#“ n	Printout of the logo with number n	

18.3 NEW COMMANDS FOR THE N78 EMULATION

18.3.1 PRINT STORED LOGO

[Name]	Print logo with number n			
[Format]	ASCII	<ESC>	“#“	n
	Hex	1Bh	23h	n
	Decimal	27d	35d	n

[Description] The stored AddOn logo with the ordinal number n is printed in the actual cursor position. The left offset can be set with the command <ESC>m<4d><offset in mms>. The ordinal number of the logo is determined with the A8 logo patch table.

18.4 NOT YET EMULATED COMMANDS

The following commands of the N78 system are not emulated at the moment.
Commands are read and cancelled.

Command	description	value range
<ESC> "a" n	Print time	
<ESC> "B" n	Trigger sound	
<ESC> "c" n,m,...,x	Set time / alarm	
<ESC> "d" n	Read clock via host interface	n:= {0}
<ESC> "E" n	Old powerdown or PCL3 start	n:= {0,1}
<ESC> "G" n,m,...,x	Old graphics line (length = printer width)	
<ESC> "h" n	Set virtual width of printer mechanism	
<ESC> "i" n	Initialize host interface	
<ESC> "K" n	EPSON column graphics	
<ESC> "M" n	Print black / grey	n:= {0,1}
<ESC> "n" n,m,...,x	Return data string to host	
<ESC> "O" n	Print out of T menue	
<ESC> "p" n,m	Set distance to dotline of printer mechanism	
<ESC> "q" n	Old command 'esc_switch_pe_konserve' (konserve = standard text)	
<ESC> "r" n,m,...,x	Configure battery charging connection	
<ESC> "s" n,m,...,x	Save batch file	
<ESC> "T" x	Print batch file no. x	x:= { A, Q, R, S)
<ESC> "U" n	Ignore the following amount of characters	
<ESC> "u" n,m,...,x	Erase batch file (with password)	
<ESC> "v" n,m,...,x	Erase statistics (with password)	
<ESC> "X" n	Read analog value	
<ESC> "x" n	Enable / disable error saving	
<ESC> "y" n	Set energy saving mode LED (flash)	
<ESC> "Z" n	Set PWM parameter	
<ESC> "z"	Set HexDump mode	
<ESC> "]" n,m	Set baudrate ESC]	
<ESC> "{" n,m,...,x	Battery test	
<ESC> "}" n	Set minimum mark length	
<ESC> "_" (n)	Wait until ticket removal	

18.5 N78 STATUS

Print filed logo

[Name]	Print logo with number n			
[Format]	ASCII	<ESC>	“k“	n
	Hex	1Bh	6Bh	n
	Decimal	27d	107d	n

[Description] Returnall actual (error) status messages. The controller sequentially returns all actual states. If no message is available a "X" is returned. This command is not performed immediately. As it is treated like a printable character, it is not performed until all previously sent characters were attended to in the parser. In this special case it is possible to enable automatic repeat of the error messages.

n = "0" Repeat function disabled

n = (1,...254d) The actual printer status is sent to the host in intervals of about n seconds.

n = (255d) Single request with no influence on the set repeat time.

18.6 N78 ERROR MESSAGES

If a status is changed or an error occurs, one or several ASCII characters are sent automatically via the serial interface, that can be evaluated by the host. If the error is eliminated the corresponding small character is sent, followed by a "X", if no further error occurs.

Message error	Message error OK	Print is interrupted	LED Dual	LED green	Meaning of error
R	-	-	green 30:1	green on	reset
X	-	-	green 30:1	green on	error eliminated
H	h	X	red 5Hz 1:1	5Hz 1:1	head lifted
P	p	X	red 5Hz 1:1	5Hz 1:1	paper end
Z	z	-	green 5Hz 3:1	5Hz 3:1	near paper end
C	c	X	red 5Hz 1:1	5Hz 1:1	cutter blocked
K	k	X	red 5Hz 1:1	5Hz 1:1	head temperature low
T	t	X	red 5Hz 1:1	5Hz 1:1	head temperature high
U	u	X	red 5Hz 1:1	5Hz 1:1	VP too low
M	m	X	red 5Hz 1:1	5Hz 1:1	VP too high
?	*-*	*-*	*green 30:1*	*green on*	*parity error*

Hint for old N34 systems:

Instead of the command <ESC>“[„ n m for the setting of power consumption and print dynamics/quality, in the N34 systems there were the commands <ESC>"Q" n for print quality and <ESC>“[„ n for power consumption.

If the command <ESC>“Q“ n is given in the N78 emulation it is accepted and then cancelled.

Then the command <ESC> “[„ expects a parameter n for the power consumption. Now the printing quality can only be set with the system parameter 18,7.

19 FLG EMULATION

The following commands are emulated:

19.1 ROW/COLUMN COMMAND

<RC10,330>

This command positions the character in the sent row (R) and colum (C). A comma has to be sent between the values of rows and columns. The above example means row 10 and column 330, the position where the characters start according to their rotation.

The values are ASCII characters. This means that the number 10 is sent as ASCII 1 followed by ASCII 0, not as a byte with the value of 10.

19.2 ROTATION COMMAND

This command sets the rotation mode for the following text until a new direction is sent. Facing the rotational direction, all characters build up to the bottom and to the right of their original points.

<NR> No rotation
<RR> rotation right (+90)
<RU> rotation upside down (+180)
<RL> rotation left (+270 oder - 90)

19.3 COMMAND HEIGHT/WIDTH

<HW2,3>

This commands defines the height and width of a character. In the above example the height is set to 2 and the width is set to 3. This means that in case of a dot size of 7x8 the character is 16 dots high and 21 dots wide.The size of characters is only limited by the size of the ticket. Take care not to build characters into the ones below them. When the height and width of characters have been changed from normal a <HW1,1> command is needed to return to normal size.

With loadable fonts HW is capped at a maximum of 4.

19.4 COMMAND FONT SIZE

<F1> Font1 characters (5x7)
<F2> Font2 characters (8x16)
<F3> OCRB (17x31)
<F4> OCRA (5x9)
<F6> large OCRB (30x52)
<F7> OCRA (15x29)
<F8> Courier (20x40)(20x33)
<F9> small OCRB (13x20)
<F10> Prestige (25x41)
<F11> Script (25x49)
<F12> Orator (46x91)
<F13> Courier (20x40)(20x42)
Fonts above F13 will work as well

19.5 CLEAR BUFFER COMMAND

<CB>

This command clears the ticket buffer and is sent before all other commands. Mostly this command is not necessary, because the printer automatically clears the buffer. The command <CB> returns all font definitions back to normal. This command should be avoided, because it degrades printer throughput.

