

GSI Lumonics

PC-MARK MT™ / PC-MARK

Command Reference

**4E Crosby Drive
Bedford, MA 01730**

**GMAX™ SYSTEMS
MULTI-AXIS BEAM HANDLING**

GSI LUMONICS LICENSE AGREEMENT

This is a legal agreement between you (either an individual or an entity) and GSI Lumonics (GSLI). By opening the sealed software packages and/or by using the SOFTWARE, you agree to be bound by the terms of this Agreement. If you do not agree to the terms of this Agreement, promptly return the disk package and accompanying items (including printed materials and binders and other containers) to GSLI.

GSLI SOFTWARE LICENSE

- 1. GRANT OF LICENSE.** The license grant shall become effective upon breaking of the seal attached to the package containing the SOFTWARE and will remain in effect unless licensee returns the package, documentation and SOFTWARE to GS within five (5) days of breaking of the seal without copying any of the same. Failure to return the package, documentation and SOFTWARE within the time period permit shall act to acknowledge Licensee's acceptance of the terms and conditions of this license.
- 2. PERMITTED USES.** This GSLI License Agreement ("License") permits you to use one copy of the specified version of the GSLI software product identified above, which may include user documentation provided in "on line" or electronic form ("SOFTWARE"), on any single computer, provided the SOFTWARE is in use on only one computer at any time. The SOFTWARE is "in use" on a computer when it is loaded into the temporary memory (i.e. RAM) or installed into the permanent memory (e.g. hard disk, CD-ROM, or other storage device). If the anticipated number of users of the SOFTWARE will exceed the number of applicable Licenses, then you must have a reasonable mechanism or process in place to ensure that the number of persons using the SOFTWARE concurrently does not exceed the number of Licenses.
- 3. UPGRADES.** If the SOFTWARE is an upgrade from another product, whether from GSLI or another supplier, you may use or transfer the SOFTWARE only in conjunction with that upgraded product, unless you destroy it. If the SOFTWARE is an upgrade from a GSLI product, you now may use that upgraded product only in accordance with this License, except that if the SOFTWARE is an upgrade from a component of a package of software programs which you licensed as a single product, the SOFTWARE may be used and transferred only as part of that single product package and according to the License Agreement provided earlier with that single product package.
- 4. COPYRIGHT.** The SOFTWARE including any images, photographs, animation, video, audio, music and text incorporated into the SOFTWARE is owned by GSLI or its suppliers and is protected by United States copyright laws and international treaty provisions. Therefore, you must treat the SOFTWARE like any other copyrighted material except that you may either (a) make one copy of the SOFTWARE solely for backup or archival purposes, or (b) transfer the SOFTWARE to a single hard disk provided you keep the original solely for backup or archival purposes. You may not copy the printed materials accompanying the SOFTWARE, nor print copies of any user documentation provided in "on line" or electronic form.
- 5. OTHER RESTRICTIONS.** This License is your proof to exercise the rights granted herein and must be retained by you. You may not rent or lease the SOFTWARE, but you may transfer your rights under this License on a permanent basis provided you transfer this License, the SOFTWARE, and all accompanying printed materials, retain no copies and the recipient agrees to the terms of this License. You may not reverse engineer, decompile or disassemble the SOFTWARE, except to the extent that the foregoing restriction is expressly prohibited by applicable law. You may not electronically transfer the SOFTWARE. You may not transmit the SOFTWARE to countries prohibited by the US Federal Government without limitation.
- 6. CUSTOMER SUPPORT.** GSLI will attempt to answer your technical support request concerning the SOFTWARE, however, this service is offered on a reasonable efforts basis only, and GSLI may not be able to resolve every support request. GSLI supports the SOFTWARE only if it is used under conditions and on operating systems for which the SOFTWARE is designed.
- 7. LIMITED WARRANTY.** GSLI warrants that (a) the SOFTWARE will perform substantially in accordance with the accompanying printed materials for a period of ninety (90) days from the date of receipt; and (b) any hardware accompanying the SOFTWARE will be free from defects in materials and workmanship under normal use and service for a period of one (1) year from the date of receipt. Any implied warranties on the SOFTWARE and hardware are limited to ninety (90) days and one (1) year, respectively. Some states/jurisdictions do not allow limitations on duration of an implied warranty, so the above limitation may not apply to you.
- 8. CUSTOMER REMEDIES.** GSLI's entire liability and your exclusive remedy shall be, at GSLI's option, either (a) return of the price paid or (b) repair or replacement of the SOFTWARE or hardware that does not meet GSLI's Limited Warranty and that is returned to GSLI with a copy of your receipt. This Limited Warranty is void if failure of the SOFTWARE or hardware has resulted from accident, abuse or misapplication. Any replacement SOFTWARE will be warranted for the remainder of the original warranty period or thirty (30) days, whichever is longer. Outside the United States, neither these remedies nor any product support services offered by GSLI are available without proof of purchase from an authorized non-U.S. source.
- 9. NO OTHER WARRANTIES.** To the maximum extent permitted by applicable law, GSLI disclaims all other warranties, either express or implied, including but not limited to implied warranties of merchantability and fitness for a particular purpose, with respect to the SOFTWARE, the accompanying written materials, and any accompanying hardware. This limited warranty gives you specific legal rights. You may have others, which vary from state/jurisdiction to state/jurisdiction.
- 10. NO LIABILITY FOR CONSEQUENTIAL DAMAGES.** To the maximum extent permitted by applicable law, in no event shall GSLI or its suppliers be liable for any damages, whatsoever (including, without limitation, damages for loss of business profit, business interruption, loss of business information, or other pecuniary loss) arising out of the use or inability to use this GSLI product, even if GSLI has been advised of the possibility of such damages. Because some States/Jurisdiction do not allow the exclusion or limitation of liability for consequential or incidental damages, the above limitation may not apply to you.
- 11. ACKNOWLEDGMENT.** By opening the package containing SOFTWARE, Licensee acknowledges that it has read this Agreement, understood the same and agrees to be bound by its terms and conditions. Licensee also agrees that this Agreement supersedes all proposals or prior agreements, oral or written, and any communication between the parties relating to the subject matter of this Agreement.

U.S. GOVERNMENT RESTRICTED RIGHTS:

The SOFTWARE and documentation are provided with RESTRICTED RIGHTS. Use, duplication or disclosure by the Government is subject to restrictions as set forth in subparagraph (c)(1) and (2) of the Commercial Computer Software - Restricted Rights at 48 CFR 52.227-19, as applicable. Manufacturer is GSLI, 60 Fordham Road Wilmington, MA 01887. If any provision of this Agreement is found to be unlawful, void or unenforceable, then that provision shall be severed from this Agreement and will not affect the validity and enforceability of any of the remaining provisions. This Agreement shall be governed by the laws of the State of Massachusetts.

TABLE OF CONTENTS

INTRODUCTION.....	1
1. Warranty.....	2
2. Customer Support.....	2
3. List of All Commands.....	3
PC-MARK MT / PC-MARK COMMANDS	7
1. System Level Commands	7
unload.....	8
set_irq_number.....	8
set_card_base.....	9
home_beam	9
set_corr_table.....	11
ret_corr_table	11
set_head_type	12
ret_head_type.....	13
ret_head_status.....	14
set_corr_file.....	15
ret_corr_file.....	15
set_config_file.....	16
ret_config_file	16
get_config.....	17
set_param_file.....	17
ret_param_file.....	18
set_appl.....	18
ret_appl.....	18
set_message	19
ret_message.....	19
set_para_set.....	20
ret_para_set	21
ret_first_grp	22
ret_next_grp	22
save	23
release.....	23
system	24
mark	24
m_mark	25
ret_busy.....	25
set_laser.....	26
ret_laser.....	26
set_power_scale.....	27
ret_power_scale	27
set_target_velocity.....	28
2. User Interface Port Commands.....	29
start_mark	29

start_mark_on_begin	30
do_mark	30
end_mark	31
mark	31
remote_execute	32
set_mark_error	32
clear_mark_error	33
check_begin	33
check_abort	33
3. Group Commands	35
create	35
release	35
set_message.....	36
ret_message	36
ret_first_obj	37
ret_next_obj	38
save.....	38
mark.....	39
m_mark.....	39
mobi	40
ret_busy	40
4. General Object Commands.....	41
create	41
release	42
ret_typ	42
set_message.....	42
ret_message	43
set_para_set	43
ret_para_set.....	43
save.....	44
set_mark_flag	44
ret_mark_flag.....	45
mark.....	45
m_mark.....	45
mobi	46
ret_busy	46
vector_count	47
ret_il_ptr	47
ret_info	48
reset	48

5. Object Transformation Commands.....	49
x_flip	49
y_flip	49
rotate	50
slant	50
scale	50
offset.....	51
mul_matrix.....	51
ret_matrix.....	52
6. Vector Object Commands.....	53
set_source	53
ret_source.....	54
set_pen_flag.....	54
ret_pen_flag.....	55
set_pens	55
ret_pens.....	56
set_vector_wrap	56
7. Text Object Commands	57
set_font.....	57
ret_font	58
set_string.....	58
ret_string.....	58
set_char_height.....	59
ret_char_height.....	59
set_char_width	59
ret_char_width	60
set_char_slant	60
ret_char_slant.....	60
set_bx_char_map.....	61
8. Straight Text Object Commands.....	63
set_orientation	63
ret_orientation.....	63
9. Radial Text Object Commands.....	65
set_radius	65
ret_radius.....	65
set_direction	66
ret_direction.....	66
10. Barcode Object Commands	67
set_string.....	67
ret_string.....	68
set_x	68
ret_x.....	68
set_n.....	69
ret_n	69
set_bw	69

ret_bw	70
set_qz.....	70
ret_qz	70
set_t	71
ret_t.....	71
set_height.....	71
ret_height	72
set_density	72
ret_density.....	72
set_inverse_flag.....	73
ret_inverse_flag	73
set_check_code_flag	73
ret_check_code_flag.....	74

APPENDIX A: Brief on PC-MARK MT75

1. Discontinued Commands.....	75
2. Multitasking Commands	75
3. Master / Slave Option (Multiple Cards)	76
4. Multiple Correction Tables.....	77
5. Operation of JOB EDITOR.....	77
6. Hardware Stability	77
7. Future Commands	77

APPENDIX B: Commands Sorted by Alphanumeric.....78

PC-MARK MT / PC-MARK COMMAND REFERENCE

INTRODUCTION

The section includes for each command:

- A description of the command.
- The command syntax.
- The names, types and purposes of any arguments.
- Return values.
- Possible errors returned.

Under **Arguments** and **Return Values**, in columns type and units, there are several sorts of entries possible:

TYPE OF UNIT	RANGE
DOS (String)	Path and file name following DOS rules
Type	Object type - HPGL, MCL, STEXT, RTEXT, BC39, BC25I, BCUPC
Field	Unitless size in the range from -32768 to 32767 or 0 to 65535
Integer	Whole number with sign in the range from -32768 to 32767
Long Integer	Whole number with sign in the range from -2147483648 to 2147483647
Word	Whole number without sign in the range from 0 to 65535
Real	Floating point number
Flag	String, either Y or N
Byte	Whole number in the range from 0 to 255

This directory of commands has been organized by type of command as shown in the following table.

COMMAND GROUP	BEGINNING ON PAGE NO.
System Level	Page 7
User Interface Port	Page 29
Group	Page 35
General Object	Page 41
Object Transformation	Page 49
Vector Object	Page 53
Text Object	Page 57
Straight Text Object	Page 63
Radial Text Object	Page 65
Barcode Object	Page 67

1. Warranty

GSI Lumonics (GSLI) warrants this product to be free from defects in workmanship for 12 months from the date of shipment. GSLI will, replace the software if it is defective within the warranty period and returned, freight pre-paid, to a service center designated by GSLI.

GSI Lumonics requests that customers obtain a Return Authorization Number prior to returning the software, and that they carefully pack it in the original packing or equivalent.

2. Customer Support

GSI Lumonics has support services available to you concerning problems with the software or manual you are using.

Before calling for assistance, please make sure you refer to any appropriate sections in the manual that may answer your questions.

If you need further assistance:

The customer service personnel will be able to give you direct assistance and answers to your questions.



CALL

U.S. (Massachusetts):	(978) 661-4300 (in the U. S.) +01 978-661-4300 (outside the U. S)
Germany (Munich):	+49 89 899134-0
Italy (Monza):	+39 39 2025387
UK (Banbury):	+44 132-787-2424
Japan (Tokyo):	+81 3 3406 7990

... ask for the GMAX Customer Service Department

3. List of All Commands

The following two tables itemizes all the commands, the type of command, a brief description and the page where you can find detailed information on the command. The commands are grouped in this table according to the type of command. You will notice that there are some commands that are redundant from one group to another. For example, there are four **mark** commands:

- One a System command.
- One a User Interface Port command.
- One a Group command.
- One a General Object command.

