Instructions available only in DIRECT mode

Instructions that can be used only if DIRECT mode has been turned on by pressing the DIRECT button

These instructions cannot be included in programs in EDIT mode.

CLEAR	Initializes the BASIC internal memory DIRECT mode only	
Format	CLEAR	
Examples	CLEAR	

NEW	Erases programs DIRECT mode only	
Format	NEW [Program SLOT]	
Arguments	Program SLOT	0-3: Erases the specified SLOT only If unspecified, all SLOTs are erased
Examples	NEW NEW 3	

LIST	Switches to EDIT mode and starts editing - DIRECT mode only - Using LIST with no argument is equal to pressing the EDIT button	
Format	LIST [Line number/ERR]	
Arguments	Line number	- If unspecified, the displayed list will start with the default line - A program SLOT can be specified, e.g., by entering 2:120
	ERR	Specifies the line where the last error occurred
Examples	LIST ERR LIST 1:	

RUN	Program execution DIRECT mode only	
Format	RUN [Program SLOT]	
Arguments	Program SLOT	Specified program SLOT to be executed (0 to 3. If omitted, 0)
Examples	RUN RUN 1	

CONT	Resumes a suspended program - DIRECT mode only - Execution is resumed from the location it was suspended at using the START button, the STOP instruction, or due to an error - If the program has been stopped and then edited, it cannot be resumed - If the program was suspended while waiting for user input, it cannot be resumed - The program may not be able to be resumed depending on the type of error that occurred	
Format	CONT	
Examples	CONT	

PROJECT (1)	Switches the default project DIRECT mode only	
Format	PROJECT "Project name"	
Arguments	Project name	Name string of project to change - New projects can be created from the TOP MENU - Project name "" specifies the default project
Supplement	Three conditions for the current project	1) Current project at start-up time (specified using Change Active Project option under the Manage Projects/Files menu) 2) Current project at non-execution time (set with the PROJECT instruction) 3) Current project at execution time (set during execution, e.g., of EXEC) When the current project is set for 1) - 3), the respective subordinate project settings will also be updated accordingly. For example, if the current project is changed through the Change Active Project setting, the project at non-execution time and the project at execution time will also be changed. Furthermore, when execution is started (by using RUN, executing a tool, or executing a program from the file viewer), the current project at non-execution time will be set as the initial value of the current project at execution time.
Examples	PROJECT ""	

PROJECT (2)	Obtains the default project Can be also used from within programs	
Format	PROJECT OUT PJ$	
Arguments	None	
Return Values	PJ$	Current project name
Examples	PROJECT OUT PJ$	

BACKTRACE	Displays the history of previous callers - DIRECT mode only - When execution has been stopped, for example with the STOP instruction, the caller history up to the point just before it was stopped will be displayed - A list of slot numbers and line numbers will be displayed	
Format	BACKTRACE	
Examples	BACKTRACE	

Basic instructions (variables and arrays)

Instructions for handling definitions of variables or arrays, array operations, etc.

=	Assigns a value or expression to a variable - A simplified notation of the LET instruction used in conventional BASIC - In this software, the LET instruction itself is omitted; only '=' should be used for assignment	
Format	=	

Examples	A=10
	A$="HELLO"	

DIM (1)	Declares arrays to use - In this product, arrays must always be declared - The subscript should begin with 0 - The number of elements must be enclosed in []. () is not allowed - Either DIM or VAR can be used	
Format	DIM Array variable name[Number of elements] ,…	
Arguments	Array variable name[Number of elements]	- Alphanumeric characters and underscores (_) are allowed - Leading numerals are not allowed - String variables are also allowed for the array variable
	Number of elements	- Specify the number of array elements to provide, enclosed in [] - Up to four dimensions can be specified, with commas (,) to separate them
Examples	DIM ATR[4] DIM DX[5], DY[5], DZ[5] DIM POS[10,5]	

DIM (2)	Declares variables to use - When OPTION STRICT is specified, each variable that will be used must be declared - Usage where DIM is used for variable definition	
Format	DIM Variable name ,…	
Arguments	Variable name	- Alphanumeric characters and underscores (_) are allowed - Leading numerals are not allowed - String variables can also be declared
Examples	DIM A, ATRB, B$	

VAR (1)	Declares variables to use When OPTION STRICT is specified, each variable that will be used must be declared	
Format	VAR Variable name ,…	
Arguments	Variable name	- Alphanumeric characters and underscores (_) are allowed - Leading numerals are not allowed - String variables can also be declared
Examples	VAR A, ATRB, B$	

VAR (2)	Declares arrays to use - In this product, arrays must always be declared - The subscript should begin with 0 - The number of elements must be enclosed in []. () is not allowed - Either DIM or VAR can be used	
Format	VAR Array variable name[Number of elements] ,…	
Arguments	Array variable name[Number of elements]	- Alphanumeric characters and underscores (_) are allowed - Leading numerals are not allowed - String variables are also allowed for the array variable
	Number of elements	- Specify the number of array elements to provide, enclosed in [] - Up to four dimensions can be specified, with commas (,) to separate them
Examples	VAR ATR[4] VAR DX[5], DY[5], DZ[5] VAR POS[10, 5]	

SWAP	Swaps the values of two variables Replacing a character string with a numerical value, or vice versa, is not allowed	
Format	SWAP Variable 1, Variable 2	
Arguments	Variable 1	Variable to be replaced
	Variable 2	Variable to be replaced with
Examples	SWAP A,B	

INC	Increments the value of a variable by +1 If the Expression argument is specified, the value of the expression will be added	
Format	INC Variable [, Expression]	
Arguments	Variable	Name of variable to increment value of
	Expression	Value to add (If omitted, 1)
Examples	INC X INC X,3	

DEC	Decrements the value of a variable by -1 If the Expression argument is specified, the value of the expression will be subtracted	
Format	DEC Variable [, Expression]	
Arguments	Variable	Name of variable to decrement value of
	Expression	Value to subtract (If omitted, 1)
Examples	DEC X DEC X,3	

COPY (1)	Copies one array to another array - For one-dimensional arrays only, if the number of elements in the copy destination is insufficient, the required element(s) will be added automatically - Both the copy source and destination ignore dimensions	
Format	COPY Copy destination array [,Copy destination offset],Copy source array [[,Copy source offset] , Number of copy elements]	
Arguments	Copy destination array	Copy destination array (to be overwritten with the content of the copy source array)
	Copy destination offset	First element to be overwritten (If this is omitted, overwriting will start with the beginning of the copy destination)
	Copy source array	Copy source array
	Copy source offset	First element to be overwritten (If this is omitted, copying will start with the beginning of the copy source)
	Number of copy elements	Number of elements to be overwritten (If this is omitted, up to the end of the copy source will be copied)
Examples	DIM SRC[10],DST[10] COPY DST,SRC	

COPY (2)	Reads a DATA sequence into an array - The data elements enumerated in the DATA instruction will be read into the array - For one-dimensional arrays only, if the number of elements in the copy destination is insufficient, the required element(s) will be added automatically	
Format	COPY Copy destination array [,Copy destination offset], "@Label string" [,Number of copy data items]	
Arguments	Copy destination array	Copy destination array (to be overwritten with the content of the DATA sequence)
	Copy destination offset	First element to be overwritten (If this is omitted, overwriting will start with the beginning of the array)
	"@Label string"	Specify the @Label name string set to the DATA instruction to be read
	Number of copy data items	- Number of data items to be read (If this is omitted, data items will be read according to the number of elements in the copy destination array) - If the number of data items is smaller than the number of arrays in the copy destination, an error will occur.
Examples	DIM DST[5] COPY DST,"@SRC" @SRC DATA 5,1,1,2,4	

SORT	Sorts arrays in ascending order	
Format	SORT [Start position, Number of elements,] Array 1 [,Array 2 ,…]	
Arguments	Start position	Position in Array 1 (0-) from which to start sorting
	Number of elements	Number of elements in Array 1 (1-) to sort
	Array 1	Array with numerical values to sort
	Array 2	- Array to sort according to the result of sorting of Array 1 - Array 1 to Array 8 can be enumerated
Examples	DIM WORK[10] SORT 0, 10, WORK	

RSORT	Sorts arrays in descending order	
Format	RSORT [Start position, Number of elements,] Array 1 [,Array 2 ,…]	
Arguments	Start position	Position in Array 1 (0-) from which to start sorting
	Number of elements	Number of elements in Array 1 (1-) to sort
	Array 1	Array with numerical values to sort
	Array 2	- Array to sort according to the result of sorting of Array 1 - Array 1 to Array 8 can be enumerated
Examples	DIM WORK[10] RSORT 0, 10, WORK	

PUSH	Adds an element to the end of an array (The number of elements will increase by 1)	
Format	PUSH Array, Expression	
Arguments	Array	Array to which the element will be added
	Expression	Value of the element to add
Examples	DIM WORK[10] PUSH WORK, 123	

POP	Removes an element from the end of an array (The number of elements will decrease by 1)	
Format	Variable=POP(Array)	
Arguments	Array	Array from which the element will be removed
Return Values	Value of the element that was removed	
Examples	DIM WORK[10] PUSH WORK, 123 A=POP(WORK)	

UNSHIFT	Adds an element to the start of an array (The number of elements will increase by 1)	
Format	UNSHIFT Array, Expression	
Arguments	Array	Array to which the element will be added
	Expression	Value of the element to add
Examples	DIM WORK[10] UNSHIFT WORK, 123	

SHIFT	Removes an element from the start of an array (The number of elements will decrease by 1)	
Format	Variable=SHIFT(Array)	
Arguments	Array	Array from which the element will be removed
Examples	DIM WORK[10] UNSHIFT WORK, 123 A=SHIFT(WORK)	

FILL	Sets all the elements in an array to the specified value - Partial changes can also be made by specifying an offset and number of elements - You can specify any type of array, including integer, real number, or string	
Format	FILL Array, Value [,Offset [,Number of elements]]	
Arguments	Array	The array that you want to overwrite with a value
	Value	The desired number or string
	Offset	The position to begin writing the value from
	Number of elements	The number of elements to write the value into
Examples	DIM WORK[10] FILL WORK,0	

Basic instructions (control and branching)

Control instructions for comparison, branching, repeats, etc.

@	Name to indicate a program or data position - It is not possible to specify the line number directly with GOTO or other instructions - Branch destinations and data positions must be all specified using labels	
Format	@Label name	
Arguments	@Label name	Alphanumerical or underscore (_) characters, prefixed with @
Examples	@MAINLOOP	

GOTO (1)	Forces branching	
Format	GOTO @Label	
Arguments	@Label	- Jump target @Label name - Label string, which is the Label name enclosed in "" (String variables are also allowed) - A program SLOT can be specified in the following format: "1:@Label name" - The target SLOT should be enabled beforehand with the USE instruction
Examples	GOTO @MAIN JP$="@MAIN":GOTO JP$	

GOSUB (1)	Calls a sub-routine	
Format	GOSUB @Label	
Arguments	@Label	- @Label name of the sub-routine to call - Label string, which is the Label name enclosed in "" (String variables are also allowed) - A program SLOT can be specified in the following format: "1:@Label name" - The target SLOT should be enabled beforehand with the USE instruction
Examples	GOSUB @SUB	

RETURN (1)	Returns from a sub-routine to the caller
Format	RETURN
Examples	RETURN

RETURN (2)	Returns a value from a sub-routine while returning to the caller Used to return values in a DEF instruction defined as function type
Format	RETURN

Examples	DEF CALC(A,B)
	 RETURN A*B
	END
	PRINT CALC(2,3)

OUT	Instruction used when multiple outputs are required - Used to declare a DEF instruction that returns multiple values - Also used in built-in instructions that return multiple values
Format	OUT
Examples	DEF SUB A OUT D,M D=A DIV 10 M=A MOD 10 END SUB 34 OUT DV,ML PRINT DV,ML

ON (1)	Branches to a label line according to the control variable value - The branch number begins with 0, unlike in conventional BASIC	
Format	ON Control variable GOTO @Label 0, @Label 1…	
Arguments	@Label 0	Jump target to use when the control variable is 0
	@Label 1	Jump target to use when the control variable is 1 : - Prepare the necessary number of branch destinations - Label strings cannot be used in the ON to GOTO labels
Examples	ON IDX GOTO @JMP_A,@JMP_B PRINT "OVER":END @JMP_A PRINT "IDX=0":END @JMP_B PRINT "IDX=1":END	

GOTO (2)	Branches to a label line according to the control variable value - The branch number begins with 0, unlike in conventional BASIC	
Format	ON Control variable GOTO @Label 0, @Label 1…	
Arguments	@Label 0	Jump target to use when the control variable is 0
	@Label 1	Jump target to use when the control variable is 1 : - Prepare the necessary number of branch destinations - Label strings cannot be used in the ON to GOTO labels
Examples	ON IDX GOTO @JMP_A,@JMP_B PRINT "OVER":END @JMP_A PRINT "IDX=0":END @JMP_B PRINT "IDX=1":END	

ON (2)	Calls a sub-routine according to a control variable value - The branch number begins with 0, unlike in conventional BASIC	
Format	ON Control variable GOSUB @Label 0, @Label 1…	
Arguments	@Label 0	Sub-routine when the control variable is 0
	@Label 1	Sub-routine when the control variable is 1 : - Prepare the necessary number of branch destinations - Label strings cannot be used in the ON to GOSUB labels
Examples	ON IDX GOSUB @SUB_A,@SUB_B PRINT "EXIT":END @SUB_A PRINT "IDX=0":RETURN @SUB_B PRINT "IDX=1":RETURN	

GOSUB (2)	Calls a sub-routine according to the control variable value - The branch number begins with 0, unlike in conventional BASIC	
Format	ON Control variable GOSUB @Label 0, @Label 1…	
Arguments	@Label 0	Sub-routine when the control variable is 0
	@Label 1	Sub-routine when the control variable is 1 : - Prepare the necessary number of branch destinations - Label strings cannot be used in the ON to GOSUB labels
Examples	ON IDX GOSUB @SUB_A,@SUB_B PRINT "EXIT":END @SUB_A PRINT "IDX=0":RETURN @SUB_B PRINT "IDX=1":RETURN	

IF (1)	Executes Process 1 if the condition is satisfied, or Process 2 if the condition is not satisfied - GOTO can be omitted immediately after THEN or ELSE - ENDIF should be used when the process spans multiple lines	
Format	IF Conditional expression THEN Process to execute when the condition is satisfied [ELSE Process to execute when the condition is not satisfied] [ENDIF]	
Conditional Expressions	Comparison Operators	== Equal to != Not equal to > Greater than < Smaller than >= Equal to or greater than <= Equal to or smaller than
	Logical Operators (for comparing multiple conditions)	(Condition 1 AND Condition 2) Both of the conditions should be satisfied (Condition 1 && Condition 2) Both of the conditions should be satisfied (Condition 1 OR Condition 2) Either one of the conditions should be satisfied (Condition 1 || Condition 2) Either one of the conditions should be satisfied * The key for the "||" characters is located to the upper left of the "?" key on the keyboard.

Examples	IF A==1 THEN PRINT "OK"
	IF A>1 THEN @JMP1 ELSE PRINT DATE$
	IF A==1 THEN
	 PRINT "Congratulations":BEEP 72
	ELSE
	 PRINT "Bad luck"
	ENDIF
	@JMP1
	END	

THEN	Control destination if the IF condition is satisfied See Comment for IF for details regarding conditional evaluation
Format	IF Conditional expression THEN Process to execute when the condition is satisfied [ELSE Process to execute when the condition is not satisfied] [ENDIF]

Examples	IF A==1 THEN PRINT "OK"
	IF A<1 THEN @JMP1 'GOTO omitted
	IF A==1 THEN
	 PRINT "Congratulations":BEEP 72
	ELSE
	 PRINT "Bad luck"
	ENDIF
	@JMP1
	END

ELSE	Control destination if the IF condition is not satisfied See Comment for IF for details regarding conditional evaluation
Format	IF Conditional expression THEN Process to execute when the condition is satisfied ELSE Process to execute when the condition is not satisfied [ENDIF]

Examples	IF A==1 THEN PRINT "OK"
	IF A<1 THEN @JMP1 ELSE PRINT DATE$
	IF A==1 THEN
	 PRINT "Congratulations":BEEP 72
	ELSE
	 PRINT "Bad luck"
	ENDIF
	@JMP1
	END

ELSEIF	Additional conditional evaluation if the IF condition is not satisfied - Used to evaluate another condition if the IF condition is not satisfied - See Comment for IF for details regarding conditional evaluation
Format	IF Conditional expression THEN Process to execute when the condition is satisfied ELSEIF Conditional expression THEN Process to execute when the condition is satisfied ENDIF

Examples	IF A==1 THEN
	 PRINT "Congratulations":BEEP 0
	ELSEIF A==2 THEN
	 PRINT "Bad luck"
	ELSE IF A==3 THEN
	 PRINT "So-so"
	ENDIF '--- Required when using ELSE IF
	ENDIF

ENDIF	Ends if processing spans multiple lines after control switching with IF See Comment for IF for details regarding conditional evaluation
Format	IF Conditional expression THEN Process to execute when the condition is satisfied ELSE Process to execute when the condition is not satisfied [ENDIF]

Examples	IF A==0 THEN
	 PRINT "A=0"
	ENDIF

IF (2)	Branches to @Label if the condition is satisfied See Comment for IF for details regarding conditional evaluation
Format	IF Conditional expression GOTO @Label [ELSE Process to execute when the condition is not satisfied]
Notes when using a string for the label	- Label strings can also be used for the label - Label strings are not allowed if GOTO immediately after ELSE is omitted × IF A==0 GOTO "@LABEL1" ELSE "@LABEL2" ○ IF A==0 GOTO "@LABEL1" ELSE @LABEL2 ○ IF A==0 GOTO "@LABEL1" ELSE GOTO "@LABEL2"

Examples	IF A==1 GOTO @MAIN
	IF X>0 GOTO @JMP1 ELSE PRINT A$
	IF Y==5 GOTO @JMP1 ELSE @JMP2
	@JMP1
	PRINT "@JMP1"
	@JMP2
	PRINT "@JMP2"
	END

GOTO (3)	Branches to @Label if the condition is satisfied - See Comment for IF for details regarding conditional evaluation
Format	IF Conditional expression GOTO @Label [ELSE Process to execute when the condition is not satisfied]
Notes when using a string for the label	- Label strings can also be used for the label - Label strings are not allowed if GOTO immediately after ELSE is omitted × IF A==0 GOTO "@LABEL1" ELSE "@LABEL2" ○ IF A==0 GOTO "@LABEL1" ELSE @LABEL2 ○ IF A==0 GOTO "@LABEL1" ELSE GOTO "@LABEL2"

Examples	IF A==1 GOTO @MAIN
	IF X>0 GOTO @JMP1 ELSE PRINT A$
	IF Y==5 GOTO @JMP1 ELSE @JMP2
	@JMP1
	PRINT "@JMP1"
	@JMP2
	PRINT "@JMP2"
	END

FOR	Repeats the process for the specified number of times - The NEXT instruction should be placed at the end of the process - If the condition is not satisfied, the process may not be executed at all	
Format	FOR Loop variable=Initial value TO End value [STEP Increment]	
Arguments	Loop variable	Variable for loop count (On each iteration of the loop, the increment is added to the count)
	Initial value	Value or expression for the loop variable at the start of the loop
	TO End value	Value or expression for the loop variable at the end of the loop
	STEP Increment	- Increment added to the loop variable at the end of the loop (If omitted, 1) - If the increment is specified as a fractional value, the intended loop count may not be achieved due to operational errors.
Examples	FOR I=0 TO 9 STEP 2 PRINT I;","; NEXT	

TO	Specifies the loop count end value - See Comment for the FOR instruction for details regarding FOR to NEXT
Format	TO End value
Examples	FOR I=0 TO 9 STEP 2 PRINT I;","; NEXT

STEP	Specifies the increment value for a FOR loop count - See Comment for the FOR instruction for details regarding FOR to NEXT
Format	STEP Increment
Examples	FOR I=0 TO 9 STEP 2 PRINT I;","; NEXT

NEXT	Instruction that indicates the end of a FOR loop - See Comment for the FOR instruction for details regarding FOR to NEXT - Using NEXT with IF in a FOR loop is not recommended - Use CONTINUE to exit the loop before the end	
Format	NEXT [Control variable]	
Arguments	Control variable	- Even if a control variable is specified, it will be ignored and the instruction will work in the same way as NEXT on its own - Specifications such as NEXT J,I are not allowed
Examples	FOR I=0 TO 9 STEP 2 PRINT I;","; NEXT	

WHILE	Repeats the process up to WEND while the condition is satisfied - Exits the loop if the condition is not satisfied or when the BREAK instruction is executed	
Format	WHILE Conditional expression	
Conditional Expressions	The same conditional expressions as in IF statements can be specified	
	Comparison Operators	== Equal to != Not equal to > Greater than < Smaller than >= Equal to or greater than <= Equal to or smaller than
	Logical operators (for comparison with multiple conditions)	(Condition 1 AND Condition 2) Both of the two conditions must be satisfied (Condition 1 && Condition 2) Both of the two conditions must be satisfied (Condition 1 OR Condition 2) Either of the two conditions must be satisfied (Condition 1 || Condition 2) Either of the two conditions must be satisfied * The key "||" can be found to the upper left of ? on your keyboard.
Examples	A=0:B=4 WHILE A<B A=A+1 WEND	

WEND	Instruction that indicates the end of a WHILE loop
Format	WEND
Examples	A=0:B=4 WHILE A<B A=A+1 WEND

REPEAT	Instruction for starting a REPEAT loop - The UNTIL instruction and a conditional expression should be placed at the end of the loop - Unlike the WHILE instruction, this executes the process before determining the condition - Exits the loop when the condition is satisfied or when the BREAK instruction is executed
Format	REPEAT
Examples	A=0:B=4 REPEAT A=A+1 UNTIL A>B

UNTIL	Repeats the process from REPEAT until the conditional expression is satisfied - The REPEAT instruction should be placed at the beginning of the loop - Unlike the WHILE instruction, this executes the process before determining the condition - Exits the loop if the condition is satisfied or when the BREAK instruction is executed	
Format	UNTIL conditional expression	
Conditional Expressions	The same conditional expressions as in IF statements can be specified	
	Comparison Operators	== Equal to != Not equal to > Greater than < Smaller than >= Equal to or greater than <= Equal to or smaller than
	Logical operators (for comparison with multiple conditions)	(Condition 1 AND Condition 2) Both of the two conditions must be satisfied (Condition 1 && Condition 2) Both of the two conditions must be satisfied (Condition 1 OR Condition 2) Either of the two conditions must be satisfied (Condition 1 || Condition 2) Either of the two conditions must be satisfied * The key for "||" can be found on the upper left of ? on your keyboard.
Examples	A=0:B=4 REPEAT A=A+1 UNTIL A<B	

CONTINUE	Forces a loop to proceed - Used in FOR ... NEXT, WHILE ... WEND, REPEAT ... UNTIL
Format	CONTINUE

Examples	FOR I=0 TO 9
	 IF I==1 THEN CONTINUE
	 IF I==7 THEN BREAK
	 PRINT I;",";
	NEXT

BREAK	Forces a loop to end - Used in FOR ... NEXT, WHILE ... WEND, REPEAT ... UNTIL
Format	BREAK

Examples	FOR I=0 TO 9
	 IF I==1 THEN CONTINUE
	 IF I==7 THEN BREAK
	 PRINT I;",";
	NEXT

END (1)	Exits the program
Format	END
Examples	END

END (2)	Exits a DEF definition for a user function or user instruction.
Format	END
Examples	DEF FUNC PRINT "FUNC" END

STOP	Suspends a running program - The program SLOT:line number of the suspended program will be displayed - The program can be resumed with the CONT instruction (However, resuming is not available in certain situations)
Format	STOP
Examples	STOP

Basic instructions (advanced control)

Instructions for user-defined functions, control of add-ons, etc.