19.6 BARCODE COMMANDS

19.6.1 COMMAND BARCODE INTERPRETATION

<BI>

This command prints the barcode interpretation (human readable code) below the printed barcode. The <BI> command is only active for the directly following barcode print. The interpretation is printed in font1 and is set automatically depending on the size of the bar code. Other barcode command sequences are listed in the respective barcode supplement.

19.6.2 COMMAND SELECT BARCODE

<AB#>string or <aB#>string:

A= U (for UPC A and EAN8)
A= E (for EAN13)
A= N (for Code 39)
A= F (for interleaved 2/5)
A= O (for Code 128 B + C)

New style - rotation commands influence barcode

a= u (for UPC A and EAN8)
a= e (for EAN13)
a= n (for Code 39)
a= f (for interleaved 2/5)
a= o (for Code 128 B + C)

B= P (for picket fence style)
B= L (for ladder style)

 # and string vary as described below

The above example is a figurative description of a barcode selection command.
The letter A means the kind of barcode selected.
The letter B means the alignment of the barcode (either picket fence or ladder styles)
The # means the size (units) of the barcode. This is an optional parameter. Without it the default size is 4.
Each unit represents an 8 dot high rectangle, which means the default barcode is 32 dots high (8x4). Barcodes can only be rotated in specific directions:
<NR> and <RU> for picket fence style barcodes
<RR> and <RL> for ladder style barcodes
Refer to the particular bar code supplement selected in order to find the exact commands and strings needed for bar code printing.

19.6.3 COMMAND EXPAND BARCODE

<X#>

This command expands a barcode's (normally based on a one dot unit) length. The number after the X is the new dot unit of the barcode size. In the above example the new width of the barcode is 2 dots. This doubles the barcode length. The command <X3> would triple the barcode length. Usually a 2 dot wide bar is all that is needed for a clearly readable barcode.

9 (nine) is the largest expansion number allowed. String = barcode data.

19.7 COMMAND LENGTH

19.7.1 COMMAND PRINTING LENGTH

<PL#>

This command allows a user to control the length of a printed ticket. On poweron the printer calculates the ticket length. It equates the printing length with this value. Therefore, a standard ticket is usually printed to about 1050 columns. To reduce the printable area on the ticket, the number of rows should be by 1 smaller than the actual ticket length. The printing length must not exceed the actual ticket length. If an oversized <PL> command is sent, the printer will set it equal to the ticket length.

Please, note that the first column always starts in the leftmost position of the ticket.

19.7.2 COMMAND PERMANENT PRINTING LENGTH

<pl#>

With this command the printer permanently stores the printing length value in the flash. The sent # is equal to the number of dot columns to be printed. If no number is sent the printing length determined during initialization will be stored. Once a value has been stored, the ticket measuring function of the printer, after poweron, is disabled. This function is useful when label stocks are printed. The back and forth motion of the ticket during initialization makes it possible to peel off a label. By having the label printing size permanently stored, there is no need to measure the label.

19.7.3 COMMAND PERMANENT TICKET LENGTH

<tl#>

This command is only applicable when the ticket length exceeds the printing. It causes the printer to permanently store the ticket length value in the flash. The sent # is equal to the number of dot columns of the ticket. If no number is sent the ticket length determined during initialization will be stored. Once a value has been stored, the ticket measuring function of the printer, after poweron, is disabled. Usually ticket length and printing length are the same. However, when using short (below 1.5") die cut labels it is often desirable to have the printing length shorter than the ticket length. Label stocks usually have a clear (non-printing) area between the labels. So the ticket (label) exceeds the printing size. In this case the size of the free space between the labels is typically the difference between printing length and ticket length.

Please note that the command <tl#> has to be sent before the command <pl#>.

19.7.4 COMMAND DELETE PERMANENT TICKET LENGTH

<dpl>

This command activates the automatic measuring function of the printer after poweron. It is used whenever a <pl #> or a <tl #> command is to be deleted and clears the stored printing length.

19.8 REPEAT COMMAND

<RE#>

The repeat command allows the user to print multiple copies without retransmitting the ticket. The # gives the number of ticket copies in addition to the first one. The repeat command can be sent anywhere in the data stream prior to the print command.

19.9 COMMAND PRINT RESIDENT LOGO

<LO#>

This command causes pre-loaded logos to be printed on the ticket. For further details see the logo supplement.

This command must be preceded by a starting point, <SP # #> command.

19.10 COMMAND DRAWING

19.10.1 COMMAND DRAW BOX

<BXr,c>

This command makes the printer draw a box "r" dot rows tall and "c" dot columns wide. We recommend a row/column command subsequently to prevent any confusion concerning the cursor position.

19.10.2 COMMAND DRAW VERTICAL LINE

<VXr>

This command makes the printer draw a vertical line with the length of "r" dots and the width of 1 dot. We recommend a row/column command subsequently to avoid confusion concerning the cursor position.

19.10.3 COMMAND DRAW HORIZONTAL LINE

<HXc>

This command makes the printer draw a horizontal line with the length of "c" dots and the width of 1 dot. We recommend a row/column command subsequently to avoid confusion concerning the cursor position.

19.10.4 COMMAND LINE THICKNESS

<LT#>

This command is used in connection with the draw line and draw field commands. It enables the user to change the default line thickness (1 dot). The character # in the command represents the number of dots of line thickness. This command has to be sent immediately before the draw line or draw field command. After the line has been drawn, the line thickness is reset to default (1 dot). If several fields with different line thickness are to be drawn, separate line thickness commands are to be sent before each of the draw field commands. The following sequence gives a field with a length of 10 dots and a width of 10 dots with a line thickness of 4: <LT4> <BX 10.10>.


The thickness of a field increases towards the center.

A vertical line grows towards the right side and a horizontal line grows towards the bottom of the ticket. The only limitation for the field drawing line is that it cannot exceed 1/2 of the size of the smallest field. For example, a field of 10x15 can have a maximum line thickness of 5.


This makes a completely black field.