For alphanumerically list of commands, see "Appendix B"

COMMAND	TYPE OF COMMAND	DESCRIPTION	PAGE No.
unload	System	Unload PC-MARK MT (or PC-MARK) to free memory.	8
set_irq_number ¹	System	Associate the HC/2 or HC/3 Master card with an IRQ.	8
set_card_base ¹	System	Associate the HC/2 card(s) with an I/O address.	9
home_beam ¹	System	Move the beam to a specified field coordinate.	9
set_corr_table ¹	System	Associate the HC/2 or HC/3 card number with a correction table number.	11
ret_corr_table ¹	System	Return the number of correction table for the specified HC/2 or HC/3 card number.	11
set_head_type ¹	System	Set the used Scan Head type.	12
ret_head_type ¹	System	Return the used Scan Head type.	13
ret_head_status ¹	System	Return the actual Head status with associated HC/2 or HC/3 card.	14
set_corr_file ¹	System	Cause the correction file to be associated with the specified table number.	12
ret_corr_file ¹	System	Return the name of correction file for the specified table number.	15
set_config_file	System	Set the grid correction file.	16
ret_config_file	System	Return name of config file set by last set_config_file command.	16
get_config	System	Return config data used to calculate the correction file.	17
set_param_file	System	Set name of laser parameter file.	17
ret_param_file	System	Return name of parameter file set by set_param_file command.	18
set_appl	System	Set the name of the application program.	18
ret_appl	System	Return name of current application set by set_appl.	18
set_message	System	Set a PC-MARK MT (or PC-MARK) message of up to 60 characters.	19
ret_message	System	Return the message set previously by set_message.	19
set_para_set	System	Set 1 of 64 sets of laser parameters.	20
ret_para_set	System	Return values of the laser parameters for a given set.	21
ret_first_grp	System	Return the name of the first PC-MARK MT (or PC-MARK) group.	22
ret_next_grp	System	Return the name of the next PC-MARK MT (or PC-MARK) group.	22
save	System	Append all groups and objects to the file specified.	23
release	System	Release all resources currently allocated for groups and objects.	23
system	System	Return system statistics.	24
mark	System	Execute all scannable objects.	24
m_mark ¹	System	Initiate the execution of all scannable objects.	25
ret_busy ¹	System	Return the status of the output during marking of a job.	25
set_laser ¹	System	Switch the laser beam on for an indefinite amount of time.	26
ret_laser ¹	System	Return the state of the laser following a set_laser.	26

set_power_scale	System	Compensate laser lamp power variations.	27
ret_power_scale	System	Return value of power scale factor.	27
set_target_velocity	System	Enable application to supply X & Y velocity of the object.	28
start_mark	User Interface Port	Start a scanning operation.	29
start_mark_on_begin	User Interface Port	Start scanning that will be gated by BEGIN_MARK signal.	30
do_mark	User Interface Port	Mark objects.	30
end_mark	User Interface Port	Issue after the last .do_mark command.	31
mark	User Interface Port	Combine .start_mark, .do_mark, .end_mark commands.	31
remote_execute	User Interface Port	Control the REMOTE_EXECUTE line of User Interface Port.	32
set_mark_error	User Interface Port	Use to assert the MARK_ERROR signal.	32
clear_mark_error	User Interface Port	Use to remove the MARK_ERROR signal.	33
check_begin²	User Interface Port	Return the sense of the BEGIN_MARK signal.	33
check_abort²	User Interface Port	Return the value of the MARK_ABORT signal.	33
create	Group	Create a group.	35
release	Group	Delete a group.	35
set_message	Group	Set a group message with up to 60 characters in length.	36
ret_message	Group	Return message previously set by command set_message.	36
ret_first_obj	Group	Return name of first object in the group.	37
ret_next_obj	Group	Return the name of the next object in the group.	38
save	Group	Save all objects to the path specified.	38
mark	Group	Execute all scannable objects in the group.	39
m_mark¹	Group	Begins marking the group and returns immediately.	39
mobi	Group	Mark immediately on begin all scannable objects in the group.	40
ret_busy¹	Group	Return the status of the output during marking of a job.	40
create	General Object	Create the object with the given object name.	41
release	General Object	Delete the named object.	42
ret_typ	General Object	Return the type of the object.	42
set_message	General Object	Set the object message.	42
ret_message	General Object	Return message set previously by set_message command.	43
set_para_set	General Object	Set a pointer to 1 of the 64 parameter sets.	43
ret_para_set	General Object	Return number of the parameter set used to execute the object.	43
save	General Object	Append the object to the file specified.	44
set_mark_flag	General Object	Set the execute flag for the object.	44
ret_mark_flag	General Object	Return the state of the execute flag for the object.	45
mark	General Object	Execute the object if the object is scannable.	45
m_mark¹	General Object	Begins marking the object and returns immediately.	45
mobi	General Object	Mark immediately on begin the object if the object is scannable.	46
ret_busy¹	General Object	Return the status of the output during marking of a job.	46
vector_count	General Object	Return the number of vectors in the object.	47
ret_il_ptr	General Object	Return a memory pointer to a list of vectors for the object.	47
ret_info	General Object	Return information about the given object.	48
reset	General Object	Reset the object transformation matrix to the units matrix.	48
x_flip	Object Transform	Mirror the object around the Y axis.	49
y_flip	Object Transform	Mirror the object around the X axis.	49
rotate	Object Transform	Rotate the object by the given angle.	50
slant	Object Transform	Skew object in X dimension while leaving Y dimension alone.	50
scale	Object Transform	The object will be magnified by the given scale factors.	50
offset	Object Transform	Translate the object by the given X and Y offset (relative).	51
mul_matrix	Object Transform	Multiply the objects transformation matrix by the given matrix.	51
ret_matrix	Object Transform	Return the objects transformation matrix.	52
set_source	Vector Object	Set the path name for the vector object.	53
ret_source	Vector Object	Return the path name for the vector object.	54
set_pen_flag	Vector Object	Pen changing on vector objects having internal pen changes.	54
ret_pen_flag	Vector Object	Return the state of the pen flag.	55
set_pens	Vector Object	Map HPGL and MCL pens to laser parameter sets.	55
ret_pens	Vector Object	Return current mapping of the pens to laser parameter sets.	56
set_vector_wrap	Vector Object	Set the Vector Wrapping Option	56

GMAX PC-MARK MT / PC-MARK

set_font	Text Object	Font to use when rendering the text for the object.	57
ret_font	Text Object	Return the font of the object.	58
set_string	Text Object	Specify the string of text that will be used.	58
ret_string	Text Object	Return the string of text that will be used.	58
set_char_height	Text Object	Set the height for text characters.	59
ret_char_height	Text Object	Return the height for text characters.	59
set_char_width	Text Object	Set the ratio of width to height for text characters.	59
ret_char_width	Text Object	Return the ratio of width to height for text characters.	60
set_char_slant	Text Object	Change amount of slant for each character of a text object.	60
ret_char_slant	Text Object	Return amount of slant for each character of a text object.	60
set_bx_char_map ²	Text Object	Set character mapping for Bitstream Fonts	61
set_orientation	Straight Text Object	Specify whether text will be laid out horizontal or vertical.	63
ret_orientation	Straight Text Object	Return the text orientation set by set_orientation .	63
set_radius	Radial Text Object	Set the radius for the radial text.	65
ret_radius	Radial Text Object	Return the radius of the radial text.	65
set_direction	Radial Text Object	Radial text will be clockwise or counter clockwise.	66
ret_direction	Radial Text Object	Return the direction set by the set_direction command.	66
set_string	BarCode Object	Specify the string that is to be barcoded.	67
ret_string	BarCode Object	Return the string that is to be barcoded.	68
set_x	BarCode Object	Specify the Narrow Element dimension of the barcode.	68
ret_x	BarCode Object	Return the Narrow Element dimension of the barcode.	68
set_n	BarCode Object	Specify the wide to narrow ratio for the barcode.	69
ret_n	BarCode Object	Return the wide to narrow ratio for the barcode.	69
set_bw	BarCode Object	Specify the black (imprint) to white (non-imprint) ratio.	69
ret_bw	BarCode Object	Return the black to white ratio.	70
set_qz	BarCode Object	Set the barcode quiet zone.	70
ret_qz	BarCode Object	Return the settings of the barcode quiet zone.	70
set_t	BarCode Object	Set the space between characters.	71
ret_t	BarCode Object	Return the space between characters.	71
set_height	BarCode Object	Specify the height of the barcode in field units.	71
ret_height	BarCode Object	Return the height of the barcode in field units.	72
set_density	BarCode Object	Set number of steps between fill bars when filling a black bar.	72
ret_density	BarCode Object	Return number of steps set by the set_density command.	72
set_inverse_flag	BarCode Object	Set whether barcode should be executed normally or inverted.	73
ret_inverse_flag	BarCode Object	Return setting set by the set_inverse_flag command.	73
set_check_code_flag	BarCode Object	Set toggle for 3 of 9 check code to be added.	73
ret_check_code_flag	BarCode Object	Determine whether or not to add 3 of 9 check code.	74

¹ PC-MARK MT Only.

² Not available for PC-MARK MT.

PC-MARK MT / PC-MARK COMMANDS

1. System Level Commands

The following table defines the System Level commands with a brief description on each command.

Command	Type of Command	Description
unload	System	Unload PC-MARK MT (or PC-MARK) to free memory.
set_irq_number ¹	System	Associate the HC/2 or HC/3 Master card with an IRQ.
set_card_base ¹	System	Associate the HC/2 card(s) with an I/O address.
home_beam ¹	System	Move the beam to a specified field coordinate.
set_corr_table ¹	System	Associate the HC/2 or HC/3 card number with a correction table number.
ret_corr_table ¹	System	Return the number of correction table for the specified HC/2 or HC/3 card number.
set_head_type ¹	System	Set the used Scan Head type.
ret_head_type ¹	System	Return the used Scan Head type.
ret_head_status ¹	System	Return the actual Head status with associated HC/2 or HC/3 card.
set_corr_file ¹	System	Cause the correction file to be associated with the specified table number.
ret_corr_file ¹	System	Return the name of correction file for the specified table number.
set_config_file	System	Set the grid correction file.
ret_config_file	System	Return name of config file set by last set_config_file command.
get_config	System	Return config data used to calculate the correction file.
set_param_file	System	Set name of laser parameter file.
ret_param_file	System	Return name of parameter file set by set_param_file command.
set_appl	System	Set the name of the application program.
ret_appl	System	Return name of current application set by set_appl.
set_message	System	Set a PC-MARK MT (or PC-MARK) message of up to 60 characters.
ret_message	System	Return the message set previously by set_message.
set_para_set	System	Set 1 of 64 sets of laser parameters.
ret_para_set	System	Return values of the laser parameters for a given set.
ret_first_grp	System	Return the name of the first PC-MARK MT (or PC-MARK) group.
ret_next_grp	System	Return the name of the next PC-MARK MT (or PC-MARK) group.
save	System	Append all groups and objects to the file specified.
release	System	Release all resources currently allocated for groups and objects.
system	System	Return system statistics.
mark	System	Execute all scannable objects.
m_mark ¹	System	Initiate the execution of all scannable objects.
ret_busy ¹	System	Return the status of the output during marking of a job.
set_laser ¹	System	Switch the laser beam on for an indefinite amount of time.
ret_laser ¹	System	Return the state of the laser following a set_laser.
set_power_scale	System	Compensate laser lamp power variations.
ret_power_scale	System	Return value of power scale factor.
set_target_velocity	System	Enable application to supply X & Y velocity of the object.

¹ PC-MARK MT Only.

unload

Description

This command causes **MMARKIT** (or **MARKIT**) to completely clear all its allocated XMS¹ memory blocks, and then to remove itself from the program area, thus releasing about 170 KByte of memory (the exact amount depends on the start-up parameters)

¹⁾ EMS for PC-MARK

Remark

This command does not remove **MMCL.exe** (or **MCL.exe**) from memory. Refer to "Installation" and the section "Unloading or Terminating **PC-MARK MT** (or **PC-MARK**)", on how to remove **MMCL.exe** (or **MCL.exe**).

Syntax

Command: **.unload**

Reply: **0 :**

Arguments

None

Return Values

None

Errors

0

set_irq_number



This command is available in **PC-MARK MT (Multitasking)** only.

Description

This command is used to set the IRQ of the **HC/2** card (or **HC/2 Master**) which is selected by jumper options on the **HC/2**. For more information on IRQ settings see the "HC/2 Reference Manual".

In case of the **HC/3** card there is no jumpers for interrupt levels. PCI BIOS reserves IRQ for every PCI card installed in the computer. This command just activates Plug and Play capability of **HC/3** card and reads IRQ level reserved by BIOS for **HC/3** card. The **irq number** is ignored and IRQ level reserved by BIOS is assigned to **HC/3** card.

Syntax

Command: **.set_irq_number [irq number]**

Reply: **0 :**

For **HC/2**:

Arguments

Name	Type	Range	Function
irq number		see HC/2 manual	IRQ selected on the HC/2

For **HC/3**:

Arguments

Name	Type	Range	Function
irq number		Any value	IRQ reserved by BIOS is assigned to HC/3

Return Values

None

Errors

0

set_card_base



This command is available in PC-MARK MT (Multitasking) only.

Description

This command is used to set a base address for each **HC/2** card which is selected by jumper options on the **HC/2**. For more information on base address settings see the “HC/2 Hardware Manual”.

In case of the **HC/3** card there are no jumpers for base address. PCI BIOS reserves IO addresses for every PCI card installed in the computer. This command just activates Plug and Play capability of **HC/3** card and reads base addresses reserved by BIOS for **HC3** cards. The **HC/2 number** and **hex base address** arguments are ignored. In case of **HC/3** Master/Slave configuration card number is defined by cabling. For more information on base address settings see the “HC/3 Hardware Manual”.

Syntax

Command: **.set_card_base [HC/2 number] [hex base address]**

Card Number	HC/2 Card
0	Single or Master HC/2
1	Second HC/2
2	Third HC/2
3	Fourth HC/2

Reply: 0 :

For **HC/2**:

Arguments

Name	Type	Range	Function
HC/2 number		0-3	Number of the HC/2.
hex base addr.		see HC/2 manual	Base address of the HC/2

For **HC/3**:

Arguments

Name	Type	Range	Function
HC/2 number		Any value	Number defined by cabling is assigned
hex base addr.		Any value	Base address reserved by BIOS is assigned

Return Values

None

Errors

0

home_beam



This command is available in PC-MARK MT (Multitasking) only.

Description

Move the beam to the field position defined by the variables X,Y. The speed is defined by the currently active (or default if none had been set) JS and SP values.

Syntax

Command: `.home_beam [x-coordinate] [y-coordinate]`

Reply: 0 :

Arguments

Name	Type	Units	Range	Function
X-coordinate	Integer	Field	-32767 - 32767	X position
Y-coordinate	Integer	Field	-32767 - 32767	Y position

Return Values

None

Errors

0

set_corr_table



This command is available in PC-MARK MT (Multitasking) only.

Description

This command associates a **HC/2** or **HC/3** with a correction table number. The table number is used to be associated with a correction file.