DEF (1)	About DEF user-defined instructions 1) USER (No arguments; no return values) 2) A=USER(X) (With argument; single return value) 3) USER(X) OUT A,B (With argument; multiple return values) Using DEF allows you to define unique instructions as shown above	
DEF	Common Supplement for DEF	- The definition range should be from DEF to END - Variables and labels defined in the DEF to END range are handled as local - GOTO outside the DEF to END range is impossible - GOSUB or ON GOSUB in the DEF to END range cannot be used They can be used if a SLOT is specified, as in GOSUB "0:@SUB" - They can be used from a different SLOT by adding the COMMON instruction
	Specifications for DEF arguments	- For arguments received with DEF, types will not be checked strictly - Variable names to be received can be specified as necessary by separating them with commas (,) - For string variables, it is also possible to attach $ to the end of a variable name
	Specifications for DEF return values	- For DEF return values, types will not be checked strictly - The type will be determined according to the value written at the beginning of the output variables - When an integer is assigned to a numerical variable, it is handled as integer type - To handle a value as real type, it should be written as in A=100.0 - If the type of a return value from DEF is different from the type expected by the recipient, an error will occur

DEF (2)	Defines a user instruction with no return values and no arguments
Format	DEF definition name
Arguments	None
Return Values	None
Examples	'--- Text display DEF FUNC PRINT "SAMPLE" END '--- Call FUNC

DEF (3)	Defines a user function with a single return value
Format	DEF Function name([Argument [,Argument…])
Arguments	Specify variable names as necessary if there is an argument or arguments to be passed to the function
Return Values	Value to return as a result should be specified after the RETURN instruction * Notation such as RETURN ANS
Examples	'---Addition DEF ADD(X,Y) RETURN X+Y END '--- Factorial calculation using recursion DEF FACTORIAL(N) IF N==1 THEN RETURN N RETURN N*FACTORIAL(N-1) END '--- Character string inversion DEF REVERSE$(T$) VAR A$="" 'Local character string VAR L=LEN(T$) 'Local WHILE L>0 A$=A$+MID$(T$,L-1,1) DEC L WEND RETURN A$ END '--- Call PRINT ADD(10,5) PRINT FACTORIAL(4) PRINT REVERSE$("BASIC")

DEF (4)	Defines a user instruction with multiple return values
Format	DEF Instruction name [Argument [,Argument…]] [OUT V1 [,V2…]]
Arguments	Specify variable names as necessary if there is an argument or arguments to be passed to the function
Return Values	Variable names to be returned as a result should be specified as necessary after OUT
Examples	'--- Addition and multiplication DEF CALCPM A,B OUT OP,OM OP=A+B OM=A*B END '--- Call CALCPM 5,10 OUT P,M PRINT P,M

COMMON	1) COMMON DEF USER 2) COMMON DEF USER(X) 3) COMMON DEF USER(X) OUT A,B COMMON can be used when a unique instruction is used from a different SLOT USE is required if a program is used between different SLOTs
Examples	COMMON DEF FOO(X, Y, Z)

CALL (1)	Calls the user-defined instruction with the specified name	
Format	CALL "Instruction name" [,Argument…] [OUT Variable 1 [,Variable 2…]]	
Arguments	Instruction name	- User-defined instruction name string to call - Being a string, it should either be enclosed in "" or specified using a string variable.
	Arguments-	Any arguments required for the specified instruction
Return Values	Variable names to return as a result should be specified as necessary after OUT	
Examples	CALL "USERCD",X,Y OUT A,B ' DEF USERCD X,Y OUT A,B A=X+Y:B=X*Y END	

CALL (2)	Calls the user-defined function with the specified name	
Format	Variable=CALL("Function name" [,Argument…])	
Arguments	Function name	- User-defined function name string to call
		- Being a string, it should either be enclosed in "" or specified using a string variable.
	Arguments	Any arguments required for the specified function should be enumerated
Examples	A=CALL("USERFC",X,Y)	
	'	
	DEF USERFC(X,Y)	
	RETURN X*Y	
	END	

CALL (3)	Calls a sprite callback
	Processes which have been specified for each sprite using SPFUNC are called together
Format	CALL SPRITE
Examples	CALL SPRITE

CALL (4)	Calls a BG callback
	Processes which have been specified for each sprite using SPFUNC are called together
Format	CALL BG
Examples	CALL BG

XON	Declares the use of a special feature	
	- These features are not available unless their use is declared beforehand	
	- When XON EXPAD is successful, RESULT will be returned as TRUE.	
	- If the system is already in the XON state, this command will not display a dialog	
Format	XON Name of feature to use	
Arguments	Name of feature to use	MOTION: Motion sensor, gyro sensor
		EXPAD: Circle Pad Pro
		MIC: Microphone
Examples	XON MOTION	

XOFF	Stops using a special feature declared with XON	
Format	XOFF Name of the feature to stop	
Arguments	Name of feature to stop	MOTION: Motion sensor, gyro sensor
		EXPAD: Circle Pad Pro
		MIC: Microphone
Examples	XOFF MOTION	

Basic instructions (data operations and others)

Instructions for reading data, vertical synchronization, comments, etc.

READ	Reads the information enumerated with the DATA instruction into the variables	
	Information should be read in the same type as that enumerated with the DATA instruction	
Format	READ Acquisition variable 1 [, Acquisition variable 2…]	
Arguments	Acquisition variables	- Variables to store read information (Multiple variables can be specified)
		- DATA in and after the line specified with the RESTORE instruction will be acquired
		- If RESTORE is omitted, acquisition will begin with the first occurrence of DATA
Examples	READ X,Y,Z,G$	
	DATA 200,120,0,"JAN"	
	DATA 210,120,0,"FEB"	

DATA	Defines data to read with READ
	- Numerical values and character strings can be mixed
	- Expressions containing only numerical constants are handled as constants, and so can be written in DATA statements
	- Constants starting with # are also allowed
	- Expressions where &&, ||, variables, and functions are mixed are not allowed
	- Character string expressions are not allowed
Format	DATA Data [, Data…]
Notation of Data	- List numerical values and character strings, separating each one with ','
	- Character strings must be enclosed in double quotations ("") ("" cannot be omitted)
Examples	READ X,Y,Z,ST$ 'Comments can be written
	DATA 123,345,56,"SAMPLE"

RESTORE	Specifies the first DATA to read with the READ instruction	
Format	RESTORE @Label	
Arguments	@Label	- @Label name given to the beginning of the DATA instruction to be read
		- A string variable to which a @Label name is assigned can also be specified
		- It is also possible to reference a label from a different SLOT by using the format RESTORE "1:@Label name"
		- The target SLOT should be enabled beforehand with USE, e.g., USE 1
Examples	RESTORE @DATATOP	
	@DATATOP	
	DATA 123,345,56,"SAMPLE"	

OPTION	Sets the operating mode of the program	
Format	OPTION Feature name	
Arguments	Feature name	STRICT: Variable declaration is required (A reference without declaration will give an error)
		DEFINT: Causes the default variable type to be Integer
Examples	OPTION STRICT	

WAIT	Stops the program until the specified number of vertically synchronized frames has been reached	
Format	WAIT [Number of frames]	
Arguments	Number of frames	Specify the number of frames to wait, starting from the present point (0: Ignore; if omitted, 1 is assumed)
Examples	WAIT 60	

VSYNC	Stops the program until the specified number of vertically synchronized frames has been reached	
	Unlike WAIT, the VSYNC count starts from the last VSYNC	
Format	VSYNC [Number of frames]	
Arguments	Number of frames	Specify the number of frames to wait, starting from the last VSYNC (0: Ignore; if omitted, 1 is assumed)
Examples	VSYNC 1	

'	Symbols for writing comments
	- Comments do not affect program execution
Format	' [String]
Examples	' ---MAIN ROUTINE---

REM	Instructions for writing comments
	- Comments do not affect program execution
Format	REM [Character string]
Examples	REM ---MAIN ROUTINE---

KEY	Assigns an arbitrary character string to a function key	
Format	KEY Number,"Character string"	
Arguments	Number	Number of the function key (1-5)
	Character string	- Character string to assign - If the whole string cannot be displayed, '…' will be displayed at the end
Examples	KEY 1,"CLS"+CHR$(13)	

TMREAD	Converts a time string to numerical values	
Format	TMREAD ["Time string"] OUT H,M,S	
Arguments	Time string	Time string in "HH:MM:SS" format (if omitted, the current time)
Return Values	Variables to store numerical values	H: Variable to receive the hours (0-23) M: Variable to receive the minutes S: Variable to receive the seconds
Examples	TMREAD "12:59:31" OUT H,M,S	

DTREAD	Converts a date string to numerical values	
Format	DTREAD ["Date string"] OUT Y,M,D [,W]	
Arguments	Date string	Date string in "YYYY/MM/DD" format (if omitted, the current date and time)
Return Values	Variables to store numerical values	Y: Variable to receive the year M: Variable to receive the month D: Variable to receive the day W: Variable to receive the day of the week (numerical value: 0 for Sunday)
Examples	DTREAD "2014/10/12" OUT Y,M,D	

CHKLABEL	Checks if there is a label that can be referenced with the specified string	
Format	Variable = CHKLABEL("@Label string"[,Flag])	
Arguments	@Label string	- It is also possible to check a different SLOT by using CHKLABEL "1:@Label name" - The target SLOT should be enabled beforehand with USE, e.g., USE 1
	Flag	0= Searches only within DEF (if omitted, 0) 1= If not found within DEF, searches for global labels
Return Values	FALSE= Does not exist, TRUE= Exists	
Examples	A=CHKLABEL("@MAIN")	

CHKCALL	Checks if there is an instruction or function that can be referenced with the specified string	
Format	Variable = CHKCALL("Character string")	
Arguments	Character string	Character string of the instruction or function to check
Return Values	FALSE= Does not exist, TRUE= Exists	
Examples	A=CHKCALL("KEYCHECK")	

CHKVAR	Checks if there is a variable that can be referenced with the specified string	
Format	Variable = CHKVAR("Character string")	
Arguments	Character string	Character string of the variable to check
Return Values	FALSE= Does not exist, TRUE= Exists	
Examples	A=CHKVAR("COUNTX")	

DIALOG (1)	Displays a dialog and waits for a button to be pressed - The result is returned with the system variable RESULT - RESULT: 1 (Confirmed), -1 (Canceled), 0 (Time out)	
Format	DIALOG "Text string"	
Arguments	Text string	Character string to display in the dialog
Supplement (common for DIALOG instructions)	- Dialog are always displayed on the Touch Screen - The total length of the text string and caption string should be 256 characters or less - If CHR$(10) or CHR$(13) is included in the text string, a line break will occur at that point - If a negative value is set for the Timeout period, texts are handled in frame units	
Examples	DIALOG "Good morning!"	

DIALOG (2)	Displays a dialog and waits for a button to be pressed	
Format	DIALOG "Text string",[Selection type],["Caption string"],[Timeout period]	
Arguments	Text string	Character string to display in the dialog
	Selection type	0: OK (default) 5: Next
	Caption string	Character string to display in the caption field at the top of the dialog
	Timeout period	Number of seconds to wait before closing the dialog automatically (If omitted, 0: Not closed)
Supplement (common for DIALOG instructions)	- Dialog are always displayed on the Touch Screen - The total length of the text string and caption string should be 256 characters or less - If CHR$(10) or CHR$(13) is included in the text string, a line break will occur at that point - If a negative value is set for the Timeout period, texts are handled in frame units	
Examples	DIALOG "Let's get started",5,"Scenario",-120	

DIALOG (3)	Displays a dialog and waits for the specified button to be pressed	
Format	Variable = DIALOG("Text string",[Selection type],["Caption string"],[Timeout period])	
Arguments	Text string	Character string to display in the dialog
	Selection type	0: OK (default) 1: No/Yes 2: Back/Next 3: Cancel/Confirm 4: Cancel/Execute 5: Next
	Caption string	Character string to display in the caption field at the top of the dialog
	Timeout period	Number of seconds to wait before closing the dialog automatically (If omitted, 0: Not closed)
Return Values	-1: Negation (Left button) 0: Timeout 1: Affirmation (Right button) * These values will also remain in the system variable RESULT.	
Supplement (common for DIALOG instructions)	- Dialog are always displayed on the Touch Screen - The total length of the text string and caption string should be 256 characters or less - If CHR$(10) or CHR$(13) is included in the text string, a line break will occur at that point - If a negative value is set for the Timeout period, texts are handled in frame units	
Examples	R=DIALOG("Would you like to try again?" ,1,"かくにん",0)	

DIALOG (4)	Displays a dialog and waits for the Touch Screen or a hardware button to be pressed	
Format	Variable = DIALOG("Text string",Button type,["Caption string"],[Timeout period])	
Arguments	Text string	Character string to display in the dialog
	Button type	|b00| ABXY buttons (1) |b01| +Control Pad (2) |b02| L,R buttons (4) |b03| Touch Screen (8) - Specify a value for which the logical OR is calculated with the above bit value and the sign is reversed - ZL and ZR buttons cannot be detected - -1 causes only ABXY to be specified - For example, to detect ABXY and +Control Pad, -3 should be specified
	Caption string	Character string to display in the caption field at the top of the dialog
	Timeout period	Number of seconds to wait before closing the dialog automatically (if omitted, 0: Not closed)
Return Values	128: A button pressed 129: B button pressed 130: X button pressed 131: Y button pressed 132: +Control Pad up pressed 133: +Control Pad down pressed 134: +Control Pad left pressed 135: +Control Pad right pressed 136: L button pressed 137: R button pressed 140: Touch Screen pressed	
Examples	R=DIALOG("ABXYLR/+Control Pad/Touch",-15,"Special",0)	

DIALOG (5)	Displays a dialog used only for inputting file names	
Format	String=DIALOG("Initial string", "Caption string" [,Maximum characters])	
Arguments	Initial string	String that is initially input
	Caption string	String to be displayed in the caption field
	Maximum characters	Up to 14 characters
Return Values	The obtained character string will be returned * If RESULT=-1, Canceled (the character string is invalid)	
Supplement (common for DIALOG instructions)	- Dialog are always displayed on the Touch Screen - The total length of the text string and caption string should be 256 characters or less - If CHR$(10) or CHR$(13) is included in the text string, a line break will occur at that point - If a negative value is set for the Timeout period, texts are handled in frame units	
Examples	T$=DIALOG("NEWNAME0","SAVE", 14)	

DIALOG (6)	Displaying special characters in DIALOG To use special character and symbols, pass the character code in the UTF-16 format to CHR$ * For details on the UTF-16 format, please refer to a technical book or similar resource.
Supplement (common for DIALOG instructions)	- Dialog are always displayed on the Touch Screen - The total length of the text string and caption string should be 256 characters or less - If CHR$(10) or CHR$(13) is included in the text string, a line break will occur at that point - If a negative value is set for the Timeout period, texts are handled in frame units

Examples to display Japanese Kanji characters	' もういちど？
	T$="もう"+CHR$(&H4E00)+CHR$(&H5EA6)+"？"
	' かくにん
	C$=CHR$(&H78BA)+CHR$(&H8A8D)
	R=DIALOG(T$,1,C$,0)

Console input/output

Instructions related to display of characters, input of strings on the screen, etc.

CLS	Clears the console screen (the screen specified with the DISPLAY instruction)
Format	CLS
Examples	DISPLAY 0 CLS

COLOR	Specifies the display colors for the console screen Constants for text colors are available (#TBLACK to #TWHITE)	
Format	COLOR Drawing color [,Background color]	
Arguments	Drawing color	0: Transparent color 1: Black, #TBLACK 2: Dark red, #TMAROON 3: Red, #TRED 4: Dark green, #TGREEN 5: Green, #TLIME 6: Dark yellow, #TOLIVE 7: Yellow, #TYELLOW 8: Dark blue, #TNAVY 9: Blue, #TBLUE 10: Dark magenta, #TPURPLE 11: Magenta, #TMAGENTA 12: Dark cyan, #TTEAL 13: Cyan, #TCYAN 14: Gray, #TGRAY 15: White, #TWHITE
	Background color	- Background color number for each character (0-15: See the drawing colors) - If only the background color needs to be changed, the drawing color can be omitted
Examples	COLOR 7,4 COLOR #TWHITE COLOR ,0	

LOCATE	Specifies the character display location on the console screen	
Format	LOCATE [X-coordinate],[Y-coordinate] [,Z-coordinate]	
Arguments	X-,Y-coordinates	- Coordinates of each character (X:0-49,Y:0-29) - If the X- and Y-coordinates are omitted, the previous coordinates for each will be kept
	Z-coordinate	- Coordinate in the depth direction (Rear:1024<Screen surface:0<Front:-256) - If omitted, the previous Z-coordinate will be kept
Examples	LOCATE 20,15 LOCATE 0,0,-200	

PRINT	Displays characters on the console screen - Omitting expressions causes only a line break to occur - ? can be used instead of PRINT	
Format	PRINT [Expression [; or, Expression…]]	
Arguments	Expression	- Variables, string variables, numerical values, or character strings to display - Formulas are also allowed, including the four arithmetic operations, and functional calculations (The calculation results will be displayed)
	; (semicolon)	Without beginning a new line after the previous display item, displays the next display item without any space
	, (comma)	- Without beginning a new line after the previous display item, places a set interval before the next display item - The display location is determined according to a system variable (the TABSTEP unit)
Examples	PRINT "RESULT(X,Y)=";DX*4+1,DY+1	

ATTR	Sets the rotation/inversion attributes of the characters to display on the console screen Constants for text attributes are available (#TROT0-270, #TREVH,V)	
Format	ATTR Display attribute	
Arguments	Display attribute	|b00| ↑Rotation by 90 degrees (specified by using two bits: b00 and b01) |b01| ↓#TROT0, #TROT90, #TROT180, #TROT270 |b02| Horizontal inversion (0=OFF, 1=ON), #TREVH |b03| Vertical inversion (0=OFF, 1=ON), #TREVV
Examples	ATTR 3:PRINT "ABC"	

SCROLL	Adjusts the display location of the whole console screen - Can give the impression of a moving view point (characters will move in the opposite direction) - Characters pushed out of the screen will disappear	
Format	SCROLL Number of characters X, Number of characters Y	
Arguments	Number of characters X	Amount of horizontal view point movement (Negative values indicate leftward movement, positive values rightward movement)
	Number of characters Y	Amount of vertical view point movement (Negative values indicate upward movement, positive values downward movement)
Examples	SCROLL 5,7	

CHKCHR	Checks the character code of a character on the console screen	
Format	Variable = CHKCHR(X-coordinate,Y-coordinate)	
Arguments	X-,Y-coordinates	Coordinates in character units (X:0-49,Y:0-29)
Return Values	UTF-16 character code	
Examples	CODE=CHKCHR(0,0)	

INPUT	Inputs numerical values or character strings from the keyboard - Waits for input until the ENTER key is input - If the number of input items is insufficient, "?Redo from start" will be displayed for re-input	
Format	INPUT ["Guiding text string";] Variable[,Variable 2…]	
Arguments	Guiding text string	- Guidance message for input (Optional) - If , (comma) is used instead of ; after the guiding text string, a ? mark will not be displayed - Only when ; is used, a string variable can be used for the guiding text string
	Variables	- Variables to receive the input (numerical values or string variables) - When specifying multiple variables, they should be delimited with commas (,)
Examples	INPUT "Your name and age";NM$,AG	