19.11 COMMAND TICKET COUNTER

19.11.1 COMMAND PRINT TICKET COUNTER

<PC>

The printer keeps track of each ticket to be printed. The seven-digit number is kept in the printer as a ticket count. Up to two ticket counts (same count both) can be printed anywhere on the ticket in any font size and any rotation. This requires the command <PC>. This command can be placed anywhere in the ticket data. However the position of the ticket pointer is used to place the ticket counter. Therefore we recommend to send a normal row/column command preceding the <PC> command. This makes it possible to print two identical counts: one on the main ticket and one on a stub. See the following instructions 19.11.2 to modify the load ticket count.

To change the count, see the load ticket count instruction below. Note that the height/width command has no effect with this command and that only two counts can be printed per ticket. A separate <PC> command must be sent for each count you want printed. For example, to print two rotated-right font3 counts, 100 columns apart, you might send the following:
<F3> <RR> <RC10,100> <PC>
<F3> <RR> <RC10,200> <PC>

19.11.2 COMMAND LOAD TICKET COUNT

<TC1234567>

This command enables the user to preload the seven-digit ticket number. It has to contain all seven digits - the number 5 would be sent as: <TC0000005>. This number is the count for the actually sent ticket. The following ticket will be counted higher by one. Using this command together with a repeat command and a print count data command makes it possible to print a lot of tickets with counter number at full speed.

19.12 COMMAND INVERTED PRINT MODE

19.12.1 COMMAND ENABLE INVERTED PRINT MODE

<EI>
This command enables the inverted print mode - print white on black.
(Actually not available with soft fonts.)

19.12.2 COMMAND DISABLE INVERTED PRINT MODE

<DI>
This command disables the inverted print mode.

19.13 COMMAND SCALE FONT

19.13.1 COMMAND SCALE DOWN FONT

<SD#> [not available for FGL21]
This command scales down an FGL font to a fraction of its original size. The font is divided by the # given in the scaling command. This command can be combined with the <HW #, #> command, in order to generate fractions or multiplications of the font size.

Example: To scale a font to 2/3 of its original size, the command <HW2,2> <SD3> is sent.


The command for height/width should be sent before the scaling command.
(Actually not available with soft fonts.)

19.13.2 COMMAND SCALE UP FONT

<TTF5;20>
This commands scales a font up from size 5 to size 20.

19.14 COMMAND PRINT INTENSITY

<Ive#>
This command adjusts the print intensity according to a given speed setting. <Ive0> is default. The # value can vary from -5 up to +5. Positive values increase the print intensity while negative values reduce it.


This function is meant to fine tune the intensity and not for compensation of inferior and/or inadequate ticket stock. Excessive manipulation of the print intensity with this command reduces the life of the printhead considerably.

When using this command with a dual printer, the command <Ive #> adjusts the print intensity on the active path. It is, therefore, recommended to send the path command immediately before the <Ive #> command, to make sure that the proper path intensity will be selected.

19.15 COMMAND PRINT/CUT TICKET

<p>
This is the normal print command. The printer will cut the ticket after printing if it has a cutter.

19.16 COMMAND PRINT TICKET/ NO CUT

<q>
This is the normal command for printing without cut. Though the printer has a cutter it will not cut the ticket after printing.

19.17 FGL STATUS

19.17.1 STATUS REQUEST

<S1> STATUS REQUEST
<S2> PROM TYPE AND TICKET COUNT STATUS REQUEST
<S3> DELAYED STATUS REQUEST (end of ticket run status)
<s3> DELAYED STATUS REQUEST (end of ticket run status)
<S5> NO STATUS
<s5> NO STATUS
<S6> ASCII STATUS
<s6> ASCII STATUS
<S7> DOWNLOAD SPACE AVAILABLE STATUS
<S8> PARTIAL ASCII STATUS
<s8> PARTIAL ASCII STATUS

19.17.2 STATUS MESSAGES

(HEX)	(DEZ)	
06H	6	TICKET ACK
0FH	15	LOW PAPER
10H	16	OUT OF PAPER
11H	17	X-ON
12H	18	POWER ON
13H	19	X-OFF
18H	*24*	*PAPER JAM*
1DH	29	CUTTER JAM


When an error occurs, the status LED blinks 1:1

20 PCL3 EMULATION

The printer also emulates PCL graphics data similar to HP deskjet 350®.

20.1 PCL COMMANDS REDIRECTED TO GeBE GRAPHICS COMMANDS

The following commands are redirected to GeBE graphics commands:

<ESC>"*""b""#""W" [data]	Transfer Raster Graphics value range 0 (empty row) up to 32767
<ESC>"*""b"(type)"M"	Raster compression (compression procedure) Typ: 0 = unencoded 1 = run-length 2 = tiff 3 = delta-row *4 = reserved* *5 = adaptive compression*
<ESC>"*""b""#""Y"	Raster Offset Command (vertical movement) System parameter 38,9 indicates, how many empty rows must exist to get them cleared. None are cleared with parameter = 0. Typical = 16

20.2 IGNORED PCL COMMANDS

The following commands are not executed (i.e. the characters up to the next ESC are read and ignored. The end of a command is always a capital letter.)

<ESC>"*" ..... All except "r""B" and "r""C"

<ESC>"*""r""#""F"	Raster Graphics Presentation Mode Command	(Orientation)
<ESC>"*""r""#""T"	Raster Height Command	(Height of graphics field)
<ESC>"*""r""#""S"	Raster Width Command	(Width of graphics field)
<ESC>"*""r""#""A"	Start Raster Graphics Command	(Graphics field offset)
<ESC>"*""p""#""Y"	Vertical Cursor Positioning in PCL Units	(=1/300 inch)
<ESC>"*""t""#""R"	Raster Graphics Resolution Command	(Graphics resolution)
<ESC>"E"	Printer Reset	(printer reset)
<ESC>"&"xxx	Number of copies, Print direction	(Number of copies, print direction)

Linking of small letters is taken into account until up to a length of 255 bytes!

20.3 SWITCHED PCL COMMANDS

<ESC>“*““r““B“ End Raster Graphics Command
<ESC>“*““r““C“ End Raster Graphics Command, reset compression mode, leftmargin = 0

The parameters 38,10, 38,11, and 38,12 determine which actions are executed when these commands are given.

Parameter:

38,10: End of graphics <ESC>“*““r““C“ and <ESC>“*““r““B“ execute a formfeed

38,11: End of graphics <ESC>“*““r““C“ and <ESC>“*““r“„“B“execute a EoT

38,12: End of graphics <ESC>“*““r““C“ and <ESC>“*““r““B“ execute the batch file no. 38,12. 0:= no batch file executed..
These commands are ignored when there is no entry in this batch file or when the parameter = 0).

