IPL5X

5 axes linear interpolator for CNC

DEVELOPER GUIDE

Rev. No.	History
0.32	Work in progress

Features

IPL5X is a 5 axes linear interpolator. All axes are computed at the same time.
Outputs timing precision and resolution have been the main focus during the development of this interface. Linear segments are handled continuously without any interruption in between.

USB

- Configuration
- Live data for driving a CNC
- Update firmware

Flash

- Stored data for driving a CNC
- Size: 2MB
- Max number of files: 32
- Max number of segments: 83885

Manual inputs

- LCD (2*20 or 4*20 characters) and Keyboard (6 keys)
- Drive a CNC without an attached PC

Outputs

- Free association between axes and outputs
- Possibility to duplicate one or more axes to multiple outputs (XG=XD=X)
- Possibility to reverse the polarity independently on each step or direction

Performances

- Interpolation frequency on each of the 5 axes: 50 kHz
    - For compatibility, this frequency can be lowered to 10,20,30 or 40kHz
- Number of internal segments buffer: 8 (USB and Flash)
- Built in acceleration and deceleration profile and ramps (16 available)

PWM

- Frequency and resolution: Interpolation frequency/128 and 8 bits
- PWM On/Off selection with unwanted startup protection
- Auto and Manual PWM selection
- Manual PWM set by an external potentiometer

Inputs

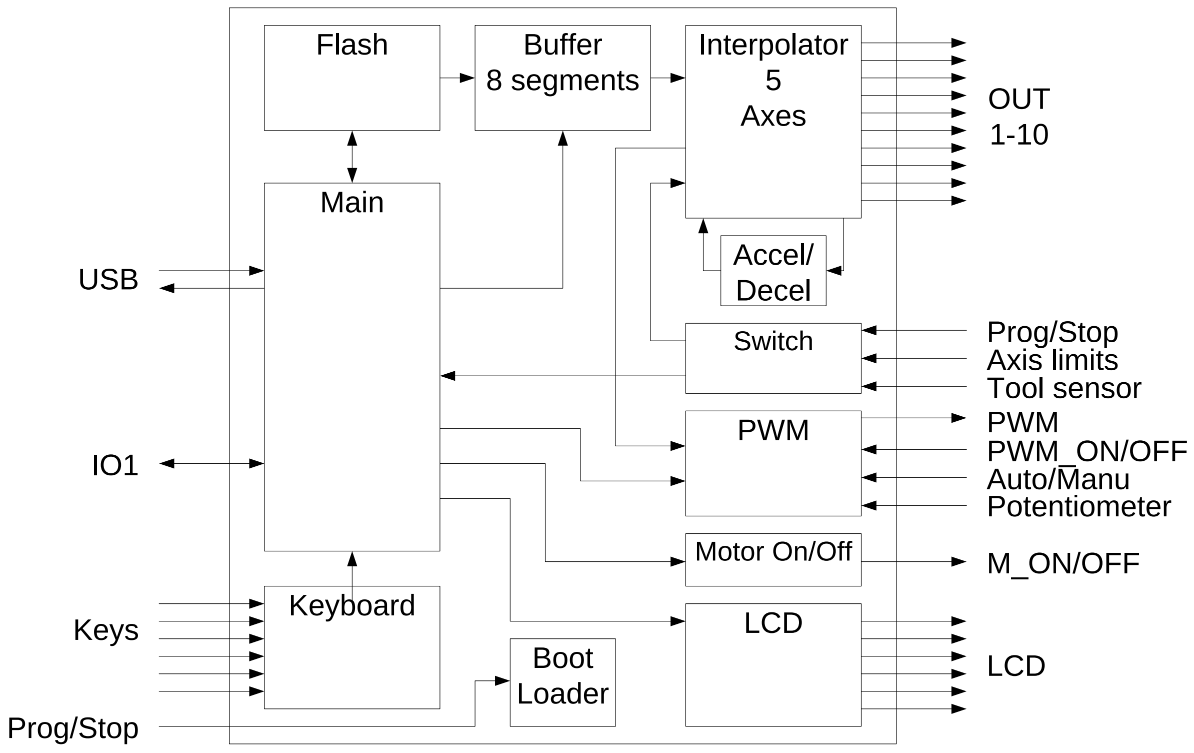
- 1 input for axis limits
- 1 input for tool sensor
- Switches can be ignored if unsused

1 available input/output for end application use or used as an PWM on/off follower

From a design perspective this list of hardware is optional:

- Flash
- LCD
- Keyboard

Block diagram



USB Details

IPL5X is a USB 2.0 device working at full speed.

It identifies itself on the system as a HID compatible device with the following IDs:

VENDOR ID	0x04D8
PRODUCT ID	0x00AA

USB input and output buffers length are 36 Bytes.

Polling time is set to 1ms.

Instructions

Every instruction sequence starts with a one-byte instruction code. Depending on the instruction, this might be followed by commands or data bytes.

Instruction Set

Instruction	Description	Instruction code
Version	Firmware version	0x56 'V'
Buffer	Buffer management	0x42 'B'
Data	Add data to buffer in USB mode	0x44 'D'
Information	Get interface status	0x49 'I'
Stop	Stop steps activities	0x53 'S'
Table	Update table details	0x54 'T'
PWM	PWM ON/OFF + Value	0x50 'P'
Motors	Motors ON/OFF	0x4D 'M'
General input/output	IO1 control	0x47 'G'
Override	Override limits/sensor switches	0x4F 'O'
LCD	Display a string on LCD	0x4C 'L'
Reset	Reboot the interface	0x5A 'Z'
Flash Model	Flash	0x46 'F'
Flash Read	Read 32 bytes of data	0x52 'R'
Flash Block Erase	Erase a block of data	0x45 'E'
Flash Write	Write 32 bytes of data	0x57 'W'
Flash Content	List of files and used blocks	0x42 'C'
MM2001 Programming	Read/Write MM2001 program	0x59 'Y'

To better document
To code
To code and document...

The 1st byte sent by the interface in response is always the instruction code.

Version

Description:

This instruction returns the name and current version of the interface in ascii format.

PC

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x56	-	-	-	-	-	-	-	-	-	-

IPL5X

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x56	'I'	'P'	'L'	'5'	'X'	' '	'v'	x	'.'	y

Byte 11
z

x: ascii major version
y: ascii minor version
z: ascii sub-minor version

Buffer management

Description:

The FIFO buffer (First In First Out) holds a maximum of 8 data information about speed, steps, PWM and acceleration.
Before any data input a reset must occur to initialize correctly the buffer. Not doing so will result in unreliable results.
Recommended procedure to write data in the buffer:

1. Reset buffer content
2. Fix data source: USB or Flash
3. USB: Fill the buffer with one or more data. See the data command.
   Flash: wait for the buffer to fill up. See the information command.
4. Execute the Buffer content
5. USB: continue to fill the buffer with data
   Flash: data will automatically be added to the buffer

PC

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x42	CMD	BLCK	ADRH	ADRL	-	-	-	-	-	-

CMD

- 0x00 Reset buffer and stop any ongoing steps activity
- 0x01 Data source is USB
- 0x02 Data source is Flash and will start to read at address BCK:ADRH:ADRL (24 bits)
- 0x80 Execute the current Buffer content

BLCK:	Block number	(0-31 <-> 0x00-0x1F)
ADRH:	High order address	(0-255 <-> 0x00-0xFF)
ADRL:	Low order address	(0-255 <-> 0x00-0xFF)