LINPUT	Gets a character string input from the keyboard - Also accepts "," and other characters that the INPUT instruction does not allow - Waits for input until the ENTER key is input	
Format	LINPUT ["Guiding text string";] String variable	
Arguments	Guiding text string	Guidance message for input (Optional)
	String variable	String variable to receive a single line input
Examples	LINPUT "ADDRESS:";ADR$	

INKEY$	Gets a character input from the keyboard (without waiting for input)
Format	String variable=INKEY$()
Arguments	None
Return Values	- A character (UTF-16) from the keyboard - If there is no input, "" will be returned
Examples	C$=INKEY$()

FONTDEF (1)	Defines a font for the specified character code	
Format	FONTDEF Character code, "Font definition string"	
Arguments	Character code	Character code (UTF-16) for which to define a font
	Font definition string	- One pixel corresponds to a 16-bit color code in the RGBA=5551 format - 5 bits for each RGB color (0-31) + alpha channel 1 bit (0: Transparent, 1: Opaque) - A color element should be handled as a 4-digit hexadecimal string - Example) White: FFFF, Black: 0001, Red: F801 - As one character occupies 8x8=64 pixels, its font definition string should consist of a total of 256 characters
See Also	Font images can be manipulated with GCOPY, GSAVE, or GLOAD page number -1	

Examples	F$="FFFF":Z$="0000"
	D$=F$*7+Z$
	D$=D$+F$*2+Z$*3+F$*2+Z$
	D$=D$+F$+Z$+F$*3+Z$+F$+Z$
	D$=D$+F$+Z$*5+F$+Z$
	D$=D$+F$+Z$+F$*3+Z$+F$+Z$
	D$=D$+F$+Z$+F$*3+Z$+F$+Z$
	D$=D$+F$*7+Z$
	D$=D$+Z$*8
	FONTDEF ASC("A"),D$	

FONTDEF (2)		Defines a font for the specified character code
Format		FONTDEF Character code, Numerical value array
Arguments	Character code	Character code (UTF-16) for which to define a font
	Font definition array	- A numerical value array with an element for each pixel should be prepared (8x8 pixels for one character = 64 elements) - One pixel corresponds to a 16-bit color code in the RGBA=5551 format - 5 bits for each RGB color (0-31) + alpha channel 1 bit (0: Transparent, 1: Opaque) - Example) White: &HFFFF, Black: &H0001, Red: &HF801
See Also		Font images can be manipulated with GCOPY, GSAVE, or GLOAD page number -1
Examples		DIM F%[64] DATA "11111110" DATA "11000110" DATA "10111010" DATA "10000010" DATA "10111010" DATA "10111010" DATA "11111110" DATA "00000000" TOP=ASC("A"):CNT=1 FOR I=0 TO CNT-1 FOR D=0 TO 7 READ F$ FOR B=0 TO 7 C=0:IF MID$(F$,B,1)=="1" THEN C=&HFFFF F%[D*8+B]=C NEXT NEXT FONTDEF TOP+I,F% NEXT

FONTDEF (3)	Resets the font definition to its initial state
Format	FONTDEF
Examples	FONTDEF

WIDTH		Changes the console character sizes - Only enlarges the characters, does not display a smooth zoomed-in view - This is an auxiliary function for people who have trouble viewing small characters
Format		WIDTH Font size
Arguments	Font size	8: 8x8 pixels (Standard) 16: 16x16 pixels (Twice as large as the normal horizontal and vertical display)
Examples		WIDTH 16 A=WIDTH()

Various kinds of input

Instructions for retrieving information from buttons, sticks, touch sensors, and microphones

BUTTON		Gets the status of hardware buttons Constants for buttons are available for return values
Format		Variable=BUTTON([Feature ID [,Terminal ID]])
Arguments	Feature ID	0: Held down 1: Moment pressed (with the repeat feature enabled) 2: Moment pressed (with the repeat feature disabled) 3: Moment released
	Terminal ID (0-3)	This should be specified to get information from another terminal via wireless communication
Return Values		|b00| +Control Pad up (1), #UP |b01| +Control Pad down (2), #DOWN |b02| +Control Pad left (4), #LEFT |b03| +Control Pad right (8), #RIGHT |b04| A button (16), #A |b05| B button (32), #B |b06| X button (64), #X |b07| Y button (128), #Y |b08| L button (256), #L |b09| R button (512), #R |b10| Not used |b11| ZR button (2048), #ZL |b12| ZL button (4096), #ZR - The buttons correspond to b0-b12 (If a button is pressed, its corresponding bit = 1) - Contents in () next to button names are decimal numerals - ZR and ZL buttons are available only when Circle Pad Pro is used

Examples		B=BUTTON()
		B=BUTTON(0,3)

BREPEAT		Sets the key repeat feature - Omitting Start time and Interval will turn off repeat - Management numbers differ from the bit values that correspond to each button in BUTTON - ZR and ZL buttons are available only when Circle Pad Pro is used
Format		BREPEAT Button ID, Start time, Interval
Arguments	Button ID	0: +Control Pad up ID 1: +Control Pad down ID 2: +Control Pad left ID 3: +Control Pad right ID 4: A button ID 5: B button ID 6: X button ID 7: Y button ID 8: L button ID 9: R button ID 10: Not used 11: ZR button ID 12: ZL button ID
	Start time	Time from when a key is pressed first to when repeat begins (in units of 1/60th of a second)
	Interval	Repeat interval after repeat begins (in units of 1/60th of a second, 0 = Repeat OFF)
Examples		BREPEAT 0,15,4

STICK		Gets information on the Circle Pad
Format		STICK [Terminal ID] OUT X,Y
Arguments	Terminal ID (0-3)	This should be specified when information is obtained from another terminal via wireless communication
Return Values	X,Y	- Variables to receive Circle Pad input magnitude (X:±1.0, Y:±1.0) - Actual return values will be around ±0.86 - For Y values, ↑ corresponds to positive and ↓ to negative
Examples		STICK OUT X,Y STICK 3 OUT X,Y

STICKEX	Gets information on the Circle Pad Pro stick Circle Pad Pro should be enabled beforehand with XON EXPAD	
Format	STICKEX [Terminal ID] OUT X,Y	
Arguments	Terminal ID (0-3)	This should be specified when information from another terminal is to be obtained via wireless communication
Return Values	X,Y	Variables to receive Circle Pad Pro input magnitude (X:±1.0, Y:±1.0)
Examples	XON EXPAD STICKEX OUT X,Y	

ACCEL	Gets information on acceleration - The motion sensor should be enabled beforehand with XON MOTION - Note that this instruction will continue to detect 1G acceleration in the gravity direction - This is useful when operation is performed while tilting	
Format	ACCEL OUT X,Y,Z	
Arguments	None	
Return Values	X,Y,Z	Variables to receive acceleration (Unit: G)
Examples	XON MOTION ACCEL OUT X,Y,Z	

GYROV	Gets information on the angular velocity of the gyro sensor Motion sensor(s) should be enabled beforehand with XON MOTION	
Format	GYROV OUT P,R,Y	
Arguments	None	
Return Values	**P**	Variable to receive Pitch (angular velocity of the X-coordinate) (Unit: radians/second)
	R	Variable to receive Roll (angular velocity of the Y-coordinate) (Unit: radians/second)
	Y	Variable to receive Yaw (angular velocity of the Z-coordinate) (Unit: radians/second)
Examples	XON MOTION GYROV OUT P,R,Y	

GYROA	Gets information on the angle of the gyro sensor Motion sensor(s) should be enabled beforehand with XON MOTION	
Format	GYROA OUT P,R,Y	
Arguments	None	
Return Values	**P**	Variable to receive Pitch (angle of the X-coordinate) (Unit: radian)
	R	Variable to receive Roll (angle of the Y-coordinate) (Unit: radian)
	Y	Variable to receive Yaw (angle of the Z-coordinate) (Unit: radian)
Examples	XON MOTION GYROA OUT P,R,Y	

GYROSYNC	Updates gyro information - Error accumulation may occur if gyro information is repeatedly retrieved - This instruction should be called to reset information appropriately - However, calling this instruction at an interval of 1 frame or less is prohibited
Format	GYROSYNC
Examples	GYROSYNC

TOUCH	Gets touch information The 5 pixels around the edge of the screen cannot be read	
Format	TOUCH [Terminal ID] OUT STTM,TX,TY	
Arguments	Terminal ID (0-3)	This should be specified when information from another terminal is to be obtained via wireless communication
Return Values	STTM	Variable to receive the time when the screen is touched (0 = No touch)
	TX,TY	- Variables to receive the touch coordinates (TX: 5-314, TY: 5-234) - Note that returned values are not in the same range as the size of the Touch Screen
Examples	TOUCH OUT TM,TX,TY	

MICSTART	Starts sampling from the microphone - The microphone should be enabled beforehand with XON MIC - Recorded into memory used for sampling in the system	
Format	MICSTART Sampling rate, Number of bits, Number of seconds	
Arguments	Sampling rate	0: 8180Hz 1: 10910Hz 2: 16360Hz 3: 32730Hz
	Number of bits	0: 8 bits 1: 16 bits
	Number of seconds	0: Loop mode 1-: Number of seconds for sampling - 8180Hz: Up to 32 sec for 8 bits, 16 sec for 16 bits - 10910Hz: Up to 24 sec for 8 bits, 12 sec for 16 bits - 16360Hz: Up to 16 sec for 8 bits, 8 sec for 16 bits - 32730Hz: Up to 8 sec for 8 bits, 4 sec for 16 bits - In loop mode, data will be overwritten from the beginning once the maximum number of seconds has been reached
Examples	XON MIC MICSTART 0,1,10	

MICSTOP	Stops sampling from the microphone
Format	MICSTOP
Examples	MICSTOP

MICDATA	Gets data from the microphone This returns sampling data from the specified position	
Format	Variable=MICDATA(Acquisition position)	
Arguments	Acquisition position	- 0- (The range is determined according to the number of bits and the maximum number of seconds) - In loop mode, the range will not be checked
Return Values	Waveform data	- For 8 bits, return values are 128-basis - For 16 bits, return values are 32768-basis
Examples	D=MICDATA(100)	

MICSAVE	Copies data from the internal sampling memory to an array	
Format	MICSAVE [[Acquisition position,] Number of samples to get,] Array name	
Arguments	Acquisition position	Position to start capturing from (0-)
	Number of samples	- Number of samples to capture (If omitted, the whole sampling buffer) - Any value greater than the product of the sampling rate and the number of seconds specified with MICSTART will give an error
	Array name	- Array to store the captured sampling data - For one-dimensional arrays, if the number of samples exceeds the number of elements, the array will be extended automatically
Examples	MICSTART 0,0,1 'rate:8180 bit:8 length:1sec DIM WAVE%[8180] 'MICSIZE MICSAVE 0,8180,WAVE%	

Files

Instructions for retrieving file lists, reading from/writing to files, etc.

FILES (1)	Displays a file list on the console	
Format	FILES ["File type"]	
Arguments	File type	To display only a certain type of file, specify the following: "TXT:" Texts and programs "DAT:" Binary data (including graphics) "//" Project list "PROJECT/" Project name should be specified
Examples	FILES	

FILES (2)	Gets a file list and stores it in an array	
Format	FILES ["File type",] String array	
Arguments	File type	To display only a certain type of file, specify the following: "TXT:" Texts and programs "DAT:" Binary data (including graphics) "//" Project list "PROJECT/" Project name should be specified
	String array	String array to store the listed file names - For one-dimensional arrays, the array will be automatically extended according to the number of files obtained
Examples	DIM NAMETBL$[100] FILES NAMETBL$	

LOAD (1)	Loads a file - A confirmation dialog will be displayed - It is impossible to load a program into the same program SLOT as a running program	
Format	LOAD "[Resource name:]File name"[,Dialog display flag]	
Arguments	Resource name:	If omitted: Current program SLOT PRG0-PRG3: Program SLOT (PRG = PRG0) GRP0-GRP5: Graphic page GRPF: Font image page
	File name	Name of file to load
	Dialog display flag	FALSE = Suppresses confirmation dialog
Examples	LOAD "PROGNAME" LOAD "GRP0:GRPDATA"	

LOAD (2)	Loads a text file into a string variable	
Format	LOAD "TXT:File name"[,Dialog display flag] OUT TX$	
Arguments	File name	Name of text file to load (prefixed with "TXT:")
	Dialog display flag	FALSE = Suppresses confirmation dialog
Return Values	TX$	String variable to store the loaded text file
Examples	LOAD "TXT:MEMOFILE" OUT TX$	

LOAD (3)	Loads a text file into a string variable	
Format	String variable = LOAD("TXT:File name" [,Dialog display flag])	
Arguments	File name	Name of text file to load (prefixed with "TXT:")
	String variable	String variable to store the loaded text file
	Dialog display flag	FALSE = Suppresses confirmation dialog
Examples	TX$=LOAD("TXT:MEMOFILE")	

LOAD (4)	Loads a binary file into a numerical value array	
Format	LOAD "DAT:File name", Numerical value array[,Dialog display flag]	
Arguments	File name	Name of binary file to load (prefixed with "DAT:")
	Numerical value array	Numerical value variable to store the loaded binary file
	Dialog display flag	FALSE = Suppresses confirmation dialog
Examples	DIM MARRAY[100] LOAD "DAT:MDATA", MARRAY	

SAVE (1)	Saves a file - When run, a confirmation dialog will be displayed - The confirmation dialog for SAVE cannot be hidden	
Format	SAVE "[Resource name:]File name"	
Arguments	Resource name:	If omitted: Current program SLOT PRG0-PRG3: Program SLOT (PRG = PRG0) GRP0-GRP5: Graphic page GRPF: Font image page
	File name	Name to save the file under
Examples	SAVE "PRG0:TEST"	

SAVE (2)	Saves a string variable to a text file	
Format	SAVE "TXT:File name", String variable	
Arguments	File name	Name to save the file under (prefixed with "TXT:")
	String variable	String variable containing the text data to be saved (UTF-8)
Examples	SAVE "TXT:MEMOFILE",TX$	

SAVE (3)	Saves a numerical value array to a binary file	
Format	SAVE "DAT:File name", Numerical value array	
Arguments	File name	Name to save the file under (prefixed with "DAT:")
	Numerical value array	Numerical value array containing the data to be saved
Examples	SAVE "DAT:TEST",MARRAY	

RENAME	Changes a file name When run, a confirmation dialog will be displayed	
Format	RENAME "[File type:]File name", "[File type:]New name"	
Arguments	File type:	"TXT:" Texts and programs (optional) "DAT:" Binary data (including graphics)
	File name	Name of file to change name of
	New name	New file name
Examples	RENAME "SAMPLE","NEWNAME"	

DELETE	Deletes a file When run, a confirmation dialog will be displayed	
Format	DELETE "[File type:]File name"	
Arguments	File type:	"TXT:" Texts and programs (optional) "DAT:" Binary data (including graphics)
Examples	DELETE "SAMPLE"	

EXEC (1)	Loads and executes a program - It is impossible to return from a program started with EXEC to the previous program - It is possible to return by using END in a program started with EXEC in another SLOT - This cannot be used to run a program in DIRECT mode	
Format	EXEC "[Resource name:]File name"	
Arguments	Resource name:	PRG0-PRG3: Program SLOT into which to load the program
	File name	File name of the program to load
Examples	EXEC "SAMPLE" EXEC "PRG0:SBGED"	

EXEC (2)	Executes a program in a different SLOT - It is impossible to return from a program executed with EXEC to the previous program - It is possible to return by using END in a program started with EXEC in another SLOT - This cannot be used to run a program in DIRECT mode	
Format	EXEC Program SLOT	
Arguments	Program SLOT	0-3: SLOT number of the program to execute
Examples	EXEC 2	

USE	Makes a program in the specified program SLOT executable	
Format	USE Program SLOT	
Arguments	Program SLOT	0-3: Program SLOT
Examples	USE 2	

CHKFILE	Checks if the specified file exists	
Format	Variable = CHKFILE("[File type:]File name")	
Arguments	File type	"TXT:" Texts and programs "DAT:" Binary data (including graphics)
	File name	Name of the file to check
Return Values	TRUE= Exists, FALSE= Does not exist	
Examples	A=CHKFILE("SBATTACK")	

Wireless communication

Instructions related to wireless communication sessions, which allow up to four systems to be connected

* In order to connect, each system must have SmileBASIC installed.

MPSTART	Starts a wireless communication session - Connection to a session is allowed when MPSTART identifiers are equal - The RESULT system variable should be used to get information on whether or not a session has successfully been established - Communication will be terminated if the system goes into sleep mode	
Format	MPSTART Maximum number of connected users, "Communication identifier string"	
Arguments	Maximum number of connected users	2-4: Number of concurrent connected users
	Communication identifier string	Any character string for authentication
Examples	MPSTART 4,"ANYSTR"	

MPEND	Ends a wireless communication session - All participants close the session synchronously - A wait dialog will be displayed
Format	MPEND
Examples	MPEND

MPSEND	Sends data to all participants in a wireless communication session - Delivery of sent data is guaranteed, but with a delay - A large number of MPSEND calls in a short period will result in an error * Communication buffer overflow - Communication will be terminated if the system goes into sleep mode	
Format	MPSEND "Character string to send"	
Arguments	Character string to send	Character string of up to 256 bytes
Examples	MPSEND "HELLO!"	

MPRECV	Receives data from MPSEND - If there is no data to receive, the sender ID will contain the value -1 - Communication will be terminated if the system goes into sleep mode	
Format	MPRECV OUT SID,RCV$	
Arguments	SID	0-3: Connection destination number from which the string will be sent
	RCV$	String variable to store the received data
Examples	MPRECV OUT SID,RCV$ PRINT SID;":";RCV$	

MPSTAT	Gets the connection status of a specified terminal in a wireless communication session Communication will be terminated if the system goes into sleep mode	
Format	Variable = MPSTAT([Terminal ID])	
Arguments	Terminal ID	0-3: ID of another terminal in the wireless communication session (If omitted, the whole session will be assumed)
Return Values	0: Not connected, 1: Connected	
Examples	RET=MPSTAT(2)	

MPNAME$	Gets the terminal name of a specified terminal in a wireless communication session Communication will be terminated if the system goes into sleep mode	
Format	String variable = MPNAME$(Terminal ID)	
Arguments	Terminal ID	0-3: ID of another terminal in the wireless communication session
Return Values	Terminal name string	
Examples	NAME$=MPNAME$(3)	

MPGET	Gets user-defined data from a specified terminal in a wireless communication session Communication will be terminated if the system goes into sleep mode	
Format	Variable=MPGET(Terminal ID, Internal management number)	
Arguments	Terminal ID	0-3: ID of another terminal in the wireless communication session
	Internal management number	0-8: Management number of the target data
Return Values	Numerical value (integer) of the specified data	
Examples	RET=MPGET(0, 5)	

MPSET	Writes to user-defined data in a wireless communication session Communication will be terminated if the system goes into sleep mode	
Format	MPSET Internal management number, Numerical value	
Arguments	Internal management number	0-8: Management number of the target data/td>
	Numerical value	Numerical value to register (Only an integer value is allowed)
Examples	MPSET 5,123	

Screen control

Instructions related to screen display modes, etc.