20.4 ELIMINATE EMPTY ROWS

Parameter 38,9 "Default maximum empty rows" determines how many successive empty rows are needed, until they are eliminated. This makes a length limitation of blank spaces possible.

21 LOGOS

The C32 system executes AddOn type GeBE standard logic files.
Logos are stored as AddOns in the printer flash and supplied with the logo index in storage sequence. So the logo stored in third place will have the logo index 3 and can be addressed with the command "print logo".

21.1 LOGO SEQUENCE

If you wish this specific logo in the second place you need to clear the logos and insert them in the desired sequence or you must switch the logos in the applied logo patch tables in the parser modules. Each printer emulation has its own logo patch table in the parser modules (parameter x,5). With the help of the patch table the sequence of logos, as stored in the flash, can be changed. The logos 0 - 31 can be switched to other positions. Up from the 32nd stored logo switching is no longer possible. This gives the basic table the following structure:

[0,1,2,3,4.....,31] 32, 33, 34 255

The command "select logo 1" addresses the first logo in the AddOn flash memory. The GeBE logo: logo 0 is filed as resident in the firmware.

If you want the fourth logo be addressed with the command "select logo 1" you must enter the following sequence for the C32 mode in parameter 14,5:

<RS>[1:14,5:0,4]

Logo 4 is now switched to position 1. Then may follow 2, 3, 4 ... and so on.
Logo 1 can no longer be addressed in this mode, unless you enter: <RS>[1:14,5:0,4,2,3,1,5]
Logo 4 and 1 switched positions, not in the flash, but as an ordinal number.

22 FONTS

The printer supports different font systems.

a. Mono space and proportional fonts
b. 8 bit fonts e.g. IBM II, Codepage 851 .....
c. Unicode font for the 'basic multilingual plane' (BMP0)
d. 16 bit font with automatic 8-16bit switch:
   z.B. Chinese GB2312
e. *Arabian fonts, including writing from the right to the left*
f. *Country specific fonts*

The C32 system processes GeBE standard font files type AddOn, proportional as well a NOT proportional (that means, even fonts where each character may have a different width). Fonts are stored as AddOns in the printer flash and supplied with the font index in storage sequence. So the font stored in third place will have the font index 3 and can be addressed with the command <ESC>P<3d>. The system font 0 with the size 8x16 is firmly stored in the firmware and cannot be changed.

22.1 FONT SEQUENCE

If you wish this specific font in the second place you need to clear the fonts and insert them in the desired sequence or you must switch the fonts in the applied font patch tables in the parser modules. Each printer emulation has its own font patch table in the parser modules (parameter x,4). With the help of the patch table the sequence of fonts, as stored in the flash, can be changed. The fonts 0 - 31 can be switched to other positions. Up from the 32nd stored font switching is no longer possible. This gives the basic table the following structure:

[0,1,2,3,4.....,31] 32, 33, 34 255

The command "select font 1" addresses the first font in the AddOn flash memory. Font 0 is filed as resident in the firmware. If you want the fourth font be addressed with the command "select font 1" you must enter the following sequence for the C32 mode in parameter 14,4:

<RS>[1:14,2:0,4]

Font 4 is now switched to position 1. Then may follow 2, 3, 4 ... and so on.
Font 1 can no longer be addressed in this mode, unless you enter: <RS>[1:14,2:0,4,2,3,1,5]
Font 4 and 1 switched positions, not in the flash, but as an ordinal number.

22.2 UNICODE FONT (OPTIONALLY LOADABLE)

Alternatively also 16 bit fonts and unicode fonts can be processed.
In this case the kind of coding has to be indicated with:

Parameter 13,9

0: 8 bits	suitable for fonts up to max. 256 characters (e.g. ASCII or GeBE standard fonts)
1: 8.16 bits	automatic switch, suitable for the GeBE font SimHei_GB23112_24x24_9
2: UTF 16 big endian	suitable for fonts up to max. 65536 characters (e.g. GeBE font GE4769)
3: UTF 16 little endian	suitable for fonts up to max. 65536 characters (e.g. GeBE font GE4770)
4: UTF 8	suitable for fonts up to max. 65536 characters (e.g. GeBE font GE4771)

If you select the 16 bit font with the command <ESC>"Pn" the printer automatically expects a 16 bit value to address the desired character. However, if the first byte is smaller than 80Hex, only the first page is started without expecting any further address byte. If you still want to use only 2 byte codings, the parameter 13,7 has to be changed accordingly from value 1 to value 0. To address the font in the 16 bit mode, the parameter 13,7 must be set to "0". The font size can be zoomed randomly.

Coding the unicode font is done with UTF8, UTF16BE, OR UTF16LE. To address the font the parameter 13.9 has to be set accordingly. Character size is 6pt. This represents a chinese character with 16x16 pixels. The font size can be zoomed randomly.

22.3 STANDARD FONTS

Font	Size	Name	Remark
0	8x16	GB4_8x16.addon	System font GeBE A8 Standard
1	16x24	GB216x24.addon	GeBE N78 Font1
2	9x22	GB2_9x22.addon	GeBE N78 Font2
3	7x16	GB2_7x16.addon	GeBE N78 Font3
4	12x24	GB212x24.addon	GeBE N78 Font4
5	14x22	GB214x22.addon	GeBE N78 Font5
6	16x24	KYR16x24.addon	GeBE N78 Cyrillic
7	8x16	KYR_8x16.addon	GeBE A8 Cyrillic
8	120x120	GB_Roboto_36.addon	Roboto condensed 36pt*
9	120x120	GB_RobotoBold_36.addon	Roboto condensed Bold 36pt*
10	65x124	GB_Mono821_36.addon	GeBE TrueType Monoblock 36pt
11	69x124	GB_Mono821Bold_36.addon	GeBE TrueType Monoblock Bold 36pt
12	5x7	F1_5x7_b.addon	FGL Font1
13	8x16	F2_8x16_b.addon	FGL Font2
14	17x31	F3_17x31_b.addon	FGL Font3
15	5x9	F4_5x9_b.addon	FGL Font4
16	30x52	F6_30x52_b.addon	FGL Font6
17	15x29	F7_15x29_b.addon	FGL Font7
18	20x30	F8_20x30_b.addon	FGL Font8
19	13x20	F9_13x20_b.addon	FGL Font9
20	25x41	F10_25x41_b.addon	FGL Font10
21	25x49	F11_25x49_b.addon	FGL Font11
22	46x91	F12_46x91_b.addon	FGL Font12
23	20x40	F13_20x40_b.addon	FGL Font13

