IchigoJam BASIC reference ver 1.4

command	description	example
LED / LED	light on the LED when n equals 1, light off when n equals 0	LED 1
WAIT / wait	wait n frames (60frame = 1sec) (if num2 eqauls 0 low power consumption mode, if num1 is minus short wait mode -261 same as WAIT1)	WAIT 60
: / colon	series the commands	WAIT 60:LED 1
1 / one	memory commands as line number. Remove line without commands	10 LED1
RUN / run	execute program in memory [F5]	RUN
LIST / list	show the program in memory [F4] (line num1: show the line, if minus to the line / line num2: show to the line, if 0 to the end / ESC to stop)	LIST 10,300
GOTO / goto	change the execution line (it's OK as using variables)	GOTO 10
END / end	end this program	END
IF / if	if num does not equals 0 execute command1, else execute command2 (you can ommit THEN / ELSE)	IF BTN() END
THEN / then	if num does not equals 0 execute command1, else execute command2 (you can ommit THEN / ELSE)	IF BTN() THEN END
ELSE / else	if num does not equals 0 execute command1, else execute command2 (you can ommit THEN / ELSE)	IF BTN() END ELSE CONT
BTN / button	return 1 if you push the botton, else 0 (num:0(embeded button)/UP/DOWN/RIGHT/LEFT/SPACE/X(88), 0:if ommit, -1:all)	LED BTN()
NEW / new	delete all program in the memory	NEW
PRINT / print	write the letter or nunmber to the screen (strings must surround ["], connect pars with [;]) Abbreviation:?	PRINT "HI!";15
? / question	write the letter or nunmber to the screen (strings must surround ["], connect pars with [;]) Abbreviation:?	?"HI!";15
LOCATE / locate	set the position to write (if y equals -1 no write mode). x as X+Y*W when set only x. show the cursor if num not equals 0. Abbreviation:LC	LOCATE 3,3
LC / locate	set the position to write (if y equals -1 no write mode). x as X+Y*W when set only x. show the cursor if num not equals 0. Abbreviation:LC	LC 3,3
CLS / clear screen	clear the screen	CLS
RND / random	return the random number 0 to num - 1	PRINT RND(6)
SAVE / save	save the program (num:0-3, 100-227:optional EEPROM, if omit using number)	SAVE 1
LOAD / load	load the program (num:0-3, 100-227:optional EEPROM, if omit using number)	LOAD
FILES / files	show the file list from num1 to num2 (all if num1 equals 0, ESC to stop)	FILES
BEEP / beep	sound the BEEP, num1 is period(1-255), num2:length(1/60sec) (you can omit num1 and num2) *to connect the sounder on SOUND(EX2)-GND	BEEP
PLAY / play	play the music specified mml as MML(Music Macro Language) just PLAY to stop the music. MML: CDEFGAB=tone R=rest, + half tone higher, - half tone lower, with num specified length, with . half stretch the length, < octave up, > octave down, Tn set the tempo (you can change with TEMPO command) initial value:120, Ln default length (1,2,3,4,8,16,32) initial value:4, On set the octave from O1C to O5B initial value:3, Nn sound a tone specified 1-255 (same as BEEP command), ' end of music, $ repeat play after this mark, space is just skip *to connect the sounder on SOUND(EX2)-GND	PLAY "$CDE2CDE2"
TEMPO / tempo	change the tempo of playing music	TEMPO 1200
+ / plus	return x plus y	PRINT 1+1
- / minus	return x minus y	PRINT 2-1
* / asterisk	return x times y	PRINT 7*8
/ / slash	return integer of x divide y	PRINT 9/3
% / percent	return reminder of x divide y	PRINT 10%3
() / blacket	return calculate the number in priority	PRINT 1+(1*2)
LET / let	set the number to 1 letter of alphabet as named memory(variable) (series put to the array) Abbreviation:var=num	LET A,1
= / equal	1. set the number to 1 letter of alphabet as named memory(variable) 2. same as == when using as number	A=1
INPUT / input	set the number to var from keyboard input (you can omit strings and comma)	INPUT "ANS?",A
TICK / tick	return the time count from CLT(count up in 1/60sec) *if nu=1, 1/(60*261)	PRINT TICK()
CLT / clear tick	clear the time count	CLT
INKEY / inkey	return from keyboard or UART (0:no input, #100:0 input from UART)	PRINT INKEY()
LEFT / left	28. using with INKEY and SCROLL	IF INKEY()=LEFT LED1
RIGHT / right	29. using with INKEY and SCROLL	IF INKEY()=RIGHT LED1
UP / up	30. using with INKEY and SCROLL	IF INKEY()=UP LED1
DOWN / down	31. using with INKEY and SCROLL	IF INKEY()=DOWN LED1