XSCREEN	Sets a screen mode - Screen modes 2 and 3 can also be used in DIRECT mode, but the Touch Screen will be switched to a keyboard after execution is started - 3D specification can be disabled in the Parental Control settings	
Format	XSCREEN Screen mode [,Number of sprite assignments ,Number of BG assignments]	
Arguments	Screen mode	0: Upper screen-3D, Touch Screen-Not used (Default) 1: Upper screen-2D, Touch Screen-Not used 2: Upper screen-3D, Touch Screen-Used (Keyboard displayed during INPUT) 3: Upper screen-2D, Touch Screen-Used (Keyboard displayed during INPUT) 4: Upper and Touch screens combined (Upper screen 2D; INPUT and DIRECT mode not allowed)
	Number of sprite assignments	- Number of sprites to assign to the upper screen: 0-512 - Touch Screen: 512 - number of SPs on the upper screen
	Number of BG allocations	- Number of BG layers to assign to the upper screen: 0-4 - Touch Screen: 4 - number of BG layers on the upper screen
Examples	XSCREEN 2,128,4	

DISPLAY (1)	Selects the screen to manipulate (Upper or Touch) - DISPLAY 1 can be specified when XSCREEN 2 or 3 is used -This command cannot be directly executed in DIRECT mode.	
Format	DISPLAY Screen ID	
Arguments	Screen ID	0: Upper screen, 1: Touch Screen
Examples	DISPLAY 0	

DISPLAY (2)	Gets the Screen ID that is currently being used - DISPLAY 1 can be specified when XSCREEN 2 or 3 is used - This command cannot be directly executed in DIRECT mode.
Format	Variable=DISPLAY()
Return Values	Screen ID (0: Upper screen, 1: Touch Screen)
Examples	A=DISPLAY()

VISIBLE	Switches screen display elements ON/OFF	
Format	VISIBLE Console,Graphic,BG,sprite	
Arguments	Console	0: Hide (#OFF), 1: Display (#ON)
	Graphic	0: Hide (#OFF), 1: Display (#ON)
	BG	0: Hide (#OFF), 1: Display (#ON)
	sprite	0: Hide (#OFF), 1: Display (#ON)
Examples	VISIBLE 1,1,1,1	

BACKCOLOR (1)	Specifies a background color	
Format	BACKCOLOR Background color code	
Arguments	Background color code	- Usually specified with the RGB function, e.g., BACKCOLOR RGB(64,128,128) - To specify a numerical value directly, a color code consisting of an 8-bit value for each RGB element should be specified
Examples	BACKCOLOR RGB(64,128,128)	

BACKCOLOR (2)	Specifies the current background color
Format	Variable=BACKCOLOR()
Return Values	Color code of the background color currently set
Examples	C=BACKCOLOR()

ACLS	Resets the draw settings to their settings when BASIC was started - The same operations as those shown after END in the Examples should be executed - Sound settings such as BGM will not be affected
Format	ACLS
Examples	ACLS END '--- XSCREEN 0 LOAD "GRP4:SYS/DEFSP.GRP" LOAD "GRP5:SYS/DEFBG.GRP" FONTDEF SPDEF DISPLAY 1 WIDTH 8 BACKCOLOR 0 FADE 0 COLOR 15,0:LOCATE 0,0,0:ATTR 0:CLS GPAGE 1,1:SPPAGE 4:BGPAGE 5 VISIBLE 1,1,1,1 DISPLAY 0 BACKCOLOR 0 FADE 0 WIDTH 8 COLOR 15,0:LOCATE 0,0,0:ATTR 0:CLS FOR I=0 TO 3:GPAGE I,I:GCLS 0:NEXT GPAGE 0,0:GPRIO 1024 SPPAGE 4:SPCLR BGPAGE 5:BGCLR VISIBLE 1,1,1,1

FADE (1)	Sets the color for the screen fader - The fader is always displayed in the front - The entire screen is filled with the fading color (taking the transparent color into consideration)	
Format	FADE Fading color [,Fading time]	
Arguments	Fading color	The color to fill the screen (specifying RGB(0,0,0,0) disables the fader)
	Fading time	The screen color changes from the current fading color to the specified fading color over a specified time period, which can be specified in units of 1/60th of a second.
Examples	FADE RGB(32,64,64,64),60	

FADE (2)	Gets the current screen fader color
Format	Value=FADE()
Return Values	Color code consisting of an 8-bit value for each ARGB element
Examples	C=FADE()

FADECHK	Gets the state of the fading animation	
Format	Variable=FADECHK()	
Return Values	TRUE= Animation in progress, FALSE= Animation suspended	
Examples	R=FADECHK()	

Graphics

Functions for drawing figures including lines and circles in pixel units

GPAGE (1)	Specifies a page for graphic display and a page for manipulation	
Format	GPAGE Display page, Manipulation page	
Arguments	Display page	0-5: GRP0-GRP5
	Manipulation page	0-5: GRP0-GRP5 * By default, GRP4 contains sprites and GRP5 contains BG images.
Examples	GPAGE 0,0	

GPAGE (2)	Gets information on the graphic page currently set	
Format	GPAGE OUT VP,WP	
Arguments	None	
Return Values	VP	Page number for display (0-5)
	WP	Page number for manipulation (0-5)
Examples	GPAGE OUT WP,GP	

GCOLOR (1)	Specifies the graphic draw color	
Format	GCOLOR Color code	
Arguments	Color code	- Usually specified with the RGB function, e.g., GCOLOR RGB(64,255,48) - To specify a numerical value directly, a color code consisting of an 8-bit value for each ARGB element should be specified - An 8-bit value for A (255: Opaque, Otherwise: Transparent) + one for each RGB element (0-255)
Examples	GCOLOR RGB(255,0,0)	

GCOLOR (2)	Specifies the graphic draw color	
Format	GCOLOR OUT C32	
Arguments	None	
Return Values	C32	Color code consisting of an 8-bit value for each ARGB element
Examples	GCOLOR OUT C32	

RGB	Gets a color code based on 8-bit RGB values - Black RGB(0,0,0) - White RGB(255,255,255) - Light gray RGB(224,224,224) - Gray RGB(128,128,128) - Dark gray RGB(64,64,64) - Red RGB(255,0,0) - Pink RGB(255,96,208) - Purple RGB(160,32,255) - Light blue RGB(80,208,255) - Blue RGB(0,32,255) - Yellow green RGB(96,255,128) - Green RGB(0,192,0) - Yellow RGB(255,224,32) - Orange RGB(255,160,16) - Brown RGB(160,128,96) - Pale pink RGB(255,208,160)	
Format	Variable = RGB([Transparency,] Red,Green,Blue)	
Arguments	Transparency	- Transparency information (255: Opaque, Otherwise: Transparent) - A transparency level in the range 0-255 can be specified for SPCOLOR
	Red, Green, Blue	Each color has an 8-bit color tone value (each 0-255)
Return Values	Variable=Color code (An 8-bit value for each ARGB element) * See GCOLOR	
Examples	GPSET 0,0, RGB(255,255,0) 'YELLOW	

RGBREAD	Gets each RGB element from a color code	
Format	RGBREAD Color code OUT [A,] R,G,B	
Arguments	Color code	Color code consisting of an 8-bit value for each ARGB element * See GCOLOR
Return Values	A	Variable to receive transparency information (Opaque: 255 - 0: Transparent)
	R,G,B	Variables to receive 8-bit color information (each 0-255)
Examples	RGBREAD C OUT R,G,B	

GCLIP	Specifies a clipping area on the graphic screen - When the range is omitted in display mode, the whole screen will be clipped - When the range is omitted in write mode, the whole graphic page is assumed	
Format	GCLIP Clip mode [,Start point X,Start point Y,End point X, End point Y]	
Arguments	Clip mode	0: Clipping for display, 1: Clipping for writing
	Start point X,Y	Start point coordinates for the clipping area
	End point X,Y	End point coordinates for the clipping area
Examples	GCLIP 0,100,100,200,200	

GPRIO	Changes the display order of the graphic screen If 3D mode is used, the whole graphic screen will be affected	
Format	GPRIO Z-coordinate	
Arguments	Z-coordinate	Coordinate in the depth direction (Rear:1024 < Screen surface:0 < Front:-256)
Examples	GPRIO -100	

GCLS	Clears the graphic screen - Instruction to fill the whole screen with black - It is also possible to specify a color code with which to fill the screen	
Format	GCLS [Color code]	
Arguments	Color code	Color code consisting of an 8-bit value for each ARGB element * See GCOLOR
Examples	GCLS RGB(32,32,32)	

GSPOIT	Gets a color from the specified coordinates on the graphic screen The return value may not be the same as the value specified at the time of drawing because it has passed through the internal color representation	
Format	Variable = GSPOIT(X-coordinate,Y-coordinate)	
Arguments	X-,Y-coordinates	Coordinates of which to get the color (X: 0-399, Y: 0-239)
Return Values	Color code consisting of an 8-bit value for each ARGB element * See GCOLOR	
Examples	C=GSPOIT(100,100)	

GPSET	Puts a pixel on the graphic screen	
Format	GPSET X-coordinate,Y-coordinate [,Color code]	
Arguments	X-,Y-coordinates	Coordinates to place the pixel at
	Color code	Color code consisting of an 8-bit value for each ARGB element * See GCOLOR
Examples	GPSET 100,50	

GLINE	Draws a straight line on the graphic screen	
Format	GLINE Start point X,Start point Y, End point X,End point Y [,Color code]	
Arguments	Start point X,Y	Start point coordinates (X: 0-399, Y: 0-239)
	End point X,Y	End point coordinates (X: 0-399, Y: 0-239)
	Color code	Color code consisting of an 8-bit value for each ARGB element * See GCOLOR
Examples	GLINE 0,0,399,239,RGB(0,255,255)	

GCIRCLE (1)	Draws a circle on the graphic screen	
Format	GCIRCLE Center point X,Center point Y, Radius [,Color code]	
Arguments	Center point X,Y	Center point coordinates (X: 0-399, Y: 0-239)
	Radius	Radius of the circle (in pixels) 1-
	Color code	Color code consisting of an 8-bit value for each ARGB element * See GCOLOR
Examples	GCIRCLE 200,120,30	

GCIRCLE (2)	Draws an arc on the graphic screen	
Format	GCIRCLE Center point X, Center point Y, Radius, Start angle, End angle [Flag [Color code]]	
Arguments	Center point X,Y	Center point coordinates (X: 0-399, Y: 0-239)
	Radius	Radius of the circle (in pixels) 1-
	Start angle, End angle	Angle of the arc 0-360
	Flag	Drawing method (0=Arc, 1=Sector)
	Color code	Color code consisting of an 8-bit value for each ARGB element * See GCOLOR
Examples	GCIRCLE 200,120,30, 0,45, 1	

GBOX	Draws a quadrangle on the graphic screen	
Format	GBOX Start point X,Start point Y, End point X,End point Y [,Color code]	
Arguments	Start point X,Y	Start point coordinates (X: 0-399, Y: 0-239)
	End point X,Y	End point coordinates (X: 0-399, Y: 0-239)
	Color code	Color code consisting of an 8-bit value for each ARGB element * See GCOLOR
Examples	GBOX 0,0,399,239	

GFILL	Draws a quadrangle on the graphic screen and fills it with a color	
Format	GFILL Start point X,Start point Y, End point X,End point Y [,Color code]	
Arguments	Start point X,Y	Start point coordinates (X: 0-399, Y: 0-239)
	End point X,Y	End point coordinates (X: 0-399, Y: 0-239)
	Color code	Color code consisting of an 8-bit value for each ARGB element * See GCOLOR
Examples	GFILL 0,0,399,239	

GPAINT	Fills the graphic screen with color If the border color is omitted, the color range at the start point coordinates will be used	
Format	GPAINT Start point X, Start point Y [,Fill Color [, Border color]]	
Arguments	Start point X,Y	Coordinates to start filling from (X: 0-399, Y: 0-239)
	Fill color	Color code consisting of an 8-bit value for each ARGB element * See GCOLOR
	Border color	Should be specified in the same way as Fill color
Examples	GPAINT 200,120,RGB(255,0,0),RGB(0,0,0)	

GCOPY	Copies an image from another graphic page	
Format	GCOPY [Transfer source page,] Start point X,Start point Y, End point X,End point Y, Transfer destination X,Transfer destination Y, Copy mode	
Arguments	Transfer source page	0-5 (GRP0-GRP5), -1 (GRPF) If omitted: Current drawing page
	Start point X,Y End point X,Y	Start and end point coordinates of the copy source range (X: 0-399, Y: 0-239)
	Transfer destination X,Y	Start point coordinates of the copy destination range (X: 0-399, Y: 0-239)
	Copy mode	TRUE = Copies the transparent color, FALSE = Does not copy the transparent color
Examples	GCOPY 0, 0,0,100,100, 200,100 ,1	

GSAVE	Copies an image (whole screen) to an array	
Format	GSAVE [Transfer source page,] [X,Y,Width,Height,] Transfer destination array, Color conversion flag	
Arguments	Transfer source page	0-5 (GRP0-GRP5), -1(GRPF) If omitted: Current drawing page
	X,Y,Width,Height	Start point X-coordinate, start point Y-coordinate, and width/height (in pixels) of the copy source range If omitted: Current drawing area
	Transfer destination array	Array variable to store the image * If the number of elements in the array is insufficient, the required element(s) will be added automatically, provided that the array is one-dimensional.
	Color conversion flag	0: Performs color conversion (Converts to 32-bit logical colors) 1: Leaves the physical codes as they are (16-bit)
Examples	DIM WORK[0] GSAVE 0,0,0,512,512,WORK,1	

GLOAD (1)	Copies image data from an array to the graphic screen	
Format	GLOAD [X,Y,Width,Height,] Image array,Color conversion flag,Copy mode	
Arguments	X,Y,Width,Height	Start point X-coordinate, start point Y-coordinate, and width/height (in pixels) of the copy destination range
	Image array	Numerical value array containing image data stored with GSAVE
	Color conversion flag	0: Performs color conversion (Converts to 32-bit logical colors) 1: Leaves the physical codes as they are (16-bit)
	Copy mode	TRUE = Copies the transparent color, FALSE = Does not copy the transparent color
Examples	GLOAD 0,0,512,512, WORK, 1, 0	

GLOAD (2)	Copies image data from an array to the graphic screen Colors will be handled as index colors from palettes	
Format	GLOAD [X,Y,Width,Height,] Image array,Palette array,Copy mode	
Arguments	X,Y,Width,Height	Start point X-coordinate, start point Y-coordinate, and width/height (in pixels) of the copy destination range
	Image array	Numerical value array containing image data stored with GSAVE
	Palette array	Numerical value array containing palette data
	Copy mode	TRUE = Copies the transparent color, FALSE = Does not copy the transparent color
Examples	GLOAD 0,0,512,512, WORK, PALETTE, 0	

GTRI	Draws a triangle on the graphic screen and fills it with a color	
Format	GTRI X1,Y1, X2,Y2, X3,Y3 [,Color code]	
Arguments	X1,Y1	Vertex 1(X: 0-399, Y: 0-239)
	X2,Y2	Vertex 2 (X: 0-399, Y: 0-239)
	X3,Y3	Vertex 3 (X: 0-399, Y: 0-239)
	Color code	Color code consisting of an 8-bit value for each ARGB element * See GCOLOR
Examples	GTRI 200,10,300,200,100,200	

GPUTCHR (1)	Draws a character string on the graphic screen	
Format	GPUTCHR X,Y, "String" [,Scale X,Scale Y][,Color code]	
Arguments	X,Y	Display position (X: 0-399, Y: 0-239)
	"String"	String to display
	Scale X,Y	Display magnification (No scaling=1.0)
	Color code	Color code consisting of an 8-bit value for each ARGB element * See GCOLOR
Examples	GPUTCHR 10,10,"あいう"	

GPUTCHR (2)	Draws a character on the graphic screen	
Format	GPUTCHR X,Y, Character code [,Scale X,Scale Y][,Color code]	
Arguments	X,Y	Display position (X: 0-399, Y: 0-239)
	Character code	Character code to display
	Scale X,Y	Display magnification (No scaling=1.0)
	Color code	Color code consisting of an 8-bit value for each ARGB element * See GCOLOR
Examples	GPUTCHR 10,10,ASC("A")	

Sprites

Functions related to display of images made up of rectangles that can be moved freely

SPPAGE (1)	Sets a graphic page to assign to sprites	
Format	SPPAGE Graphic page	
Arguments	Graphic page	0-5 (GRP0-GRP5) By default, the page for sprites is 4 (GRP4)
Examples	SPPAGE 4	

SPPAGE (2)	Gets the graphic page that has been assigned to sprites
Format	Variable=SPPAGE()
Return Values	Graphic page number (0-5)
Examples	P=SPPAGE()

SPCLIP	Specifies a clipping area in the sprite - If the range is omitted, the whole screen will be assumed	
Format	SPCLIP [Start point X,Start point Y,End point X, End point Y]	
Arguments	Start point X,Y	Start point coordinates for the clipping area (X: 0-399, Y: 0-239)
	End point X,Y	End point coordinates for the clipping area (X: 0-399, Y: 0-239)
Examples	SPCLIP 100,100,200,200	

SPDEF (1)	Resets the sprite character definition template to its initial state
Format	SPDEF
Arguments	None
Common Supplement for SPDEF	- The sprite definition template is a common component for both the upper screen and the Touch Screen - This is provided in order to simplify SPSET definition
Examples	SPDEF

SPDEF (2)	Creates a template for sprite character definition	
Format	SPDEF Definition number, U,V [,W,H [,Origin X,Origin Y]] [,Attribute]	
Arguments	Definition number	Definition number of the template: 0-4095
	U,V	Coordinates of the original image to define (U: 0-511, V: 0-511)
	W,H	Size of the original image to define If omitted: 16,16 * U+W and/or V+H values greater than 512 will give an error.
	Origin X,Y	Reference point for the coordinates of the sprite If omitted: 0,0
	Attribute	|b00| Display (0=OFF, 1=ON) #SPSHOW |b01| ↑Rotation by 90 degrees (Specified with two bits: b01 and b02) |b02| ↓#SPROT0, #SPROT90, #SPROT0180, #SPROT270 |b03| Horizontal inversion (0=OFF, 1=ON), #SPREVH |b04| Vertical inversion (0=OFF, 1=ON), #SPREVV |b05| Additive synthesis (0=OFF, 1=ON), #SPADD If omitted, 0x01 (Only display is set to ON)
Examples	SPDEF 0,192,352,32,32,16,16,1	

SPDEF (3)	Creates templates for sprite character definition collectively from an array	
Format	SPDEF Numerical value array	
Arguments	Numerical value array	Numerical value array containing sprite template data - One sprite template consists of the following 7 elements: U,V,W,H,Origin X,Origin Y,Attribute - The number of elements must be a multiple of 7 - A specific number of sprite templates (the number of elements divided by 7) will be defined in order, starting with 0
Examples	SPDEF SRCDATA	

SPDEF (4)	Creates templates for sprite character definition collectively from a DATA sequence	
Format	SPDEF "@Label string"	
Arguments	@Label string	Label of the DATA instruction that enumerates the sprite template data - The @Label name should be enclosed in "" or specified with a string variable - The first data should be the number of sprites to define, followed by enumeration of the data for each sprite (7 data elements per sprite) - One sprite template consists of the following 7 elements: U,V,W,H,Origin X,Origin Y,Attribute
Examples	SPDEF "@SRCDATA"	

SPDEF (5)	Gets information on a sprite character definition template	
Format	SPDEF Definition number OUT U,V [,W,H [,HX,HY]] [,A]	
Arguments	Definition number	Definition number of the template: 0-4095
Return Values	U,V	Variables to receive the image coordinates
	W,H	Variable to receive the image size
	HX,HY	Variable to receive the reference point for the sprite coordinates
	A	Variable to receive the attribute
Examples	SPDEF 2 OUT U,V,ATR	

SPDEF (6)	Copies a template for sprite character definition - Unnecessary elements can be omitted (Separating commas (',') are required) - Arguments are used to adjust the copied template	
Format	SPDEF Definition number,Source definition number,[U],[V],[W],[H],[Origin X],[Origin Y],[Attribute]	
Arguments	Definition number	Definition number of the template: 0-4095
	Source definition number	Definition number of the copy source: 0-4095
	U,V	Coordinates of the original image to define (U: 0-511, V: 0-511)
	W,H	Size of the original image to define If omitted: 16,16 * U+W and/or V+H values greater than 512 will give an error.
	Origin X,Y	Reference point for the coordinates of the sprite If omitted: 0,0
	Attribute	|b00| Display (0=OFF, 1=ON) #SPSHOW |b01| ↑Rotation by 90 degrees (Specified with two bits: b01 and b02) |b02| ↓#SPROT0, #SPROT90, #SPROT0180, #SPROT270 |b03| Horizontal inversion (0=OFF, 1=ON), #SPREVH |b04| Vertical inversion (0=OFF, 1=ON), #SPREVV |b05| Additive synthesis (0=OFF, 1=ON), #SPADD If omitted, 0x01 (Only display is set to ON)
Examples	SPDEF 0,255,192,352,32,32,16,16,1 SPDEF 1,255,,,32,32,,,	

SPSET (1)	Creates a sprite (using a definition template) - SPSET makes a sprite available for use - Executing SPSET will initialize rotation and all other information - All values of SPVAR will be 0 - When any SPHIT instruction for collision detection is to be used, SPCOL should be called after SPSET	
Format	SPSET Management number,Definition number	
Arguments	Management number	Number of the sprite to create: 0-511
	Definition number	Definition number of the template defined with SPDEF: 0-4095
Examples	SPSET 1,500	

SPSET (2)	Creates a sprite (using image and other information specified directly) Can be used to set a sprite separately without using the values from SPDEF - SPSET makes a sprite available for use - Executing SPSET will initialize rotation and all other information - All values of SPVAR will be 0 - When any SPHIT instruction for collision detection is to be used, SPCOL should be called after SPSET	
Format	SPSET Management number ,U,V [,W,H] ,Attribute	
Arguments	Management number	Number of the sprite to create: 0-511
	U,V	Coordinates of the original image to define (U: 0-511, V: 0-511)
	W,H	Size of the original image to define (If omitted: 16,16) * U+W and/or V+H values greater than 512 will give an error.
	Attribute	|b00| Display (0=OFF, 1=ON) #SPSHOW |b01| ↑Rotation by 90 degrees (Specified with two bits: b01 and b02) |b02| ↓#SPROT0, #SPROT90, #SPROT0180, #SPROT270 |b03| Horizontal inversion (0=OFF, 1=ON), #SPREVH |b04| Vertical inversion (0=OFF, 1=ON), #SPREVV |b05| Additive synthesis (0=OFF, 1=ON), #SPADD If omitted, 0x01 (Only display is set to ON)
Examples	SPSET 54,0,0,32,32,1	

SPSET (3)	Finds an available sprite number and creates a sprite (using a definition template) Finds an available sprite number from the whole range - SPSET makes a sprite available for use - Executing SPSET will initialize rotation and all other information - All values of SPVAR will be 0 - When any SPHIT instruction for collision detection is to be used, SPCOL should be called after SPSET	
Format	SPSET Definition number OUT IX	
Arguments	Definition number	Definition number of the template defined with SPDEF: 0-4095
Return Values	IX	Variable to receive the generated number: 0-511 (-1 = No available number)
Examples	SPSET 500 OUT IX	

SPSET (4)	Finds an available sprite number and creates a sprite (using image and other information specified directly) Finds an available sprite number from the whole range - SPSET makes a sprite available for use - Executing SPSET will initialize rotation and all other information - All values of SPVAR will be 0 - When any SPHIT instruction for collision detection is to be used, SPCOL should be called after SPSET	
Format	SPSET U,V,W,H,Attribute OUT IX	
Arguments	U,V	Coordinates of the original image to define (U: 0-511, V: 0-511)
	W,H	Size of the original image to define (If omitted: 16,16) * U+W and/or V+H values greater than 512 will give an error.
	Attribute	|b00| Display (0=OFF, 1=ON) #SPSHOW |b01| ↑Rotation by 90 degrees (Specified with two bits: b01 and b02) |b02| ↓#SPROT0, #SPROT90, #SPROT0180, #SPROT270 |b03| Horizontal inversion (0=OFF, 1=ON), #SPREVH |b04| Vertical inversion (0=OFF, 1=ON), #SPREVV |b05| Additive synthesis (0=OFF, 1=ON), #SPADD
Return Values	IX	Variable to receive the generated number: 0-511 (-1 = No available number)
Examples	SPSET 0,0,32,32,1 OUT IX	