The following Unicode fonts are optionally available:

1. **GB1_Unicode_16x16:** (37625 characters / 1327 KB)
   This font contains practically all customary Unicode characters of the first basic level (BMP0). In the original resolution the Chinese characters have a size of 16x16 pixels.
   Consisting of pages: 00-06, 09-11, 1E-1F, 30-33, 4E-9F, AC-D7, F9-FE
2. **GB1_UnicodeEUCJK_18x18:** (22568 characters / 1232 KB)
   This font contains all Chinese, Japanese, Korean and European characters. In the original resolution the Chinese characters have a size of 20x20 pixels.
   Consisting of pages: 00-05, 1F, 4E-9F
3. **GB1_UnicodeRest_18x18:** (18357 Zeichen / 869 KB)
   Except CJK, this font contains all European and remaining characters of the complete Unicode Font.
   Consisting of pages: 00-06, 09-11, 1E-27, 30-33, AC-D7, F9-FE
4. **SimHei GB2312:** (476 KB)
   This font contains approx. 6,700 Chinese characters and European standard characters.
   Consisting of pages: 00, B0 – F7

Please note that our Unicode is very substantial. However, like all existing Unicode fonts it is not complete. On demand we send you a pdf with the survey of all available characters.

22.3.1 GeBE STANDARD FONTS 0 - 7

Table of fonts 0, 1-5

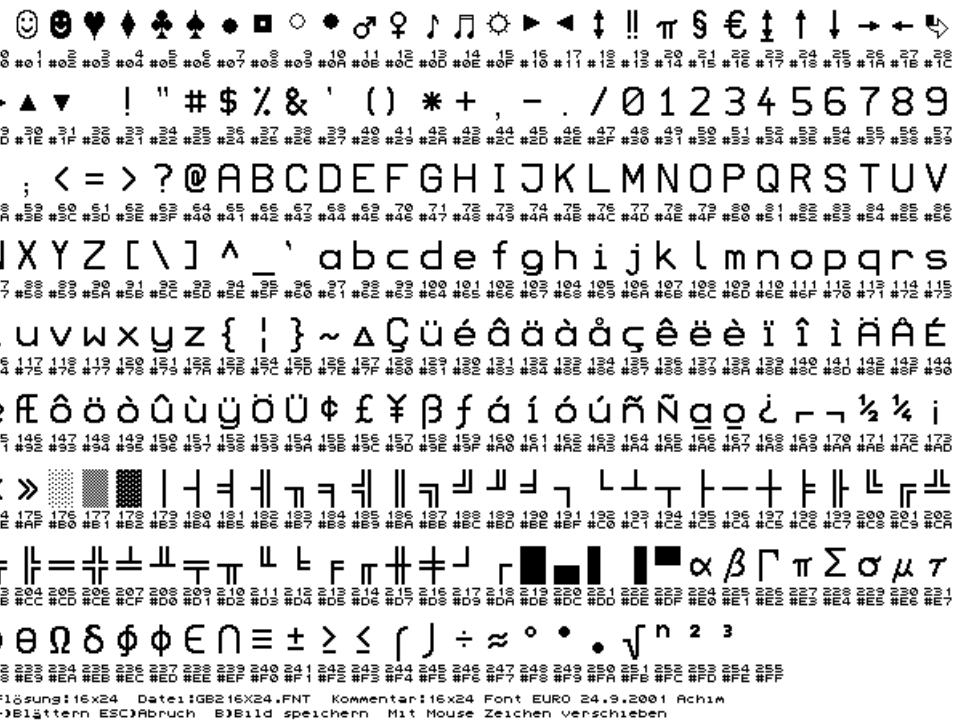
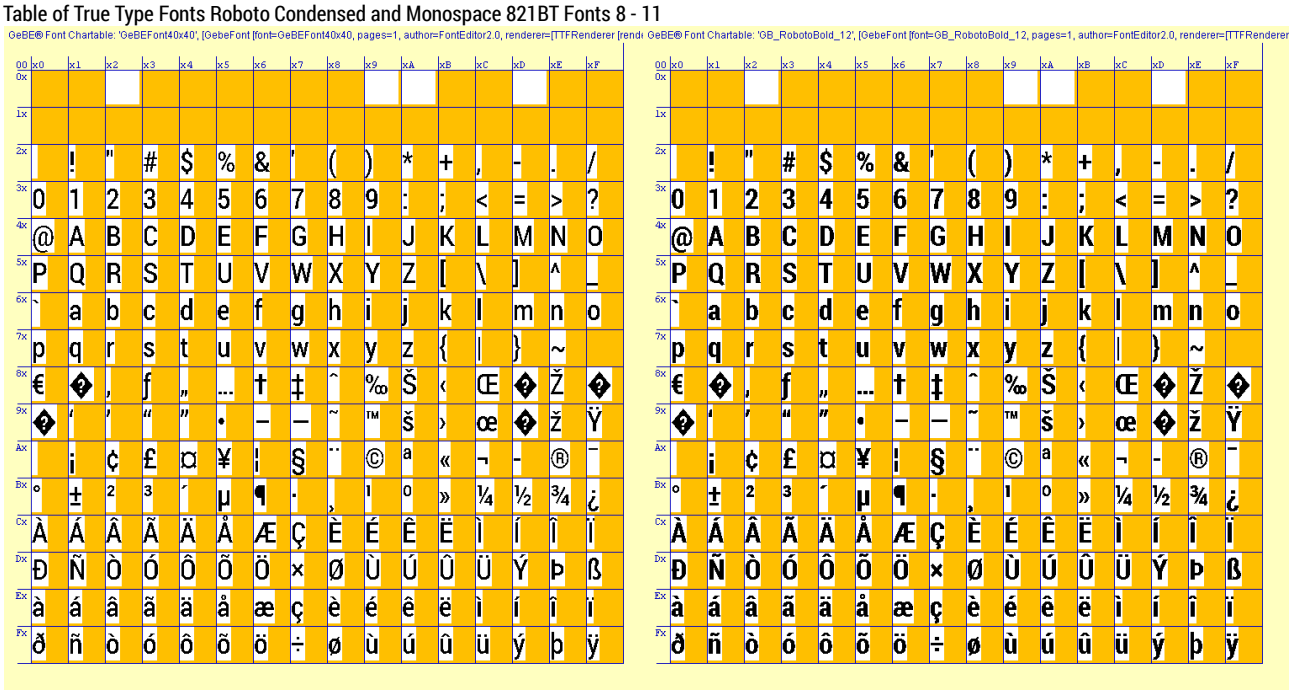