SPACE / space	32. using with INKEY and SCROLL	IF INKEY()=SPACE LED1
CHR / character	In PRINT, return the letter string specified the num (you can set series with comma)	PRINT CHR$(65)
ASC / ascii	return the letter code from string	PRINT ASC("A")
SCROLL / scroll	scroll the screen (0/UP:up、1/RIGHT:right、2/DOWN:down、3/LEFT:left)	SCROLL 2
SCR / screen	return the letter code located x, y on the screen (if omit x and y, using current position) Alias:VPEEK	PRINT SCR(0,0)
VPEEK / V peek	return the letter code located x, y on the screen (if omit x and y, using current position) Alias:VPEEK	PRINT VPEEK(0,0)
== / equal equal	return 1 if x equals y else 0	IF A==B LED 1
!= / not equal to	return 1 if x does not equal y else 0	IF A!=B LED 1
<> / less than and greater than	return 1 if x does not equal y else 0	IF A<>B LED 1
<= / less than or equal	return 1 if x <= y, else 0	IF A<=B LED 1
< / less than	return 1 if x < y else 0	IF A<B LED 1
>= / greater than or equal	return 1 if x >= y else 0	IF A>=B LED 1
> / greater than	return 1 if x < y else 0	IF A>B LED 1
AND / and	return 1 if x and y else 0	IF A=1 AND B=1 LED 1
&& / and	return 1 if x and y else 0	IF A=1 && B=1 LED 1
OR / or	return 1 if x or y else 0	IF A=1 OR B=1 LED 1
ll / or	return 1 if x or y else 0	IF A=1 ll B=1 LED 1
NOT / not	return 1 if x equals 0 else 0	IF NOT A=1 LED 1
! / not	return 1 if x equals 0 else 0	IF !(A=1) LED 1
REM / remark	not execute after this command (comment)	REM START
' / single quote	not execute after this command (comment)	START
FOR / for	set num1 to var, execute the loop to the NEXT until var reach num2 by step num3 (you can omit STEP, nest limit:6)	FOR I=0 TO 10:?I:NEXT
TO / to	set num1 to var, execute the loop to the NEXT until var reach num2 by step num3 (you can omit STEP, nest limit:6)	FOR I=0 TO 10:?I:NEXT
STEP / step	set num1 to var, execute the loop to the NEXT until var reach num2 by step num3 (you can omit STEP, nest limit:6)	FOR I=0 TO 10:?I:NEXT
NEXT / next	set num1 to var, execute the loop to the NEXT until var reach num2 by step num3 (you can omit STEP, nest limit:6)	FOR I=0 TO 10:?I:NEXT
POS / position	return a positon of cursor (num 0:X+Y*W, 1:X, 2:Y, ommit:0)	?POS(0),POS(1)
DRAW / draw	draw a line or a point (n 0:clear, 1:set, 2:reverse, ommit:1)	DRAW 1,5,10,15
POINT / point	return a point on the screen or not	?POINT(1,5)
OUT / out	output num2 to the output pin specified num1 (num1:OUT1-11, you can set all states when you omit num2, if	OUT 1,1

	num2 equals -1 the output pin switch into the input pin, if num2 equals -2 the output pin switch into the input with pull-up)	
IN / in	return 1 if when input terminal pin is high else 0 (num:0-11 (IN0/1/4/9 pull up, IN5-8,10-11:if switched, IN0,9:button), you can get all states when you omit num)	LET A,IN(1)
ANA / analog	return the value 0-1023 specified voltage of input terminal (2:IN2, 5-8:IN5-8(OUT1-4), 0,9:BTN, 0:omitted)	?ANA()
PWM / PWM	output num2(0.01msec) length pulse in num3(if omit 2000) period to the output pin specified num1 (num1:OUT2-5, OUT2-4 same period)	PWM 2,100
CLV / clear variables	clear (set to zero) variables and array variables	CLV
CLEAR / clear	clear (set to zero) variables and array variables	CLEAR
CLK / clear keys	clear key buffer and key status	CLK
CLO / clear output	initialize the input and output pins	CLO
ABS / absolute	return the absolute value	?ABS(-2)
[] / array	array variables (from [0] to [101] 102 series variables) you can set in series using LET[0],1,2,3	[3]=1
GOSUB / gosub	move to linenum and execute after this command when RETURN/RTN (nest limit:30)	GOSUB 100
GSB / gosub	move to linenum and execute after this command when RETURN/RTN (nest limit:30)	GSB 100
RETURN / return	back to linunum last GOSUB/GSB called	RETURN
RTN / return	back to linunum last GOSUB/GSB called	RTN
DEC / deci	In PRINT, return strings from num1 with beam specified num2 (you can omit num2)	?DEC$(99,3)
# / hash	return the number specified in hexadecimal *if you omit THEN after this, use : to separate	#FF
HEX / hex	In PRINT, return hexadecimal strings from num1 with beam specified num2 (you can omit num2)	?HEX$(255,2)
` / back quote	return the number specified in binary number	`1010