Remark

The number of tables that can be used is determined by the -t parameter of **MMARKIT**. See “Description of MMARKIT Parameters” in the Programmers Manual for details.

Syntax

Command: **.set_corr_table [HC/2 number] [table number]**

Reply: **0 :**

Arguments

Name	Type	Range	Function
HC/2 number		0-3	Number of the HC/2 or HC/3 (0 for single/master card).
table number		0-3	Correction table number

Return Values

None

Errors

0	101	105	109	110
---	-----	-----	-----	-----

ret_corr_table



This command is available in PC-MARK MT (Multitasking) only.

Description

This command returns the associate table number of the specified **HC/2** or **HC/3** number.

Syntax

Command: **.ret_corr_table [HC/2 number]**

Reply: **0 : [table number]**

Arguments

Name	Type	Range	Function
HC/2 number		0-3	Number of the HC/2 or HC/3 (0 for single/master card).

Return Values

Name	Type	Range	Function
table number		0-3	Correction table number

Errors

0

set_head_type



This command is available in PC-MARK MT (Multitasking) only.

Description

This command is used to set the correct Scan Head type. Normally this command will be inserted into the **MMCL.ini**.

REMARK: This command is only used to support the **.ret_head_status** and doesn't influence the software or the Scan Head.

Syntax

Command: **.set_head_type [type]**

Type	Scan Head
2	HCI (HPLK System)
3	HPM15M2
4	HMP10A
8	HPM10G3
16	HPM10R
6,7,12	reserved

Reply: 0 :

Arguments

Name	Type	Range	Function
head type		2,3,4,6,7	Type of Scan Head

Return Values

None

Errors

0	0101
---	------

ret_head_type



This command is available in PC-MARK MT (Multitasking) only.

Description

This command returns the used Scan Head type.

Syntax

Command: **.ret_head_type**

Reply: **0** : (see table)

Type	Scan Head
0	no Scan Head type configured
2	HCI (HPLK System)
3	HPM15M2
4	HMP10A
8	HPM10G3
16	HPM10R
6,7,12	reserved

Arguments

None

Return Values

Name	Type	Range	Function
head type		2,3,4,6,7	Type of Scan Head

Errors

0

ret_head_status



This command is available in PC-MARK MT (Multitasking) only.

Description



NOTE

This command is used to get the actual Scan Head status.

Before you can use this command you have to specify the correct Scan Head type with the `.set_head_type` command.

Syntax

Command: `.ret_head_status[n]`

Reply: `0 :`

Arguments

Name	Type	Range	Function
n		0-3	Card Number of the connected HC/2.or HC/3

Return Values

Name	Type	Range	Function
o	Flag	0/1	[0] means the galvos are out of the field or if the Scan Head speed has been forced to high, causing the electronics to shut down. This condition can last several minutes. [1] means that the power supply to the Scan Head and the general condition of the Head is OK.
t	Flag	0/1	[0] means the galvo temperature controller has not reached its stable operating point, so the position error may be higher. [1] means the galvo temperature controller has reached its stable operating point with minimum position error.
w	Flag	0/1	[0] signifies that the galvos have not settled in the window where the position error has reached a minimum set point value. [1] signifies that the galvos have settled in the window where the position error has reached a minimum set point value.

Errors

0	126	0101	1003	1005
---	-----	------	------	------

set_corr_file



This command is available in PC-MARK MT (Multitasking) only.

Description

This command causes a specified grid correction file to be associated with the specified correction table number.

Remark

A description of the correction file format can be found in the Programmer's Manual.

Syntax

Command: **.set_corr_file [correction_file] [table number]**

Reply: **0 :**

Arguments

Name	Type	Range	Function
correction_file	String	DOS, path and file name	The files include information for field correction.
table number		0-3	Correction table number of the associate HC/2.or HC/3

Return Values

None

Errors

0	101	105	109	110
---	-----	-----	-----	-----

ret_corr_file



This command is available in PC-MARK MT (Multitasking) only.

Description

This command will return the name of the correction file which is associate with the specified table number.

Syntax

Command: **.ret_corr_file [table number]**

Reply: **0 :**

Arguments

Name	Type	Range	Function
table number		0-3	Correction table number of the associate HC/2.or HC/3

Return Values

Name	Type	Range	Function
correction_file	String	DOS, path and file name	The files include information for field correction.

Errors

0	101	105	109	110
---	-----	-----	-----	-----

set_config_file

Description

This command allows you to set the grid correction file. The grid file must be in GS format. Two files are actually read, one with the extension `.inf` and one with the extension `.asc`.



NOTE

The application program has to make sure that the `.set_config_file` command was successful (no error).

Remark

A description of the correction file format can be found in Programmer's Manual.

Syntax

Command: `.set_config_file [correction_file]`

Reply: 0 :

Arguments

Name	Type	Range	Function
correction_file	String	DOS, path and file name	The files include information for field correction.

Return Values

None

Errors

0

ret_config_file

Description

This command will return the name of the config file set by the last `set_config_file` command.



NOTE

The name which was set last will be returned, even if the `set_config_file` command returned an error because the file was not found.

Syntax

Command: `.ret_config_file`

Reply: 0 : [config_file]

Arguments

None

Return Values

Name	Type	Range	Function
config_file	String	DOS, file name	The file name of the correction file

Errors

0

get_config

Description

This command returns the configuration data which were used to calculate the correction file.

Syntax

Command: **.get_config**

Reply: **0 : [angle] [d] [e] [f] [m] [comment]**

Arguments

None

Return Values

Name	Type	Units	Range	Function
angle	Real	°(deg)	0 to 180	Optical scan angle.
d	Real	meters	0 to max Real	Distance between the Y-mirror and the marking field.
d	Real	meters	0 to max Real	Distance between the scan mirrors.
f	Real	meters	0 to max Real	Length and width of the marking field.
m	Real	meters	0 to max Real	Calibration margin
comment	String			Comment text describing configuration.

Errors

0	112
---	-----

set_param_file

Description

This command sets the name of the laser parameter file. No files are read as a result of this command. This command can be used by applications that need to communicate this information to another application.

Syntax

Command: **.set_param_file [para_file]**

Reply: **0 :**

Arguments

Name	Type	Range	Function
para_file	String	DOS, path and file name	Name of parameter file.

Return Values

None

Errors

0	101	109	110
---	-----	-----	-----

ret_param_file

Description

This command will return the name of the parameter file as set by the `set_param_file` command.

Syntax

Command: `.ret_param_file`

Reply: `0 : [para_file]`

Arguments

None

Return Values

Name	Type	Range	Function
para_file	String	DOS, file name	Name of the parameter file

Errors

0	101	109	110
---	-----	-----	-----

set_appl

Description

This command is used to set the name of the application program. The application program is one that will handle the special details of the particular job.

Syntax

Command: `.set_appl [application]`

Reply: `0 :`

Arguments

Name	Type	Range	Function
application	String	DOS, path and file name	Name of the application

Return Values

None

Errors

0	101	109	110
---	-----	-----	-----

ret_appl

Description

This command will return the name of the current application program as set by the `set_appl` command.

Syntax

Command: `.ret_appl`

Reply: `0 : [application]`

Arguments

None

Return Values

Name	Type	Range	Function
application	String	DOS, file name	application file

Errors

0	101	109	110
---	-----	-----	-----

set_message

Description

This command allows the setting of the application message. The message is a string of up to 60 characters in length that can be used by any application. It is not interpreted in any way by **PC-MARK MT** (or **PC-MARK**).

Syntax

Command: **.set_message [message]**

Reply: **0 :**

Arguments

Name	Type	Range	Function
message	String	0 - 60 characters	Application message

Return Values

None

Errors

0	101	109	110
---	-----	-----	-----

ret_message

Description

This command returns the message as set previously by the **set_message** command.

Syntax

Command: **.ret_message**

Reply: **0 : [message]**

Arguments

None

Return Values

Name	Type	Range	Function
message	String	0 - 60 characters	Application message.

Errors

0	101	109	110
---	-----	-----	-----

set_para_set

Description

The command allows the setting of the laser parameters for 1 of the 64 sets of laser parameters.

Syntax

Command: **.set_para_set (see table)**

Reply: **0 :**

Arguments

Name	Units	Default	Range	Function
mark step size	LSB	50	0 to 65535	Micro-vector size for imprinting vectors.
jump step size	LSB	100	0 to 65535	Micro-vector size for non-imprinting vectors.
step period	μsec	168	0 to 65535	Micro-vector output rate.
mark delay	μsec	500	0 to 65535	End of imprinting vector delay.
jump delay	μsec	500	0 to 65535	End of non-imprinting vector delay.
Stroke delay	μsec	500	0 to 65535	End of imprinting vector sequence delay.
laser on delay	μsec	500	0 to 65535	Turn on delay for laser at start of stroke.
laser off delay	μsec	500	0 to 65535	Turn off delay for laser at end of stroke.
q-switch width	μsec	500	0 to 65535	Pulse width of Q-switch pulse.
q-switch period	μsec	500	0 to 65535	Rep-rate of the Q-switch pulse train.
laser power	bytes	50	0 to 255	Digital output to control laser power.
autosegmentation		1	1 or 0	Enable/disable autosegmentation.
break angle	°(deg)	180	0 to 180	Included angle that breaks a stroke.
dither width	LSB	0	0 to 65535	Diameter of the circle described by the laser.
dither feed	LSB	0	0 to 65535	Distance between subsequent circles. Defines the rate of feed.
power change delay	msec	0	0 to 65535	Settle time for laser power supply.
FPS		0	1 or 0	Enable first pulse suppression if required.
message			0-235 characters	String of characters.

Return Values

None

Errors

0	101	109	110
---	-----	-----	-----

ret_para_set

Description

This command returns the values of the laser parameters for a given set.

Syntax

Command: **.ret_para_set [number]**

Reply **0** : (see table)

Arguments

None

Return Values

Name	Units	Range	Function
mark step size	LSB	0 to 65535	Micro-vector size for imprinting vectors.
jump step size	LSB	0 to 65535	Micro-vector size for non-imprinting vectors.
step period	μsec	0 to 65535	Micro-vector output rate.
mark delay	μsec	0 to 65535	End of imprinting vector delay.
jump delay	μsec	0 to 65535	End of non-imprinting vector delay.
stroke delay	μsec	0 to 65535	End of imprinting sequence delay.
laser on delay	μsec	0 to 65535	Turn on delay for laser at start of stroke.
laser off delay	μsec	0 to 65535	Turn off delay for laser at end of stroke.
q-switch width	μsec	0 to 65535	Pulse width of Q-switch pulse.
q-switch period	μsec	0 to 65535	Rep-rate of the Q-switch pulse train.
laser power		0 to 255	Digital output to control laser power.
autosegmentation		1 or 0	Enable/disable autosegmentation.
break angle	°(deg)	0 to 180	Included angle that breaks a stroke.
dither width	LSB	0 to 65535	Diameter of the circle described by the laser.
dither feed	LSB	0 to 65535	Distance between subsequent circles. Defines the rate of feed.
power change delay	ms	0 to 65535	Settle time for laser power supply.
FPS		1 or 0	Enable first pulse suppression if required.
message		0-235 characters	String of characters.

Errors

0	101	102	109	110
----------	------------	------------	------------	------------

ret_first_grp

Description

This command will return the name of the first group. This command, when used in conjunction with `ret_next_grp`, allows the application to determine the names of all the groups.

Syntax

Command: `.ret_first_grp`

Reply: `0 : [grp_name]`

Arguments

None

Return Values

Name	Type	Range	Function
grp_name	String	1 - 8 characters	Name of a group.

Errors

0	107
---	-----

ret_next_grp

Description

This command will return the name of the next group. Repeating this command until error code 107 is returned will return the names of all the groups.

Syntax

Command: `.ret_next_grp`

Reply: `0 : [grp_name]`

Arguments

None

Return Values

Name	Type	Range	Function
grp_name	String	1 - 8 characters	Name of a group.

Errors

0	107
---	-----

save

Description

This command will append all groups and objects to the file specified.

Remark

The file is not created, so if it currently does not exist, the save will fail.

Syntax

Command: **.save [path]**

Reply: 0 :

Arguments

Name	Type	Range	Function
path	String	DOS, existing file	Path and file name.

Return Values

None

Errors

0	101	105	109	110
---	-----	-----	-----	-----

release

Description

This command will delete all groups and objects and release all resources currently allocated for groups and objects.

Remark

It is a good idea to issue this command before starting a new job.

Syntax

Command: **.release**

Reply: 0 :

Arguments

None

Return Values

None

Errors

0

system

Description

This command will return system statistics to the application.

Syntax

Command: **.system**

Reply: **0** : [mem_avail] [mcl_mem] [version]
 [laser_stat.] [vec_mem_avail] [vec_mem_total]

Arguments

None

Return Values

Name	Type	Units	Range	Function
mem_avail	long Integer	bytes	0 - 640000	The number of bytes available to MMARKIT (or MARKIT) for storing objects.
mcl_mem	long Integer	bytes	0 - 640000	The number of bytes available for use by MMCL (or MCL) for its internal buffer.
version	String	Date	Date	The date that MMARKIT (or MARKIT) was built.
laser_stat.	Integer	TBD	TBD	Currently not used.
vec_mem_av.	long Integer	5 bytes Vector	0 - space on EMS / DISK ¹⁾	Space remaining for this many vectors.
vec_mem_tot	long Integer	5 bytes Vector	0 - space in XMS ¹⁾	Total amount of vector memory allocated.

¹⁾ EMS / DISK for PC-MARK

Errors

0

mark

Description

This command will execute all scannable objects.

Syntax

Command: **.mark**

Reply: **0** :

Arguments

None

Return Values

None

Errors

0	111	112	115	117	118	121	123
---	-----	-----	-----	-----	-----	-----	-----

m_mark



This command is available in PC-MARK MT (Multitasking) only.

Description

This command will set MARK_IN_PROGRESS signal true and initiate execution of all scannable objects. It returns immediately to allow the application program to continue other tasks while marking is done in the background.

