

SCRIPTING MANUAL (BASIC)

Avisaro 2.0 **Base Module / Box**

Version 1.9
2008.05.21

CHANGE HISTORY

2007-09-01		Initial version
2008-03-20		Added CAN and AUX commands
2008-05-21		Grouped commands for faster search

CONTACT

Avisaro AG
Vahrenwalderstr. 7 (tch)
30165 Hannover

Germany

Telephone:

+49-511-7809390

Telefax:

+49-511-35319624

eMail:

support@avisaro.com

TABLE OF CONTENT

Change History.....	2
Contact	3
Table of content.....	4
Introduction.....	Fehler! Textmarke nicht definiert.
Load and run.....	6
LOAD	6
LIST	6
RUN	6
GENERAL BASIC SCRIPTING SYNTAX	8
Valid Labels	8
Separating Lines	8
Basic Commands – Structural instructions.....	9
FOR i=start TO end [STEP n]...NEXT	9
DO...LOOP [UNTIL]	9
IF...THEN...ELSEIF...ELSE...end IF	9
WHILE <booleanexpression>...WEND	9
GOSUB <label>	9
GOTO <label>	9
RETURN	10
SLEEP <expression>	10
REM	10
BASIC Commands – special purpose	11
EXEC <string>	11
BASIC commands – string and variable functions	12
DIM x(y)	12
LOAD <address>, <variable>	12
SAVE <address>, <variable>	12
ABS(n)	12
ASC(n)	12
CHR\$(n)	12
DATA	12
INSTR([Start,] SearchString, FindString)	13
LCASE\$(stringexpression)	13
LEFT\$(stringexpression, n)	13
LEN(stringexpression)	13
LTRIM\$(stringexpression)	13
MID\$(stringexpression, start, length)	13
READ variable1[, variable2[, ...]]	14
RESTORE	14
RIGHT\$(stringexpression, n)	14
RTRIM\$(stringexpression)	14
SPACE\$(n)	14
SQR(n)	14
STR\$(numeric-expression)	14
STRING\$(m, n), STRING\$(m, stringexpression)	15

TAB(n)	15
UCASE\$(stringexpression)	15
VAL(stringexpression)	15
BASIC commands – Status and Time Information.....	16
TIMES\$	16
DATE\$	16
TIME	16
FREEMEM	16
LASTERR	16
BYTESREAD	16
VERSION <expression>	16
BASIC commands – file handling.....	17
OPEN "access", filenum, "filename"	17
GET filehandle, variable	17
PUT filehandle, variable, [max_length]	17
CLOSE [filenum[, filenum]]	18
KILL "filename"	18
SEEK filenum, filepos	18
STATUS <handle>	18
EOF(filenum)	18
LOC (filenum)	19
LOF (filenum)	19
BASIC commands – networking.....	20
CONNECT <handle> <ipaddress> <port> <txdelay>	20
LISTEN <handle> <port> <txdelay>	20
UDP <handle> <tx_ip> <tx_port> <rx_port> <tx_delay> <use_checksums>	20
RESOLV (stringexpression)	21
GET filehandle, variable	21
PUT handle, variable, [max_length]	21
CLOSE [filenum[, filenum]]	21
STATUS <handle>	22
BASIC commands – In/Output, CAN, RS232, I2C, IO.....	23
INMODE <expression>	23
OUTMODE <expression>	23
AUXOPEN <handle>, <p1>, <p2>, <p3>, <p4>	23
INPUT <variable>	24
PRINT <expression>[, ;][<expression>][...]	24
KEYS	24
SETLEDS <expression>	24
GETCAN <bytearray>	25
CANINFO (num)	25
PUTCAN <bytearray>	26
SETCAN <bytearray>,<index>,<value>	26

LOAD AND RUN

The Avisaro Base module provides three commands to control the scripting engine:

LOAD		
Description	Loads a new BASIC script into the module	
Details	The script is loaded but not executed. Use RUN to execute.	
Returns	No return value	
Parameter	<i>none</i>	Waits for the script to be entered on the console. Enter script manually, via 'send text file' feature of the console or via 'autoexec.txt' (see manual for details). Exit entry mode by entering the stop sequence (usually: +++).
	<i>filename</i>	The BASIC script is loaded from SD flash card.

LIST		
Description	Lists the BASIC script stored in the module	
Details		
Returns	The BASIC script is displayed	
Parameter	<i>none</i>	

RUN		
Description	Starts a BASIC script.	
Details		
Returns	No return value	
Parameter	<i>none</i>	Starts execution of the BASIC script.
	auto	Sets a flag to enable the autostart function. The next time the module is switch on, basic will start to run.
	wait	Starts BASIC while stopping the console. When BASIC finishes, the console is active again.

	await	Sets the autostart flag and the wait mode.
	manual	Removes the autostart flag.