Table of Cyrillic fonts 6 and 7

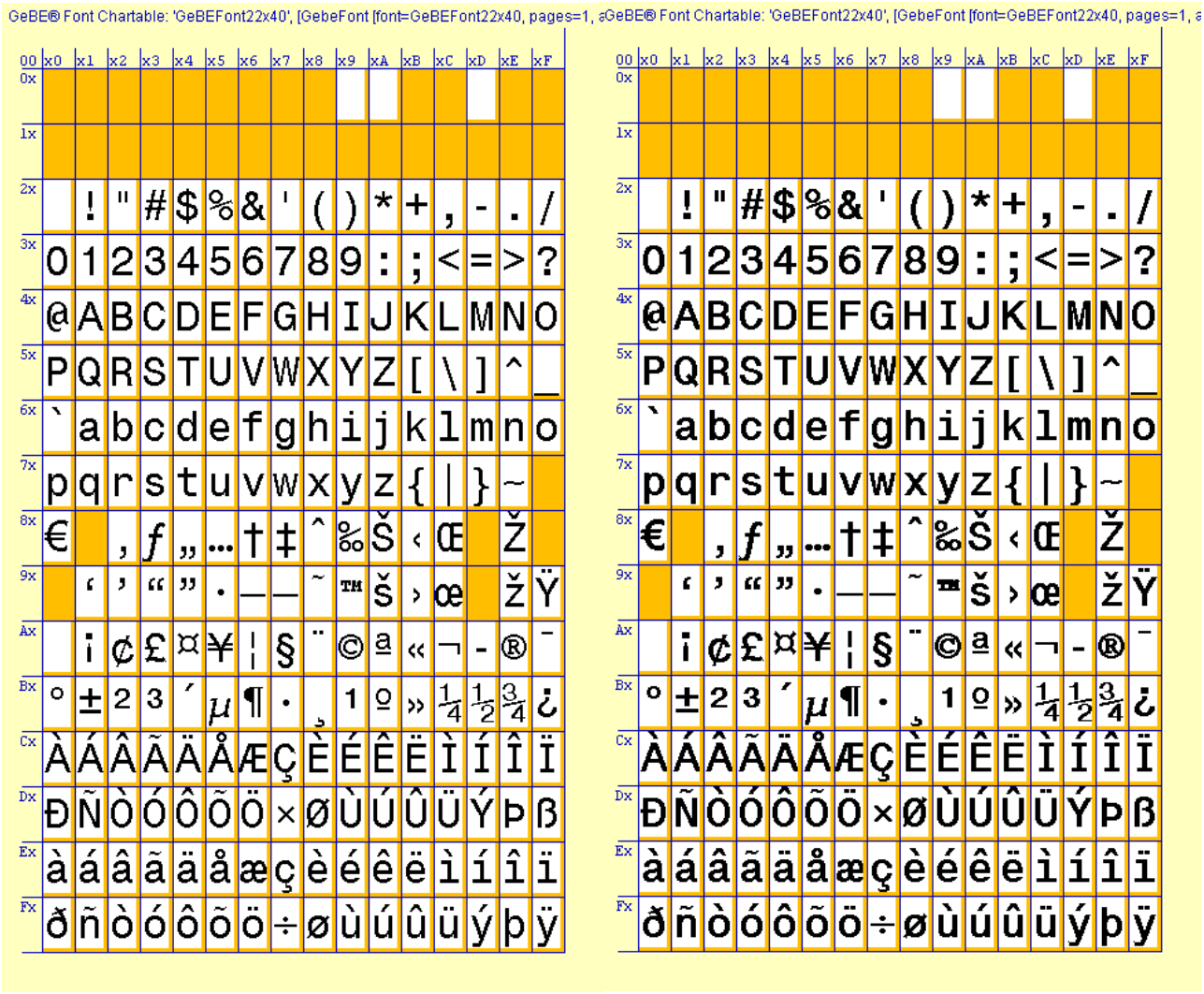
22.3.2 GeBE STANDARD FONTS 8 - 11

Table of True Type Fonts Roboto Condensed and Monospace 821BT Fonts 8 - 11



Roboto Condensed 12pt, resp. 36pt

Roboto Condensed Bold 12pt, resp. 36Pt



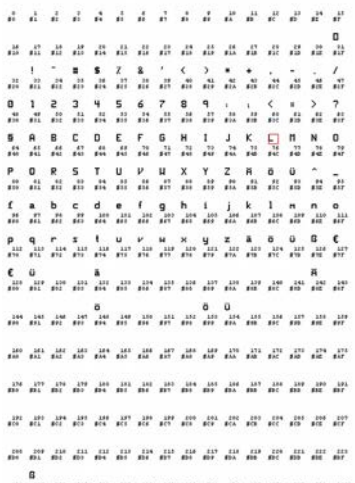
Monospace 821BT 12pt, resp. 36pt

Monospace 821BT Bold 12pt, resp. 36pt

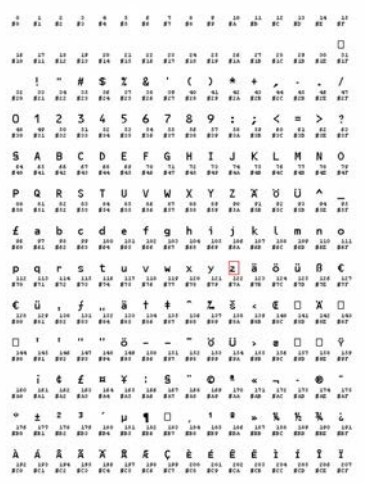
22.3.3 FONTS 12 - 23

Font 12 5x7

Font 15 5x9



Font 13 8x16



Font 19 13x20

Font 17 15x29

Font 18 20x30

Font 14 17x31

Font 23 20x40

Font 20 25x41

Font 21 25x49

Font 16 30x52

Font 22 46x91

23 ATTACHMENT A: ADVANCED C32 DEBUG STATUS MESSAGE

The advanced C32 debug status message is intended for diagnose and support.
GeBE reserves the right for any changes like flag assignments.
For your application apply the C32 standard status messages.