Must be called ONCE ONLY before calling `.ret_busy` .

Syntax

Command: `.m_mark`

Reply: 0 :

Arguments

None

Return Values

None

Errors

0	104	105	110	111	112	115	116	117	118	121	123
---	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

ret_busy



This command is available in PC-MARK MT (Multitasking) only.

Description

This command returns the status of the output during marking of a job.

It must be called at least once although it may be called as often as required following an initial `.m_mark` .

If marking has finished , `.ret_busy` causes the MARK_IN_PROGRESS signal to go false.

Syntax

Command: `.ret_busy`

Reply if there are more vectors: 0 : [Y]

Reply if there are no more vectors and marking is completely finished: 0 : [N]

Arguments

None

Return Values

None

Errors

0	104	105	110	111	112	115	116	117	118	121	123
---	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

set_laser

MT

This command is available in PC-MARK MT (Multitasking) only.

Description

This command switches the laser beam on or off. If the argument is set to [1] the laser beam is switched on until a command follows which causes the laser to be switched off.

The laser will also be turned off by a `.mark_abort` either received from the hardware or through a software command.

Syntax

Command: `.set_laser [on/off]`

Reply: `0 :`

Arguments

Name	Type	Range	Function
on	Flag	0	Laser beam is on.
off	Flag	1	Laser beam is on.

Return Values

None

Errors

0

ret_laser

MT

This command is available in PC-MARK MT (Multitasking) only.

Description

This command returns the state of the laser (on/off).

If the `.ret_laser` command is called during marking, it will not show the state of the laser due to that mark.

If the laser has been switched off by a `.mark_abort`, then the `.ret_laser` will correctly show the state to be [1].

Syntax

Command: `.ret_laser`

Reply: `0 : (see table)`

Arguments

None

Return Values

Name	Type	Function
0	Flag	Laser beam is on.
1	Flag	Laser beam is off.

Errors

0

set_power_scale

Description

This command makes it possible to compensate for laser lamp power variations (due to aging, etc.), without the need to change Scribe parameters in individual jobs.

If you choose a factor of 2.0, for instance, then the actual value output to the laser interface will be twice that given in the current Scribe. The output value is limited to the normal control range.

The power scale held internally defaults to 1.0, so that existing Jobs are not affected.

Syntax

Command: `.set_power_scale [factor]`

Reply: `0 :`

Arguments

The value of **factor** can be from 0.0 to 65.0

Return Values

None

Errors

ret_power_scale

Description

This command will return the value of the power scale factor set by `set_power_scale`.

Syntax

Command: `.ret_power_scale`

Reply: `0 :`

Arguments

None

Return Values

None

Errors

set_target_velocity

Description

This command allows one to follow and execute an object moving with constant velocity through the scanning field such as items on a conveyer-belt.

This command enables the application to supply the X and Y velocities of the object. Before starting to execute a given object, the application informs **PC-MARK MT** (or **PC-MARK**) of the object's velocity in the X and Y directions. **PC-MARK MT** (or **PC-MARK**) in turn sends the information to the based **MMCL** (or **MCL**) interpreter. As the execution proceeds, **MMCL** (or **MCL**) uses the velocities and the time elapsed to follow the object.

Syntax

Command: `.set_target_velocity [x] [y]`

Reply: `0 :`

Arguments

Both X and Y parameters are REQUIRED, and represent the velocities in MARKING-FIELD-BITS PER SECOND. Either or both may be zero, and both have the range of pascal longint.

The velocity values can be calculated fairly simply, given the following...
 the scanning field size in bits, **FB** = 65536 bits in both X and Y directions;
 the scanning field size AT THE OBJECT in meters, **F_{obj}**
 the object's actual velocity in meters/second, **V_{obj}**;
 ...then the target velocity **V_{targ}** in bits/second is given by

$$V_{\text{targ}} = \text{FB} * V_{\text{obj}} / F_{\text{obj}} .$$

The command may be issued to **PC-MARK MT** (or **PC-MARK**) as often as needed. **MMCL** (or **MCL**) automatically resets its tracking counters to zero, but note that the latest target velocities remain in effect across imprints: i.e., it **IS** necessary to issue...

`.set_target_velocity [0] [0]`

...when you wish to execute a stationary object after having executed some moving objects (unless, of course, you reload the **PC-MARK MT** (or **PC-MARK**) software).

Return Values

None

Errors

0

2. User Interface Port Commands

The following table defines the User Interface Port commands with a brief description on each command.

Command	Type of Command	Description
start_mark	User Interface Port	Start a scanning operation.
start_mark_on_begin	User Interface Port	Start scanning that will be gated by BEGIN_MARK signal.
do_mark	User Interface Port	Mark objects.
end_mark	User Interface Port	Issue after the last .do_mark command.
mark	User Interface Port	Combine .start_mark, .do_mark, .end_mark commands.
remote_execute	User Interface Port	Control the REMOTE_EXECUTE line of User Interface Port.
set_mark_error	User Interface Port	Use to assert the MARK_ERROR signal.
clear_mark_error	User Interface Port	Use to remove the MARK_ERROR signal.
check_begin ²	User Interface Port	Return the sense of the BEGIN_MARK signal.
check_abort ²	User Interface Port	Return the value of the MARK_ABORT signal.

start_mark

Description

This command will execute an operation. This command (or **.start_mark_on_begin** must be issued before a **.do_mark** command will be accepted. It is an error to issue a **.do_mark** command before a **.start_mark** (or **.start_mark_on_begin**) command has been issued.

The **MARK_IN_PROGRESS** signal will be asserted when **.start_mark** is issued. If **MARK_ABORT** has been asserted when **.start_mark** is issued then **.start_mark** will quit without asserting **MARK_IN_PROGRESS** and return with an error status. It will still be necessary to issue an **.end_mark** command before another **.start_mark** command will be allowed.

Syntax

Command: **.start_mark**

Reply: **0 :**

Arguments

None

Return Values

None

Errors

0

start_mark_on_begin

Description

`.start_mark_on_begin` is issued to start executing an operation that will be gated by the **BEGIN_MARK** signal. `.start_mark_on_begin` takes a **timeout** argument that specifies how long **PC-MARK MT** (or **PC-MARK**) should wait for the **BEGIN_MARK** signal to be asserted before returning an error.

Syntax

Command: `.start_mark_on_begin [timeout]`

Reply: 0 :

Arguments

The **timeout** argument is in units of seconds and has a range of 0 to 65535 seconds (over 18 hours). A **timeout** value of 0 has the special meaning of "wait forever"

PC-MARK MT (or **PC-MARK**) will "hang" if **BEGIN_MARK** is never asserted.

**NOTE***Return Values*

None

Errors

0

do_mark

Description

This command is used to execute objects. It will return immediately with an error if either `.start_mark` or `.start_mark_on_begin` has not been issued. `.do_mark` will quit execution and return with an error if either **MARK_ABORT** or **LASER_FAULT** has been asserted.

Syntax

Command: `.do_mark`

Reply: 0 :

Arguments

None

Return Values

None

Errors

0

end_mark

Description

This command must be issued after the last `.do_mark` of a series of `.do_mark` commands has been issued. It must also be issued if `.start_mark` or `.start_mark_on_begin` returns an error code. If a `.start_mark` was used to begin the session then `.end_mark` will simply cause `MARK_IN_PROGRESS` to be removed. If a `.start_mark_on_begin` was used to begin the session then `.end_mark` will wait for `BEGIN_MARK` to be removed and will then remove `MARK_IN_PROGRESS`. If `BEGIN_MARK` has not been removed within 30 seconds after issuing the `.end_mark` command `MARK_IN_PROGRESS` will be removed unconditionally and `.end_mark` will return an error code.

If `MARK_ABORT` is asserted then `.end_mark` will unconditionally remove `MARK_IN_PROGRESS` and return an error status.

The external system may remove `MARK_ABORT` any time after `MARK_IN_PROGRESS` has been removed. If another `.start_mark` or `.start_mark_on_begin` is issued before `MARK_ABORT` is removed then that imprint will be aborted too.

Syntax

Command: `.end_mark`

Reply: 0 :

Arguments

None

Return Values

None

Errors

0

mark

Description

This command performs a combination of the `.start_mark`, `.do_mark`, and `.end_mark` commands.

Syntax

Command: `.mark`

Reply: 0 :

Arguments

None

Return Values

None

Errors

0

remote_execute

<i>Description</i>	This command is used to manage the REMOTE_EXECUTE line of the User Interface Port.	
<i>Syntax</i>	Command: .remote_execute [sig_type] [pulse_width]	
	Reply: 0 :	
<i>Arguments</i>	<p>sig_type: may take on the values T to assert the signal, F to remove the signal, P to pulse the signal.</p> <p>pulse_width: ignored for signal types of T and F. Must be in the range of 0 to 65535 μs for signal type P (pulse). The pulse width is limited by the software overhead and is on the order of 50 μs.</p>	
<i>Examples</i>	<pre>.remote_execute [T] [0] assert the REMOTE_EXECUTE signal .remote_execute [F] [0] remove the REMOTE_EXECUTE signal .remote_execute [P] [250] pulse the REMOTE_EXECUTE signal for 250 μs</pre>	
<i>Return Values</i>	None	
<i>Errors</i>	<table border="1"><tr><td>0</td></tr></table>	0
0		

set_mark_error

<i>Description</i>	This command can be used by the application to assert the MARK-ERROR signal if the application detects an application-specific error that requires the process to be terminated. The application is not required to assert this signal when a command returns an error code. It is up to the application to determine what application errors must be signaled to other components.	
<i>Syntax</i>	Command: .set_mark_error	
	Reply: 0 :	
<i>Arguments</i>	None	
<i>Return Values</i>	None	
<i>Errors</i>	<table border="1"><tr><td>0</td></tr></table>	0
0		

clear_mark_error

<i>Description</i>	This command can be used to remove the MARK_ERROR signal by the application. Note that .end_mark will remove MARK_ERROR if it was previously asserted so the application need only issue this command for application-specific timing requirements.
<i>Syntax</i>	Command: .clear_mark_error Reply: 0 :
<i>Arguments</i>	None
<i>Return Values</i>	None
<i>Errors</i>	<input type="text" value="0"/>

check_begin



This command is not available in PC-MARK MT (Multitasking).

<i>Description</i>	This command returns the sense of the BEGIN_MARK signal.
<i>Syntax</i>	Command: .check_begin Reply: 0 :
<i>Arguments</i>	None
<i>Return Values</i>	None
<i>Errors</i>	<input type="text" value="0"/>

check_abort



This command is not available in PC-MARK MT (Multitasking).
It returns error code 1022 "MCL - Bad Opcode"

<i>Description</i>	This command returns the value of the MARK_ABORT signal. The command return value (not the error code) will be 0 if MARK_ABORT is not asserted (high) and will be 1 if MARK_ABORT is asserted (low). The application software does not need to use this command but may take advantage of it if desired.
<i>Syntax</i>	Command: .check_abort Reply: 0 :
<i>Arguments</i>	None
<i>Return Values</i>	None
<i>Errors</i>	<input type="text" value="0"/>

3. Group Commands

The following table defines the Group commands with a brief description on each command.

Command	Type of Command	Description
create	Group	Create a group.
release	Group	Delete a group.
set_message	Group	Set a group message with up to 60 characters in length.
ret_message	Group	Return message previously set by command <code>set_message</code> .
ret_first_obj	Group	Return name of first object in the group.
ret_next_obj	Group	Return the name of the next object in the group.
save	Group	Save all objects to the path specified.
mark	Group	Execute all scannable objects in the group.
m_mark ¹	Group	Begins marking the group and returns immediately.
mobi	Group	Mark immediately on begin all scannable objects in the group.
ret_busy ¹	Group	Return the status of the output during marking of a job.

¹ PC-MARK MT Only.

All Group commands are preceded by the name of the group the command refers to. This has to be the name of an existing group generated by the `create` command.

create

Description

This command will create the group with the given legal group name. The number of characters can be up to 255 maximum. It has to begin with a character and then contain special characters and numbers.

Syntax

Command: `.groupname.create`
 Reply: `0 : [group] created`

Arguments

None

Return Values

None

Errors

0	103	110	119
---	-----	-----	-----

release

Description

This command will delete the named group and all objects that are contained within the group.

Syntax

Command: `.groupname.release`
 Reply: `0 :`

Arguments

None

Return Values

None

Errors

0	104	110	119
---	-----	-----	-----

set_message

Description

This command allows the setting of the group message. The message is a string of up to 60 characters in length that can be used by any application.

It is not interpreted in any way by **PC-MARK MT** (or **PC-MARK**).

Syntax

Command: **.groupname.set_message [message]**

Reply: **0 :**

Arguments

Name	Type	Range	Function
message	String	0 - 60 characters	Group message.

Return Values

None

Errors

0	101	104	110
---	-----	-----	-----

ret_message

Description

This command returns the message as set previously by the **set_message** command.

Syntax

Command: **.groupname.ret_message**

Reply: **0 : [message]**

Arguments

None

Return Values

Name	Type	Range	Function
message	String	0 - 60 characters	Group message.

Errors

0	104
---	-----

ret_first_obj

Description

This command returns the name of the first object in the group. This command when used in conjunction with the `ret_next_obj` command allows the application to determine the names of all the objects in a group.

Syntax

Command: `.groupname.ret_first_obj`

Reply: 0 : [name] [type] [mark_flag] [para_set]
 [x-offset] [y-offset] [x-size] [y-size]

Arguments

None

Return Values

Name	Type	Units	Range	Function
name	String	ASCII	1 - 8 characters	Name of the object.
type	Type	Type	HPGL, MCL, ...	Type of the object.
mark_flag	Flag		Y/N	Object is scannable Y/N.
para_set	Integer	num	1 - 64	Number of the parameter set.
x-offset	Integer	Field	-32767 - 32767	X position of object.
y-offset	Integer	Field	-32767 - 32767	Y position of object.
x-size	Integer	Field	0 - 65535	X dimension of object.
y-size	Integer	Field	0 - 65535	Y dimension of object.