ADRH:ADRL represents the address of each bytes in the block

IPL5X

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x42	RET	-	-	-	-	-	-	-	-	-

RET

- 0x00 Unknown command
- 0xFF Ok

Example: USB data to move motor 1 of 10 steps, forward direction in 1sec

Reset

PC

Byte 0	Byte 1
0x42	0x00

IPL5X

Byte 0	Byte 1
0x42	0xFF

Source=USB

PC

Byte 0	Byte 1
0x42	0x01

IPL5X

Byte 0	Byte 1
0x42	0xFF

Fill Buffer with one data ('D'=0x44, 1 sec=0x0000C34F, 10 steps on M1=10*2=0x00000014)

PC

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x44	00	4F	C3	00	00	14	00	00	00	00
Byte 11	**Byte 12**	**Byte 13**	**Byte 14**	**Byte 15**	**Byte 16**	**Byte 17**	**Byte 18**	**Byte 19**	**Byte 20**	**Byte 21**
00	00	00	00	00	00	00	00	00	00	00
Byte 22	**Byte 23**	**Byte 24**	**Byte 25**							
00	00	00	00							

IPL5X

Byte 0	Byte 1
0x44	0x01

The data command returns a lot more information but it has been shorten for this example. The return value of 0x01 means that the data has been stored.

Note: more data could be sent to fill up the buffer. Once the buffer is full, every data pushed to it will be refused and the response will change to 0xFF.

Execute

PC

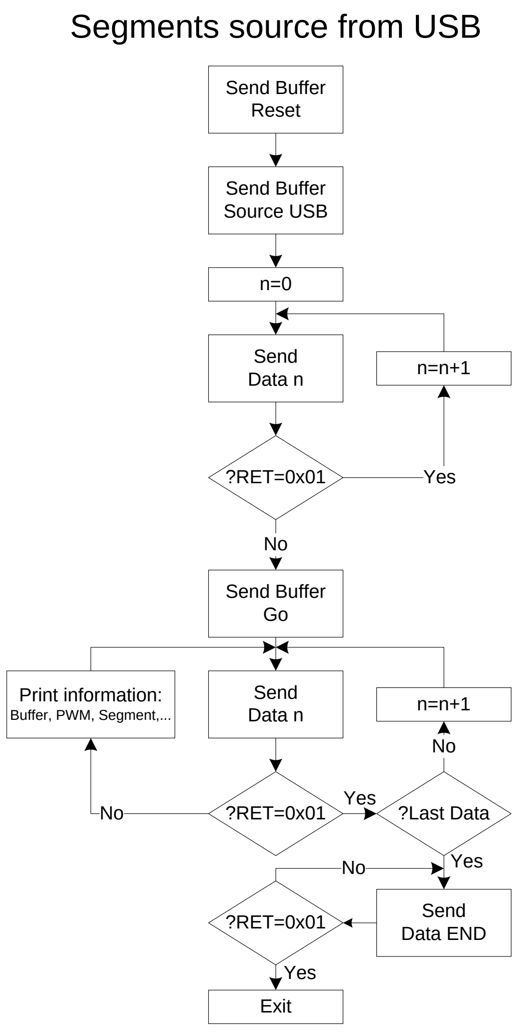
Byte 0	Byte 1
0x42	0x80

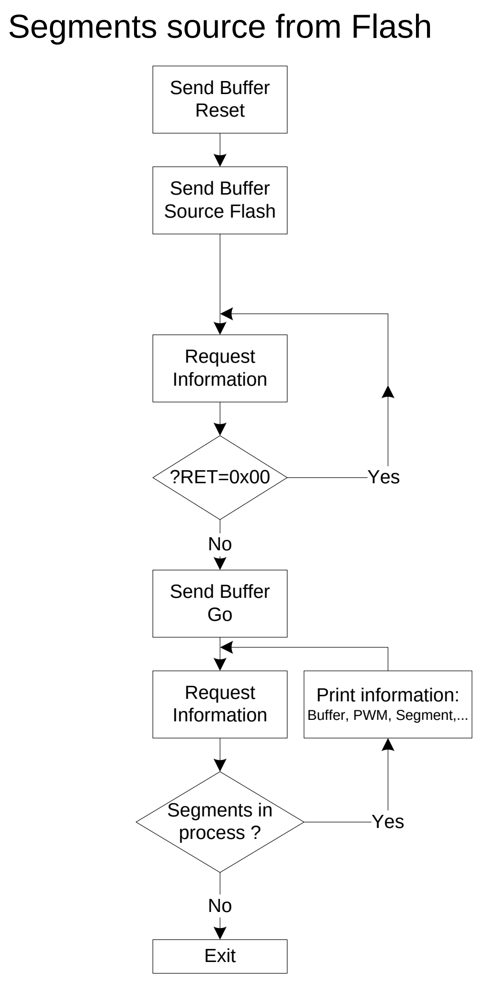
Once sent, motor1 will begin to move.

IPL5X

Byte 0	Byte 1
0x42	0xFF

Note: Any subsequent data write will be stored in the buffer (if not full) and executed after the previous data are processed. At the end of the data stream, DATA END should be sent to stop the interpolation process.





Data

Description:

This instruction data provides the necessary information while using USB to move to a given position in a certain amount of time.

PC

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x44	CMD	NBRL	NBRM	NBRH	NBRU	S1L	S1M	S1H	S1U	S2L
Byte 11	**Byte 12**	**Byte 13**	**Byte 14**	**Byte 15**	**Byte 16**	**Byte 17**	**Byte 18**	**Byte 19**	**Byte 20**	**Byte 21**
S2M	S2H	S2U	S3L	S3M	S3H	S3U	S4L	S4M	S4H	S4U

CMD Bit 6=0 & Bit 5=0

Byte 22	Byte 23	Byte 24	Byte 25	Byte 26	Byte 27	Byte 28	Byte 29	Byte 30	Byte 31	Byte 32
S5L	S5M	S5H	S5U	-	-	-	-	-	-	-

CMD Bit 6=1 & Bit 5=0

Byte 22	Byte 23	Byte 24	Byte 25	Byte 26	Byte 27	Byte 28	Byte 29	Byte 30	Byte 31	Byte 32
S5L	S5M	S5H	S5U	PWM	-	-	-	-	-	-

CMD Bit 6=0 & Bit 5=1

Byte 22	Byte 23	Byte 24	Byte 25	Byte 26	Byte 27	Byte 28	Byte 29	Byte 30	Byte 31	Byte 32
S5L	S5M	S5H	S5U	F_ACC	F_DEC	DECL	DECM	DECH	DECU	-

CMD Bit 6=1 & Bit 5=1

Byte 22	Byte 23	Byte 24	Byte 25	Byte 26	Byte 27	Byte 28	Byte 29	Byte 30	Byte 31	Byte 32
S5L	S5M	S5H	S5U	PWM	F_ACC	F_DEC	DECL	DECM	DECH	DECU

CMD

- Bit 7
    - 1 -> end of data: this will stop all step activities, shut down the PWM and turn off motors (if in auto mod).
    - 0 -> valid data to be stored in the buffer
- Bit 6
    - 1 -> PWM value for auto mode is present
    - 0 -> PWM value for auto mode is not present
- Bit 5
    - 1 -> acceleration/deceleration data is present
    - 0 -> acceleration data is not present
- Bit 4: direction bit for M5
- Bit 3: direction bit for M4
- Bit 2: direction bit for M3
- Bit 1: direction bit for M2
- Bit 0: direction bit for M1