GENERAL BASIC SCRIPTING SYNTAX

VALID LABELS

Line numbers with decimal points are valid

Example: 100.5 PRINT "Hello world!"

Usually use the colon to define a label

Example: Label1: PRINT "This is a valid label!"

Label2:

SEPARATING LINES

Lines can be separated by using a colon

Example: ?"Hi!":?"Hi again!":?"Hi yet again!"

To continue a line, add the underscore (‘_’) character at the end

Example: PRINT "Hi, my name is " + _
"Cool Guy!"

BASIC COMMANDS – STRUCTURAL INSTRUCTIONS

FOR I=START TO END [STEP N]...NEXT	
	<ul style="list-style-type: none"> • Control flow block that repeats a number of statements • ie. FOR i = start to end [STEP n] • see also STEP

DO...LOOP [UNTIL]	
	<ul style="list-style-type: none"> • Control block that loops until expression returns true • UNTIL <expression> is optional

IF...THEN...ELSEIF...ELSE...END IF	
	<ul style="list-style-type: none"> • Control flow block that evaluates the boolean expression and returns • either true or false. ie. IF 10 < 20 THEN ?"True"

WHILE <BOOLEANEXPRESSION>...WEND	
	<ul style="list-style-type: none"> • Control flow block repeating a number of statements until expression returns false • see also WEND

GOSUB <LABEL>	
	<ul style="list-style-type: none"> • Branches to <label> • see also RETURN

GOTO <LABEL>	
	<ul style="list-style-type: none"> • Branches forward or backwards depending on <label> • see also LABELS

RETURN	<ul style="list-style-type: none"> • Returns to the next instruction after the last GOSUB call • see also GOSUB
---------------	---

SLEEP <EXPRESSION>	
	<ul style="list-style-type: none"> • Delays the program for <expression> system ticks • The system tick is 50Hz • Thus, waiting for 1 second results in SLEEP(50)

REM	
	Comment block, or use '

BASIC COMMANDS – SPECIAL PURPOSE

EXEC <STRING>	
	<ul style="list-style-type: none">• Takes <string> as command line and executes the command• Example: EXEC "dir" -> Shows the root directory• Return code of executed command sets the LASTERR pseudo variable

BASIC COMMANDS – STRING AND VARIABLE FUNCTIONS

DIM X(Y)	
	<ul style="list-style-type: none"> Without AS, will always malloc y BYTES, not ints
LOAD <ADDRESS>, <VARIABLE>	
	<ul style="list-style-type: none"> Loads variable from data flash position 'address' variable can be an array, a string, an array-element (byte), or a single variable
SAVE <ADDRESS>, <VARIABLE>	
	<ul style="list-style-type: none"> Stores variable into data flash position 'address' variable can be an array, a string, an array-element (byte), or a single variable
ABS(N)	<ul style="list-style-type: none"> Function returns the absolute value of integer expression n see also SQR
ASC(N)	<ul style="list-style-type: none"> Function returns the ASCII value of the character see also CHR\$
CHR\$(N)	<ul style="list-style-type: none"> Function returns the character of ASCII value n see also ASC
DATA	<ul style="list-style-type: none"> Used to store numeric and string constants used by the READ statement. see also READ, RESTORE

INSTR([START,] SEARCHSTRING, FINDSTRING)	
	<ul style="list-style-type: none"> • Start searching at [Start] position of Search String • SearchString is the string to be searched • FindString is the string to be looked for • Returns 0 if match not found

LCASE\$(STRINGEXPRESSION)	
	<ul style="list-style-type: none"> • Function returns a string that has been lowercased • see also UCASE\$

LEFT\$(STRINGEXPRESSION, N)	
	<ul style="list-style-type: none"> • Function that returns a string containing the leftmost n characters of the string expression. • see also RIGHT\$, MID\$

LEN(STRINGEXPRESSION)	
	<ul style="list-style-type: none"> • Function returns the length of the string

LTRIM\$(STRINGEXPRESSION)	
	<ul style="list-style-type: none"> • Function that returns a string with leading spaces removed • see also RTRIM\$

MID\$(STRINGEXPRESSION, START, LENGTH)	
	<ul style="list-style-type: none"> • Function that returns a portion of the string expression • see also LEFT\$, RIGHT\$

READ VARIABLE1[, VARIABLE2[, ...]]	
	<ul style="list-style-type: none"> • Reads values from DATA statement and assigns the values to the variables. • see also DATA, RESTORE

RESTORE	
	<ul style="list-style-type: none"> • Allows DATA statements to be reread from the beginning • see also DATA, READ

RIGHT\$(STRINGEXPRESSION, N)	
	<ul style="list-style-type: none"> • Function that returns a string containing the rightmost n characters of the string expression. • see also LEFT\$