Errors

0	104	105	110	111	112	115	117	123
---	-----	-----	-----	-----	-----	-----	-----	-----

ret_next_obj

Description

This command returns the name of the next object in the group. This command when used in conjunction with the `ret_first_obj` command allows the application to determine the names of all the objects in a group.

Syntax

Command: `.groupname.ret_next_obj`

Reply: **0** : [name] [type] [mark_flag] [para_set]
 [x-offset] [y-offset] [x-size] [y-size]

Arguments

None

Return Values

Name	Type	Units	Range	Function
name	String	ASCII	1 to 8 characters	Name of the object.
type	Type	Type	HPGL, MCL, ...	Type of the object.
mark_flag	Flag		Y/N	Object is scannable Y/N.
para_set	Integer	num	1 to 64	Number of the parameter set.
x-offset	Integer	Field	-32767 to 32767	X position of object.
y-offset	Integer	Field	-32767 to 32767	Y position of object.
x-size	Integer	Field	0 to 65535	X dimension of object.
y-size	Integer	Field	0 to 65535	Y dimension of object.

Errors

0	104	105	110	111	112	115	117	118	123
----------	------------	------------	------------	------------	------------	------------	------------	------------	------------

save

Description

This command will save all objects to the path specified. Note that the file is not created, so if it currently does not exist, the save will fail.

Syntax

Command: `.groupname.save [path]`

Reply: **0** :

Arguments

Name	Type	Range	Function
path	String	DOS, existing file	Path and file name.

Return Values

None

Errors

0	101	105	109	110
----------	------------	------------	------------	------------

mark

Description This command will execute all scannable objects in the group.

Syntax Command: **.groupname.mark**

Reply: 0 :

Arguments None

Return Values None

Errors

0	104	105	111	112	115	116	117	118	121	123
---	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

m_mark



This command is available in PC-MARK MT (Multitasking) only.

Description This command will set MARK_IN_PROGRESS signal true and begins marking the group. It returns immediately to allow the application program to continue other tasks while marking is done in the background.
Must be called ONCE ONLY before calling **.ret_busy**.

Syntax Command: **.groupname.m_mark**

Reply: 0 :

Arguments None

Return Values None

Errors

0	104	105	110	111	112	115	116	117	118	121	123
---	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

mobi

Description

This command will mark immediately on begin all scannable objects in the group. If no BEGIN_MARK signal is received within the specified period, **PC-MARK MT** (or **PC-MARK**) returns an error message. If you specify 0, **PC-MARK MT** (or **PC-MARK**) waits forever.

Syntax

Command: **.groupname.mobi [n]**

Reply: **0 :**

Arguments

Name	Units	Range	Function
n	sec	0 to 65535	Time-out period in seconds.

Return Values

None

Errors

0	104	105	111	112	115	116	117	118	121	123
---	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

ret_busy



This command is available in PC-MARK MT (Multitasking) only.

Description

This command returns the status of the output during marking of a job. It must be called at least once although it may be called as often as required following an initial **.m_mark**. If marking has finished, **.ret_busy** causes the MARK_IN_PROGRESS signal to go false.

Syntax

Command: **.groupname.ret_busy**

Reply if there are more vectors: **0 : [Y]**

Reply if there are no more vectors and marking is completely finished:

0 : [N]

Arguments

None

Return Values

None

Errors

0	104	105	110	111	112	115	116	117	118	121	123
---	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

4. General Object Commands

The following table defines the General Object commands with a brief description on each command.

Command	Type of Command	Description
create	General Object	Create the object with the given object name.
release	General Object	Delete the named object.
ret_typ	General Object	Return the type of the object.
set_message	General Object	Set the object message.
ret_message	General Object	Return message set previously by set_message command.
set_para_set	General Object	Set a pointer to 1 of the 64 parameter sets.
ret_para_set	General Object	Return number of the parameter set used to execute the object.
save	General Object	Append the object to the file specified.
set_mark_flag	General Object	Set the execute flag for the object.
ret_mark_flag	General Object	Return the state of the execute flag for the object.
mark	General Object	Execute the object if the object is scannable.
m_mark ¹	General Object	Begins marking the object and returns immediately.
mobi	General Object	Mark immediately on begin the object if the object is scannable.
ret_busy ¹	General Object	Return the status of the output during marking of a job.
vector_count	General Object	Return the number of vectors in the object.
ret_il_ptr	General Object	Return a memory pointer to a list of vectors for the object.
ret_info	General Object	Return information about the given object.
reset	General Object	Reset the object transformation matrix to the units matrix.

¹ PC-MARK MT Only.

All Object commands are preceded by:

- The name of the group created by the group **create** command.
then
- The name of the object which is created by the Object **create** command.

create

Description

This command will create the object with the given object name. The object name must be unique within the group.

Syntax

Command: **.groupname.objectname.create [type]**

Reply: **0 : [object] created**

Arguments

There are 5 different type of objects:

MCL
HPGL
STEXT
RTEXT
BarCode

Refer to the section **Objects** under **JOBS**, for more information.

Return Values

None

Errors

0	103	106	108	109	110	119
---	-----	-----	-----	-----	-----	-----

release

Description This command will delete the named object.

Syntax Command: **.groupname.objectname.release**

Reply: 0 :

Arguments None

Return Values None

Errors

0	104	110
---	-----	-----

ret_typ

Description This command will return the type of the object.

Syntax Command: **.groupname.objectname.ret_typ**

Reply: 0 : [type]

Arguments None

Return Values

Name	Type	Range	Function
type	Type	HPGL, MCL, ...	Type of object.

Errors

0	104
---	-----

set_message

Description This command allows the setting of the object message. The message is a string of up to 60 characters in length that can be used by any application. It is not interpreted in any way by PC-MARK MT (or PC-MARK).

Syntax Command: **.groupname.objectname.set_message [comment]**

Reply: 0 :

Arguments

Name	Type	Range	Function
comment	String	1 - 60 characters	Object message.

Return Values None

Errors

0	104	109	110
---	-----	-----	-----

ret_message

Description

This command returns the message as set previously by the `set_message` command.

Syntax

Command: `.groupname.objectname.ret_message`

Reply: 0 : [comment]

Arguments

None

Return Values

Name	Type	Range	Function
comment	String	1 - 60 characters	Object message.

Errors

0	104	109	110
---	-----	-----	-----

set_para_set

Description

The command acts as a pointer to one of the 64 parameter sets. The parameters defined in this set will be used when this object is executed.

Syntax

Command: `.groupname.objectname.set_para_set [number]`

Reply: 0 :

Arguments

Name	Type	Units	Default	Range	Function
number	Integer	num	1	1 - 64	Number of the parameter set.

Return Values

None

Errors

0	101	102	104	109	110
---	-----	-----	-----	-----	-----

ret_para_set

Description

This command returns the number of the parameter set to be used when this object is executed.

Syntax

Command: `.groupname.objectname.ret_para_set`

Reply: 0 : [number]

Arguments

None

Name	Type	Units	Range	Function
number	Integer	num	1 - 64	Number of the parameter set.

Errors

0	101	102	104	109	110
---	-----	-----	-----	-----	-----

save

Description

This command will append the object to the file specified. Note that the file is not created, so if it currently does not exist, the save will fail.

Syntax

Command: `.groupname.objectname.save [path]`

Reply : 0 :

Arguments

Name	Type	Range	Function
path	String	DOS, existing file	Path and file name.

Return Values

None

Errors

0	101	104	105	109	110
---	-----	-----	-----	-----	-----

set_mark_flag

Description

This command will set the execute flag for the object. If the execute flag is set to Y the object is scannable.

Syntax

Command: `.groupname.objectname.set_mark_flag [mark_flag]`

Reply: 0 :

Arguments

Name	Type	Range	Function
mark_flag	Flag	Y/N	Object is scannable Y/N.

Return Values

None

Errors

0	101	102	104	109	110
---	-----	-----	-----	-----	-----

ret_mark_flag

Description

This command returns the state of the execute flag for the object.
If the execute flag is Y then the object is scannable.

Syntax

Command: **.groupname.objectname.ret_mark_flag**

Reply: 0 :

Arguments

None

Return Values

Name	Type	Range	Function
mark_flag	Flag	Y/N	Object is scannable Y/N.

Errors

0	101	102	104	109	110
---	-----	-----	-----	-----	-----

mark

Description

This command will execute the object if the object is scannable.

Syntax

Command: **.groupname.objectname.mark**

Reply: 0 :

Arguments

None

Return Values

None

Errors

0	104	105	110	111	112	115	116	117	118	121	123
---	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

m_mark



This command is available in PC-MARK MT (Multitasking) only.

Description

This command will set MARK_IN_PROGRESS signal true and begins marking the object. It returns immediately to allow the application program to continue other tasks while marking is done in the background.

Must be called ONCE ONLY before calling **.ret_busy**.

Syntax

Command: **.groupname.objectname.m_mark**

Reply: 0 :

Arguments

None

Return Values

None

Errors

0	104	105	110	111	112	115	116	117	118	121	123
---	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

mobi

Description

This command will mark the object immediately on detection of the BEGIN_MARK signal (if the object is scannable). If no BEGIN_MARK signal is received within the specified period, **MMARKIT** (or **MARKIT**) returns an error message. If you specify 0, **MMARKIT** (or **MARKIT**) waits forever.

Syntax

Command: **.groupname.objectname.mobi [n]**

Reply: **0 :**

Arguments

Name	Units	Range	Function
n	sec	0 to 65535	Time-out period in seconds.

Return Values

None

Errors

0	104	105	111	112	115	116	117	118	121	123
---	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

ret_busy



This command is available in PC-MARK MT (Multitasking) only.

Description

This command returns the status of the output during marking of a job. It must be called at least once although it may be called as often as required following an initial **.m mark**. If marking has finished, **.ret_busy** causes the MARK_IN_PROGRESS signal to go false.

Syntax

Command: **.groupname.objectname.ret_busy**

Reply if there are more vectors: **0 : [Y]**

Reply if there are no more vectors and marking is completely finished:

0 : [N]

Arguments

None

Return Values

None

Errors

0	104	105	110	111	112	115	116	117	118	121	123
---	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

vector_count

Description

This command will return the number of vectors in the object. Each vector is five bytes long and is in **MCL** format.

Syntax

Command: **.groupname.objectname.vector_count**

Reply: **0 : [vectors]**

Arguments

None

Return Values

Name	Type	Units	Range	Function
vectors	long Integer	5 bytes per Vector	0 to XMS ¹⁾	The amount of MCL-vectors by the object.

¹⁾ EMS / DISK for PC-MARK

Errors

0	104	110	112
---	-----	-----	-----

ret_il_ptr

Description

This command will return a memory pointer (in segment and offset form) to a list of vectors for the object. Since vectors for an object may not always fit in memory, this command may return only partial lists of vectors. The return values indicate how many vectors have been returned and how many are remaining.

Syntax

Command: **.groupname.objectname.ret_il_ptr**

Reply: **0 : [offset] [segment] [returned] [remaining]**

Arguments

None

Return Values

Name	Type	Units	Range	Function
offset	word	Pointer offset	valid offset	This is the pointer offset to the vector buffer.
segment	word	Pointer segment	valid segment	This is the pointer segment to the vector buffer.
returned	word	5 byte Vector	0 to max vectors	This indicates how many vectors have been returned for this call.
remaining	word	5 byte Vector	0 to max vectors	This indicates how many vectors are remaining in the object.

Errors

0	104	105	110	112	115	117	118	123
---	-----	-----	-----	-----	-----	-----	-----	-----

ret_info

Description

This command returns information about the given object.

Syntax

Command: **.groupname.objectname.ret_info**

Reply: **0** : [name] [type] [mark_flag] [para_set]
 [x-offset] [y-offset] [x-size] [y-size]

Arguments

None

Return Values:

Name	Type	Units	Range	Function
name	String	ASCII	1 - 8 characters	Name of the object.
type	Type		HPGL, MCL, ...	Type of object.
mark_flag	Flag		Y / N	Object is scannable Y/N.
para_set	Integer	num	1 - 64	Number of the parameter set.
x-offset	Integer	Field	-32767 - 32767	X position of object.
y-offset	Integer	Field	-32767 - 32767	Y position of object.
x-size	Integer	Field	0 - 65535	X dimension of object.
y-size	Integer	Field	0 - 65535	Y dimension of object.

Errors

0	104	105	110	111	112	115	117	118	123
---	-----	-----	-----	-----	-----	-----	-----	-----	-----

reset

Description

This command resets the object transformation matrix to the units (identity) matrix. This has the effect of undoing all sizing, scaling, rotating and positioning that has occurred since the object was created.

Syntax

Command: **.groupname.objectname.reset**

Reply: **0** :

Arguments

None

Return Values

None

Errors

0	104
---	-----

5. Object Transformation Commands

The following table defines the Object Transformation commands, a brief description on each command and the page location for detailed information on the command.

Command	Type of Command	Description
x_flip	Object Transform	Mirror the object around the Y axis.
y_flip	Object Transform	Mirror the object around the X axis.
rotate	Object Transform	Rotate the object by the given angle.
slant	Object Transform	Skew object in X dimension while leaving Y dimension alone.
scale	Object Transform	The object will be magnified by the given scale factors.
offset	Object Transform	Translate the object by the given X and Y offset (relative).
mul_matrix	Object Transform	Multiply the objects transformation matrix by the given matrix.
ret_matrix	Object Transform	Return the objects transformation matrix.

All Object commands are preceded by:

- The name of the group created by the group **create** command.
then
- The name of the object which is created by the Object **create** command.

x_flip

Description

The command mirrors the object around the Y axis.

Syntax

Command: **.groupname.objectname.x_flip**

Reply: **0 :**

Arguments

None

Return Values

None

Errors

0	101	102	104	109	110
---	-----	-----	-----	-----	-----

y_flip

Description

The command mirrors the object around the X axis.