NBRU:NBRH:NBRM:NBRL (32 bits) -> total number of interpolation pulse (0 is invalid)
NBR= Upper rounded value of (Time_sec*Main_frequency_Hz)
S1U:S1H:S1M:S1L (32 bits) -> two times the number of M1 steps during the time NBR

- S1=2 * Lower rounded value of M1steps

S2U:S2H:S2M:S2L (32 bits) -> two times the number of M2 steps during the time NBR

- S2=2 * Lower rounded value of M2steps

S3U:S3H:S3M:S3L (32 bits) -> two times the number of M3 steps during the time NBR

- S3=2 * Lower rounded value of M3steps

S4U:S4H:S4M:S4L (32 bits) -> two times the number of M4 steps during the time NBR

- S4=2 * Lower rounded value of M4steps

S5U:S5H:S5M:S5L (32 bits) -> two times the number of M5 steps during the time NBR

- S5=2 * Lower rounded value of M5steps

PWM (8 bits) -> auto mode PWM value=0x00..0xFF=0..255

- The PWM value should be sent only when it needs to be changed. The value is loaded at this specific segment and will be used by the system until another value is sent. If the PWM_ON/OFF button is on position ON, PWM will be enabled.

F_ACC -> Index of the frequency in the acceleration table to use at beginning of acceleration
F_DEC -> Index of the frequency in the acceleration table to use as a target for deceleration
DECU:DECH:DECM:DECL (32 bits) -> Number of pulses after which deceleration starts

IPL5X	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
	0x44	ST	BUF	SEGL	SEGM	SEGH	FLAG	INPUT	PWMA	PWMM	KEYB
	Byte 11	**Byte 12**	**Byte 13**	**Byte 14**	**Byte 15**	**Byte 16**	**Byte 17**	**Byte 18**	**Byte 19**	**Byte 20**	**Byte 21**
	STOP	NBRL	NBRM	NBRH	NBRU	-	-	-	-	-	-

ST

- 0x00 Error not in USB mode
- 0x01 Data stored in the buffer
- 0xFF Buffer full -> data has not been stored

Byte 2 - 21: refer to instruction “Information” for more details

Examples

A working excel file is available with all the formulas to generate the data instruction.

Flash and Data:

To execute from Flash, you need to write the content of Data one after the other without the leading byte 0 0x44. The length of this content should be adjusted to only what’s needed, ie do not include the bytes with “-“. It is important to finish by the command end of data which represents end of file.

Example

This example turns on PWM to 50% and wait for the tool to spin up for 5sec (at 50kHz), then move the motor 1 forward for 100 steps in 1sec, then turns the PWM off and finally stop.
Content to write in Flash starting at address BLCK=0x00, ADRH=0x00, ADRL=0x00:

000000: 40 90 D0 03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 7F 00 50 C3 00 00 64

000020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 40 01 00 00 00 00 00 00 00 00 00 00 00

000040: 00 00 00 00 00 00 00 00 00 00 00 00 00 80

Information

Description:

This instruction provides the status of the interface.

PC

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x49	-	-	-	-	-	-	-	-	-	-

IPL5X

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x49	ST	BUF	SEGL	SEGM	SEGH	FLAG	INPUT	PWMA	PWMM	KEYB
Byte 11	**Byte 12**	**Byte 13**	**Byte 14**	**Byte 15**	**Byte 16**	**Byte 17**	**Byte 18**	**Byte 19**	**Byte 20**	**Byte 21**
STOP	NBRL	NBRM	NBRH	NBRU	-	-	-	-	-	-

ST

- 0x00 Buffer not full
- 0xFF Buffer full

BUF : buffer usage

- Value between 0x00 and 0x07
  - 0x00=0% (empty), 0x01=14%, 0x02=29%, 0x03=43%
  - 0x04=57%, 0x05=71%, 0x06=86%, 0x07=100% (full)

SEGH:SEGM:SEGL (24 bits) : segment number being processed

- 0x000000 no segment in process
- 0x000001 1st segment
- 0x000002 2nd segment
- …

FLAG

- Bit 0 : 1 -> segments and steps are being processed, 0 -> nothing in progress
- Bit 1 : 1 -> stop instruction is pending, 0 -> no stop in progress
- Bit 2 : internal use
- Bit 3 : 1 -> source USB, 0 -> source Flash
- Bit 4 : 1 -> PWM on, 0 -> PWM off
- Bit 5 : 1 -> origin/end switches override
- Bit 6 : 1 -> Motor on, 0 -> Motor off
- Bit 7 : 1 -> origin/end switches reversed

INPUT : external switches status

- Bit 0 : 1 -> PWM manual mode, 0 -> PWM auto mode
- Bit 1 : 1 -> axis limits contact open, 0 -> axis limits contact close
- Bit 2 : 1 -> tool sensor contact open, 0 -> tool sensor contact close
- Bit 3 : 1 -> PWM_ON/OFF is in position OFF, 0->ON
- Bit 4 : value of IO1 +5V=1, GND=0
- Bit 5 : 1 -> PROG not pressed, 0 -> PROG pressed
- Bit 6 : none
- Bit 7 : none

PWMA : PWM value in auto mode

- Value from 0x00=0d=0% to 0xFF=255d=100%

PWMM : PWM value in manual mode

- Value from 0x00=0d=0% to 0xFF=255d=100%

KEYB : Keyboard status

- Bit 0 : 1 -> ESC
- Bit 1 : 1 -> OK
- Bit 2 or Bit 3: 0 -> Keyboard detected, 1 -> Keyboard not detected
- Bit 4 : 1 -> LEFT
- Bit 5 : 1 -> DOWN
- Bit 6 : 1 -> UP
- Bit 7 : 1 -> RIGHT

STOP

- Bit 1: 0 Step activity running=Interpolator enabled
  1 Step activity stopped=Interpolator disabled
- Bit 2: 0 No stop command has been received
  1 Stop command has been received
- Bit 4: 1 Stop is due to a origin/end switch
- Bit 5: 1 Stop is due to a sensor switch
- Bit 6: 1 Stop is due to PROG/STOP button
- Bit 7: 1 Stop is due to end of data processed from the buffer

NBRU:NBRH:NBRM:NBRL (32 bits) number of pulses left for the current segment

To calculate the current absolute position:

1. Sum up all the steps for each axis from segment 1 to SEGH:SEGM:SEGL-1
2. Add to the previous for each axis:

$$\mathrm{STEPc} = \frac{\mathrm{STEPt} * (\mathrm{NBRt} - \mathrm{NBRc} - 1)}{\mathrm{NBRt}}$$

STEPc = the current number of steps done on the segment SEGH:SEGM:SEGL
STEPt = the total number steps for segment SEGH:SEGM:SEGL
NBRt = the total number of pulses for segment SEGH:SEGM:SEGL
NBRc = NBRU:NBRH:NBRM:NBRL = number of pulses left for the segment SEGH:SEGM:SEGL

Stop

Description:

This command stops a current segment in process.
There is 2 stop options fast or slow which use or not deceleration.
Stop conditions are:

- Stop commands
- Emergency push button (STOP/PROG)
- Begin/End switches
- End of data

The return value gives an indication of how many steps have been done until the full stop happened.
A stop condition will automatically shut down the PWM.