SPSET (5)	Finds an available sprite number in a certain range and creates a sprite (using a definition template) Finds an available number in the specified range - SPSET makes a sprite available for use - Executing SPSET will initialize rotation and all other information - All values of SPVAR will be 0 - When any SPHIT instruction for collision detection is to be used, SPCOL should be called after SPSET	
Format	SPSET Upper limit,Lower limit, Definition number OUT IX	
Arguments	Upper limit, Lower limit	Range in which to find an available number (0-511)
	Definition number	Definition number of the template defined with SPDEF: 0-4095
Return Values	IX	Variable to receive the generated number: 0-511 (-1 = No available number)
Examples	SPSET 100,120, 500 OUT IX	

SPSET (6)	Finds an available sprite number in a certain range and creates a sprite (using image and other information specified directly) Finds an available number in the specified range - SPSET makes a sprite available for use - Executing SPSET will initialize rotation and all other information - All values of SPVAR will be 0 - When any SPHIT instruction for collision detection is to be used, SPCOL should be called after SPSET	
Format	SPSET Upper limit,Lower limit, U,V,W,H,Attribute OUT IX	
Arguments	Upper limit,Lower limit	Range in which to find an available number (0-511)
	U,V	Coordinates of the original image to define (U: 0-511, V: 0-511)
	W,H	Size of the original image to define (If omitted: 16,16) * U+W and/or V+H values greater than 512 will give an error.
	Attribute	|b00| Display (0=OFF, 1=ON) #SPSHOW |b01| ↑Rotation by 90 degrees (Specified with two bits: b01 and b02) |b02| ↓#SPROT0, #SPROT90, #SPROT0180, #SPROT270 |b03| Horizontal inversion (0=OFF, 1=ON), #SPREVH |b04| Vertical inversion (0=OFF, 1=ON), #SPREVV |b05| Additive synthesis (0=OFF, 1=ON), #SPADD If omitted, 0x01 (Only display is set to ON)
Return Values	IX	Variable to receive the generated number: 0-511 (-1 = No available number)
Examples	SPSET 100,120, 0,0,32,32,1 OUT IX	

SPCLR	Stops using the specified sprite and releases the memory If memory is not released after use with sprites, there will be no available memory for SPSET	
Format	SPCLR Management number	
Arguments	Management number	Management number of the sprite to stop using: 0-511
Examples	SPCLR 56	

SPSHOW	Starts displaying a sprite If used before SPSET, an error will occur	
Format	SPSHOW Management number	
Arguments	Management number	Management number of the sprite to display: 0-511
Examples	SPSHOW 43	

SPHIDE	Hides a sprite - This only hides the sprite; it continues to exist - If used before SPSET, an error will occur	
Format	SPHIDE Management number	
Arguments	Management number	Management number of the sprite to hide: 0-511
Examples	SPHIDE 43	

SPHOME (1)	Specifies the reference point (home position) for the coordinates of a sprite - Position reference point for the SPOFS instruction - Center point for rotation and scaling - Center coordinates for collision detection - If used before SPSET, an error will occur	
Format	SPHOME Management number,Position X,Position Y	
Arguments	Management number	Management number of the sprite for which to set the reference point: 0-511
	Position X,Y	Relative coordinates with the top left corner of the sprite as the origin (0,0)
Examples	SPHOME 34,16,16	

SPHOME (2)	Gets the reference point (home position) for the coordinates of a sprite If used before SPSET, an error will occur	
Format	SPHOME Management number OUT HX,HY	
Arguments	Management number	Management number of the sprite: 0-511
Return Values	HX,HY	Variable to receive the coordinates of the reference point
Examples	SPHOME 10 OUT HX,HY	

SPOFS (1)	Changes (moves) the coordinates of a sprite If used before SPSET, an error will occur	
Format	SPOFS Management number, X, Y [,Z]	
Arguments	Management number	Management number of the target sprite: 0-511
	X,Y	Screen coordinates where the sprite will be displayed
	Z	Coordinate in the depth direction (Rear:1024 < Screen surface:0 < Front:-256)
Examples	SPOFS 23,50,80 SPOFS 23,,,1000 SPOFS 23,150,180,0	

SPOFS (2)	Gets the coordinates of a sprite If used before SPSET, an error will occur	
Format	SPOFS Management number OUT X,Y[,Z]	
Arguments	Management number	Management number of the target sprite: 0-511
Return Values	X,Y	Variable to receive the coordinates
	Z	Variable to receive the depth information
Examples	SPOFS 12 OUT X,Y,Z	

SPROT (1)	Specifies the rotation angle of a sprite If used before SPSET, an error will occur	
Format	SPROT Management number,Angle	
Arguments	Management number	Management number of the target sprite: 0-511
	Angle	Rotation angle: 0-360 (clockwise)
Examples	SPROT 23,45	

SPROT (2)	Gets the rotation angle of a sprite If used before SPSET, an error will occur	
Format	SPROT Management number OUT DR	
Arguments	Management number	Management number of the target sprite: 0-511
Return Values	DR	Variable to receive the angle
Examples	SPROT 23 OUT DR	

SPROT (3)	Gets the rotation angle of a sprite (Function type) If used before SPSET, an error will occur	
Format	Variable=SPROT(Management number)	
Arguments	Management number	Management number of the target sprite: 0-511
Return Values	Current angle (0-360)	
Examples	A=SPROT(23)	

SPSCALE (1)		Changes the scale (display magnification) of a sprite - For collision detection that takes scale into account, SPCOL should first be executed If used before SPSET, an error will occur
Format		SPSCALE Management number, Magnification X, Magnification Y
Arguments	Management number	Management number of the target sprite: 0-511
	Magnification X,Y	0.5 (50%) - 1.0 (100%) - 2.0 (200%) -
Examples		SPSCALE 56, 0.75, 0.75

SPSCALE (2)		Gets the display magnification of a sprite If used before SPSET, an error will occur
Format		SPSCALE Management number OUT SX,SY
Arguments	Management number	Management number of the target sprite: 0-511
Return Values	SX,SY	Variable to receive the magnification
Examples		SPSCALE 45 OUT SX,SY

SPCOLOR (1)		Sets the display color of a sprite If used before SPSET, an error will occur
Format		SPCOLOR Management number, Color code
Arguments	Management number	Management number of the target sprite: 0-511
	Color code	32-bit color code in the ARGB=8888 format - The lower the value of A, the higher the transparency level - The actual display color will be the color code multiplied by the original pixel color
Examples		SPCOLOR 1,RGB(16, 255,0,0) 'A=16,R=255,G=0,B=0

SPCOLOR (2)		Gets the display color of a sprite If used before SPSET, an error will occur
Format		SPCOLOR Management number OUT C32
Arguments	Management number	Management number of the target sprite: 0-511
Return Values	C32	Variable that returns the current color code (32-bit ARGB)
Examples		SPCOLOR 1 OUT C

SPCHR (1)		Changes the character definition of a sprite (using the specified template) If used before SPSET, an error will occur
Format		SPCHR Management number, Definition number
Arguments	Management number	Management number of the sprite to change the definition of: 0-511
	Definition number	Definition number of the template registered using the SPDEF instruction: 0-4095
Examples		SPCHR 0,500

SPCHR (2)		Changes the character definition of a sprite (using a definition specified directly) - Arguments other than the management number can be omitted - If used before SPSET, an error will occur
Format		SPCHR Management number,[U],[V],[W],[H],[Attribute]
Arguments	Management number	Management number of the target sprite: 0-511
	U,V	Coordinates of the original image to define (U: 0-511, V: 0-511)
	W,H	Size of the original image to define (If omitted: 16,16) * U+W and/or V+H values bigger than 512 will give an error
	Attribute	|b00| Display (0=OFF, 1=ON) #SPSHOW |b01| ↑Rotation by 90 degrees (Specified with two bits: b01 and b02) |b02| ↓#SPROT0, #SPROT90, #SPROT0180, #SPROT270 |b03| Horizontal inversion (0=OFF, 1=ON), #SPREVH |b04| Vertical inversion (0=OFF, 1=ON), #SPREVV |b05| Additive synthesis (0=OFF, 1=ON), #SPADD If omitted, 0x01 (Only display is set to ON)
Examples		SPCHR 5,64,64,16,16,1 SPCHR 6,,,32,32,1 'UV skip

SPCHR (3)		Gets information on the character definition of a sprite If used before SPSET, an error will occur
Format		SPCHR Management number OUT U,V [,W,H [,A]]
Arguments	Management number	Management number of the target sprite: 0-511
Return Values	U,V	Variables to store the coordinates of the original image
	W,H	Variables to store the size of the original image
	A	Variables to store the attribute
Examples		SPCHR 5 OUT U,V,W,H,ATR

SPCHR (4)		Gets the character definition number of a sprite If used before SPSET, an error will occur
Format		SPCHR Management number OUT DEFNO
Arguments	Management number	Management number of the target sprite: 0-511
Return Values	DEFNO	Variable to receive the definition number
Examples		SPCHR 5 OUT DEFNO

SPLINK		Links one sprite to another sprite - Only the coordinates will be linked (The rotation angle and magnification information will not) - Only a sprite with a lower management number can be specified as the link destination (parent) - The display coordinates of the child will be determined in relation to the parent - In this coordinate system, the top left corner of the screen will not be the origin - There are no restrictions on link hierarchies - If used before SPSET, an error will occur
Format		SPLINK Management number, Link destination management number
Arguments	Management number	Management number of the link source (child) sprite: 0-511
	Link destination management number	Management number of the link destination (parent) sprite: 0-511 * Management numbers that are not lower than that of the link source will give errors.
Examples		SPLINK 15,4

SPUNLINK		Unlinks a sprite If used before SPSET, an error will occur
Format		SPUNLINK Management number
Arguments	Management number	Management number of the sprite to unlink: 0-511
Examples		SPUNLINK 15

SPANIM (1)		Displays animation with a sprite (using animation data specified with an array) If used before SPSET, an error will occur - Animation waits for a specified time, according to the value input - Animation starts from the frame following SPANIM execution - Up to 32 pieces of data will be accepted for each target element - If a negative value is specified for time, linear interpolation from the previous value will occur
Format		SPANIM Management number,"Animation target",Data array [,Loop]
Arguments	Management number	Management number of the sprite for which to set the animation: 0-511
	Animation target	Numerical value or character string to control the elements that should change - 0 or "XY": XY-coordinates - 1 or "Z": Z-coordinates - 2 or "UV": UV-coordinates (Coordinates of the definition source image) - 3 or "I": Definition number - 4 or "R": Rotation angle - 5 or "S": Magnification XY - 6 or "C": Display color - 7 or "V": Variable (Value of sprite internal variable 7) - Adding 8 to the target numerical value will cause the value to be treated as being relative to the run time - Suffixing the character string with "+" will also cause the value to be treated as being relative to the run time
	Data array	One-dimensional numerical value array storing the animation data
	Loop	Loop count: (1-) The value 0 specifies an endless loop
Data Arrays		Animation data should be provided in a numerical value array in the following order (Up to 32 pieces of data): Time 1, Item 1,[Item2,] Time 2,Item 1,[Item 2,]…

Examples		DIM PANIM[6]
		PANIM[0] = -60 'frame(-60=smooth)
		PANIM[1] = 200 'offset X,Y
		PANIM[2] = 100
		PANIM[3] = -30 'frame
		PANIM[4] = 50 'offset
		PANIM[5] = 20
		SPSET 0,0
		SPANIM 0,"XY",PANIM

SPANIM (2)		Displays animation with a sprite (using animation data specified with the DATA instruction) If used before SPSET, an error will occur - Animation waits for a specified time, according to the value input - Animation starts from the frame following SPANIM execution - Up to 32 pieces of data will be accepted for each target element - If a negative value is specified for time, linear interpolation from the previous value will occur
Format		SPANIM Management number,"Animation target","@Label string" [,Loop]
Arguments	Management number	Management number of the sprite for which to set the animation: 0-511
	Animation target	Numerical value or character string to control the elements that should change - 0 or "XY": XY-coordinates - 1 or "Z": Z-coordinates - 2 or "UV": UV-coordinates (Coordinates of the definition source image) - 3 or "I": Definition number - 4 or "R": Rotation angle - 5 or "S": Magnification XY - 6 or "C": Display color - 7 or "V": Variable (Value of sprite internal variable 7) - Adding 8 to the target numerical value will cause the value to be treated as being relative to the run time - Suffixing the character string with "+" will also cause the value to be treated as being relative to the run time
	@Label string	- First label of the DATA instruction storing the animation data - This should be specified as a character string by enclosing the @Label name in "" (or as a string variable)
	Loop	Loop count: (1-) The value 0 specifies an endless loop
Data		Animation data should be provided in the DATA instruction in the following order: DATA Number of key frames (maximum: 32) DATA Time 1,Item 1[,Item 2] DATA Time 2,Item 1[,Item 2] :
Examples		@MOVDATA DATA 2 'counter DATA -60,200,100 'frame,offset DATA -30,50,20 'frame,offset SPSET 0,0 SPANIM 0,"XY","@MOVDATA"

SPANIM (3)		Displays animation with a sprite (using animation data specified directly as arguments) If used before SPSET, an error will occur - Animation waits for a specified time, according to the value input - Animation starts from the frame following SPANIM execution - Up to 32 pieces of data will be accepted for each target element - If a negative value is specified for time, linear interpolation from the previous value will occur
Format		SPANIM Management number,"Animation target",Time 1,Item 1[,Item 2] [,Time 2,Item 1[,Item 2]]… [,Loop]
Arguments	Management number	Management number of the sprite for which to set the animation: 0-511
	Animation target	Numerical value or character string to control the elements that should change - 0 or "XY": XY-coordinates - 1 or "Z": Z-coordinates - 2 or "UV": UV-coordinates (Coordinates of the definition source image) - 3 or "I": Definition number - 4 or "R": Rotation angle - 5 or "S": Magnification XY - 6 or "C": Display color - 7 or "V": Variable (Value of sprite internal variable 7) - Adding 8 to the target numerical value will cause the value to be treated as being relative to the run time - Suffixing the character string with "+" will also cause the value to be treated as being relative to the run time
	Time,Item	- Animation data itself (As many Time/Item sets as needed should be listed. Maximum: 32)
	Loop	Loop count: (1-) The value 0 specifies an endless loop
Examples		SPSET 0,0 SPANIM 0,"XY", -60,200,100, -30,50,20

SPSTOP		Stops animation of a sprite If used before SPSET, an error will occur
Format		SPSTOP [Management number]
Arguments	Management number	Management number of the target sprite: 0-511 * If the management number is omitted, animation of all sprites will be stopped.
Examples		SPSTOP

SPSTART	Starts animation of a sprite (If used before SPSET, an error will occur)	
Format	SPSTART [Management number]	
Arguments	Management number	Management number of the target sprite: 0-511 * If the management number is omitted, animation of all sprites will be started.
Examples	SPSTART	

SPCHK	Gets the animation status of a sprite If used before SPSET, an error will occur	
Format	Variable=SPCHK(Management number)	
Arguments	Management number	Management number of the target sprite: 0-511
Return Values	|b00| XY-coordinates (1), #CHKXY |b01| Z-coordinates (2), #CHKZ |b02| UV-coordinates (4), #CHKUV |b03| Definition number (8), #CHKI |b04| Rotation (16), #CHKR |b05| Magnification XY (32), #CHKS |b06| Display color (64), #CHKC |b07| Variable (128), #CHKV For each bit, a target is assigned (If 0 is assigned for all bits, animation is being stopped)	
Examples	ST=SPCHK(5) '|b00|#CHKXY '|b01|#CHKZ '|b02|#CHKUV '|b03|#CHKI '|b04|#CHKR '|b05|#CHKS '|b06|#CHKC '|b07|#CHKV	

SPVAR (1)	Writes to a sprite internal variable - Sprite internal variables (Each sprite has eight variables that the user can use) - Can also be used before SPSET (When SPSET is executed, all eight variables will be 0)	
Format	SPVAR Management number,Internal variable number,Numerical data	
Arguments	Management number	Management number of the target sprite: 0-511
	Internal variable number	Number of the internal variable: 0-7
	Numerical value	Numerical value to register with the internal variable (0-
Examples	SPVAR 0,7,1	

SPVAR (2)	Reads a sprite internal variable (Function type) - Sprite internal variables (Each sprite has eight variables that the user can use) - Can also be used before SPSET (When SPSET is executed, all eight variables will be 0)	
Format	Variable=SPVAR(Management number,Internal variable number)	
Arguments	Management number	Management number of the target sprite: 0-511
	Internal variable number	Number of the internal variable: 0-7
Return Values	Value written with SPVAR	
Examples	V=SPVAR(54,0)	

SPVAR (3)	Reads a sprite internal variable - Sprite internal variables (Each sprite has eight variables that the user can use) - Can also be used before SPSET (When SPSET is executed, all eight variables will be 0)	
Format	SPVAR Management number,Internal variable number OUT V	
Arguments	Management number	Management number of the target sprite: 0-511
	Internal variable number	Number of the internal variable: 0-7
Return Values	V	Numerical value variable that returns the value of the internal variable
Examples	SPVAR 54,0 OUT V	

SPCOL (1)	Sets sprite collision detection information - Must be called before any SPHIT instruction is used - SPCOLVEC should also be called - If used before SPSET, an error will occur	
Format	SPCOL Management number [,Scale adjustment]	
Arguments	Management number	Management number of the target sprite: 0-511
	Scale adjustment	FALSE = Ignores this specification (If omitted = FALSE) TRUE = Synchronizes the detection area with SPSCALE * This specification will be effective for SPSCALE instructions set after the SPCOL instruction.
Examples	SPCOL 3,TRUE	

SPCOL (2)	Sets sprite collision detection information (including mask specification) - Must be called before any SPHIT instruction is used - SPCOLVEC should also be called - If used before SPSET, an error will occur	
Format	SPCOL Management number,[Scale adjustment],Mask	
Arguments	Management number	Management number of the target sprite: 0-511
	Scale adjustment	FALSE = Ignores this specification (If omitted = FALSE) TRUE = Synchronizes the detection area with SPSCALE * This specification will be effective for SPSCALE instructions set after the SPCOL instruction.
	Mask	0 - &HFFFFFFFF (32 bits) * For collision detection, the AND of the bits is determined, and if it is 0, it is regarded as no collision (If omitted, &HFFFFFFFF).
Examples	SPCOL 3,TRUE,31 SPCOL 3,,31	

SPCOL (3)	Sets sprite collision detection information (including range specification) - Must be called before any SPHIT instruction is used - SPCOLVEC should also be called - If used before SPSET, an error will occur	
Format	SPCOL Management number,Start point X,Start point Y,Width,Height,[Scale adjustment],Mask	
Arguments	Management number	Management number of the target sprite: 0-511
	Start point X,Y	- Start point coordinates of the detection area: X,Y (-32768 to 32767) - Relative coordinates with SPHOME as the origin (0,0)
	Width,Height	Width and height of the detection area: W,H (0-65535)
	Scale adjustment	FALSE = Ignores this specification (If omitted = FALSE) TRUE = Synchronizes the detection area with SPSCALE * This specification will be effective for SPSCALE instructions set after the SPCOL instruction.
	Mask	0 - &HFFFFFFFF (32 bits) * For collision detection, the AND of the bits is determined, and if it is 0, it is regarded as no collision (If omitted, &HFFFFFFFF).
Examples	SPCOL 3,0,0,32,32,TRUE,255 SPCOL 3,0,0,32,32,,255	

SPCOL (4)	Gets sprite collision detection information (scale adjustment and mask) If used before SPSET, an error will occur	
Format	SPCOL Management number OUT Scale adjustment [,Mask]	
Arguments	Management number	Management number of the target sprite: 0-511
Return Values	Scale adjustment	Variable to receive the scale value
	Mask	Variable to receive the mask value
Examples	SPCOL 3 OUT SC,MSK	

SPCOL (5)	Gets sprite collision detection information (range) If used before SPSET, an error will occur	
Format	SPCOL Management number OUT Start point X,Start point Y,Width,Height	
Arguments	Management number	Management number of the target sprite: 0-511
Return Values	Start point X,Y	Variables to receive the start point coordinates of the detection area
	Width,Height	Variables to receive the width and height of the detection area
Examples	SPCOL 3 OUT X,Y,W,H	

SPCOL (6)	Gets sprite collision detection information (range and scale adjustment) If used before SPSET, an error will occur	
Format	SPCOL Management number OUT Start point X,Start point Y,Width,Height,Scale adjustment	
Arguments	Management number	Management number of the target sprite: 0-511
Return Values	Start point X,Y	Variables to receive the start point coordinates of the detection area
	Width,Height	Variables to receive the width and height of the detection area
	Scale adjustment	Variable to receive the scale value
Examples	SPCOL 3 OUT X,Y,W,H,SC	

SPCOL (7)	Gets sprite collision detection information (all information) If used before SPSET, an error will occur	
Format	SPCOL Management number OUT Start point X,Start point Y,Width,Height,Scale adjustment,Mask	
Arguments	Management number	Management number of the target sprite: 0-511
Return Values	Start point X,Y	Variables to receive the start point coordinates of the detection area
	Width,Height	Variables to receive the width and height of the detection area
	Scale adjustment	Variable to receive the scale value
	Mask	Variable to receive the mask value
Examples	SPCOL 3 OUT X,Y,W,H,SC,MSK	