RTRIM\$(STRINGEXPRESSION)	
	<ul style="list-style-type: none"> • Function that returns a string with trailing spaces removed • see also LTRIM\$

SPACE\$(N)	<ul style="list-style-type: none"> • Function returns a string with n number of spaces • see also TAB
-------------------	---

SQR(N)	<ul style="list-style-type: none"> • Function that returns the square root of n • A negative expression is undefined, use ABS to safeguard from this • see also ABS
---------------	--

STR\$(NUMERIC-EXPRESSION)	
	<ul style="list-style-type: none"> • Converts numeric-expression to string • see also VAL

STRING\$(M, N), STRING\$(M, STRINGEXPRESSION)	
	<ul style="list-style-type: none"> • m, an integer expression, returning the length of string n, an integer expression (0..255) which is an ASCII value to fill string • A string expression can be used instead, the first character is used to fill the string.

TAB(N)	<ul style="list-style-type: none"> • Exact duplicate of SPACE\$ (for now...) • This is actually a device I/O function • see also SPACE\$
---------------	---

UCASE\$(STRINGEXPRESSION)	
	<ul style="list-style-type: none"> • Function returns a string that has been uppercased • see also LCASE\$

VAL(STRINGEXPRESSION)	
	<ul style="list-style-type: none"> • Function returns the numeric value of the string of digits

BASIC COMMANDS – STATUS AND TIME INFORMATION

TIME\$	
	<ul style="list-style-type: none"> • Gets the time as string --> HH:MM:SS

DATE\$	
	<ul style="list-style-type: none"> • Gets the date as string --> YYYY/MM/DD

TIME	
	<ul style="list-style-type: none"> • Gets date/time as timestamp based on 01.01.2007 00:00:00

FREEMEM	
	<ul style="list-style-type: none"> • Gets number of bytes that are available from the heap

LASTERR	
	<ul style="list-style-type: none"> • Get the last error code if there is one, or ERR_OK

BYTESREAD	
	<ul style="list-style-type: none"> • Get the number of bytes transferred by the last 'read' operation

VERSION <EXPRESSION>	
	<ul style="list-style-type: none"> • User program tells the compiler/VM which version it expects.

BASIC COMMANDS – FILE HANDLING

OPEN "ACCESS", FILENUM, "FILENAME"	
	<ul style="list-style-type: none"> • Access can be "R", "W" or "A" (read, write or append) • Filenum is any free and valid file handle (0...100) • Filename is the name of the file. • This function sets the LASTERR variable

GET FILEHANDLE, VARIABLE	
	<ul style="list-style-type: none"> • Reads from file to a buffer or variable • variable can be an array, a string, an array-element (byte), or a single variable

PUT FILEHANDLE, VARIABLE, [MAX_LENGTH]	
	<ul style="list-style-type: none"> • Write variable to file, socket or other output channel • If filehandle is -3 data is sent to the IO protocol driver asynchronously • if filehandle is -2 data is sent to the IO protocol driver as synchronous message • if filehandle is -4 data is sent to AUX RS232 (UART#3) • Variable can be a complete array, a constant or 'TIME\$', 'DATE\$' • Byte constants (0...255), int constants and int variables • Byte constants must begin with '#' • Other numeric constants are considered to be 'integer' • Array must have the 'max_length' value that specifies number of elements to write

CLOSE [FILENUM[, FILENUM]]	
	<ul style="list-style-type: none"> • If arguments are omitted, then all open files are closed • see also OPEN • This function sets the LASTERR variable

KILL "FILENAME"	
	<ul style="list-style-type: none"> • Deletes file

SEEK FILENUM, FILEPOS	
	<ul style="list-style-type: none"> • Sets position of the file for next read/write

STATUS <HANDLE>	
Description:	<ul style="list-style-type: none"> • Returns UDP/TCP-socket or file handle status
Return codes:	<p>For Files: (handle 0...100)</p> <ul style="list-style-type: none"> • if open for reading: 1 • if open for writing: 2 <p>Number of free packet buffers (handle -4)</p> <ul style="list-style-type: none"> • n 0...x (0 == no free packets) <p>For all:</p> <ul style="list-style-type: none"> • 0 if handle does not exist, is closed or error

EOF(FILENUM)	<ul style="list-style-type: none"> • Test file for End-of-file • Returns a non-zero value if the file is at the end.
---------------------	--

LOC (FILENUM)

- Returns the current file position
- If filenum is 0, this function returns the number of used bytes on disk
- see also LOF

LOF (FILENUM)

- Returns the length of file, in bytes.
- if filenum is 0, this function returns size of entire disk
- see also LOC