PC

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x53	CMD	FDEC	-	-	-	-	-	-	-	-

CMD

- 0x00 Fast stop: stops immediately the steps activity even if the current move was using acceleration
- 0x01 Slow stop: stops after a deceleration the steps activity. Deceleration will take some time to complete, you need to poll the "Step activity" bit for a status
  - FDEC is used as the deceleration factor. It is determined the same way than the parameter F_DEC from the Data instruction.
- 0x03 Get status information

IPL5X

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x53	RET	SEGL	SEGM	SEGH	NBRL	NBRM	NBRH	NBRU	-	-

RET

- Bit 1: 0 Step activity running=Interpolator enabled
  1 Step activity stopped=Interpolator disabled
- Bit 2: 0 No stop command has been received
  1 Stop command has been received
- Bit 4: 1 Stop is due to a origin/end switch
- Bit 5: 1 Stop is due to a sensor switch
- Bit 6: 1 Stop is due to PROG/STOP button
- Bit 7: 1 Stop is due to end of data processed from the buffer

SEGH:SEGM:SEGL (24 bits) : segment number being processed

- 0x000000 no segment in process
- 0x000001 1st segment
- 0x000002 2nd segment
- …

NBRU:NBRH:NBRM:NBRL (32 bits) number of pulses left for the segment after the full stop

Note: During auto motor power up, step activity will be running and segment will be 0x000000.

To calculate the absolute position after a stop:

3. Sum up all the steps for each axis from segment 1 to SEGH:SEGM:SEGL-1
4. Add to the previous for each axis:

$$\mathrm{STEPc} = \frac{\mathrm{STEPt} * (\mathrm{NBRt} - \mathrm{NBRc} - 1)}{\mathrm{NBRt}}$$

STEPc	= the current number of steps done on the segment SEGH:SEGM:SEGL
STEPt	= the total number steps for segment SEGH:SEGM:SEGL
NBRt	= the total number of pulses for segment SEGH:SEGM:SEGL
NBRc	= NBRU:NBRH:NBRM:NBRL = number of pulses left for the segment SEGH:SEGM:SEGL

TABLE

Description:

This instruction gives access to the tables settings.
There are 3 different tables that can be stored in EEPROM.
To read or write one table, 3 reads/writes should be done to get the low, high and upper part.
After writing a table description, you need to load it in memory using the Write table number command (0x01).

Read the table number in use from EEPROM

PC

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x54	CMD	-	-	-	-	-	-	-	-	-

CMD	0x00	Read table number from EEPROM

IPL5X

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x54	CMD	NBR	-	-	-	-	-	-	-	-

CMD	0x00	Read table number from EEPROM
NBR	Table number in use from 0 to 2	

Write the table number to use to EEPROM and load it in memory

PC

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x54	CMD	NBR	-	-	-	-	-	-	-	-

CMD	0x01	Write table number to EEPROM
NBR	Table number to use from 0 to 2	

IPL5X

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x54	RET	-	-	-	-	-	-	-	-	-

RET:

- 0xFF Table number change FAILED !!! and not loaded to memory
- 0x00 Table number changed and load to memory completed successfully

Read the low part of a given table

PC

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x54	CMD	NBR	-	-	-	-	-	-	-	-

CMD	0x03	Read low
NBR	Table number to access from 0 to 2	

IPL5X

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x54	CMD	NBR	N [1]	N [2]	N [3]	N [4]	N [5]	N [6]	N [7]	N [8]
Byte 11	**Byte 12**	**Byte 13**	**Byte 14**	**Byte 15**	**Byte 16**	**Byte 17**	**Byte 18**	**Byte 19**	**Byte 20**	**Byte 21**
A1	A2	A3	A4	A5	OUT1	OUT2	OUT3	OUT4	OUT5	OUT6
Byte 22	**Byte 23**	**Byte 24**	**Byte 25**	**Byte 26**	**Byte 27**	**Byte 28**	**Byte 29**	**Byte 30**	**Byte 31**	**Byte 32**
OUT7	OUT8	OUT9	OUT10	FREQ	SLOPE	PWM	MOFF	FLAGS	IO1	LANG
Byte 33	**Byte 34**	**Byte 35**								
0x00	0x00	0x00								

CMD	0x03	Read low
NBR	Table number (0-2)	
N[1..8]	Table name composed of 8 ASCII characters	

- The end of the string should be completed with spaces
- If this table contains no valid information, the name must be filled with 0x00

A1..5 represent the type of the 5 axes, valid values are:

0x00-0x03	0x04	0x05	0x06	0x08	0x09	0x0A	0x0C	0x0D	0x0E	
Not used	X	XL	XR	Y	YL	YR	Z	ZL	ZR	
	0x10	**0x11**	**0x12**	**0x20**	**0x21**	**0x22**	**0x23**	**0x30**	**0x31**	**0x32**
	A	B	C	R	RX	RY	RZ	U	V	W

OUTx represent the available outputs to control axes

- Bit 7
  - 0 = DIR
  - 1 = STEP
- Bit 6
  - 0 = normal signal
  - 1 = reversed signal
- Bit 2..0
  - Axe number: 1-5
- If an axe is not used, the value must be set to 0
- Note: the same axe can have multiple dir or/and step outputs

FREQ represents the interpolation frequency

- 1=10kHz, 2=20kHz, 3=30kHz, 4=40kHz, 5=50kHz

SLOPE represents the available acceleration slope

- Value from 0x00:fastest to 0x0F:slowest

PWM max PWM value that can be reached in auto or manual

- Value from 0x00 to 0xFF

MOFF Time in seconds between the end of steps activity and motors power off

- 0x00 disable auto motors on/off
- 0x01-0x7F : 1 - 127 seconds

FLAGS

- Bit 0: limits/sensor switches 0 = disabled, 1 = enabled
- Bit 1: Table orientation 0 = normal, 1 = reversed
- Bit 2: 1 = M_ON/OFF pin reversed

- Bit 3-7: Future use

IO1

- Bit 0: 0=output, 1=input
- Bit 1: output default value 0 or 1
- Bit 2: 1 => output follow PWM -> PWM on IO1=1, PWM off IO1=0
- Bit 3-6: Future use
- Bit 7: debug output

LANG

- 0=French, 1=English

Example: Read the low part of the table description number 1

PC

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x54	0x03	0x1	-	-	-	-	-	-	-	-

IPL5X

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x54	0x03	0x1	‘M’	‘M’	‘2’	‘0’	‘0’	‘1’	‘ ‘	‘ ‘
Byte 11	**Byte 12**	**Byte 13**	**Byte 14**	**Byte 15**	**Byte 16**	**Byte 17**	**Byte 18**	**Byte 19**	**Byte 20**	**Byte 21**
0x05	0x06	0x09	0x0A	0x00	0x03	0x83	0x01	0x81	0x04	0x84
Byte 22	**Byte 23**	**Byte 24**	**Byte 25**	**Byte 26**	**Byte 27**	**Byte 28**	**Byte 29**	**Byte 30**	**Byte 31**	**Byte 32**
0x02	0x82	0x00	0x00	0x01	0x07	0xCC	0x0A	0x00	0x04	0x00
Byte 33	**Byte 34**	**Byte 35**								
0x00	0x00	0x00								

Table name	-> “MM2001 “	
Axe 1 =	XL DIR	-> OUT3
	XL STEP	-> OUT4
Axe 2 =	XR DIR	-> OUT7
	XR STEP	-> OUT8
Axe 3 =	YL DIR	-> OUT1
	YL STEP	-> OUT2
Axe 4 =	YR DIR	-> OUT5
	YR STEP	-> OUT6
Axe 5 =	0x00	
Not affected	-> OUT9, 10	
FREQ	-> 0x01 => 10kHz	
Slope	-> 0x07	
PWM Max	-> 0xCC	=> 80%