Syntax

Command: **.groupname.objectname.y_flip**

Reply: **0 :**

Arguments

None

Return Values

None

Errors

0	101	102	104	109	110
---	-----	-----	-----	-----	-----

rotate

Description

This command rotates the object by the given angle.
The center of the rotation is at the origin (0,0).

Syntax

Command: **.groupname.objectname.rotate [angle]**

Reply: 0 :

Arguments

Name	Type	Units	Default	Range	Function
angle	Real	°(deg)	0	0 - 360	Rotation angle of the object.

Return Values

None

Errors

0	101	102	104	109	110
---	-----	-----	-----	-----	-----

slant

Description

This command will skew the object in the X dimension while leaving the Y dimension unaffected.

Syntax

Command: **.groupname.objectname.slant [angle]**

Reply: 0 :

Arguments

Name	Type	Units	Default	Range	Function
angle	Real	°(deg)	0	0 - 360	Skew angle of the object.

Return Values

None

Errors

0	101	102	104	109	110
---	-----	-----	-----	-----	-----

scale

Description

The object will be magnified by the given scale factors.

Syntax

Command: **.groupname.objectname.scale [x-factor]
[y-factor]**

Reply: 0 :

Arguments

Name	Type	Default	Range	Function
X-factor	Real	1	min Real to max Real	Factor for the X axis.
Y-factor	Real	1	min Real to max Real	Factor for the Y axis.

Return Values

None

Errors

0	101	102	104	109	110
---	-----	-----	-----	-----	-----

offset

Description

This command will translate the object by a given X and Y offset (relative).

Syntax

Command: `.groupname.objectname.offset [x-offset] [y-offset]`

Reply: 0 :

Arguments

Name	Type	Units	Default	Range	Function
x-offset	long Integer	Field	0	-65536 to 65536	Relative offset in X direction.
y-offset	long Integer	Field	0	-65536 to 65536	Relative offset in Y direction.

Return Values

None

Errors

0	101	102	104	109	110
---	-----	-----	-----	-----	-----

mul_matrix

Description

This command will multiply the objects transformation matrix by the given transformation matrix. This allows the application to encode several object transformations in one command.

(see separate section for transformation matrix).

Syntax

Command: `.groupname.objectname.mul_matrix [a] [b] [c] [d] [xo] [yo]`

Reply: 0 :

Arguments

Name	Type	Default	Range
a	Real	1	min Real to max Real.
b	Real	0	min Real to max Real.
c	Real	0	min Real to max Real.
d	Real	1	min Real to max Real.
xo	Real	0	min Real to max Real.
yo	Real	0	min Real to max Real.

Return Values

None

Errors

0	101	102	104	109	110
---	-----	-----	-----	-----	-----

ret_matrix

Description

This command will return the objects transformation matrix.

Syntax

Command: **.groupname.objectname.ret_matrix**

Reply: 0 : [a] [b] [c] [d] [xo] [yo]

Arguments

None

Return Values

Name	Type	Units	Range
a	Real		min Real to max Real.
b	Real		min Real to max Real.
c	Real		min Real to max Real.
d	Real		min Real to max Real.
xo	Real	field	min Real to max Real.
yo	Real	field	min Real to max Real.

Errors

0	104
---	-----

6. Vector Object Commands

The following table defines the Vector Object commands, a brief description on each command and the page location for detailed information on the command.

Command	Type of Command	Description
set_source	Vector Object	Set the path name for the vector object.
ret_source	Vector Object	Return the path name for the vector object.
set_pen_flag	Vector Object	Pen changing on vector objects having internal pen changes.
ret_pen_flag	Vector Object	Return the state of the pen flag.
set_pens	Vector Object	Map HPGL and MCL pens to laser parameter sets.
ret_pens	Vector Object	Return current mapping of the pens to laser parameter sets.
set_vector_wrap	Vector Object	Set the Vector Wrapping Option

All Object commands are preceded by:

- The name of the group created by the group **create** command.
then
- The name of the object which is created by the Object **create** command.

set_source

Description

This command sets the path and file name for the vector object. The vectors (in HPGL or MCL depending on the object type) are read from this file.

Syntax

Command: **.groupname.objectname.set_source [path, filename]**

Reply: **0 :**

Arguments:

Name	Type	Range	Function
path, filename	String	DOS, existing file	Path and file name of the source file.

Return Values:

None

Errors:

0	104	105	106	110	111	115	117	118	119	123
---	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

ret_source

Description

This command returns the path and file name for the vector object. The vectors (in HPGL or MCL depending on the object type) are read from this file.

Syntax

Command: **.groupname.objectname.ret_source**

Reply: **0 : [path, filename]**

Arguments

None

Return Values

Name	Type	Range	Function
path, filename	String	DOS, existing file	Path and file name of the source file.

Errors

0	104	106
---	-----	-----

set_pen_flag

Description

Vector objects may have internal pen changes. HPGL has an SP command that allows the changing of the pen. MMCL (or MCL) has an equivalent command. It is possible to map the HPGL and MCL pens to any of 64 sets of laser parameters (see the **set_pens** command). If the **set_pen_flag** is Y then the mapping of HPGL and MCL pens to laser parameters will be used, otherwise the laser parameter set specified by the **set_para_set** command will be used for all pens. This means the set pens command in the HPGL file will be ignored.

Syntax

Command: **.groupname.objectname.set_pen_flag [flag]**

Reply: **0 :**

Arguments

Name	Type	Default	Range	Function
flag	Flag	Y	Y/N	Y: Use the pens to set the laser parameters. N: Use the laser parameter set.

Return Values

None

Errors

0	101	102	104	106	109	110
---	-----	-----	-----	-----	-----	-----

ret_pen_flag

Description

This command returns the state of the pen flag.

Syntax

Command: **.groupname.objectname.ret_pen_flag**

Reply: **0 : [flag]**

Arguments

None

Return Values

Name	Type	Range	Function
flag	Flag	Y/N	Y: Use the pens to set the laser parameters. N: Use the laser parameter set.

Errors

0	104	106
---	-----	-----

set_pens

Description

This command maps HPGL or MCL pens to laser parameter sets. Each of the eight parameters represents one of the possible pens. The value of the parameter indicates which of the 64 sets of laser parameters should be used when that pen is indicated. Note that this mapping only takes place if the **set_pen_flag** is set to Y.

Syntax

Command: **.group.object.set_pens [1] [2] [3] [4] [5] [6] [7] [8]**

Reply: **0 :**

Arguments

Name	Type	Units	Default	Range	Function
1	Integer	num	1	1 - 64	Parameter set for pen 1.
2	Integer	num	1	1 - 64	Parameter set for pen 2.
3	Integer	num	1	1 - 64	Parameter set for pen 3.
4	Integer	num	1	1 - 64	Parameter set for pen 4.
5	Integer	num	1	1 - 64	Parameter set for pen 5.
6	Integer	num	1	1 - 64	Parameter set for pen 6.
7	Integer	num	1	1 - 64	Parameter set for pen 7.
8	Integer	num	1	1 - 64	Parameter set for pen 8.

Return Values

None

Errors

0	101	102	104	106	109	110
---	-----	-----	-----	-----	-----	-----

ret_pens

Description

This command returns the current mapping of the pens to laser parameter sets.

Syntax

Command: **.groupname.objectname.ret_pens**

Reply: 0 : [1] [2] [3] [4] [5] [6] [7] [8]

Arguments

None

Return Values

Name	Type	Units	Default	Range	Function
1	Integer	num	1	1 - 64	Parameter set for pen 1.
2	Integer	num	1	1 - 64	Parameter set for pen 2.
3	Integer	num	1	1 - 64	Parameter set for pen 3.
4	Integer	num	1	1 - 64	Parameter set for pen 4.
5	Integer	num	1	1 - 64	Parameter set for pen 5.
6	Integer	num	1	1 - 64	Parameter set for pen 6.
7	Integer	num	1	1 - 64	Parameter set for pen 7
8	Integer	num	1	1 - 64	parameter set for pen 8

Errors

0	104	106
---	-----	-----

set_vector_wrap

Description

This command allows you to choose whether vectors should be allowed to wrap around the scanning field or not.

The default state is that wrapping is **not** allowed, and when some operation results in vectors running off the edge of the field, like this, a **vectors-out-of-range** error is reported.

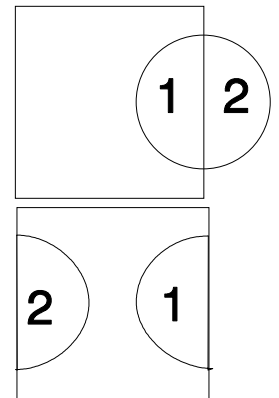
However, if you issue the command:

.set_vector_wrap [Y]

then vectors are allowed to wrap around to the opposite side of the scanning field, with no error.

Wrapping can be disallowed again with the command:

.set_vector_wrap [N]



Syntax

Command: **.groupname objectname.set_vector_wrap [Y/N]**

Reply: 0 : [flag]

Arguments

Y will allow vector warp.

N will not allow vector wrap.

Return Values

None

Errors

0

7. Text Object Commands

The following table defines the Text Object commands, a brief description on each command.

Command	Type of Command	Description
set_font	Text Object	Font to use when rendering the text for the object.
ret_font	Text Object	Return the font of the object.
set_string	Text Object	Specify the string of text that will be used.
ret_string	Text Object	Return the string of text that will be used.
set_char_height	Text Object	Set the height for text characters.
ret_char_height	Text Object	Return the height for text characters.
set_char_width	Text Object	Set the ratio of width to height for text characters.
ret_char_width	Text Object	Return the ratio of width to height for text characters.
set_char_slant	Text Object	Change amount of slant for each character of a text object.
ret_char_slant	Text Object	Return amount of slant for each character of a text object.
set_bx_char_map ²	Text Object	Set character mapping for Bitstream Fonts

² Not available for PC-MARK MT.

All Object commands are preceded by:

- The name of the group created by the group **create** command.
- then**
- The name of the object which is created by the Object **create** command.

set_font



Description

No Bitstream® font option is currently available in PC-MARK MT (Multitasking). It returns an error code indicating that Bitstream® operations are disabled.

Syntax

This command specifies which font to use when rendering the text for the object. The path must represent a valid MCL or Bitstream® (BX) font file.

Command: **.groupname.objectname.set_font [path]**

Reply: **0 :**

Arguments

Name	Type	Range	Function
path	String	DOS, existing file	Name of the font format.

Return Values

None

Errors

0	101	104	105	106	109	110	116
---	-----	-----	-----	-----	-----	-----	-----

ret_font

Description

This command returns the font of the object.

Syntax

Command: **.groupname.objectname.ret_font**

Reply: **0 : [path]**

Arguments

None

Return Values

Name	Type	Range	Function
path	String	DOS, existing file	Name of the font format.

Errors

0	104	106
---	-----	-----

set_string

Description

This command specifies the string of text that will be rendered. A font must be set before the string can be set.

Syntax

Command: **.groupname.objectname.set_string [text]**

Reply: **0 :**

Arguments

Name	Type	Default	Range	Function
text	String	Empty	0-235 characters	String of characters.

Return Values

None

Errors

0	104	105	106	109	110	111	115	116	117	118	122	123	124
---	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

ret_string

Description

This command returns the string of the text that will be rendered.

Syntax

Command: **.groupname.objectname.ret_string**

Reply: **0 : [text]**

Arguments

None

Return Values

Name	Type	Range	Function
text	String	0-235 characters	String of characters.

Errors

0	104	106
---	-----	-----

set_char_height

Description

This command sets the height for text characters.

Syntax

Command: `.groupname.objectname.set_char_height [factor]`

Reply: 0 :

Arguments

Name	Type	Default	Range	Function
factor	Real	1	0 - max real	Height of a character.

Return Values

None

Errors

0	101	104	106	109	110
---	-----	-----	-----	-----	-----

ret_char_height

Description

This command returns the height for text characters.

Syntax

Command: `.groupname.objectname.ret_char_height`

Reply: 0 : [factor]

Arguments

None

Return Values

Name	Type	Range	Function
factor	Real	0 - max real	Height of a character.

Errors

0	104	106
---	-----	-----

set_char_width

Description

This command sets the ratio of width to height for text characters.

Syntax

Command: `.groupname.objectname.set_char_width [factor]`

Reply: 0 :

Arguments

Name	Type	Default	Range	Function
factor	Real	1	0 - max real	Width factor of a character.

Return Values

None

Errors

0	101	104	106	109	110
---	-----	-----	-----	-----	-----

ret_char_width

Description

This command returns the ratio of width to height for text characters.

Syntax

Command: **.groupname.objectname.ret_char_width**

Reply: **0 : [factor]**

Arguments

None

Return Values

Name	Type	Range	Function
factor	Real	0 - max real	Width factor of a character.

Errors

0	104	106
---	-----	-----

set_char_slant

Description

This command changes the amount of slant to be applied to each character of a text object.

Syntax

Command: **.groupname.objectname.set_char_slant [slant]**

Reply: **0 :**

Arguments

Name	Type	Units	Default	Range	Function
slant	Real	°(deg)	0	0 - 45	Slant of the characters.

Return Values

None

Errors

0	101	104	106	109	110
---	-----	-----	-----	-----	-----

ret_char_slant

Description

This command returns the amount of slant to be applied to each character of a text object.

Syntax

Command: **.groupname.objectname.ret_char_slant**

Reply: **0 : [slant]**

Arguments

None

Return Values

Name	Type	Units	Range	Function
slant	Real	°(deg)	0 - 45	slant of the characters

Errors

0	104	106
---	-----	-----

set_bx_char_map



No Bitstream® font option is currently available in PC-MARK MT (Multitasking). It returns an error code indicating that Bitstream® operations are disabled.

Description

This command allows **PC-MARK** to read a Map file.

PC-MARK now allows access to any character from the Bitstream International Character Set through a Character Map that sets the correspondence between ASCII and BICS codes. You can set up Maps to suit your applications.

Note

Does not work, returns an error code indicating that Bitstream® operations are disabled.