SPCOLVEC	Sets a movement speed for sprite collision detection - It is recommended to also call this instruction when setting SPCOL - If used before SPSET, an error will occur	
Format	SPCOLVEC Management number [,Movement amount X,Movement amount Y]	
Arguments	Management number	Management number of the target sprite: 0-511
	Movement amount X,Movement amount Y	- If omitted, the amount will be automatically calculated as follows: - When linear interpolation of "XY" in SPANIM is being performed: Movement distance from the previous frame - Otherwise: 0,0
Examples	SPCOLVEC 93	

SPHITSP (1)	Detects sprite collision - SPCOL and SPCOLVEC should be called beforehand - If used before SPSET, an error will occur	
Format	Variable = SPHITSP(Management number [,First ID,Last ID])	
Arguments	Management number	Management number of the sprite to detect collision with: 0-511
	First ID,Last ID	Range of sprites to detect (0-511)
Return Values	Management number of the colliding sprite (When no collision, -1)	
Examples	H=SPHITSP(0)	

SPHITSP (2)	Detects sprite collision: collision between the specified sprites - SPCOL and SPCOLVEC should be called beforehand - If used before SPSET, an error will occur	
Format	Variable = SPHITSP(Management number ,Opponent management number)	
Arguments	Management number	Management number of the sprite to detect collision with: 0-511
	Opponent management number	Management number of the opponent sprite: 0-511
Return Values	FALSE = No collision, TRUE = Collision	
Examples	H=SPHITSP(0,34)	

SPHITRC (1)	Detects collision between a moving quadrangle and any sprite - SPCOL and SPCOLVEC should be called beforehand - If used before SPSET, an error will occur	
Format	SPHITRC(Start point X,Start point Y,Width,Height[,[Mask],Movement amount X,Movement amount Y])	
Arguments	Start point X,Y	Top left coordinates of the quadrangle to detect collision with
	Width,Height	Width and height of the quadrangle to detect collision with
	Mask	0 - &HFFFFFFFF (32 bits) * For collision detection, the AND of the bits is determined, and if it is 0, it is regarded as no collision (If omitted, &HFFFFFFFF).
	Movement amount X,Y	Movement amount of the quadrangle to detect collision with
Return Values	Management number of the colliding sprite (When no collision, -1)	
Examples	H=SPHITRC(0,0,16,16)	

SPHITRC (2)	Detects collision between the specified sprite and a quadrangle - SPCOL and SPCOLVEC should be called beforehand - If used before SPSET, an error will occur	
Format	SPHITRC(Management number,Start point X,Start point Y,Width,Height[,[Mask],Movement amount X,Movement amount Y])	
Arguments	Management number	Management number of the collision opponent sprite: 0-511
	Start point X,Y	Top left coordinates of the quadrangle to detect collision with
	Width,Height	Width and height of the quadrangle to detect collision with
	Mask	0 - &HFFFFFFFF (32 bits) * For collision detection, the AND of the bits is determined, and if it is 0, it is regarded as no collision (If omitted, &HFFFFFFFF).
	Movement amount X,Y	Movement amount of the quadrangle to detect collision with
Return Values	FALSE = No collision, TRUE = Collision	
Examples	H=SPHITRC(1,0,0,16,16)	

SPHITRC (3)	Detects collision between the specified range of sprites and a quadrangle - SPCOL and SPCOLVEC should be called beforehand - If used before SPSET, an error will occur	
Format	SPHITRC(First ID,Last ID, Start point x,Start point y,Width,Height[,[Mask],Movement amount X, Movement amount Y])	
Arguments	First ID,Last ID	Range of sprites to detect (0-511)
	Start point X,Y	Top left coordinates of the quadrangle to detect collision with
	Width,Height	Width and height of the quadrangle to detect collision with
	Mask	0 - &HFFFFFFFF (32 bits) * For collision detection, the AND of the bits is determined, and if it is 0, it is regarded as no collision (If omitted, &HFFFFFFFF).
	Movement amount X,Y	Movement amount of the quadrangle to detect collision with
Return Values	Management number of the colliding sprite (When no collision, -1)	
Examples	H=SPHITRC(0,0,16,16)	

SPHITINFO (1)	Gets information on collision detection results (Time of collision) If used before SPSET, an error will occur	
Format	SPHITINFO OUT TM	
Arguments	None	
Return Values	TM	- Variable that returns time of collision: real-type number from 0 to 1 - Position at collision detection + speed x collision time = collision X-Y coordinates
Examples	SPHITINFO OUT TM	

SPHITINFO (2)	Gets information on collision detection results (Time of collision and coordinates) If used before SPSET, an error will occur	
Format	SPHITINFO OUT TM,X1,Y1,X2,Y2	
Arguments	None	
Return Values	TM	- Variable that returns time of collision: real-type number from 0 to 1 - Position at collision detection + speed x collision time = collision X-Y coordinates
	X1,Y1	Variable that returns the X-Y coordinates of object 1 at the time of collision
	X2,Y2	Variable that returns the X-Y coordinates of object 2 at the time of collision
Examples	SPHITINFO OUT TM,X1,Y1,X2,Y2	

SPHITINFO (3)	Gets information on collision detection results (Time of collision, coordinates and speed) If used before SPSET, an error will occur	
Format	SPHITINFO OUT TM,X1,Y1,VX1,VY1,X2,Y2,VX2,VY2	
Arguments	None	
Return Values	Time of collision	- Variable that returns time of collision: real-type number from 0 to 1 - Position at collision detection + speed x collision time = collision X-Y coordinates
	X1,Y1	Variable that returns the X-Y coordinates of object 1 at the time of collision
	VX1,VY1	Variable that returns the speed of object 1 at the time of collision
	X2,Y2	Variable that returns the X-Y coordinates of object 2 at the time of collision
	VX2,VY2	Variable that returns the speed of object 2 at the time of collision
Examples	SPHITINFO OUT TM,X1,Y1,VX1,VY1,X2,Y2,VX2,VY2	

SPFUNC	Assigns a process to a sprite - An instruction for advanced users that is used when callback processing is required - All sprite processes are executed with CALL sprite - Instead of @Label, a user process defined using DEF can also be specified - At the processing target, the management number can be obtained using the CALLIDX system variable - If used before SPSET, an error will occur	
Format	SPFUNC Management number, @Label	
Arguments	Management number	Management number of the target sprite: 0-511
	@Label	Label of the processing target (or a user-defined process) to be called
Examples	SPFUNC 0,@PROG	

SPUSED	Checks if the specified sprite is in use	
Format	Variable=SPUSED(Management number)	
Arguments	Management number	Management number of the target sprite: 0-511
Return Values	TRUE = In use, FALSE = Available	
Examples	S=SPUSED(4)	

BG

Functions for displaying tiled 16x16-pixel rectangular images

BGPAGE (1)	Sets a graphic page to assign to BG	
Format	BGPAGE Graphic page	
Arguments	Graphic page	0-5 (GRP0-GRP5) By default, the graphic page for BG is 5 (GRP5)
Examples	BGPAGE 5	

BGPAGE (2)	Gets the graphic page that has been assigned to BG	
Format	Variable=BGPAGE()	
Return Values	Graphic page number (0-5)	
Examples	P=BGPAGE()	

BGSCREEN	Sets the BG screen size per layer	
Format	BGSCREEN Layer,Width,Height	
Arguments	Layer	Target layer number: 0-3
	Width,Height	- Width and height in character units (Width x Height should be equal to or less than 16383) - Initial state: 25 x 15 (Right size to fill the upper screen with BG)
Examples	BGSCREEN 0,128,127	

BGCLR	Clears the BG screen	
Format	BGCLR [Layer]	
Arguments	Layer	Target layer number: 0-3 (If omitted, all layers)
Examples	BGCLR	

BGSHOW	Shows the BG screen	
Format	BGSHOW Layer	
Arguments	Layer	Target layer number: 0-3
Examples	BGSHOW 0	

BGHIDE	Hides the BG screen	
Format	BGHIDE Layer	
Arguments	Layer	Target layer number: 0-3
Examples	BGHIDE 0	

BGCLIP	Specifies the display area of the BG screen	
Format	BGCLIP Layer [,Starting point X,Starting point Y,End point X,End point Y]	
Arguments	Layer	Target layer number: 0-3
	Start point X,Y	Start point coordinates (in pixels) of the display area
	End point X,Y	- End point coordinates (in pixels) of the display area - If the start and end points are omitted, the whole layer will be the display area
Examples	BGCLIP 0,20,20,379,219	

BGHOME (1)	Sets the display origin of a layer - Origin for rotation and scaling of the BG screen	
Format	BGHOME Layer,Position X,Position Y	
Arguments	Layer	Target layer number: 0-3
	Position X,Y	Origin coordinates in pixel units
Examples	BGHOME 0,200,120	

BGHOME (2)	Gets the display origin of a layer	
Format	BGHOME Layer OUT HX,HY	
Arguments	Layer	Target origin number: 0-3
Return Values	HX,HY	Variables to receive the coordinates of the reference point
Examples	BGHOME 0 OUT HX,HY	

BGOFS (1)	Changes the display offset of the BG screen	
Format	BGOFS Layer,X,Y,[Z]	
Arguments	Layer	Target layer number: 0-3
	X,Y	Display offset coordinates in pixels
	Z	Coordinate in the depth direction (Rear:1024 < Screen surface:0 < Front:-256)
Examples	BGOFS 0,-100,-100	

BGOFS (2)	Gets BG coordinates	
Format	BGOFS Layer OUT X,Y[,Z]	
Arguments	Layer	Target layer number: 0-3
Return Values	X,Y	Variables to receive the coordinates
	Z	Variable to receive the depth information
Examples	BGOFS 0 OUT X,Y,Z	

BGROT (1)	Rotates the BG screen	
Format	BGROT Layer,Angle	
Arguments	Layer	Target layer number: 0-3
	Angle	Rotation angle (clockwise): 0-360
Examples	BGROT 0,180	

BGROT (2)	Gets rotation information from the BG screen	
Format	BGROT Layer OUT R	
Arguments	Layer	Target layer number: 0-3
Return Values	Angle	R: 0-360
Examples	BGROT 0 OUT R	

BGSCALE (1)	Scales the BG screen - When scaled down, BGs exceeding 3600 in total will not be displayed - If this display limit is exceeded, the BG screen will be distorted	
Format	BGSCALE Layer,Enlargement scale X,Enlargement scale Y	
Arguments	Layer	Target layer number: 0-3
	Enlargement scale X,Y	0.5 (50%) - 1.0 (100%) - 2.0(200%) -
Examples	BGSCALE 0,1.5,2.0	

BGSCALE (2)	Gets scale-up/down information from the BG screen	
Format	BGSCALE Layer OUT SX,SY	
Arguments	Layer	Target layer number: 0-3
Return Values	SX,SY	0.5 (50%) - 1.0 (100%) - 2.0(200%) -
Examples	BGSCALE 0 OUT SX,SY	

BGPUT	Places a BG character on the BG screen No image will be displayed for character number 0	
Format	BGPUT Layer,X,Y,Screen data	
Arguments	Layer	Target layer number: 0-3
	X,Y	Coordinates to place the character at (0 - the value specified with BGSCREEN minus 1)
	Screen Data	|b00| ↑ | | Character number (0-4095, repeated at the cycle of 1024) |b11| ↓ |b12| ↑Rotation by 90 degrees (Specified with two bits: b12 and b13) |b13| ↓[00 = 0 degrees, 01 = 90 degrees, 10 = 180 degrees, 11 = 270 degrees] |b14| Horizontal inversion (0=OFF, 1=ON) |b15| Vertical inversion (0=OFF, 1=ON) - 16-bit numerical value that specifies the character number and the rotation information - A 4-digit hexadecimal string can also be specified ("0000"-"FFFF")
Examples	BGPUT 0,0,0,5 BGPUT 0,20,15,"80FF"	

BGFILL		Fills the BG screen with a BG character
Format		BGFILL Layer,Start point X,Start point Y,End point X,End point Y,Screen data
Arguments	Layer	Target layer number: 0-3
	Start point X,Y	Start point coordinates (Each coordinate: 0 - the value specified with the BGSCREEN instruction minus 1)
	End point X,Y	End point coordinates (Each coordinate: 0 - the value specified with the BGSCREEN instruction minus 1)
	Screen Data	|b00| ↑ | | Character number (0-4095, repeated at the cycle of 1024) |b11| ↓ |b12| ↑Rotation by 90 degrees (Specified with two bits: b12 and b13) |b13| ↓[00 = 0 degrees, 01 = 90 degrees, 10 = 180 degrees, 11 = 270 degrees] |b14| Horizontal inversion (0=OFF, 1=ON) |b15| Vertical inversion (0=OFF, 1=ON) - 16-bit numerical value that specifies the character number and the rotation information - A 4-digit hexadecimal string can also be specified ("0000"-"FFFF")
Examples		BGFILL 0,0,0,19,15,1024 BGFILL 0,5,5,10,10,"C040"

BGGET		Gets information on a BG character on the BG screen
Format		Variable=BGGET(Layer, X, Y [,Coordinate system flag])
Arguments	Layer	Target layer number: 0-3
	X,Y	Coordinates to get the BG character from (Coordinate values differ depending on the coordinate system flag described below)
	Coordinate system flag (If omitted, 0)	0: Treats X-, Y-coordinates as the BG screen coordinates (in character units) 1: Treats X-, Y-coordinates as the screen coordinates (in pixel units)
Return Values		|b00| ↑ | | Character number (0-4095, repeated at cycles of 1024) |b11| ↓ |b12| ↑Rotation by 90 degrees (Specified with two bits: b12 and b13) |b13| ↓#BGROT0, #BGROT90, #BGROT0180, #BGROT270 |b14| Horizontal inversion (0=OFF, 1=ON), #BGREVH |b15| Vertical inversion (0=OFF, 1=ON), #BGREVV Screen data
Examples		C=BGGET(0,12,14)

BGANIM (1)		Displays animation with BG (using animation data specified with an array) - Animation waits for a specified time, according to the value input - Animation starts from the frame following BGANIM - Up to 32 pieces of data will be accepted for each target element - If a negative value is specified for time, linear interpolation from the previous value will occur
Format		BGANIM Layer,"Animation target",Data array [,Loop]
Arguments	Layer	Number of the layer for which to set the animation: 0-3
	Animation target	Numerical value or character string to control the elements that should change - 0 or "XY": XY-coordinates - 1 or "Z": Z-coordinates - 4 or "R": Rotation angle - 5 or "S": Magnification XY - 6 or "C": Display color - 7 or "V": Variable (Value of BG internal variable 7) - Adding 8 to the target numerical value will cause the value to be treated as being relative to the run time - Suffixing the character string with "+" will also cause the value to be treated as being relative to the run time
	Data array	One-dimensional numerical value array storing the animation data
	Loop	Loop count: (1-) The value 0 specifies an endless loop
Data Arrays		Animation data should be provided in a numerical value array in the following order (Up to 32 pieces of data): Time 1, Item 1,[Item2,] Time 2,Item 1,[Item 2,]…

Examples		DIM PANIM[6]
		PANIM[0] = -60 'frame(-60=smooth)
		PANIM[1] = 200 'offset X,Y
		PANIM[2] = 100
		PANIM[3] = -30 'frame
		PANIM[4] = 50 'offset
		PANIM[5] = 20
		BGANIM 0,"XY",PANIM

BGANIM (2)		Displays animation using the BG (Specifying animation data with the DATA instruction) - Animation waits for a specified time, according to the value input - Animation starts from the frame following BGANIM - Up to 32 pieces of data will be accepted for each target element - If a negative value is specified for time, linear interpolation from the previous value will occur
Format		BGANIM Layer,"Animation target","@Label string" [,Loop]
Arguments	Layer	Number of the layer for which to set the animation: 0-3
	Animation target	Numerical value or character string to control the elements that should change - 0 or "XY": XY-coordinates - 1 or "Z": Z-coordinate - 4 or "R": Rotation angle - 5 or "S": Magnification XY - 6 or "C": Display color - 7 or "V": Variable (Value of BG internal variable 7) - Adding 8 to the target numerical value will cause the value to be treated as being relative to the run time - Suffixing the character string with "+" will also cause the value to be treated as being relative to the run time
	@Label string	- First label of the DATA instruction storing the animation data - This should be specified as a character string by enclosing the @Label name in " (or as a string variable)
	Loop	Loop count: (1-) The value 0 specifies an endless loop
Data		Animation data should be provided in the DATA instruction in the following order: DATA Number of key frames (maximum: 32) DATA Time 1,Item 1[,Item 2] DATA Time 2,Item 1[,Item 2] :
Examples		@MOVDATA DATA 2 'counter DATA -60,200,100 'frame,offset DATA -30,50,20 'frame,offset BGANIM 0,"XY","@MOVDATA"

BGANIM (3)	Displays animation using the BG (Specifying animation data with arguments directly) - Animation waits for a specified time, according to the value input - Animation starts from the frame following BGANIM - Up to 32 pieces of data will be accepted for each target element - If a negative value is specified for time, linear interpolation from the previous value will occur	
Format	BGANIM Layer,"Animation target",Time 1,Item 1[,Item 2] [,Time 2,Item 1[,Item 2]]… [,Loop]	
Arguments	Layer	Number of the layer for which to set the animation: 0-3
	Animation target	Numerical value or character string to control the elements that should change - 0 or "XY": XY-coordinates - 1 or "Z": Z-coordinate - 4 or "R": Rotation angle - 5 or "S": Magnification XY - 6 or "C": Display color - 7 or "V": Variable (Value of BG internal variable 7) - Adding 8 to the target numerical value will cause the value to be treated as relative to the run time - Suffixing the character string with "+" will also cause the value to be treated as relative to the run time/td>
	Time, Item	- Animation data itself (Up to 32 necessary data items can be listed)
	Loop	Loop count: (1-) The value 0 specifies an endless loop
Examples	BGANIM 0,"XY", -60,200,100, -30,50,20	

BGSTOP	Stops BG animation	
Format	BGSTOP [Layer]	
Arguments	Layer	Target layer number: 0-3 * If the layer is omitted, animation of all layers will be stopped.
Examples	BGSTOP	

BGSTART	Starts BG animation	
Format	BGSTART [Layer]	
Arguments	Layer	Target layer number: 0-3 * If the layer is omitted, animation of all layers will be started.
Examples	BGSTART	

BGCHK	Gets BG animation status	
Format	Variable=BGCHK(Layer)	
Arguments	Layer	Number of the layer to check: 0-3
Return Values	|b00| XY-coordinates (1), #CHKXY |b01| Z-coordinate (2), #CHKZ |b02| |b03| |b04| Rotation (16), #CHKR |b05| Magnification XY (32), #CHKS |b06| Display color (64), #CHKC |b07| Variable (128), #CHKV A target is assigned for each bit (If 0 is assigned for all bits, animation is being stopped)	
Examples	ST=BGCHK(0) '|b00|#CHKXY '|b01|#CHKZ '|b04|#CHKR '|b05|#CHKS '|b06|#CHKC '|b07|#CHKV	

BGVAR (1)	Writes to a BG internal variable User variables; there are eight variables for each BG layer	
Format	BGVAR Layer,Internal variable number,Numerical value	
Arguments	Layer	Target layer number: 0-3
	Internal variable number	Number of the internal variable: 0-7
	Numerical value	Numerical value to register with the internal variable
Examples	BGVAR 0,7,1	

BGVAR (2)	Reads a BG internal variable (function type) User variables; there are eight variables for each BG layer	
Format	Variable=BGVAR(Layer number,Internal variable number)	
Arguments	Layer	Target layer number: 0-3
	Internal variable number	Number of the internal variable: 0-7
Return Values	Value written with BGVAR	
Examples	V=BGVAR(0,5)	

BGVAR (3)	Reads a BG internal variable - User variables; there are eight variables for each BG layer	
Format	BGVAR Layer,Internal variable number OUT V	
Arguments	Layer	Target layer number: 0-3
	Internal variable number	Number of the internal variable: 0-7
Return Values	V	Numerical value variable that returns the value of the internal variable
Examples	BGVAR 0,5 OUT V	

BGCOPY	Copies from the BG screen in character units	
Format	BGCOPY Layer,Start point X,Start point Y, End point X,End point Y, Transfer destination X,Transfer destination Y	
Arguments	Layer	Target layer number: 0-3
	Start point X,Y End point X,Y	Start and End point coordinates of the copy source (0 - the value specified with BGSCREEN minus 1)
	Transfer destination X,Y	Start point coordinates of the copy destination (0 - the value specified with BGSCREEN minus 1)
Examples	BGCOPY 2,0,0,32,32,0,0	

BGLOAD	Copies BG data from an array to the BG screen	
Format	BGLOAD Layer, [Start point X,Start point Y,Width,Height,] Numerical value array	
Arguments	Layer	Layer number of the copy destination range: 0-3
	Start point X,Start point Y	Start point coordinates (character coordinates) of the copy destination range
	Width, Height	- Width and height (in character units) of the copy destination range - If the range specification is omitted, the whole BG screen will be the display area.
	Numerical value array	Numerical value array containing the BG data stored with BGSAVE
Examples	BGLOAD 0, 0,0,30,10, BGARRAY	