Auto motors on/off enabled, stop after 0x0A=10 seconds of inactivity

FLAGS	-> 0x00	=> limits/sensor switches disabled
		=> Table orientation normal
IO1	-> 0x04	=> IO1 set to output and follow PWM on/off
LANG	-> 0x00	=> French

Write the low part of a given table

PC

Byte 0	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**	**Byte 8**	**Byte 9**	**Byte 10**
0x54	CMD	NBR	N [1]	N [2]	N [3]	N [4]	N [5]	N [6]	N [7]	N [8]
Byte 11	**Byte 12**	**Byte 13**	**Byte 14**	**Byte 15**	**Byte 16**	**Byte 17**	**Byte 18**	**Byte 19**	**Byte 20**	**Byte 21**
A1	A2	A3	A4	A5	OUT1	OUT2	OUT3	OUT4	OUT5	OUT6
Byte 22	**Byte 23**	**Byte 24**	**Byte 25**	**Byte 26**	**Byte 27**	**Byte 28**	**Byte 29**	**Byte 30**	**Byte 31**	**Byte 32**
OUT7	OUT8	OUT9	OUT10	FREQ	SLOPE	PWM	MOFF	FLAGS	IO1	LANG
Byte 33	**Byte 34**	**Byte 35**								
-	-	-								

CMD	0x04	Write low
NBR	Table number (0-2)	
N[1..8]	Table name composed of 8 ASCII characters	

- The end of the string should be completed with spaces
- If this table contains no valid information, the name must be filled with 0x00

A1..5 represent the type of the 5 axes, valid values are:

0x00-0x03	**0x04**	**0x05**	**0x06**	**0x08**	**0x09**	**0x0A**	**0x0C**	**0x0D**	**0x0E**	
Not used	X	XL	XR	Y	YL	YR	Z	ZL	ZR	
	0x10	**0x11**	**0x12**	**0x20**	**0x21**	**0x22**	**0x23**	**0x30**	**0x31**	**0x32**
	A	B	C	R	RX	RY	RZ	U	V	W

OUTx represent the available outputs to control axes

- Bit 7 0 = DIR
  1 = STEP
- Bit 6 0 = normal signal
  1 = reversed signal
- Bit 2..0 Axe number: 1-5
  - If an axe is not used, the value must be set to 0x00
- Note: the same axe can have multiple dir or/and step outputs

FREQ represents the interpolation frequency

- 1=10kHz, 2=20kHz, 3=30kHz, 4=40kHz, 5=50kHz

SLOPE represents the available acceleration slope

- Value from 0x00:fastest to 0x0F:slowest

PWM max PWM value that can be reached in auto or manual

- Value from 0x00 to 0xFF

MOFF Time in seconds between the end of steps activity and motors power off

- 0x00 disable auto motors on/off
- 0x01-0x7F : 1 - 127 seconds

FLAGS

- Bit 0: Limits/Sensor switches 0 = disabled, 1 = enabled
- Bit 1: Table orientation 0 = normal, 1 = reversed
- Bit 2: Motors on/off pin reversed

IO1

- Bit 0: 0=output, 1=input
- Bit 1: output default value 0 or 1
- Bit 2: output follow PWM -> PWM on IO1=1, PWM off IO1=0
- Bit 3-6: Future use
- Bit 7: debug output

LANG

- 0=French, 1=English

IPL5X

Byte 0	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**	**Byte 8**	**Byte 9**	**Byte 10**
0x54	RET	-	-	-	-	-	-	-	-	-

RET:	0xFF	Write FAILED !!!
	0x00	Table low written

Read the high part of a given table

PC

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x54	CMD	NBR	-	-	-	-	-	-	-	-

CMD	0x05	Read high
NBR	Table number to access from 0 to 2	

IPL5X

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x54	CMD	NBR	S1L	S1H	S2L	S2H	S3L	S3H	S4L	S4H
Byte 11	**Byte 12**	**Byte 13**	**Byte 14**	**Byte 15**	**Byte 16**	**Byte 17**	**Byte 18**	**Byte 19**	**Byte 20**	**Byte 21**
S5L	S5H	V1ML	V1MH	V2ML	V2MH	V3ML	V3MH	V4ML	V4MH	V5ML
Byte 22	**Byte 23**	**Byte 24**	**Byte 25**	**Byte 26**	**Byte 27**	**Byte 28**	**Byte 29**	**Byte 30**	**Byte 31**	**Byte 32**
V5MH	V1MAL	V1MAH	V2MAL	V2MAH	V3MAL	V3MAH	V4MAL	V4MAH	V5MAL	V5MAH

NBR	Table number (0-2)
SxM:SxL (16 bits)	-> Number of steps to do 1mm
VxMM: VxML (16 bits)	-> Max number of steps without acceleration in 1 second
VxMAM: VxMAL (16 bits)	-> Max number of steps with acceleration in 1 second

Example: Read the high part of the table description number 1

PC

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x54	0x05	0x1	-	-	-	-	-	-	-	-

IPL5X

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x54	0x05	0x1	0xC8	0x00	0xA0	0x00	0x80	0x02	0x90	0x01
Byte 11	**Byte 12**	**Byte 13**	**Byte 14**	**Byte 15**	**Byte 16**	**Byte 17**	**Byte 18**	**Byte 19**	**Byte 20**	**Byte 21**
0x00	0x00	0xE8	0x03	0x10	0x04	0xC0	0x11	0xF8	0x0C	0x00
Byte 22	**Byte 23**	**Byte 24**	**Byte 25**	**Byte 26**	**Byte 27**	**Byte 28**	**Byte 29**	**Byte 30**	**Byte 31**	**Byte 32**
0x00	0xD0	0x07	0xB0	0x07	0x00	0x33	0x20	0x30	0x00	0x00

S1M:S1L = 0x00C8	-> 200steps for 1mm
S2M:S2L = 0x00A0	-> 160steps for 1mm
S3M:S3L = 0x0280	-> 640steps for 1mm
S4M:S4L = 0x0190	-> 400steps for 1mm
S5M:S5L = 0x0000	-> 0x00 Unused

V1MM:V1ML=0x03E8	-> 0x03E8/0x00C8=5mm/s
V2MM:V2ML=0x0410	-> 0x0410/0x00A0=6,5mm/s
V3MM:V3ML=0x11C0	-> 0x11C0/0x0280=7,1mm/s
V4MM:V4ML=0x0CF8	-> 0x03E8/0x0190=8,3mm/s
V5MM:V5ML=0x0000	-> 0x00 Unused

V1MAM:V1MAL=0x07D0	-> 0x07D0/0x00C8=10mm/s
V2MAM:V2MAL=0x07B0	-> 0x07B0/0x00A0=12,3mm/s
V3MAM:V3MAL=0x3300	-> 0x3300/0x0280=20,4mm/s
V4MAM:V4MAL=0x3020	-> 0x3020/0x0190=30,8mm/s
V5MAM:V5MAL=0x0000	-> 0x00 Unused