Syntax

Command: **.groupname.objectname.set_bx_char_map [filename]**

Reply: **0 : [slant]**

Arguments

PC-MARK comes with a built-in character map that corresponds to the map-file **STANDARD.BCM**. This file is supplied with the **PC-MARK** suite, and should be used as a model for your maps.



Note

You do not need to issue the .set_bx_char_map command if you wish to use only the built-in mapping.

Return Values

None

Errors

0

8. Straight Text Object Commands

The following table defines the Straight Text Object commands, a brief description on each command and the page location for detailed information on the command.

Command	Type of Command	Description
set_orientation	Straight Text Object	Specify whether text will be laid out horizontal or vertical.
ret_orientation	Straight Text Object	Return the text orientation set by set_orientation.

All Object commands are preceded by:

- The name of the group created by the group **create** command.
then
- The name of the object which is created by the Object **create** command.

set_orientation

Description

This command specifies whether straight text should be laid out in horizontal or in vertical manner.

Syntax

Command: **.groupname.objectname.set_orientation [orient]**

Reply: **0 :**

Arguments

Name	Type	Default	Range	Function
orient	String	H	H/V	H - lay text out horizontally; V - lay text out vertically.

Return Values

None

Errors

0	101	104	106	109	110
---	-----	-----	-----	-----	-----

ret_orientation

Description

This command returns the text orientation.

Syntax

Command: **.groupname.objectname.ret_orientation**

Reply: **0 : [orient]**

Arguments

None

Return Values

Name	Type	Range	Function
orient	String	H/V	H - lay text out horizontally; V - lay text out vertically.

Errors

0	104	106
---	-----	-----

9. Radial Text Object Commands

The following table defines the Radial Text Object commands, a brief description on each command and the page location for detailed information on the command.

Command	Type of Command	Description
set_radius	Radial Text Object	Set the radius for the radial text.
ret_radius	Radial Text Object	Return the radius of the radial text.
set_direction	Radial Text Object	Radial text will be clockwise or counter clockwise.
ret_direction	Radial Text Object	Return the direction set by the set_direction command.

These commands can only be applied to **RTEXT** type of objects.

All Object commands are preceded by:

- The name of the group created by the group **create** command.
then
- The name of the object which is created by the Object **create** command.

set_radius

Description

This command specifies the radius for the radial text. The radius is in field units but is not constrained to lie on the scanning field.

Syntax

Command: **.groupname.objectname.set_radius [radius]**

Reply: **0 :**

Arguments

Name	Type	Units	Default	Range	Function
radius	long Integer	Field	4096	0 - max long Integer	Radius of the radial text.

Return Values

None

Errors

0	101	104	106	109	110
---	-----	-----	-----	-----	-----

ret_radius

Description

This command returns the radius of the radial text.

Syntax

Command: **.groupname.objectname.ret_radius**

Reply: **0 : [radius]**

Arguments

None

Return Values

Name	Type	Units	Range	Function
radius	long Integer	Field	0 - max long Integer	Radius of the radial text.

Errors

0	104	106
---	-----	-----

set_direction

Description

This command specifies whether the radial text will be rendered in a clockwise or counter clockwise direction.

Syntax

Command: `.groupname.objectname.set_direction [direction]`

Reply: 0 :

Arguments

Name	Type	Default	Range	Function
direction	String	CW	CW/CCW	CW/CCW direction of the text.

Return Values

None

Errors

0	101	104	106	109	110
---	-----	-----	-----	-----	-----

ret_direction

Description

This command returns whether the radial text will be rendered in a clockwise or counter clockwise direction.

Syntax

Command: `.groupname.objectname.ret_direction`

Reply: 0 : [direction]

Arguments

None

Return Values

Name	Type	Range	Function
direction	String	CW/CCW	CW/CCW direction of the text.

Errors

0	104	106
---	-----	-----

10. Barcode Object Commands

The following table defines the Barcode Object commands, a brief description on each command and the page location for detailed information on the command.

Command	Type of Command	Description
set_string	BarCode Object	Specify the string that is to be barcoded.
ret_string	BarCode Object	Return the string that is to be barcoded.
set_x	BarCode Object	Specify the Narrow Element dimension of the barcode.
ret_x	BarCode Object	Return the Narrow Element dimension of the barcode.
set_n	BarCode Object	Specify the wide to narrow ratio for the barcode.
ret_n	BarCode Object	Return the wide to narrow ratio for the barcode.
set_bw	BarCode Object	Specify the black (imprint) to white (non-imprint ratio).
ret_bw	BarCode Object	Return the black to white ratio.
set_qz	BarCode Object	Set the barcode quiet zone.
ret_qz	BarCode Object	Return the settings of the barcode quiet zone.
set_t	BarCode Object	Set the space between characters.
ret_t	BarCode Object	Return the space between characters.
set_height	BarCode Object	Specify the height of the barcode in field units.
ret_height	BarCode Object	Return the height of the barcode in field units.
set_density	BarCode Object	Set number of steps between fill bars when filling a black bar.
ret_density	BarCode Object	Return number of steps set by the set_density command.
set_inverse_flag	BarCode Object	Set whether barcode should be executed normally or inverted.
ret_inverse_flag	BarCode Object	Return setting set by the set_inverse_flag command.
set_check_code_flag	BarCode Object	Set toggle for 3 of 9 check code to be added.
ret_check_code_flag	BarCode Object	Determine whether or not to add 3 of 9 check code.

For the following commands, the type of Object has to be **BarCode**.

All Object commands are preceded by:

- The name of the group created by the group **create** command.
then
- The name of the object which is created by the Object **create** command.

set_string

Description

This command specifies the string that is to be barcoded. BC39, BC25I and BCUPC type barcodes do not allow all ASCII characters to be included in a barcode.

Syntax

Command: **.groupname.objectname.set_string [string]**

Reply: **0 :**

Arguments

Name	Type	Default	Range	Function
string	String	empty string	0-235 characters	The sequence of characters to be barcoded.

Return Values

None

Errors

0	101	104	106	109	110	122	123
---	-----	-----	-----	-----	-----	-----	-----

ret_string

Description

This command returns the string that is to be barcoded.

Syntax

Command: **.groupname.objectname.ret_string**

Reply: **0 : [string]**

Arguments

None

Return Values

Name	Type	Default	Range	Function
string	String	empty string	0-235 characters	The sequence of characters to be barcoded.

Errors

0	104	106
---	-----	-----

set_x

Description

Die This command specifies the Narrow Element dimension of the barcode (commonly referred to as X). This is the width of the narrow element of the barcode in field units.

Syntax

Command: **.groupname.objectname.set_x [x]**

Reply: **0 :**

Arguments

Name	Type	Units	Default	Range	Function
x	Integer	Field	500	0 - 10000	Width of the narrow element.

Return Values

None

Errors

0	101	104	106	109	110
---	-----	-----	-----	-----	-----

ret_x

Description

This command returns the Narrow Element dimension of the barcode.

Syntax

Command: **.groupname.objectname.ret_x**

Reply: **0 : [x]**

Arguments

None

Return Values

Name	Type	Units	Default	Range	Function
x	Integer	Field		0 - 10000	Width of the narrow element.

Errors

0	104	106
---	-----	-----

set_n

Description

This command specifies the wide to narrow ratio for the barcode (commonly referred to as N). The wide element dimension is equal to X (the narrow element dimension) times N.

Syntax

Command: `.groupname.objectname.set_n [n]`

Reply: 0 :

Arguments

Name	Type	Default	Range	Function
N	Real	2.3	1.0 - 5.0	Wide to narrow ratio.

Return Values

None

Errors

0	101	104	106	109	110
---	-----	-----	-----	-----	-----

ret_n

Description

This command returns the wide to narrow ratio for the barcode.

Syntax

Command: `.groupname.objectname.ret_n`

Reply: 0 : [n]

Arguments

None

Return Values

Name	Type	Range	Function
n	Real	1.0 - 5.0	Wide to narrow ratio.

Errors

0	104	106
---	-----	-----

set_bw

Description

This command specifies the black (imprint) to white (non-imprint) ratio. This ratio allows correcting for any bleeding that might normally make imprinting bars wider than non-imprinting bars that should be equal length.

Syntax

Command: `.groupname.objectname.set_bw [bw]`

Reply: 0 :

Arguments

Name	Type	Default	Range	Function
bw	Real	1	0.0 - 100.0	Ratio of black/white.

Return Values

None

Errors

0	101	104	106	109	110
---	-----	-----	-----	-----	-----

ret_bw

Description

This command returns the black to white ratio.

Syntax

Command: **.groupname.objectname.ret_bw**

Reply: **0 : [bw]**

Arguments

None

Return Values

Name	Type	Range	Function
bw	Real	0.0 - 100.0	Ratio of black/white.

Errors

0	104	106
---	-----	-----

set_qz

Description

This command allows the setting of the barcode quiet zone.

Syntax

Command: **.groupname.objectname.set_qz [qz]**

Reply: **0 :**

Arguments

Name	Type	Units	Default	Range	Function
qz	word	Field	5000	0 - 20000	Quiet zone.

Return Values

None

Errors

0	101	104	106	109	110
---	-----	-----	-----	-----	-----

ret_qz

Description

This command returns the settings of the barcode quiet zone.

Syntax

Command: **.groupname.objectname.ret_qz**

Reply: **0 : [qz]**

Arguments

None

Return Values

Name	Type	Units	Range	Function
qz	word	Field	0 - 20000	Quiet zone.

Errors

0	104	106
---	-----	-----

set_t

Description

This command specifies the space between characters.

Syntax

Command: **.groupname.objectname.set_t [t]**

Reply: 0 :

Arguments

Name	Type	Units	Default	Range	Function
t	word	Field	500	0 - 20000	Space between the characters.

Return Values

None

Errors

0	101	104	106	109	110
---	-----	-----	-----	-----	-----

ret_t

Description

This command returns the space between characters.

Syntax

Command: **.groupname.objectname.ret_t**

Reply: 0 : [t]

Arguments

None

Return Values

Name	Type	Units	Range	Function
t	word	Field	0 - 20000	Space between the characters.

Errors

0	104	106
---	-----	-----

set_height

Description

This command specifies the height of the barcode in field units.

Syntax

Command: **.groupname.objectname.set_height [height]**

Reply: 0 :

Arguments

Name	Type	Units	Default	Range	Function
height	word	Field	3000	0 - 32000	Height of the characters.

Return Values

None

Errors

0	101	104	106	109	110
---	-----	-----	-----	-----	-----

ret_height

Description

This command returns the height of the barcode in field units.

Syntax

Command: **.groupname.objectname.ret_height**

Reply: **0 : [height]**

Arguments

None

Return Values

Name	Type	Units	Range	Function
height	word	Field	0 - 32000	Height of the characters.

Errors

0	104	106
---	-----	-----

set_density

Description

This command specifies the number of steps between fill bars when filling a black bar.

Syntax

Command: **.groupname.objectname.set_density [density]**

Reply: **0 :**

Arguments

Name	Type	Units	Default	Range	Function
density	word	Field	3	0 - 10000	Steps between vectors of a bar.

Return Values

None

Errors

0	101	104	106	109	110
---	-----	-----	-----	-----	-----

ret_density

Description

This command returns the number of steps between fill bars when filling a black bar.

Syntax

Command: **.groupname.objectname.ret_density**

Reply: **0 : [density]**

Arguments

None

Return Values

Name	Type	Units	Range	Function
density	word	field	0 - 10000	Steps between vectors of a bar.

Errors

0	104	106
---	-----	-----

set_inverse_flag

Description

This command specifies whether the barcode should be executed normally or inverted. If barcode is to be inverted, all black bars will be non-imprinted and all white bars will be imprinted.

Syntax

Command: `.groupname.objectname.set_inverse_flag [flag]`

Reply: 0 :

Arguments

Name	Type	Default	Range	Function
flag	Flag	N	Y/N	Y = invert the barcode; N = don't invert the barcode.

Return Values

None

Errors

0	101	104	106	109	110
---	-----	-----	-----	-----	-----

ret_inverse_flag

Description

This command returns whether the barcode will be executed normally or inverted.

Syntax

Command: `.groupname.objectname.ret_inverse_flag`

Reply: 0 : [flag]

Arguments

None

Return Values

Name	Type	Range	Function
flag	Flag	Y/N	Y = invert the barcode. N = don't invert the barcode.

Errors

0	104	106
---	-----	-----

set_check_code_flag

Description

This command specifies whether the barcode 3 of 9 check code will be automatically added to the barcode or not.

Syntax

Command: `.groupname.objectname.set_check_code_flag [flag]`

Reply: 0 :

Arguments

Name	Type	Default	Range	Function
flag	Flag	Y	Y/N	Y = add the check code. N = don't add the check code.

Return Values

None

Errors

0	101	104	106	109	110
---	-----	-----	-----	-----	-----

ret_check_code_flag

Description

This command determines whether the check code will be automatically added to the barcode or not.

Syntax

Command: `.groupname.objectname.ret_check_code_flag`

Reply: 0 : [flag]

Arguments

None

Return Values

Name	Type	Range	Function
flag	Flag	Y/N	Y = add the check code. N = don't add the check code

Errors

0	104	106
---	-----	-----

APPENDIX A:

Brief on PC-MARK MT

This section is intended for current **PC-MARK** users and describes commands affected by the implementation of multitasking in **MMCL** and **MMARKIT**.

All commands which are not listed work as they did previously.



PC-MARK MT (Multitasking) is the software platform for the multitasking HC/2 or HC/3 card. It does not work with the standard HelperCard.

1. Discontinued Commands

1. **check_abort** (User I/O Port Command)
This command is not available: it returns error code 1022 "MCL - Bad Opcode".
2. **set_font [filename]** (Text Object Command)
This command cannot be used to select a BITSTREAM font: it returns an error code indicating that BITSTREAM operations are disabled.
3. **set_bx_char_map [filename]** (Text Object Command)
This command does not work: it returns an error code indicating that BITSTREAM operations are disabled.