BIN / binary	In PRINT, return binary number strings from num1 with beam spcified num2 (you can omit num2)	?BIN$(255,8)
& / ampersand	return x logical and y (bit calculation)	?3&1
I / pipe	return x logical or y (bit calculation)	?3I1
^ / hat	return x logical exclusive or y (bit calculation)	?A^1
>> / shift right	return x shift down y bits (bit calculation)	?A>>1
<< / shift left	return x shift up y bits (bit calculation)	?A<<1
~ / tlda	return bit inverted x (bit calculation)	?~A
COS / cosine	return cosine number times 256 by degrees *for ver1.4 or later	?COS(90)
SIN / sine	return sine number times 256 by degrees *for ver1.4 or later	?SIN(90)
STOP / stop	stop the program	STOP
CONT / continue	continue the same line or stop line	CONT
SOUND / sound	return 1 if sound playing else 0	?SOUND()
FREE / free	return free memory of program (up to 1024 bytes)	?FREE()
VER / version	n=0 or default: return the version number of IchigoJam BASIC (n=0 or default) 1: return the language number of IchigoJam BASIC(1:Japanese 2:Mongol, 3:Vietnam)	?VER()
RENUM / renumber	renumber the line number of program from num1 step num2 (num1:10, num2:10 if omit, you may have to change manually line number specified in GOTO/GOSUB)	RENUM
LRUN / load and run	LOAD num and RUN	LRUN 1
FILE / file	return the number of last using FILE	?FILE()
LINE / line	return the line number of last execution	?LINE()
SRND / S random	initialize the seed of random	SRND 0
HELP / help	display the memory map	HELP
PEEK / peek	read 1 byte number from the memory in address specified num2	?PEEK(#700)
POKE / poke	write 1 byte num1 specified to the memory in address specified num2	POKE #700,#FF
COPY / copy	memory copy from num1 to num2 length specified num3 (if num3 is minus, copy direction is inverted)	COPY #900,0,256
CLP / clear pattern	initialize the character pattern memory (#700-#7FF)	CLP
" / double quote	return the address of strings on the memory	A="ABC"
STR / stings	In PRINT, return strings from address specified num1 (num2:length you can omit)	PRINT STR$(A)
LEN / length	return the length of strings	PRINT LEN("ABC")
@ / at mark	Put in front of the line, you can as destination line number (type GOTO @LOOP)	@LOOP
VIDEO / video	switch the video signal enabled or disabled (num0:0 disabled / 1 enable[F8] / 2 invert black and white / greater than bigmode, num2:when num0 is 0 clock down mode clock=1/num2)	VIDEO 0
RESET / reset	reboot the IchigoJam	RESET
SLEEP / sleep	sleep the IchigoJam (after push the button, execute and run file 0)	SLEEP
UART / UART	set UART mode (num1 0:off 1:with PRINT 2:with PRINT/LC/CLS/SCROLL 3:with PRINT and enter code is \r\n, echo back input with +4, disable screen put with +8, initial value:2) (num2 0:UART recv off, 1:on, +2:ignore ESC, +4:CR change(13→10), initial value:1)	UART 0
BPS / BPS	num1: set UART speed (0:115,200bps -1:57600bps -2:38400bps or 9600 and so on ... if -100 or less, -100 times speed. For example, if it is -2604, it is 260400 initial value:0), num2: set I2C speed(unit kHz, 0:400kHz default value:0)	BPS 9600
OK / OK	show OK and error messages (1: message on if omit, 2: message off)	OK 2
I2CR / I two C read	read from I2C device num1:I2C address, num2:address of command, num2:length of command, num4:address of return data, num5:length of return data (if command is 1byte you can omit num2, if no commands num2/num3 you can omit)	R=I2CR(#50,#114A,2,#114C,2)
I2CW / I two C write	write to I2C device num1:I2C address, num2:address of command, num2:length of command, num4:address of return data, num5:length of return data (you can omit num4/num5, if command is 1byte you can omit num2)	R=I2CW(#50,#114A,2,#114C,2)
IOT.IN / IoT in	get a received number from the sakura.io module	R=IoT.IN()
IOT.OUT / IoT out	send a number to send via the sakura.io module as the 0 channel	IoT.OUT 100
WS.LED / WS LED	The WS2812B connected to LED lights up by num1 with values set in the order of green, red, and blue from the beginning of the array. If the num2 is specified, it repeats that number. *for ver 1.4	WS.LED 3
SWITCH / switch	switch the video output, TV or LCD (0:TV 1:LCD)	SWITCH
USR / user	Call the machine language (Arm Cortex-M0) of the address specified by the num1 with the num2 as a parameter (0 for num2 omitted)	A=USR(#700,0)