Write the high part of a given table

PC

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x54	CMD	NBR	S1L	S1M	S2L	S2M	S3L	S3M	S4L	S4M
Byte 11	**Byte 12**	**Byte 13**	**Byte 14**	**Byte 15**	**Byte 16**	**Byte 17**	**Byte 18**	**Byte 19**	**Byte 20**	**Byte 21**
S5L	S5M	V1ML	V1MM	V2ML	V2MM	V3ML	V3MM	V4ML	V4MM	V5ML
Byte 22	**Byte 23**	**Byte 24**	**Byte 25**	**Byte 26**	**Byte 27**	**Byte 28**	**Byte 29**	**Byte 30**	**Byte 31**	**Byte 32**
V5MM	V1MAL	V1MAM	V2MAM	V2MAH	V3MAM	V3MAH	V4MAM	V4MAH	V5MAM	V5MAH

CMD	0x06	Write high
NBR	Table number (0-2)	

SxM:SxL (16 bits)	-> Number of steps to do 1mm
VxMM: VxML (16 bits)	-> Max number of steps without acceleration in 1 second
VxMAM: VxMAL (16 bits)	-> Max number of steps with acceleration in 1 second

IPL5X

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x54	RET	-	-	-	-	-	-	-	-	-

RET:

- 0xFF Write FAILED !!!
- 0x00 Table high written

Read the upper part of a given table

PC

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x54	CMD	NBR	-	-	-	-	-	-	-	-

CMD	0x07	Read upper
NBR	Table number to access from 0 to 2	

IPL5X

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x54	CMD	NBR	ORI1L	ORI1M	ORI2L	ORI2M	ORI3L	ORI3M	ORI4L	ORI4M
Byte 11	**Byte 12**	**Byte 13**	**Byte 14**	**Byte 15**	**Byte 16**	**Byte 17**	**Byte 18**	**Byte 19**	**Byte 20**	**Byte 21**
ORI5L	ORI5M	-	-	-	-	-	-	-	-	-
Byte 22										
-										

NBR Table number (0-2)
ORIxM:ORIxL (16 bits) -> Number of steps to move from the Origin switches to the real Origin position on each axes

Example: Read the upper part of the table description number 1

PC

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x54	0x07	0x1	-	-	-	-	-	-	-	-

IPL5X

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x54	0x07	0x1	0xC8	0x00	0xA0	0x00	0x80	0x02	0x90	0x01
Byte 11	**Byte 12**	**Byte 13**	**Byte 14**	**Byte 15**	**Byte 16**	**Byte 17**	**Byte 18**	**Byte 19**	**Byte 20**	**Byte 21**
0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
Byte 22										
0x00										

ORI1M:ORI1L = 0x00C8	-> 200steps
ORI2M:ORI2L = 0x00A0	-> 160steps
ORI3M:ORI3L = 0x0280	-> 640steps
ORI4M:ORI4L = 0x0190	-> 400steps
ORI5M:ORI5L = 0x0000	-> 0x00 Unused

Write the upper part of a given table

PC

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x54	CMD	NBR	ORI1L	ORI1M	ORI2L	ORI2M	ORI3L	ORI3M	ORI4L	ORI4M
Byte 11	**Byte 12**	**Byte 13**	**Byte 14**	**Byte 15**	**Byte 16**	**Byte 17**	**Byte 18**	**Byte 19**	**Byte 20**	**Byte 21**
ORI5L	ORI5M	-	-	-	-	-	-	-	-	-
Byte 22										
-										

CMD	0x08	Write upper
NBR	Table number (0-2)	

ORIxM:ORIxL (16 bits) -> Number of steps to move from the Origin switches to the real Origin position on each axes

IPL5X

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x54	RET	-	-	-	-	-	-	-	-	-

RET:

- 0xFF Write FAILED !!!
- 0x00 Table upper written

PWM

Description:

This instruction writes the PWM value used in auto mode.
It can also enable/disable the PWM output. Enable only works if the PWM_ONOFF switch is in position ON.
The return values give information on the current status.

PC

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x50	CMD	VAL	-	-	-	-	-	-	-	-

CMD

- 0x00 Set PWM off
- 0x01 Set PWM on
- 0x02 Read the current status

VAL

- PWM value used while in auto mode: 0x00..0xFF=0..255

IPL5X

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x50	ST	AMO	PWMA	PWMM	-	-	-	-	-	-

ST

- 0x00 PWM off
- 0x01 PWM on

AMO

- Bit 0
  - 0 -> mode manual
  - 1 -> mode auto
- Bit 1
  - 0 -> PWM_ON/OFF is in position OFF
  - 1 -> PWM_ON/OFF is in position ON
- Bit 2-7 not used

PWMA:	PWM value used in auto mode: 0x00..0xFF=0..255
PWMM:	PWM value used in manual mode: 0x00..0xFF=0..255. This value is taken from the analog to digital conversion of the potentiometer.

Example: Set PWM on and PWM auto to 0xAA

PC

Byte 0	Byte 1	Byte 2
0x50	0x01	0xAA

IPL5X

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4
0x50	0x01	0x02	0xAA	0x55

PWM on
Mode manual
PWM_ON/OFF is in position ON
PWM auto=0xAA
PWM man=0x55
Current PWM output=0x55=85 -> (85*100)/255=33,3%

Motors ON/OFF

Description:

This instruction enables/disables the motors power supply by setting an output value high/low on pin M_ON/OFF.
By default the motors are ON when the pin M_ON/OFF is set to 1 (high level) but this behavior can be changed by modifying the current table settings (Tables->low part->FLAGS.2).
The auto motors on/off modifications done by this instruction are only valid until the interface is power cycle. If you want to store definitively this parameter, you should use the table instruction.

PC

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x4D	CMD	MOFF	-	-	-	-	-	-	-	-

CMD

- 0x00 Set Motors OFF
- 0x01 Set Motors ON
- 0x02 Read the current status
- 0x03 Set auto motors on/off value

MOFF only used if CMD=0x03
Time in seconds between the end of steps activity and motors power off

- 0x00 disable auto motors on/off
- 0x01-0x7F : 1 - 127 seconds

IPL5X

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x4D	RET	MOFF	-	-	-	-	-	-	-	-

RET

- 0x00 Motors are OFF
- 0x01 Motors are ON

MOFF Time in seconds between the end of steps activity and motors power off

- 0x00 auto motors on/off disabled
- 0x01-0x7F : 1 - 127 seconds, auto motors on/off enabled

Example: Set pin M_ON/OFF to high (1)

PC

Byte 0	Byte 1
0x4D	0x01

IPL5X

Byte 0	Byte 1
0x4D	0x01

Override limits and sensor switches

Description:

This instruction could be used for 2 purposes:

- move the axis back to the table origin using the limits switches
- move the axis out if they've reached the limits or sensor switches

If the current table is configured to NOT use the limits or sensor switches, then override is always ON and cannot be changed.

PC

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x4F	CMD	-	-	-	-	-	-	-	-	-

CMD

- 0x00 Do not override = use the limits and sensor switches
- 0x01 Override = move without taking care about the switches status
- 0x02 Read the current status

IPL5X

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x4F	RET	-	-	-	-	-	-	-	-	-

RET

- Bit 0
  - 0 -> no override = use the limits and sensor switches
  - 1 -> override = do not use the limits and sensor switches
- Bit 1
  - 0 -> current table does not support the use of the switches
  - 1 -> current table is configured to use the limits and sensor switches
- Bit 2 : 1 -> axis limit contact open, 0 -> axis limits contact close
- Bit 3 : 1 -> tool sensor contact open, 0 -> tool sensor contact close

Example: Override limit and sensor switches

PC

Byte 0	Byte 1
0x4F	0x01

IPL5X

Byte 0	Byte 1
0x4F	0x03

Override limits and sensor switches = do not take care of the switches status
Table is configured to use the limits and sensor switches

General input/output IO1

Description:

This instruction controls the IO1 pin.
If the settings are changed, they are only valid until the interface is power cycle. If you want to store these parameters, you should use the table instruction.