2. Multitasking Commands

What follows are descriptions of the commands **g.o.mark**, **g.o.m_mark**, **g.o.ret_busy** and **mark_abort** and how to take advantage of the multitasking capabilities offered through these commands.

1. The command **g.o.mark** has not changed - the system begins to mark and the computer will be more or less hung up until the mark has been finished. However, certain P.C. functions will continue to run in the background, for instance the real time clock, keyboard strobing etc. With the original **HelperCard** this was not possible.
2. **g.o.m_mark** is short for "Multitasking_MARK" of an object and functions initially the same as **g.o.mark**. However, after marking begins, **PC-MARK MT** returns control to the application program which can then continue operation. The application program can start functions such as the reading and storing of files, calculations and so on. The Vectors will be processed in the background. Also, the real time clock and other P.C. functions continue to run.
3. The commands **g.o.m_mark** and **g.o.ret_busy** allow applications to start the mark and then carry on with other business (needed for double buffering for instance.) Multiple objects can be created, modified, simultaneous to the outputting of one of the objects. With a properly written application, this effectively allows multiple jobs to be created. The job/object relationship is done by the application.

One possible use of this is to allow rapid swapping between heads for instance for front side and rear component marking. (Note: this is possible as long as different correction

files are not needed. The incorporation of separate correction files would need the use of special multi-head software.

Another application would be for very rapid step and repeat. By duplicating objects and applying the `mul_matrix` command, it would be possible to simultaneously move alternate objects across the field whilst marking.



NOTE

It is an error if the application tries to execute `g.o2.m_mark` before it has seen "0: [N]" from `g.o1.ret_busy` .

4. `mark_abort` - this command allows the customer's application to stop the deflection system if (say) a laser error or other emergency state has been discovered. Its effect is the same as asserting the MARK_ABORT signal on the **HC/2** or **HC/3**.



NOTE

The hardware .MARK_ABORT signal can be activated at any time. Software `mark_abort` command, however, can obviously only be issued when the application has control, for instance during a `g.o.m_mark / g.o.ret_busy` sequence.

For compatibility reasons, after **PC-MARK MT** receives the `.mark` command, it does NOT return control to the application until after the end of the mark.

After a `mark_abort` has been processed, **MMCL** does not necessarily know where the galvos are (because the galvos are driven from the output of the FIFO, and **MMCL** fills the input). Consequently, the application should issue `end_mark` after `mark_abort`, to make **MMCL** drive the galvos to an absolute position (0,0) it can rely on.

3. Master / Slave Option (Multiple Cards)

With the Master / Slave version of the **HC/2** card (**HC/2 M-S**) it is possible to run up to four interconnected **HC/2 M-S** cards in one PC in order to control up to four **GSLI Scan Heads** simultaneously. Each **Scan Head** needs his own **HC/2 M-S**. One of the cards serves as a master and will control the first head, the laser modulation, and the timing while the other cards serve as slaves and control one additional head each. Each head will be associated with its own calibration file through software thus providing that all heads will execute the exact same pattern. The **HC/3** card is backward compatible with **HC/2** card and supports Master/Slave configuration as well.

The different requirements for the Configuration of a Master / Slave system are described in the "PC-MARK MT Programmers Manual".

4. Multiple Correction Tables

With PC-MARK MT it is now possible to keep up to four different correction tables in memory and switch between them in runtime by simply issuing the appropriate command. Besides the use in Master / Slave configurations it is also possible to immediately switch to a different correction table for example when using a pointing laser or changing working-distance.

1. **set_corr_file**. This command causes the given correction file to be loaded and associated with the specified Table Number. Up to four files can be mapped to the four Table Numbers (0 through 3). It is then possible to make any of these pre-loaded files active for an **HC/2** or **HC/3** card at any time by using the **set_corr_table** command (see below).
2. **set_corr_table**. This command associates the **HC/2** or **HC/3** card with the specified table number and thereby with the associated correction file. In Master / Slave configurations with multiple **HC/2 M-S** cards or **HC/3** (up to 4) each card can be associated with a different correction file.

5. Operation of JOB EDITOR

Note that the **JOB EDITOR** software can now be run under **PC-MARK MT**, including **START_ON_EXTERNAL_SIGNAL**. It should be clear that **JOB EDITOR** is not a multitasking application. However, at least the real time clock and keyboard are not locked out - a significant improvement over the older **HelperCard**. **MARK_ABORT** now also works very cleanly.

If the escape key is hit (just once) during marking, **JOB EDITOR** will come out of "mark-continuous" cleanly at the end of the current marking job.

6. Hardware Stability

1. The software/hardware is very stable. The interrupts in the computer are only turned off for a very short period allowing other devices to get in without crashing. The tricks which previously had to be done with the original **HelperCard** to keep the real time clock (approximately) right are now no longer necessary.
2. The FIFO concept offers an automatic watchdog function. If the application crashes the computer or if the system stops for any reason, the FIFO will go empty. This causes (effectively) a hardware **MARK_ABORT** which switches off the laser.

7. Future Commands

return_head_status - this command will eventually be implemented for all head configurations. Until a full function test is made on this command, it will not be included in the manual.

APPENDIX B: Commands

Sorted by Alphanumeric

COMMAND	TYPE OF COMMAND	DESCRIPTION	PAGE No.
check_abort	User Interface Port	Return the value of the MARK_ABORT signal.	33
check_begin ²	User Interface Port	Return the sense of the BEGIN_MARK signal.	33
clear_mark_error ²	User Interface Port	Use to remove the MARK_ERROR signal.	33
create	Group	Create a group.	35
create	General Object	Create the object with the given object name.	41
do_mark	User Interface Port	Mark objects.	30
end_mark	User Interface Port	Issue after the last .do_mark command.	31
get_config	System	Return config data used to calculate the correction file.	17
home_beam ¹	System	Move the beam to a specified field coordinate.	9
m_mark ¹	System	Initiate the execution of all scannable objects.	25
m_mark ¹	Group	Begins marking the group and returns immediately.	39
m_mark ¹	General Object	Begins marking the object and returns immediately.	45
mark	System	Execute all scannable objects.	24
mark	User Interface Port	Combine .start_mark, .do_mark, .end_mark commands.	31
mark	Group	Execute all scannable objects in the group.	39
mark	General Object	Execute the object if the object is scannable.	45
mobi	Group	Mark immediately on begin all scannable objects in the group.	40
mobi	General Object	Mark immediately on begin the object if the object is scannable.	46
mul_matrix	Object Transform	Multiply the objects transformation matrix by the given matrix.	51
offset	Object Transform	Translate the object by the given X and Y offset (relative).	51
release	System	Release all resources currently allocated for groups and objects.	23
release	Group	Delete a group.	35
release	General Object	Delete the named object.	42
remote_execute	User Interface Port	Control the REMOTE_EXECUTE line of User Interface Port.	32
reset	General Object	Reset the object transformation matrix to the units matrix.	48
ret_appl	System	Return name of current application set by set_appl.	18
ret_busy ¹	System	Return the status of the output during marking of a job.	25
ret_busy ¹	Group	Return the status of the output during marking of a job.	40
ret_busy ¹	General Object	Return the status of the output during marking of a job.	46
ret_bw	BarCode Object	Return the black to white ratio.	70
ret_char_height	Text Object	Return the height for text characters.	59
ret_char_slant	Text Object	Return amount of slant for each character of a text object.	60
ret_char_width	Text Object	Return the ratio of width to height for text characters.	60
ret_check_code_flag	BarCode Object	Determine whether or not to add 3 of 9 check code.	74
ret_config_file	System	Return name of config file set by last set_config_file command.	16
ret_corr_file ¹	System	Return the name of correction file for the specified table number.	15
ret_corr_table ¹	System	Return the number of correction table for the specified HC/2 or HC/3 card number.	11
ret_density	BarCode Object	Return number of steps set by the set_density command.	72
ret_direction	Radial Text Object	Return the direction set by the set_direction command.	66
ret_first_grp	System	Return the name of the first PC-MARK MT (or PC-MARK) group.	22
ret_first_obj	Group	Return name of first object in the group.	37
ret_font	Text Object	Return the font of the object.	58
ret_head_status ¹	System	Return the actual Head status with associated HC/2 or HC/3 card.	14
ret_head_type ¹	System	Return the used Scan Head type.	13
ret_height	BarCode Object	Return the height of the barcode in field units.	72
ret_il_ptr	General Object	Return a memory pointer to a list of vectors for the object.	47

ret_info	General Object	Return information about the given object.	48
ret_inverse_flag	BarCode Object	Return setting set by the set_inverse_flag command.	73
ret_laser ¹	System	Return the state of the laser following a set_laser.	26
ret_mark_flag	General Object	Return the state of the execute flag for the object.	45
ret_matrix	Object Transform	Return the objects transformation matrix.	52
ret_message	System	Return the message set previously by set_message.	19
ret_message	Group	Return message previously set by command set_message.	36
ret_message	General Object	Return message set previously by set_message command.	43
ret_n	BarCode Object	Return the wide to narrow ratio for the barcode.	69
ret_next_grp	System	Return the name of the next PC-MARK MT (or PC-MARK) group.	22
ret_next_obj	Group	Return the name of the next object in the group.	38
ret_orientation	Straight Text Object	Return the text orientation set by set_orientation.	63
ret_para_set	System	Return values of the laser parameters for a given set.	21
ret_para_set	General Object	Return number of the parameter set used to execute the object.	43
ret_param_file	System	Return name of parameter file set by set_param_file command.	18
ret_pen_flag	Vector Object	Return the state of the pen flag.	55
ret_pens	Vector Object	Return current mapping of the pens to laser parameter sets.	56
ret_power_scale	System	Return value of power scale factor.	27
ret_qz	BarCode Object	Return the settings of the barcode quiet zone.	70
ret_radius	Radial Text Object	Return the radius of the radial text.	65
ret_source	Vector Object	Return the path name for the vector object.	54
ret_string	Text Object	Return the string of text that will be used.	58
ret_string	BarCode Object	Return the string that is to be barcoded.	68
ret_t	BarCode Object	Return the space between characters.	71
ret_typ	General Object	Return the type of the object.	42
ret_x	BarCode Object	Return the Narrow Element dimension of the barcode.	68
rotate	Object Transform	Rotate the object by the given angle.	50
save	System	Append all groups and objects to the file specified.	23
save	Group	Save all objects to the path specified.	38
save	General Object	Append the object to the file specified.	44
scale	Object Transform	The object will be magnified by the given scale factors.	50
set_appl	System	Set the name of the application program.	18
set_bw	BarCode Object	Specify the black (imprint) to white (non-imprint) ratio.	69
set_bx_char_map ²	Text Object	Set character mapping for Bitstream Fonts	61
set_card_base ¹	System	Associate the HC/2 card(s) with an I/O address.	9
set_char_height	Text Object	Set the height for text characters.	59
set_char_slant	Text Object	Change amount of slant for each character of a text object.	60
set_char_width	Text Object	Set the ratio of width to height for text characters.	59
set_check_code_flag	BarCode Object	Set toggle for 3 of 9 check code to be added.	73
set_config_file	System	Set the grid correction file.	16
set_corr_file ¹	System	Cause the correction file to be associated with the specified table number.	12
set_corr_table ¹	System	Associate the HC/2 or HC/3 card number with a correction table number.	11
set_density	BarCode Object	Set number of steps between fill bars when filling a black bar.	72
set_direction	Radial Text Object	Radial text will be clockwise or counter clockwise.	66
set_font	Text Object	Font to use when rendering the text for the object.	57
set_head_type ¹	System	Set the used Scan Head type.	12
set_height	BarCode Object	Specify the height of the barcode in field units.	71
set_inverse_flag	BarCode Object	Set whether barcode should be executed normally or inverted.	73
set_irq_number ¹	System	Associate the HC/2 or HC/3 Master card with an IRQ.	8
set_laser ¹	System	Switch the laser beam on for an indefinite amount of time.	26
set_mark_error	User Interface Port	Use to assert the MARK_ERROR signal.	32
set_mark_flag	General Object	Set the execute flag for the object.	44
set_message	System	Set a PC-MARK MT (or PC-MARK) message of up to 60 characters.	19
set_message	Group	Set a group message with up to 60 characters in length.	36

set_message	General Object	Set the object message.	42
set_n	BarCode Object	Specify the wide to narrow ratio for the barcode.	69
set_orientation	Straight Text Object	Specify whether text will be laid out horizontal or vertical.	63
set_para_set	System	Set 1 of 64 sets of laser parameters.	20
set_para_set	General Object	Set a pointer to 1 of the 64 parameter sets.	43
set_param_file	System	Set name of laser parameter file.	17
set_pen_flag	Vector Object	Pen changing on vector objects having internal pen changes.	54
set_pens	Vector Object	Map HPGL and MCL pens to laser parameter sets.	55
set_power_scale	System	Compensate laser lamp power variations.	27
set_qz	BarCode Object	Set the barcode quiet zone.	70
set_radius	Radial Text Object	Set the radius for the radial text.	65
set_source	Vector Object	Set the path name for the vector object.	53
set_string	Text Object	Specify the string of text that will be used.	58
set_string	BarCode Object	Specify the string that is to be barcoded.	67
set_t	BarCode Object	Set the space between characters.	71
set_target_velocity	System	Enable application to supply X & Y velocity of the object.	28
set_vector_wrap	Vector Object	Set the Vector Wrapping Option	56
set_x	BarCode Object	Specify the Narrow Element dimension of the barcode.	68
slant	Object Transform	Skew object in X dimension while leaving Y dimension alone.	50
start_mark	User Interface Port	Start a scanning operation.	29
start_mark_on_begin	User Interface Port	Start scanning that will be gated by BEGIN_MARK signal.	30
system	System	Return system statistics.	24
unload	System	Unload PC-MARK MT (or PC-MARK) to free memory.	8
vector_count	General Object	Return the number of vectors in the object.	47
x_flip	Object Transform	Mirror the object around the Y axis.	49
y_flip	Object Transform	Mirror the object around the X axis.	49

¹ PC-MARK MT Only.

² Not available for PC-MARK MT.