23.1 ENABEL/DISABLE ADVANCED C32 STATUS MESSAGE

The advanced C32 status message can be enabled/disabled in parameter 12,5.

Parameter:

12,5: 1: enable / 0: disable debug status messages

12,6: periodical debug status message in [ms] / 0: disabled

23.2 MESSAGE STRUCTURE

[AILength#[TimestampmsGMID:Pos_Counter,PM,SysParams,HostIF,Print,Print_Sensors,Print_EoT,
Renderer#i__]#0xCRC:AI]

Example: Output in ASCII format:

[AI0x00000052:#[0x0000000Ems0x00000000:0x00000001,0x00000001,0x00000000,0x0000BE40,0x00000001,0x00000000,0x00000000,0x00000000#i__]#0x2D500C85:AI]

Pos counter: Line counter since printer switch-on (32 bit counter).

23.3 MODULE PM POWER MANAGER

Bit	Description
0	All modules in operative status
1	Emergency operation with factory set parameters
2	Imminent power off
3	Fast charging
4	Trickle charging
5	Battery fully charged
6	Suspend
7	Resume
8 – 31	Not used = 0

23.4 MODULE SysParams

Bit	Description
0	Module is active
1	EEPROM access not possible
2	at least 1 EEPROM system parameter set destroyed
3	at least 1 EEPROM system parameter set damaged
4	at least 1 Factory system parameter set damaged
5 – 31	Not used = 0

23.5 MODULE HostIF

Bit	Description
0	Parity error, framing error
1	Overrun error
2 – 31	Not used = 0

23.6 MODULE PRINT

Bit	Description
0	ISR run-time overflow
1	Logical Paper End, Expansion of the printing area after paper end detection
2	Debugging condition (stall condition) compiled
3	Abortion condition compiled
4	Ticket shortened
5	Top of Form reached
6	Position counter = 0
7	Start of a ticket segment
8	Blackmark identified
9	S.L.T. fixed
10	Backwards feed
11	Preheat active
12	Execution of shared assignments
13	Autoload advanced notice
14	Autoload is executed
15	Autoload feed failed (charging condition missing or untimely paper end)
16	Flush assignments
17	AutoFlush suspended
18	Cutter jam
19	Damage Head Spec Characteristics
20	Auto lush blocked
21	AccelerationCapped „Acceleration capped due to algorithm constraints“
22	ReadSensorsError „Sensor values out of date“
23	End of ticket command in execution
24	Eot Notification
25 – 31	*=0 not yet defined*

23.7 MODUL PRINT SENSORS

Bit	Description
0	printhead open
1	AUX1 Sensor recognized (meaning depending on application)
2	AUX2 Sensor recognized (meaning depending on application)
3	AUX3 Sensor recognized (meaning depending on application)
4	AUX4 Sensor recognized (meaning depending on application)
5	AUX5 Sensor recognized (meaning depending on application)
6	AUX6 Sensor recognized (meaning depending on application)
7	AUX7 Sensor recognized (meaning depending on application)
8	NPE recognized (Near Paper End)
9	physical paper end, measured at paper end by paper sensor
10	Vp too low
11	Vp too high
12	head temperature too low
13	head temperature too high
14	motor temperature too low
15	motor temperature too high
16	printhead stripped
17	feed button pressed
18	button1 pressed
19	button2 pressed
20	head has reached (at least) the first preheating temperature
21	head has reached (at least) the preheating temperature
22	head alignment too low
23	head alignment too high
24	battery temperature too low
25	battery temperature too high
26 – 31	= 0 not yet defined

23.8 MODULE PRINT EoT

Bit	Description
0	
1 – 31	Not used = 0

23.9 MODULE RENDERER

Bit	Description
0 – 31	Not used = 0

23.10 REQUEST ANALOGUE VALUES

(old debug command)

<RS> “x“ “a“ 0d

The command issues the analogue values for all 8 available analogue channels:
Output as 16 bit value per AD channel.

The resolution of the converter is only 12 bits.
That means, the upper 4 bits are always 0.

Channel 8 :	AUX 7	(additional VP measuring / battery temperature)
Channel 9:	AUX 1	(Blackmark)
Channel 10:	Vp	
Channel 11 :	Head Temp	
Channel 12:	Paper End	
Channel 13:	AUX 5	(Motor temperature / pressure sensor)
Channel 14:	Near Paper End	
Channel 15:	AUX 2	(2. Blackmark / Head Alignment)

Example: <RS>xa<0d>

Reply to Module 12 message, with the message ID: GMID = 0x000C and the message code = 0x0001

[aR:0x00000062#[0x00016E83ms0x000C0001: 0x000006AC,0x0809, 0x0FFF, 0x0D91, 0x07B7, 0x0146, 0x07EE, 0x0FFF, 0x04AE #w_+]#0x0A0080BD:aR]

Reply are all 8 ADC channels from channel 8 -15 (depending on hardware)

Debug command for detecting and reading important timing parameters on module Stats:
Only available with DEBUG fimware versions!

Reading with <RS> 'xt' <n>,
Resetting with <RS> 'dt' <n>

<n> = 0: all Tracing data
<n> = 1: Motor1-S.L.T. [us]
<n> = 2: Printer-call-delay [us]
<n> = 3: Printer-runtime PRINTISRSTATE_IDLE [us]
<n> = 4: Printer-runtime PRINTISRSTATE_RESUME [us]
<n> = 5: Printer-runtime PRINTISRSTATE_PRELINE [us]
<n> = 6: Printer-runtime PRINTISRSTATE_STARTLINE [us]
<n> = 7: Printer-runtime PRINTISRSTATE_PREHEAT [us]
<n> = 8: Printer-runtime PRINTISRSTATE_DOSTROBES [us]
<n> = 9: Printer-runtime PRINTISRSTATE_CLOSELINE [us]
<n> = 10: Printer-runtime PRINTISRSTATE_AWAIT_STATUS [us]
<n> = 11: Printer-runtime PRINTISRSTATE_FIRSTLINE [us]
<n> = 12: Printer-runtime PRINTISRSTATE_NOLINE [us]
<n> = 13: Printer-runtime PRINTISRSTATE_SUSPEND [us]

Depending on above mentioned timing, 3 values will be issued: moving average, miniumum and maximum average. This data area is empty for not yet recorded data. (Actually no average will be calculated for several values - therefore the moving average is equal to the actual value.)

Automatic reset of all data with each POR (no storage in EEPROM).