PC

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x47	CMD	VAL	-	-	-	-	-	-	-	-

CMD

- 0x00 Set IO1 pin low (GND)
- 0x01 Set IO1 pin high (+5V)
- 0x02 Read IO1 pin setting
- 0x03 Change IO1 pin setting

VAL used only if CMD=0x03

- Bit 0: 0=output, 1=input
- Bit 1: output default value 0 or 1
- Bit 2: output follow PWM -> PWM on IO1=1, PWM off IO1=0
- Bit 3-6: Future use
- Bit 7: debug output

IPL5X

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x47	RET	VAL	-	-	-	-	-	-	-	-

RET

- 0x00 IO1 pin is low (GND)
- 0x01 IO1 pin is high (+5V)

VAL if CMD=0x03

- Bit 0: 0=output, 1=input
- Bit 1: output default value 0 or 1
- Bit 2: =1 => output follow PWM -> PWM on IO1=1, PWM off IO1=0
- Bit 3-6: Future use
- Bit 7: debug output

Example: Set pin IO1 to high (+5V)

PC

Byte 0	Byte 1
0x47	0x01

IPL5X

Byte 0	Byte 1	Byte 2
0x47	0x01	0x01

RET=0x01 -> IO1 pin is high (+5V)
VAL=0x01 -> IO1 pin is set as an output with a default value set to 0

LCD

Description:

Modify the display on the LCD

Write a string to the LCD at the given position

PC

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	…	Byte 23
0x4C	LINE	POS	LEN	C0	…	C19

LINE: 0x00=1st line, 0x01=2nd line, 0x10=3rd line, 0x11=4th line
POS: X position on the line from 0 to 19 (0x00 to 0x13)
LEN: number of characters of the string from 1 to 20 (0x01 to 0x14)
C0, …, C19: String with a maximum of 20 characters

IPL5X

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x4C	-	-	-	-	-	-	-	-	-	-

Note: Line 3 and 4 are only valid if the LCD has 4 physical display lines

Example: Write string "IPL5X" on LINE 1,POS 7

PC

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
0x4C	0X01	0x07	0x05	0x49	0x50	0x4C	0x35	0x58

IPL5X

Byte 0
0x4C

Interface display commands

PC

Byte 0	Byte 1
0x4C	CMD

CMD

- 0x02 -> Disable interface default display
- 0x03 -> Enable interface default display
- 0x04 -> Read current status
- 0x05 -> Clear display

IPL5X

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x4C	RET	-	-	-	-	-	-	-	-	-

RET

- 0x00 -> Disable
- 0x01 -> Enable

Create a graphic character

PC

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	…	Byte 10
0x4C	CMD	CHAR	C0	C1	…	C7

- CMD: 0x06
- CHAR: Character hex code between 0 to 7 (0x00 to 0x7)
- C0, …, C7: defining the 8 lines of the character

IPL5X

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x4C	-	-	-	-	-	-	-	-	-	-

Example: Create character "|" with hex code 2

PC

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x4C	0X06	0x02	0x01	0x01	0x01	0x01	0x01	0x01	0x01	0x01

IPL5X

Byte 0
0x4C

Write the previously defined character on the LCD LINE 1,POS 7

PC

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
0x4C	0X01	0x07	0x01	0x02	-	-	-	-

IPL5X

Byte 0
0x4C

Reset

Description:

Hard reset the interface in Normal or Boot Loader mode.

PC

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x5A	0x55	0xAA	BLDR	-	-	-	-	-	-	-

BLDR:	0x00 Reboot in Normal mode
	0x01 Reboot in Boot Loader mode

There is no response of the interface. The USB connection will be dropped and reestablished at the end of the reboot.

Flash model

Description:

This instruction returns information about the installed flash component.
The Flash must not be accessed during step activities where the Flash is used to provide data.

PC

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x46	-	-	-	-	-	-	-	-	-	-

IPL5X

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x46	ID	TYPE	SIZE	WPS	-	-	-	-	-	-

ID:	Manufacturer ID
TYPE:	Memory type
SIZE:	Memory capacity
WPS:	Write Protect Status (should always be 0x00 for good operations)

Example 1: AMIC A25L016 Flash

PC

Byte 0
0x46

IPL5X

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4
0x46	0x37	0x30	0x15	0x00

Example 2: No Flash

PC

Byte 0
0x46

IPL5X

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4
0x46	0xFF	0xFF	0xFF	0xFF

Flash Read

Description:

This instruction reads 32 bytes of the Flash at a given location inside a block (64Kbytes).
2048 (0x800) read commands are needed to get the full content of a block.
The Flash must not be accessed during step activities where the Flash is used to provide data.

IPL5X

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x52	BLCK	ADRH	ADRL	-	-	-	-	-	-	-

BLCK:	Block number to read	(0-31 <-> 0x00-0x1F)
ADRH:	High order address to read	(0-255 <-> 0x00-0xFF)
ADRL:	Low order address to read	(0-255 <-> 0x00-0xFF)

Note:

ADRH:ADRL represents the address of each bytes of data within the block. To read the next set of data ADRH:ADRL should be incremented by 32.

IPL5X

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	…	Byte 33	Byte 34	Byte 35
0x52	BLCK	ADRH	ADRL	D0	D1	D2	…	D29	D30	D31

D0-D31: Data at the location ADRH:ADRL in BCK

Example:

Read flash data at address 0x0000 in block 0x01

PC

Byte 0	Byte 1	Byte 2	Byte 3
0x52	0x01	0x00	0x00

IPL5X

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	…	Byte 33	Byte 34	Byte 35
0x52	0x01	0x00	0x00	0xAA	0x55	0xFF	…	0xFF	0xFF	0xFF

Read the next flash data: address 0x0020 (+32) in block 0x01

PC

Byte 0	Byte 1	Byte 2	Byte 3
0x52	0x01	0x00	0x20

IPL5X

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	…	Byte 33	Byte 34	Byte 35
0x52	0x01	0x00	0x20	0xBB	0x66	0xFF	…	0xFF	0xFF	0xFF

Flash Block Erase

Description:

This instruction erases one of the 32 blocks of data in the Flash. Each block represents 64Kbytes.
A block must be erased before writing data to it:

- erase command sets all bits to '1' in a block
- write command sets bits to '0' only

One block erase could take up to 1.3s.

The Flash must not be accessed during step activities where the Flash is used to provide data.

IPL5X

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x45	BLCK	-	-	-	-	-	-	-	-	-

BLCK: Block number to erase (0-31 <-> 0x00-0x1F)

IPL5X

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x45	RET	-	-	-	-	-	-	-	-	-

RET:

- 0x00 Erase FAILED !!! This error should be notified to the user
- 0x01 No need to erase this block
- 0xFF Erase OK

Example: Erase block 0x01

PC

Byte 0	Byte 1
0x45	0x01

IPL5X

Byte 0	Byte 1
0x45	0xFF

Erase successful and verified

Flash Write

Description:

This instruction writes 32 bytes in the Flash at a given location inside a block (64Kbytes).
A verify is executed after the write is completed to ensure that reliable data are stored.
2048 (0x800) write commands are needed to set the full content of a block.
A block must be erased before writing data to it. An erase command sets all bits to 1 in a block. A write sequence sets bits to '0' only. Unused bytes should be written to 0xFF.