BGSAVE	Copies the contents of the BG screen to a numerical value array	
Format	BGSAVE Layer, [Start point X,Start point Y,Width,Height,] Numerical value array	
Arguments	Layer	Layer number of the copy source: 0-3
	Start point X,Start point Y	Start point coordinates (character coordinates) of the copy source range
	Width, Height	- Width and height (in character units) of the copy source range - If the range specification is omitted, the whole BG screen will be the display area
	Numerical value array	- Numerical value array to which to copy the data - For one-dimensional arrays only, if the array is insufficient, the required element(s) will be added automatically
Examples	DIM BGARRAY[30*10] BGSAVE 0, 0,0,30,10, BGARRAY	

BGCOORD	Converts display coordinates to BG screen coordinates, or vice versa	
Format	BGCOORD Layer,Source X-coordinate,Source Y-coordinate[,Mode]OUT DX,DY	
Arguments	Layer	Layer number: 0-3
	Source X-, Y-coordinates	Coordinates to convert (BG character coordinates or display coordinates)
	Mode	Conversion mode: 0-2 0: Converts BG screen coordinates to display coordinates 1: Converts display coordinates to BG screen coordinates (in character units) 2: Converts display coordinates to BG screen coordinates (in pixel units)
	DX,DY	Variable to store the converted coordinates (BG character coordinates or display coordinates)
Examples	BGCOORD 0,BGX,BGY,0 OUT DX,DY	

BGCOLOR (1)	Sets the BG display color	
Format	BGCOLOR Layer, Color code	
Arguments	Management number	Layer number: 0-3
	Color code	- 32-bit color code in the ARGB=8888 format - The RGB function is useful for this specification: RGB(R,G,B) - Unlike with sprites, the alpha value is not valid (Semitransparent representation is not allowed) - The actual display color will be the color code multiplied by the original pixel color.
Examples	BGCOLOR 1,RGB(255,0,0) 'R=255,G=0,B=0	

BGCOLOR (2)	Gets the BG display color	
Format	BGCOLOR Layer OUT C32	
Arguments	Layer	Layer number: 0-3
Return Values	C32	Variable that returns the current color code (32-bit ARGB)
Examples	BGCOLOR 1 OUT C	

BGFUNC	Assigns a callback process to a BG layer - An instruction for advanced users that is used when callback processing is required - All BG layer processes are executed with CALL BG - Instead of @Label, a user process defined using DEF can also be specified - At the processing target, a management number can be obtained using a CALLIDX system variable	
Format	BGFUNC Layer, @Label	
Arguments	Layer	Layer number: 0-3
	@Label	The label of the process target (or a user-defined process)
Examples	BGFUNC 0,@LAYERSUB0	

Sound

Functions for playing back music and sound effects, setting effectors, and generating synthesized voice

BEEP	Generates a simple alarm sound or sound effect	
Format	BEEP [Sound effect number][,Frequency][,Volume][,Pan pot]	
Arguments	Sound effect number	- Type of sound to generate: Preset sound 0-133 - A list of preset sounds can be viewed by pressing the SMILE button
	Frequency	- Frequency value to change to: -32768 to 32767 (One halftone per 100)
	Volume	- Volume level for playback: 0-127
	Pan pot	- Stereo pan pot specification: 0 (Left) - 64 (Center) - 127 (Right)
Examples	BEEP 20	

BGMCHK	Checks music playback status	
Format	Variable=BGMCHK([Track number])	
Arguments	Track number	Track number: 0-7 (If omitted, 0)
Return Values	FALSE = Stopped, TRUE = Playing	
Examples	RET=BGMCHK(0)	

BGMCLEAR	Clears a user-defined piece of music	
Format	BGMCLEAR [User-defined tune number]	
Arguments	User-defined tune number	Tune number: 128-255 (If omitted, all defined tunes will be cleared)
Examples	BGMCLEAR	

BGMPLAY (1)	Plays music (Plays back registered BGM) - Up to 8 tunes can be played simultaneously (The total maximum number of sounds that can be generated simultaneously is 16) - See the second page for information on how to play music using MML	
Format	BGMPLAY [Track number,] Tune number [,Volume]	
Arguments	Track number	Track number to play back: 0-7 (If omitted, number 0)
	Tune number	- Preset tune: 0-42 - User-defined tune: 128-255 - A list of preset sounds can be viewed by pressing the SMILE button.
	Volume	Volume level for playback: 0-127
Examples	BGMPLAY 0	

BGMPLAY (2)	Plays music (Plays back the input MML data) - MML playback is performed in track 0 - The MML tune will overwrite user-defined tune number 255 - Executing immediately after BGMPLAY will cause a delay of approx. 2 frames	
Format	BGMPLAY "MML string"	
Arguments	MML string	- Pressing the Help button for "MML" will display descriptions of MML commands - You can register a character string to play by listing the following symbols: :0 - :15 Channel specification T1 - T512 Tempo specification CDEFGAB Scale (C# is a halftone higher; C- is a halftone lower) N0 - N127 Key value specification (O4C=60) 1 - 192 Individual tone length specification (C1 = Whole note, C4. = Dotted quarter note) L1 - L192 Default tone length (. should be used for dotted notes) R Rest O0 - O8 Octave number specification < > One octave up or down V0 - V127 Volume level value specification ( ) Volume up or down @0 - @255 Tone change (0 - 127: Equivalent to GM, 224-: User-defined waveform) P0 - P127 Pan pot (Left: P0-63 Center: P64 Right: P65-127) [Repeat start] Number of times Repeat end (If the number of times is omitted, the loop will be endless) & Connects the preceding/succeeding notes _ Portamento
Examples	BGMPLAY "T120O4L4CC8D8EE8F8GA8G8E2"	

BGMSET	Predefines a user-defined piece of music Executing immediately after BGMPLAY will cause a delay of approx. 2 frames	
Format	BGMSET User-defined tune number,"MML string"	
Arguments	User-defined tune number	User-defined tune number: 128-255
MML string	Pressing the Help button for "MML" will display the description of MML commands	
Examples	BGMSET 128,"CDEFG"	

BGMSETD	Predefines a user-defined tune - The DATA instruction should be used for internal registration of MML (DATA "CDEFGAB") - The end of DATA is determined according to the numerical value (DATA 0) - Internally, this is handled in the same way as RESTORE - RESTORE must be used to READ the data after BGMSETD - Executing immediately after BGMPLAY will cause a delay of approx. 2 frames	
Format	BGMSETD User-defined tune number,"@Label string"	
Arguments	User-defined tune number	User-defined tune number: 128-255
	@Label string	- A label string where an MML string has been registered with DATA - Should be specified by enclosing the string in " or by assigning it to a string variable - Pressing the Help button for "MML" will display the description of MML commands
Examples	BGMSETD 128,"@MMLTOP"	

BGMVAR (1)	Writes to an MML internal variable	
Format	BGMVAR Track number, Variable number, Value	
Arguments	Track number	Target MML track number: 0-7
	Variable number	Internal variable to which to write a value: 0-7 ($0-$7 in MML)
	Value	Value to write to the variable
Examples	BGMVAR 0,5,10	

BGMVAR (2)	Reads an MML internal variable	
Format	Variable=BGMVAR(Track number, Variable number)	
Arguments	Track number	Target MML track number: 0-7
	Variable number	Internal variable from which to read the value: 0-7 ($0-$7 in MML)
Return Values	Content of the specified variable during playback (When the music is stopped, -1)	
Examples	MC=BGMVAR(0,5)	

BGMSTOP (1)	Stops playing music	
Format	BGMSTOP [Track number [,Fading time]]	
Arguments	Track number	Target track number: 0-7 (If omitted, all tracks will be stopped)
	Fading time	Seconds (Decimal fractions are allowed; 0 = Stop immediately; if omitted, handled as 0)
Examples	BGMSTOP	

BGMSTOP (2)	Stops playing music - Forces ongoing sounds such as release sounds to stop - Executing this will cause user-defined BGM 255 to be overwritten
Format	BGMSTOP -1
Arguments	-1: Value for forcibly stopping sound
Examples	BGMSTOP -1

BGMVOL	Sets the volume for the specified track	
Format	BGMVOL [Track number,] Volume	
Arguments	Track number	Target track number: 0-7 (If omitted, 0)
	Volume	Volume level to set: 0-127
Examples	BGMVOL 0,64	

WAVSET	Defines the sound of an MML user-defined musical instrument	
Format	WAVSET Definition number,A,D,S,R,"Waveform string" [,Reference pitch]	
Arguments	Definition number	- User-defined musical instrument number: 224-255 - This number is specified with the MML @ command.
	A,D,S,R	Envelope definition parameters A: Attack (0-127) D: Decay (0-127) S: Sustain (0-127) R: Release (0-127)
	Waveform string	- Hexadecimal string - Two characters represent one sample value (8 bits) - &H00 - &H80 (128) - &HFF (255) - 16, 32, 64, 128, 256, or 512 samples can be specified - The number of characters should be twice the number of samples
	Reference pitch	If omitted, 69 (O4A)
Examples	W$="7F7F7F7FFFFFFFFF7F7F7F7FFFFFFFFF"*4 WAVSET 224,3,10,30,5,W$,69	

WAVSETA	Defines the sound of an MML user-defined musical instrument from an array - Used for sound definition from an array obtained with MICSAVE - 8180Hz sampling rate, 8 bits fixed	
Format	WAVSETA Definition number,A,D,S,R,Numerical value array [,Reference pitch][,Start subscript][,End subscript]	
Arguments	Definition number	- User-defined musical instrument number: 224-255 - This number is specified with the MML @ command
	A,D,S,R	Envelope definition parameters A: Attack (0-127) D: Decay (0-127) S: Sustain (0-127) R: Release (0-127)
	Numerical value array	Array obtained with the MICSAVE instruction (Up to 16384 samples)
	Reference pitch	If omitted, 69 (O4A)
	Start subscript	Subscript of the element in the numerical value array at which to start reading (If omitted, 0)
	End subscript	Subscript of the element in the numerical value array at which to stop reading (If omitted, the last element)
Examples	WAVSETA 224,0,95,100,20,SMPDATA	

EFCOFF	Turns off the effector setting
Format	EFCOFF
Examples	EFCOFF

EFCON	Turns on the effector setting The effect type should be selected with the EFCSET instruction
Format	EFCON
Examples	EFCON

EFCSET (1)	Selects a music effect type	
Format	EFCSET Type number	
Arguments	Type number	0: No effect (Same as EFCOFF) 1: Reverb (Bathroom) 2: Reverb (Cave) 3: Reverb (Space)
Examples	EFCSET 2	

EFCSET (2)	Sets effect parameters (For advanced users)	
Format	EFCSET Initial reflection time,Reverberation sound delay time,Reverberation sound decay time,Reverberation sound filter coefficient 1,Reverberation sound filter coefficient 2,Initial reflection sound gain,Reverberation sound gain	
Arguments	Initial reflection time	0-2000 (msec)
	Reverberation sound delay time	0-2000 (msec)
	Reverberation sound decay time	1-10000 (msec)
	Reverberation sound filter coefficient 1	0.0-1.0
	Reverberation sound filter coefficient 2	0.0-1.0
	Initial reflection sound gain	0.0-1.0
	Reverberation sound gain	0.0-1.0
Examples	EFCSET 997,113,1265,0.1,0,0.2,0.1	

EFCWET	Sets the respective effect amounts for BEEP, BGM, and TALK	
Format	EFCWET BEEP effect value, BGM effect value, TALK effect value	
Arguments	BEEP effect value	Effect amount for BEEP (0-127)
	BGM effect value	Effect amount for BGM (0-127)
	TALK effect value	- Effect setting for TALK (Less than 64: OFF; 64 or greater: ON) - For TALK, the only available setting is ON/OFF; the amount does not change
Examples	EFCWET 0,100,64	

TALK	Generates synthesized speech Alphanumeric symbols are read out character-by-character	
Format	TALK "Voice string"	
Arguments	Voice string	Synthesized speech string (Characters will be read out directly)
Special commands	A special command enclosed with <> is available for use in strings <S Speed>: Speech speed (Speed: 0-65536, default: 32768) <P Pitch>: Tone pitch (Pitch: 0-65536, default: 32768)	
Examples	TALK "ABCDE<P50000><S20000>FGHIJKLM"	

TALKCHK	Checks the status of speech synthesis
Format	Variable=TALKCHK()
Return Values	FALSE = Stopped, TRUE = Playing
Examples	RET=TALKCHK()

TALKSTOP	Stops the synthesized speech currently playing
Format	TALKSTOP
Examples	TALKSTOP

Mathematics

Instructions for mathematical operations including trigonometric functions and logarithms

FLOOR	Gets the integer part (by rounding down to the whole number) - The largest integer that is not greater than the specified value will be obtained - FLOOR(12.5) will be 12, while FLOOR(-12.5) will be -13	
Format	Variable = FLOOR(Numerical value)	
Arguments	Numerical value	Source numerical value
Return Values	Integer value after rounding down	
See Also	ROUND: Round-off, CEIL: Round-up	
Examples	A=FLOOR(12.345)	

ROUND	Gets the integer part (by rounding off to the nearest whole number)	
Format	Variable = ROUND(Numerical value)	
Arguments	Numerical value	Source numerical value
Return Values	Integer value after rounding off	
See Also	FLOOR: Round-down, CEIL: Round-up	
Examples	A=ROUND(12.345)	

CEIL	Gets the integer part (by rounding up to the whole number) - The smallest integer that is not less than the specified value will be obtained - CEIL(12.5) will be 13, while CEIL(-12.5) will be -12	
Format	Variable = CEIL(Numerical value)	
Arguments	Numerical value	Source numerical value
Return Values	Integer value after rounding up	
See Also	ROUND: Round-off, FLOOR: Round-down	
Examples	A=CEIL(12.345)	

ABS	Gets the absolute value	
Format	Variable = ABS(Numerical value)	
Arguments	Numerical value	Numerical value for which to get the absolute value
Return Values	Absolute value	
Examples	A=ABS(-12.345)	

SGN	Gets the sign	
Format	Variable = SGN(Numerical value)	
Arguments	Numerical value	Numerical value for which to get the sign
Return Values	0 or ±1	
Examples	A=SGN(12.345)	

MIN (1)	Gets the smallest value in the specified numerical value array	
Format	Variable = MIN(Numerical value array)	
Arguments	Numerical value array	Name of a numerical value array storing multiple numerical values
Return Values	Smallest number in the passed arguments	

Examples	DIM TMP[2]
	TMP[0]=50:TMP[1]=3
	A=MIN(TMP)	

MIN (2)	Gets the smallest value from the specified multiple numerical values	
Format	Variable = MIN(Numerical value [,Numerical value…])	
Arguments	Numerical values enumerated directly	Enumerate multiple numerical values separated by commas
Return Values	Smallest number in the passed arguments	
Examples	A=MIN(1,2,3,4)	

MAX (1)	Gets the largest value in the specified numerical value array	
Format	Variable = MAX(Numerical value array)	
Arguments	Numerical value array	Name of a numerical value array storing multiple numerical values
Return Values	Largest number in the passed arguments	

Examples	DIM TMP[2]
	TMP[0]=50:TMP[1]=3
	A=MAX(TMP)	

MAX (2)	Gets the largest value from the specified multiple numerical values	
Format	Variable = MAX(Numerical value [,Numerical value…])	
Arguments	Numerical values enumerated directly	Enumerate multiple numerical values separated by commas
Return Values	Largest number in the passed arguments	
Examples	A=MAX(1,2,3,4)	

RND	Gets an integer random number (0 - the maximum value minus 1)	
Format	Variable = RND([Seed ID,] Maximum value)	
Arguments	Seed ID	Random number series: 0-7
	Maximum value	Upper limit of the random number to be obtained
Return Values	Integer random number in the range 0 - the maximum value minus 1	
Examples	A=RND(100)	

RNDF	Gets a real-type random number (a real-type random number greater than 0 and less than 1.0)	
Format	Variable = RNDF([Seed ID])	
Arguments	Seed ID	Random number series: 0-7
Return Values	Real-type random number greater than 0 and less than 1	
Examples	A=RNDF()	

RANDOMIZE	Initializes a random number series	
Format	RANDOMIZE Seed ID [, Seed value]	
Arguments	Seed ID	Random number series type: 0-7
	Seed value	If 0 or omitted, initialization will be performed using available entropy information
Examples	RANDOMIZE 0	

SQR	Finds the positive square root	
Format	Variable = SQR(Numerical value)	
Arguments	Numerical value	Numerical value for which to find the square root
Return Values	Positive square root found	
Examples	A=SQR(4)	

EXP	Exponentiates e (natural logarithm base)	
Format	Variable = EXP([Numerical value])	
Arguments	Numerical value	Exponent (* If omitted, e will be returned.)
Return Values	Exponentiation result	
Examples	A=EXP(2)	

LOG	Finds the logarithm	
Format	Variable = LOG(Numerical value [,Base])	
Arguments	Numerical value	Antilogarithm
	Base	Base (* If omitted, the natural logarithm will be found.)
Return Values	Result found	
Examples	A=LOG(2,2)	

POW	Exponentiates a value	
Format	Variable = POW(Numerical value, Multiplier)	
Arguments	Numerical value	Numerical value to exponentiate
	Multiplier	Exponentiation multiplier
Return Values	Exponentiation result	
Examples	A=POW(1,4)	

PI	Gets the circumference ratio	
Format	Variable = PI()	
Return Values	Circumference ratio value (3.14159265)	
Examples	A=PI()	

RAD	Finds the radian value from a degree value	
Format	Variable = RAD(Numerical value)	
Arguments	Numerical value	Degrees: 0-360
Return Values	Radian value found from the degree value	
Examples	R=RAD(45)	

DEG	Finds the degree value from a radian value	
Format	Variable = DEG(Numerical value)	
Arguments	Numerical value	Radian value
Return Values	Degree value found from the radian value	
Examples	A=DEG(0.5*PI())	

SIN	Returns the sine value	
Format	Variable = SIN(Angle)	
Arguments	Angle	Radian
Return Values	Value found	
Examples	A=SIN(RAD(45))	

COS	Returns the cosine value	
Format	Variable = COS(Angle)	
Arguments	Angle	Radian
Return Values	Value found	
Examples	A=COS(RAD(45))	

TAN	Returns the tangent value	
Format	Variable = TAN(Angle)	
Arguments	Angle	Radian
Return Values	Value found	
Examples	A=TAN(RAD(45))	

ASIN	Returns the arc sine value	
Format	Variable = ASIN(Numerical value)	
Arguments	Numerical value	-1.0 to 1.0
Return Values	Arc sine (radian) value found	
Examples	A=ASIN(0)	

ACOS	Returns the arc cosine value	
Format	Variable = ACOS(Numerical value)	
Arguments	Numerical value	-1.0 to 1.0
Return Values	Arc cosine (radian) value found	
Examples	A=ACOS(1)	

ATAN (1)	Returns the arc tangent value (from numerical values)	
Format	Variable = ATAN(Numerical value)	
Arguments	Numerical value	Numerical value from which to find the angle
Return Values	Arc tangent (radian) value found	
Examples	A=ATAN(1)	

ATAN (2)	Returns the arc tangent value (from XY-coordinates)	
Format	Variable = ATAN(Y-coordinate,X-coordinate)	
Arguments	Y-,X-coordinates	- X-,Y-coordinates from the origin - The Y-coordinate should be input first
Return Values	Arc tangent (radian) value found	
Examples	A=ATAN(1,1)	

SINH	Returns the hyperbolic sine value	
Format	Variable = SINH(Numerical value)	
Arguments	Numerical value	Real-type number for which to find the hyperbolic sine
Return Values	Hyperbolic sine value found	
Examples	A=SINH(1)	

COSH	Returns the hyperbolic cosine value	
Format	Variable = COSH(Numerical value)	
Arguments	Numerical value	Real-type number for which to find the hyperbolic cosine
Return Values	Hyperbolic cosine value found	
Examples	A=COSH(1)	

TANH	Returns the hyperbolic tangent value	
Format	Variable = TANH(Numerical value)	
Arguments	Numerical value	Real-type number for which to find the hyperbolic tangent
Return Values	Hyperbolic tangent value found	
Examples	A=TANH(0.5)	

CLASSIFY	Determines whether a given number is an ordinary numerical value, infinity, or not-a-number (NaN)	
Format	Variable = CLASSIFY(Numerical value)	
Arguments	Numerical value	Real number to check
Return Values	0 = Ordinary numerical value, 1 = Infinity, 2 = NaN	
Examples	A=CLASSIFY(0.5)	

Operations on strings

Instructions for specifying display formats for strings, extracting strings, etc.