24 ATTACHMENT B: THE BOOTLOADER

24.1 DOWNLOADING FIRMWARE, FONTS, LOGOS OR BATCH FILES

The printer is equipped with a bootloader to change firmware, fonts or addOns. The bootloader is generally designed to boot via more than one interface. Actually only USB is implemented.
In USB operation the bootloader is carried out as a particular USB printer class device with its own USB PID. Activating the bootloader means logging off the printer device and logging in the bootloader as new device. As soon as the bootloader is stopped, it is logged off in the computer as a device and the printer device is again logged in.

Starting the system is always also starting the bootloader. Then the paper is checked. If paper is available, the application is started immediately. If no paper is available the bootloader waits for 2 seconds for bootloader commands. In remote operation with a download software these 2 seconds are enough to activate the bootloader process. If you press the feed button during start up of the system and paper is not available, this waiting period is extended by further 30 seconds. This is sufficient time to activate the bootloader process manually.

If the bootloader process is not activated within this time, the firmware is automatically started. If no valid firmware exists, the printer remains in bootloader mode.

When an application is updated, the flash area needed for the application is cleared after receipt of the first data portion. This may cause several seconds of idle time. During this time you ought to avoid USB requests.

If an error occurs during download the controller remains in the bootloader mode and waits for a new download.

For detailed informationen please refer to the Software Manual Bootloader SoMAN-C32-E-0955.

25 ATTACHMENT C: CRC CALCULATION UNIT

25.1 RELATED DOCUMENTS

The following pages are excepted from the STMicroelectronics reference manual RM0090 for microcontroller memory and peripherals applying to the whole STM32F4xx family, unless otherwise specified.

Further information is available from STMicroelectronics web site (http://www.st.com):

- STM32F40x and STM32F41x datasheets
- STM32F42x and STM32F43x datasheets
- For information on the Arm® Cortex®-M4 with FPU, refer to the STM32F3xx/F4xxx Cortex®-M4 with FPU programming manual (PM0214).

25.2 CRC INTRODUCTION

The CRC (cyclic redundancy check) calculation unit is used to get a CRC code from a 32-bit data word and a fixed generator polynomial. Among other applications, CRC-based techniques are used to verify data transmission or storage integrity. In the scope of the EN/IEC 60335-1 standard, they offer a means of verifying the Flash memory integrity. The CRC calculation unit helps compute a signature of the software during runtime, to be compared with a reference signature generated at linktime and stored at a given memory location.

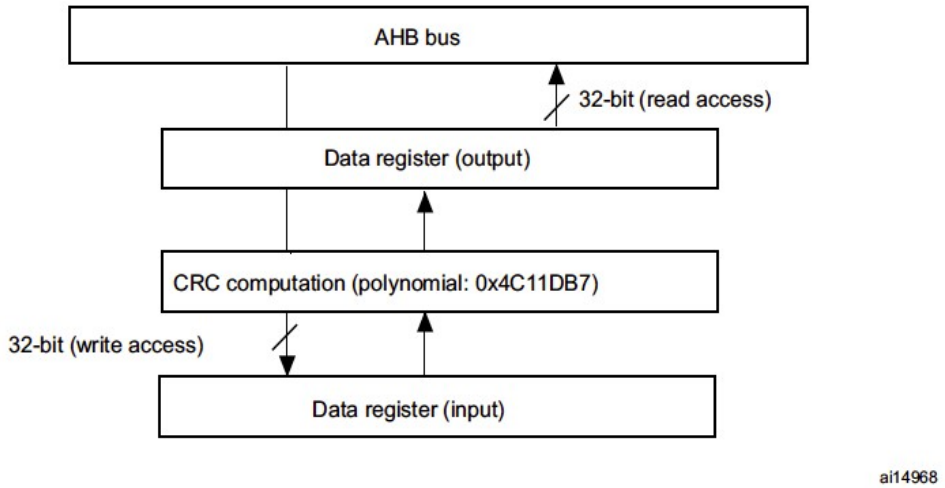
25.3 CRC MAIN FEATURES

Uses CRC-32 (Ethernet) polynomial: 0x4C11DB7

– $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$

- Single input/output 32-bit data register
- CRC computation done in 4 AHB clock cycles (HCLK)
- General-purpose 8-bit register (can be used for temporary storage)

The CRC calculation unit block diagram is shown in figure below:



25.4 CRC FUNCTIONAL DESCRIPTION

The CRC calculation unit mainly consists of a single 32-bit data register, which:
• is used as an input register to enter new data in the CRC calculator (when writing into the register)
• holds the result of the previous CRC calculation (when reading the register)

Each write operation into the data register creates a combination of the previous CRC value and the new one (CRC computation is done on the whole 32-bit data word, and not byte per byte).
The write operation is stalled until the end of the CRC computation, thus allowing back-to-back write accesses or consecutive write and read accesses.
The CRC calculator can be reset to 0xFFFF FFFF with the RESET control bit in the CRC_CR register. This operation does not affect the contents of the CRC_IDR register.

25.5 CRC REGISTERS

The CRC calculation unit contains two data registers and a control register.
The CRC registers have to be accessed by words (32 bits).

25.5.1 CRC REGISTER (CRC_DR)

Address offset: 0x00
Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DR [31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DR [15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0
Data register bits
Used as an input register when writing new data into the CRC calculator.
Holds the previous CRC calculation result when it is read.

25.5.2 INDEPENDENT DATA REGISTER (CRC_IDR)

Address offset: 0x04
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								IDR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8
Reserved, must be kept at reset value.
Bits 7:0
General-purpose 8-bit data register bits
Can be used as a temporary storage location for one byte.
This register is not affected by CRC resets generated by the RESET bit in the CRC_CR register.

25.5.3 CONTROL REGISTER (CRC_CR)

Address offset: 0x08
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															RESET
															w

Bits 31:1
Reserved, must be kept at reset value.
Bit 0
RESET bit
Resets the CRC calculation unit and sets the data register to 0xFFFF FFFF.
This bit can only be set, it is automatically cleared by hardware.

25.5.4 CRC REGISTER MAP

The following table provides the CRC register map and reset values.

Offset	Register	31-24	23-16	15-8	7	6	5	4	3	2	1	0
0x00	**CRC_DR**	Data register										
	Reset value	0xFFFF FFFF										
0x04	**CRC_IDR**	Reserved			Independent data register							
	Reset value				0x00							
0x08	**CRC_CR**	Reserved										RESET
	Reset value											0