The Flash must not be accessed during step activities where the Flash is used to provide data.

PC

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	…	Byte 33	Byte 34	Byte 35
0x57	BLCK	ADRH	ADRL	D0	D1	D2	…	D29	D30	D31

BLCK:	Block number to write	(0-31 <-> 0x00-0x1F)
ADRH:	High order address to write	(0-255 <-> 0x00-0xFF)
ADRL:	Low order address to write	(0-255 <-> 0x00-0xFF)

D0-D31: Data to write at the location ADR:ADRL in BCK

Note:

ADRH:ADRL represents the address of each bytes of data within the block. To write the next set of data ADRH:ADRL should be incremented by 32.

IPL5X

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x57	RET	-	-	-	-	-	-	-	-	-

RET:

- 0x00 Write FAILED !!! This error should be notified to the user
- 0xFF Write OK

Example 1: Write data at address0x0020 in block 0x01

Content of the flash before write (Read at address 0x0020 in block 0x01)

PC

Byte 0	Byte 1	Byte 2	Byte 3
0x52	0x01	0x00	0x20

IPL5X

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	…	Byte 33	Byte 34	Byte 35
0x52	0x01	0x00	0x20	0xFF	0xFF	0xFF	…	0xFF	0xFF	0xFF

Write 0xAA and 0x55 at the beginning of address 0x0020 in block 0x01

PC

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	…	Byte 33	Byte 34	Byte 35
0x57	0x01	0x00	0x20	0xAA	0x55	0xFF	…	0xFF	0xFF	0xFF

IPL5X

Byte 0	Byte 1
0x57	0xFF

Write successful and verified

Content of the flash after the write (Read at address 0x0020 in block 0x01)

PC

Byte 0	Byte 1	Byte 2	Byte 3
0x52	0x01	0x00	0x20

IPL5X

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	…	Byte 33	Byte 34	Byte 35
0x52	0x01	0x00	0x20	0xBB	0x66	0xFF	…	0xFF	0xFF	0xFF

Example 2: Add data to a previous written zone at address0x0020 in block 0x01

Read at Flash at address 0x0020 in block 0x01

PC

Byte 0	Byte 1	Byte 2	Byte 3
0x52	0x01	0x00	0x20

IPL5X

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	...	Byte 34	Byte 35
0x52	0x01	0x00	0x20	0xAA	0x55	0xFF	0xFF	...	0xFF	0xFF

Add data after the last existing byte (0x55)

Write the new completed location at the beginning of address 0x0020 in block 0x01

PC

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	...	Byte 34	Byte 35
0x57	0x01	0x00	0x20	0xAA	0x55	0xBB	0x66	...	0xFF	0xFF

IPL5X

Byte 0	Byte 1
0x57	0xFF

Write successful and verified

Content of the flash after the write (Read at address 0x0020 in block 0x01)

PC

Byte 0	Byte 1	Byte 2	Byte 3
0x52	0x01	0x00	0x20

IPL5X

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	...	Byte 34	Byte 35
0x52	0x01	0x00	0x20	0xAA	0x55	0xBB	0x66	...	0xFF	0xFF

MM2001 Programming

Description:

Read/Write the PIC program, data and config word of the MM2001 using the in circuit LVP feature.
Please refer to the datasheets from Microchip for details on how to program the different PIC versions.
PIC16F87X EEPROM Memory Programming Specification :
http://ww1.microchip.com/downloads/en/DeviceDoc/39025f.pdf
PIC16F87XA Flash Memory Programming Specification :
http://ww1.microchip.com/downloads/en/DeviceDoc/39589C.pdf
WARNING: after erasing or when programming the config word, make sure that LVP=1 in the config word before exiting the LVP mode, otherwise you won't be able to enter it again !

Note: To enter the MM2001 in programming mode, J8 must be set to the PRGM position. When finished, J8 MUST be set back to the NORMAL position.

Enter LVP mode

PC

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x59	CMD	-	-	-	-	-	-	-	-	-

CMD	0x00	Enter LVP mode

IPL5X

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x59	-	-	-	-	-	-	-	-	-	-

Exit LVP mode

PC

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x59	CMD	-	-	-	-	-	-	-	-	-

CMD	0x01	Exit LVP mode

IPL5X

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x59	-	-	-	-	-	-	-	-	-	-

Read command + data

PC

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x59	CMD	RCMD	-	-	-	-	-	-	-	-

CMD	0x02	Read data
RCMD	PIC16F87X & PIC16F87XA:	
		0x04=Read Data from Program Memory
		0x05=Read Data from Data Memory

IPL5X

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x59	DATH	DATL	-	-	-	-	-	-	-	-

DATH:DATL	Returned 14 bits value

Write command

PC

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x59	CMD	WCMD	-	-	-	-	-	-	-	-

CMD	0x03	Write command
WCMD	PIC16F87X & PIC16F87XA:	
		0x06=Increment Address
		0x08=Begin Erase/Programming Cycle
		0x18=Begin Programming Only Cycle
	PIC16F87X:	
		0x01=Bulk Erase Setup1
		0x07=Bulk Erase Setup2
	PIC16F87XA:	
		0x09=Bulk Erase Program Memory
		0x0B=Bulk Erase Data Memory
		0x1F=Chip Erase
		0x17=End Programming

IPL5X

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x59	-	-	-	-	-	-	-	-	-	-

Write command + data

PC

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x59	CMD	RCMD	DATH	DATL	-	-	-	-	-	-

CMD	0x04	Write command + data
RCMD	PIC16F87X & PIC16F87XA:	
		0x00=Load Configuration
		0x02=Load Data for Program Memory
		0x03=Load Data for Data Memory
DATH:DATL		14 bits value to Load

IPL5X

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x59	-	-	-	-	-	-	-	-	-	-

Example: Read the Device ID to identify the PIC type and version

Enter LVP mode

PC

Byte 0	Byte 1
0x59	0x00

IPL5X

Byte 0
0x59

Load Configuration to move PC to 0x2000

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4
PC	0x59	0x04	0x00	0x00	0x00

	Byte 0
IPL5X	0x59

Increment Address 6 times to move PC to 0x2006

	Byte 0	Byte 1	Byte 2
PC	0x59	0x03	0x06

	Byte 0
IPL5X	0x59

	Byte 0	Byte 1	Byte 2
PC	0x59	0x03	0x06

	Byte 0
IPL5X	0x59

	Byte 0	Byte 1	Byte 2
PC	0x59	0x03	0x06

	Byte 0
IPL5X	0x59

	Byte 0	Byte 1	Byte 2
PC	0x59	0x03	0x06

	Byte 0
IPL5X	0x59

	Byte 0	Byte 1	Byte 2
PC	0x59	0x03	0x06

	Byte 0
IPL5X	0x59

	Byte 0	Byte 1	Byte 2
PC	0x59	0x03	0x06

	Byte 0
IPL5X	0x59

Read Device ID value at address 0x2006

	Byte 0	Byte 1	Byte 2
PC	0x59	0x02	0x04
IPL5X	0x59	DATH=0x09	DATL=0x26

Device ID=DATH:DATL/32=0x49 -> PIC16F874
Revision=DATL and 0x1F=0x06 -> Rev 6

Exit LVP mode

	Byte 0	Byte 1
PC	0x59	0x01

	Byte 0
IPL5X	0x59