ASC	Gets a character code for the specified character (or string variable)	
Format	Variable = ASC("Character")	
Arguments	Character	Character string (or string variable) storing the character to check
Return Values	Character code (UTF-16) for the specified character	
Examples	A=ASC("A")	

CHR$	Returns the character for the specified character code	
Format	String variable = CHR$(Character code)	
Arguments	Character code	Number (UTF-16) that corresponds to a character
Return Values	Character that corresponds to the character code	
Examples	S$=CHR$(65)	

VAL	Gets a numerical value from a character string	
Format	Variable = VAL("Character string")	
Arguments	Character string	A character string representing a number (e.g., "123"), or a string variable
Return Values	Numerical value interpreted from the character string	
Examples	A=VAL("123")	

STR$	Gets a character string from a numerical value	
Format	String variable = STR$(Numerical value [,Number of digits])	
Arguments	Numerical value	Numerical value to convert to a character string
	Number of digits	- Should be specified when right-justification with a certain number of digits is desired - When the number of digits in the numerical value is greater than the specified number of digits, the specification will be ignored
Return Values	Character string generated from the numerical value (123 → "123")	
Examples	S$=STR$(123)	

HEX$	Gets a hexadecimal string from a numerical value	
Format	String variable = HEX$(Numerical value [,Number of digits])	
Arguments	Numerical value	Numerical value from which to get a hexadecimal string (The fractional part should be truncated)
	Number of digits	- Number of digits in the hexadecimal string to output - If specified, the string will be padded with leading zeros before being returned
Return Values	Hexadecimal string generated from the numerical value (255 → "FF")	
Examples	S$=HEX$(65535,4)	

FORMAT$	Stringizes values by using display formats to shape them	
Format	Variable$ = FORMAT$("Format string", Value ,…)	
Arguments	Format string (Multiple formats can be enumerated)	%S: Outputs the content of the string variable %D: Outputs integers in decimal %X: Outputs integers in hexadecimal %F: Outputs real numbers
	Supplemental specifications for format strings	The following supplemental specifications can be used after % to shape output - Specification of the number of digits: A value indicating the number of digits should be specified (%8D, %4X) - Specification of the number of fractional digits: Should be specified as (number of digits in the integer part).(number of digits in the fractional part) (%8.2F) - Space-padding: A space character and the number of digits should be specified (% 4D → 0) - Zero-padding: 0 and the number of digits should be specified (%08D → 00000000) - Left alignment: A "-" sign and the number of digits should be specified (%-8D) - Displaying the + sign: A "+" sign and the number of digits should be specified (%+8D)
	Value	- Source value to shape - An adequate number of values corresponding to the elements specified in the formats should be enumerated, separated by commas (,)
Return Values	Character string generated	
Examples	S$=FORMAT$("%06D",A)	

LEN	Gets the number of characters in a character string/Gets the number of elements in an array	
Format	>Variable = LEN("Character string" or Array variable)	
Arguments	For a character string	Character string, or the name of the string variable, in which to check the number of characters
	For an array variable	Name of the array variable in which to check the number of elements
Return Values	- For a character string: Number of characters (All characters will be counted as one character) - For an array variable: Number of elements	
Examples	A=LEN("ABC123")	

MID$	Extracts a character string with the specified number of characters from the specified position in the specified character string	
Format	String variable = MID$("Character string", Start position, Number of characters)	
Arguments	Character string	Source character string
	Start position	Position (in character units) from which to start extracting a character string
	Number of characters	Number of characters to extract
Return Values	Character string extracted	
Examples	S$=MID$("ABC",0,2)	

LEFT$	Extracts a character string with the specified number of characters from the left end of the specified character string	
Format	String variable = LEFT$("Character string", Number of characters)	
Arguments	Character string	Source character string
	Number of characters	Number of characters to extract
Return Values	Character string extracted	
Examples	S$=LEFT$("ABC",2)	

RIGHT$	Extracts a character string with the specified number of characters from the right end of the specified character string	
Format	Variable$ = RIGHT$("Character string", Number of characters)	
Arguments	Character string	Source character string
	Number of characters	Number of characters to extract
Return Values	Character string extracted	
Examples	S$=RIGHT$("ABC",2)	

INSTR	Searches for the target character string in another character string	
Format	Variable = INSTR([Start position,] "Character string to search in", "Character string to search")	
Arguments	Start position	- Position (in character units, larger than or equal to 0) in the source character string from which to start searching - If omitted, the search will be started from the beginning of the source string
	Character string to search in	Source character string
	Character string to search for	Character string to search for in the source character string
Return Values	- If the search string is found: Position in the source string (in character units) - Otherwise: -1	
Examples	A=INSTR(0, "ABC","B")	

SUBST$	Substitutes one character string with another string	
Format	String variable = SUBST$("Character string", Start position, [Number of characters,] "Substitute string")	
Arguments	Character string	Source character string
	Start position	Position in the source character string from which to start substitution (0 - Number of characters minus 1)
	Number of characters	- Number of characters to substitute with another string - If omitted, all characters after the substitution start position will be replaced with the substitute string
	Substitute string	The specified number of characters from the start position will be substituted with this string
Return Values	Character string after the substitution	
Examples	A$=SUBST$("ABC",0,2,"XY")	

Source code manipulation

Functions for writing program strings into specified SLOTs

PRGEDIT	Specifies the program SLOT to manipulate, and the current line	
Format	PRGEDIT Program SLOT [,Line number]	
Arguments	Program SLOT	- Program SLOT to manipulate: 0-3 - Specifying the SLOT currently running will give an error
	Line number	- Line to manipulate (Current line) - If this is omitted, the first line will be the current line - If -1 is specified for the line number, the current line will be the last line
Examples	PRGEDIT 0	

PRGGET$	Gets the current single line as a character string
Format	String variable=PRGGET$()
Return Values	Source character string of the current line (or an empty string if there is no applicable line)
Examples	A$=PRGGET$()

PRGSET	Substitutes the contents of the current line with the specified string If PRGGET$ has returned an empty string, a line will be added	
Format	PRGSET "Character string"	
Arguments	Character string	Character string to substitute the current line with
Examples	PRGSET "'Comment"	

PRGINS	Inserts a line in the current line For a character string including the line feed code CHR$(10), multiple lines will be inserted	
Format	PRGINS "Character string" [,Flag]	
Arguments	Character string	Source character string to insert
	Flag	1 = Inserts a line after the current line 0 = Inserts a line before the current line (If omitted = 0, before the current line)
Examples	PRGINS "PRINT "+CHR$(34)+"HELLO"+CHR$(34)	

PRGDEL	Deletes the current line	
Format	PRGDEL [Number of lines to delete]	
Arguments	Number of lines to delete	- Number of lines to delete (If omitted, one line will be deleted) - If a negative value is specified, all lines will be deleted
Examples	PRGDEL	

PRGSIZE	Gets the number of lines in the source code	
Format	Variable=PRGSIZE([Program SLOT [,Type of value to get]])	
Arguments	Program SLOT	Program SLOT from which to get the number of lines: 0-3
	Type of value to get	0 = Number of lines, 1 = number of characters, 2 = number of free characters (Default: 0)
Return Values	Type-appropriate value	
Examples	A=PRGSIZE(0)	

PRGNAME$	Program file name File that has been handled with the LOAD/SAVE instruction	
Format	String variable=PRGNAME$([Program SLOT])	
Arguments	Program SLOT	Program SLOT from which to get the file name: 0-3
Return Values	- Program file name - When a program is running, the SLOT in which it is running - When no program is running, the "SLOT of the last program run" - The "SLOT of the last program run" is usually SLOT 0 - If a running program has been suspended with the STOP instruction or the START button, or if an error has occurred, the SLOT at that time will be the “SLOT of the last program run” and will remain so until the next RUN	
Examples	PRINT PRGNAME$(0)	

Bit Operations

Instructions for performing a bit operation to numerical values

MOD	Gets the remainder of Numerical value 1 divided by Numerical value 2	
Format	Variable=Numerical value 1 MOD Numerical value 2	
Arguments	Numerical value 1	Number (or expression) to divide
	Numerical value 2	Number (or expression) to divide by (Dividing by zero gives an error)
Examples	A=200 MOD 5	

DIV	Gets the integer value of Numerical value 1 divided by Numerical value 2	
Format	Variable=Numerical value 1 DIV Numerical value 2	
Arguments	Numerical value 1	Number (or expression) to divide
	Numerical value 2	Number (or expression) to divide by (Dividing by zero gives an error)
Examples	A=200 DIV 5	

AND	Logical AND of Numerical value 1 and Numerical value 2 (Multiplication of bits)	
Format	Variable=Numerical value 1 AND Numerical value 2	
Arguments	Numerical value 1	Bit string 1
	Numerical value 2	Bit string 2
Examples	A=200 AND &HE7	

OR	Logical OR of Numerical value 1 and Numerical value 2 (Addition of bits)	
Format	Variable=Numerical value 1 OR Numerical value 2	
Arguments	Numerical value 1	Bit string 1
	Numerical value 2	Bit string 2
Examples	A=128 OR &HA3	

XOR	Exclusive OR of Numerical value 1 and Numerical value 2 (If the values are the same, 0; if not, inversion)	
Format	Variable=Numerical value 1 XOR Numerical value 2	
Arguments	Numerical value 1	Bit string 1
	Numerical value 2	Bit string 2
Examples	A=100 XOR &H4C	

<<	Shifts a numerical value to the left by the specified number of bits	
Format	Variable=Numerical value << Number of times	
Arguments	Numerical value	Bit string 1
	Number of times	Number of bit shifts
Examples	A=100 << 2	

>>	Shifts a numerical value to the right by the specified number of bits	
Format	Variable=Numerical value >> Number of times	
Arguments	Numerical value	Bit string 1
	Number of times	Number of bit shifts
Examples	A=100 >> 2	

MML

Commands for MML (Music Macro Language)

MML (1)	Commands for controlling whole tunes	
	Channel specification	:0 - :15 (A colon [:] followed by a channel number should be specified)
	Tempo specification	T1 - T512
Examples	'--- Chord of Do Mi Sol with tempo 120 BGMPLAY "T120:0CCC:1EEE:2GGG"	

MML (2)	Commands and notations for controlling tone length	
	Specification of default tone length	L1 - L192 These specifications will change the subsequent default tone length
	Individual tone length specification♦	To play a tone with a length other than the default tone length, change the tone length by inputting the length specification after the pitch symbol e.g., Specify the length of Do directly C1 (Whole note of Do) C2 (Half note of Do) C4 (Quarter note of Do) C8 (Eighth note of Do) C16 (Sixteenth note of Do) C32 (Thirty-second note of Do) C1. - C32. (Dotted note representations of Do) * Triplets should be specified as C12C12C12, C24C24C24, etc.
	Playing technique	& Connects the preceding/succeeding notes () _ Portamento ()
	Note duration ratio (gate) setting	Q0 - Q8 Smaller numbers give a greater impression of breaks between successive tones

MML (3)	Commands for controlling (tone) pitch.	
	Scale specification	C (Do) D (Re) E (Mi) F (Fa) G (Sol) A (La) B (Ti)
	Halftone higher	C# D# E# F# G# A# B#
	Halftone lower	C- D- E- F- G- A- B-
	Rest♦	R * Can be used in the same way as scales. e.g., R4 (Quarter-note rest)
	Octave specification	O0 - O8 Octave number specification
	One octave up	<
	One octave down	>
	Inversion of octave specification	! * Specifying this causes the <> symbols to be handled in reverse.
	Key value specification♦	N0 - N127 * O4C=60; one increment/decrement per halftone.
Examples	'--- Do Do Re Mi Fa Sol Sol La Ti Ti Ti Ti BGMPLAY "CCDEFGGABBBB"	

MML (4)	Commands for controlling sound volume and localization	
	Volume specification♦	V0 - V127 Volume level value specification
	One volume level up	(
	One volume level down	)
	Pan pot♦	P0 - P127 Determines the position between the speakers from which the sound is heard (localization) Left: P0 - P63 Center: P64 Right: P65 - P127
	Envelope setting	@E + A,D,S,R values Sets the change in volume from sound generation to attenuation A (Attack time): 0-127 D (Decay time): 0-127 S (Sustain level): 0-127 R (Release time): 0-127 * The smaller each time value, the slower it is. e.g., @E127,100,30,100
	Envelope resetting	@ER Releases the envelope

MML (5)	Commands for controlling tone changes	
	Instrument sound changes	@0 - @127 Equivalent to GM (Can be checked using the SMILETOOL) @128 Standard drum set @129 Electric drum set @144 - @150 PSG sound sources @151 Noise sound source @224 - @255 User-defined waveforms (Those registered with WAVSET) @256 - Sound effects provided for BEEP
	@128 drum set (@129)	B1 Acoustic Bass Drum 2 (909BD) C2 Acoustic Bass Drum 1 (808BDTom) C2# Side Stick(808RimShot) D2 Acoustic Snare (808SD) D2# Hand Clap E2 Electric Snare (909SD) F2 Low Floor Tom (808TomLF) F2# Closed Hi-hat(808CHH) G2 High Floor Tom (808TomF) G2# Pedal Hi-hat(808CHH) A2 Low Tom (808TomL) A2# Open Hi-hat(808OHH) B2 Low-Mid Tom (808TomLM) C3 High Mid Tom (808TomHM) C3# Crash Cymbal 1(808Cymbal) D3 High Tom (808TomH) D3# Ride Cymbal 1 E3 Chinese Cymbal F3 Ride Bell F3# Tambourine G3 Splash Cymbal G3# Cowbell(808Cowbell) A3 Crash Cymbal 2 A3# Vibra-slap B3 Ride Cymbal 2 C4 High Bongo C4# Low Bongo D4 Mute Hi Conga (808CongaMute) D4# Open Hi Conga(808CongaHi) E4 Low Conga (808CongaLo) F4 High Timbale F4# Low Timbale G4 High Agogo G4# Low Agogo A4 Cabasa A4# Maracas(808Maracas) B4 Short Whistle C5 Long Whistle C5# Short Guiro D5 Long Guiro D5# Claves(808Claves) E5 Hi Wood Block F5 Low Wood Block F5# Mute Cuica G5 Open Cuica G5# Mute Triangle A5 Open Triangle

MML (6)	Special effect commands used to give a subtle fluctuation to sounds and volume * @MA, @MP, and @ML cannot be used at the same time.	
	Start modulation	@MON
	Stop modulation	@MOF
	Detuning (fine frequency adjustment) setting♦	@D-128 to @D127 (-128 is a tone lower; +127 is a tone higher)
	Tremolo setting	@MA + Depth, Range, Speed, Delay values (0-127 for each) e.g., @MA64,1,16,32
	Vibrato setting	@MP + Depth, Range, Speed, Delay values (0-127 for each) e.g., @MP64,1,16,32
	Auto pan pot setting	@ML + Depth, Range, Speed, Delay values (0-127 for each) e.g., @ML100,1,8,0

MML (7)	Special music playback commands.	
	Repeat start	[
	Repeat end	]Number of times * If the number of times is omitted, the loop will be endless.♦ e.g., Complicated repeat to play CCC CCC DEF CCC CCC DEF BGMPLAY "[[CCC]2DEF]2"
	MML internal variable specification	$0 - $7 MML internal variables * In commands marked with ♦, these variables can be specified in place of numerical values. e.g., $0=64 V$0 instead of V64
	Value assignment to MML internal variables	$0=value - $7=value Assign values (0-255) to the variables * Variables being played back can be assigned a value or referenced with the BGMVAR instruction.
	Macro definition	{Label name=MML} Can be used to reuse a melody or phrase repeatedly - Channel specification within the defined MML is not allowed - For label names, up to eight alphanumerical characters can be used - Reusing a label name for another definition is not allowed
	Macro use	{Label name} MML corresponding to the defined label will be expanded
Examples	'--- Play a rhythm using a macro BGMPLAY "T240@128O2{PT0=CDEDCDE<G}[{PT0}]4"	

Error Table

System Variable for Error Handling	If an error has occurred, the relevant information is stored in the system variable.
	ERRNUM (Error number)
	ERRLINE (Number of the line where the error occurred)
Common Errors	3: Syntax error (syntax does not follow the grammar rules)
	4: Illegal function call (the number of arguments specified in an instruction or function is wrong)
	5: Stack overflow (an overflow has occurred in the stack)
	6: Stack underflow (an underflow has occurred in the stack)
	7: Divide by zero (division by zero was attempted)
	8: Type mismatch (an inconsistent variable type is specified)
	9: Overflow (the calculation result exceeded the allowed range)
	10: Out of range (a value outside the allowed range was specified)
	11: Out of memory (sufficient memory area is not available)
	12: Out of code memory (sufficient code memory area is not available)
	13: Out of DATA (DATA that can be READ is insufficient)
	14: Undefined label (the specified label could not be found)
	15: Undefined variable (the specified variable could not be found)
	16: Undefined function (the specified instruction/function could not be found)
	17: Duplicate label (the same label has been defined twice)
	18: Duplicate variable (the same variable has been defined twice)
	19: Duplicate function (the same instruction/function has been defined twice)
	20: FOR without NEXT (a FOR has no NEXT)
	21: NEXT without FOR (a NEXT has no FOR)
	22: REPEAT without UNTIL (a REPEAT has no UNTIL)
	23: UNTIL without REPEAT (an UNTIL has no REPEAT)
	24: WHILE without WEND (a WHILE has no WEND)
	25: WEND without WHILE (a WEND has no WHILE)
	26: THEN without ENDIF (a THEN has no ENDIF)
	27: ELSE without ENDIF (an ELSE has no ENDIF)
	28: ENDIF without IF (an ENDIF has no IF)
	29: DEF without END (a DEF has no END)
	30: RETURN without GOSUB (a RETURN has no GOSUB)
	31: Subscript out of range (array subscripts are not within the allowed range)
	32: Nested DEF (a DEF has been defined within another DEF)
	33: Can't continue (the program cannot resume with CONT)
	34: Illegal symbol string (a label string has been incorrectly described)
	35: Illegal file format (the file is in a format that SmileBASIC cannot support)
	36: Mic is not available (a microphone instruction was used without using XON MIC)
	37: Motion sensor is not available (a motion instruction was used without using XON MOTION)
	38: Use PRGEDIT before any PRG function (one of the PRG instructions was used without using PRGEDIT)
	39: Animation is too long (animation definition is too long)
	40: Illegal animation data (animation data is incorrect)
	41: String too long (string is too long)
	42: Communication buffer overflow (an overflow has occurred in the buffer for sending MPSEND)
	43: Can't use from DIRECT mode (an instruction that does not work in DIRECT mode was used)
	44: Can't use in program (an instruction that cannot be used in a program was used)
	45: Can't use in tool program (an instruction that cannot be used from a tool program was used)
	46: Load failed (failed to read the file)
	47: Illegal MML (the MML [Music Macro Language] is incorrect)

System Variable

What is a System Variable?	System variables are variables reserved in the system managed by SmileBASIC.
	* Although they are primarily read-only, it is possible to populate values in some (they are writable).
Examples	CSRX 'Cursor position X
	CSRY 'Cursor position Y
	CSRZ 'Cursor position Z (depth)
	FREEMEM 'Amount of free user memory available (in KB)
	VERSION 'System version (&HXXYYZZZZ)
	TABSTEP 'TAB movement amount (writable)
	SYSBEEP 'System sound effects (writable, TRUE=allowed)
	ERRNUM 'Error number
	ERRLINE 'Line where an error occurred
	ERRPRG 'Program SLOT where an error occurred
	PRGSLOT 'Current program SLOT for the PRG instruction
	RESULT 'Dialog result (TRUE/FALSE/-1=Suspended)
	MAINCNT 'Number of frames since SmileBASIC was launched
	MICPOS 'Current sampling location
	MICSIZE 'Number of samples in the sampling buffer
	MPCOUNT 'Number of participants in a session
	MPHOST 'Host ID
	MPLOCAL 'User ID
	TRUE 'Always 1
	FALSE 'Always 0
	TIME$ 'Time string (HH:MM:SS)
	DATE$ 'Date string (YYYY/MM/DD)
	HARDWARE 'Hardware information (1=new3DS)
	CALLIDX 'Number called by SPFUNC and BGFUNC

Constants

#	- 32-bit numerical value definitions prepared in the system - Used instead of a numerical value in order to specify a color or handle a button - e.g., IF BUTTON() AND (#A OR #B) THEN

Examples

```
'--- Generic
#ON    '1
#OFF   '0
#YES   '1
#NO    '0
#TRUE  '1
#FALSE '0
'--- RGB
#AQUA    '&HFF00F8F8
#BLACK   '&HFF000000
#BLUE    '&HFF0000FF
#CYAN    '&HFF0000F8
#FUCHSIA '&HFFF800F8
#GRAY    '&HFF808080
#GREEN   '&HFF008000
#LIME    '&HFF00F800
#MAGENTA '&HFFF800F8
#MAROON  '&HFF800000
#NAVY    '&HFF000080
#OLIVE   '&HFF808000
#PURPLE  '&HFF800080
#RED     '&HFFF80000
#SILVER  '&HFFC0C0C0
#TEAL    '&HFF008080
#WHITE   '&HFFF8F8F8
#YELLOW  '&HFFF8F800
'--- TEXTCOLOR
#TBLACK   '1
#TMAROON  '2
#TRED     '3
#TGREEN   '4
#TLIME    '5
#TOLIVE   '6
#TYELLOW  '7
#TNAVY    '8
#TBLUE    '9
#TPURPLE  '10
#TMAGENTA '11
#TTEAL    '12
#TCYAN    '13
#TGRAY    '14
#TWHITE   '15
'--- BUTTON
#UP    '&H0001
#DOWN  '&H0002
#LEFT  '&H0004
#RIGHT '&H0008
#A     '&H0010
#B     '&H0020
#X     '&H0040
#Y     '&H0080
#L     '&H0100
#R     '&H0200
#ZL    '&H0800
#ZR    '&H1000
'--- ATTR
#TROT0   '&H00
#TROT90  '&H01
#TROT180 '&H02
#TROT270 '&H03
#TREVH   '&H04
#TREVV   '&H08
'---SPSET/SPCHR ATTR
#SPSHOW   '&H01, Display
#SPROT0   '&H00, Rotate by 0 degree
#SPROT90  '&H02, Rotate by 90 degrees
#SPROT180 '&H04, Rotate by 180 degrees
#SPROT270 '&H06, Rotate by 270 degrees
#SPREVH   '&H08, Right/left
#SPREVV   '&H10, Up/down
#SPADD    '&H20, Additive synthesis
'--- BG ATTRE
#BGROT0   '&H0000
#BGROT90  '&H0800
#BGROT180 '&H1000
#BGROT270 '&H2000
#BGREVH   '&H4000
#BGREVV   '&H8000
'--- SPCHK/BGCHK
#CHKXY '&H01
#CHKZ  '&H02
#CHKUV '&H04
#CHKI  '&H08
#CHKR  '&H10
#CHKS  '&H20
#CHKC  '&H40
#CHKV  '&H80
```