BASIC COMMANDS – NETWORKING

CONNECT <HANDLE> <IPADDRESS> <PORT> <TXDELAY>		
Description:	<ul style="list-style-type: none"> Allocates a socket and connects to remote IP stack 	
Parameters:	<handle>	is the connection handle, any number between 100...199 (inclusive)
	<ipaddress>	is the address of the remote IP stack (a 32 bit number)
	<port>	is the port number on that remote station is listening

LISTEN <HANDLE> <PORT> <TXDELAY>		
Description:	<ul style="list-style-type: none"> Allocates a socket and switch that socket to LISTEN mode 	
Parameters:	<handle>	is the connection handle, any number between 100...199 (inclusive)
	<port>	is the port number on that we are listening

UDP <HANDLE> <TX_IP> <TX_PORT> <RX_PORT> <TX_DELAY> <USE_CHECKSUMS>		
Description:	<ul style="list-style-type: none"> Opens UDP socket 	
Parameters:	<tx_ip>	IP address of the remote station (use RESOLV to get that number)
	<tx_port>	The port number where transmitted packets will be sent to
	<rx_port>	The port number where this socket listens for incoming packets
	<tx_delay>	Number of milliseconds after that transmitted data is sent out
	<use_checksums>	Can be 1 or 0. 1: sockets checks and generates UDP checksums for TX and RX. 0: socket does not use any checksums

RESOLV (STRINGEXPRESSION)	
	<ul style="list-style-type: none"> • Calculates binary value from IP address • That value can be used for CONNECT, LISTEN and so on • False or nonexistent parameters yield to ZERO
Examples	resolv ("192.168.0.1") evaluates to -1062731775 resolv ("www.google.com") evaluates to -782923933

GET FILEHANDLE, VARIABLE	
	<ul style="list-style-type: none"> • Reads from file to a buffer or variable • variable can be an array, a string, an array-element (byte), or a single variable

PUT HANDLE, VARIABLE, [MAX_LENGTH]	
	<ul style="list-style-type: none"> • Write variable to file, socket or other output channel • If filehandle is -3 data is sent to the IO protocol driver asynchronously • if filehandle is -2 data is sent to the IO protocol driver as synchronous message • if filehandle is -4 data is sent to AUX RS232 (UART#3) • Variable can be a complete array, a constant or TIME\$, DATE\$ • Byte constants (0...255), int constants and int variables • Byte constants must begin with '#' • Other numeric constants are considered to be 'integer' • Array must have the 'max_length' value that specifies number of elements to write

CLOSE [FILENUM[, FILENUM]]	
	<ul style="list-style-type: none"> • If arguments are omitted, then all open files are closed • see also OPEN • This function sets the LASTERR variable

STATUS <HANDLE>	
Description:	<ul style="list-style-type: none"> Returns UDP/TCP-socket or file handle status
Return codes:	<p>For TCP: (handle 101...200)</p> <ul style="list-style-type: none"> TCP_STATE_LISTENING 1 TCP_STATE_SYN_RECEIVED 2 TCP_STATE_SYN_SENT 3 TCP_STATE_FINW1 4 TCP_STATE_FINW2 5 TCP_STATE_CLOSING 6 TCP_STATE_LAST_ACK 7 TCP_STATE_TIMED_WAIT 8 TCP_STATE_CONNECTED 9 <p>For UDP: (handle 201...300)</p> <ul style="list-style-type: none"> UDP_STATE_OPENED 1 <p>For WLAN (handle -4)</p> <ul style="list-style-type: none"> WLAN connected 1 <p>Number of free packet buffers (handle -4)</p> <ul style="list-style-type: none"> n 0...x (0 == no free packets) <p>For all:</p> <ul style="list-style-type: none"> 0 if handle does not exist, is closed or error

BASIC COMMANDS – IN/OUTPUT, CAN, RS232, I2C, IO

INMODE <EXPRESSION>	
	<ul style="list-style-type: none"> • directs INPUT • 1: no input • 2: input comes from IO protocol (synchronous) • 3: input comes from IO protocol (asynchronous) • other number : input comes from open file specified by number

OUTMODE <EXPRESSION>	
Description:	<ul style="list-style-type: none"> • directs PRINT outputs • 1: no output • 2: output goes to IO protocol (synchronous) • 3: output goes to IO protocol (asynchronous) • other number : output goes into open file specified by number

AUXOPEN <HANDLE>, <P1>, <P2>, <P3>, <P4>	
Description:	<p>Activates auxilliary port. Handle can be one of the following:</p> <ul style="list-style-type: none"> • <handle> = 4: Serial port #3 (see hardware manual). Additional parameters have the following meanings <ul style="list-style-type: none"> ▪ p1 --> Baud Rate ▪ p2 --> Parity, E, O, or N (as ASCII value, use ASC) ▪ p3 --> Stopbits, 1 or 2 ▪ p4 --> Bits per character, 5, 6, 7 or 8 • <handle> = 5: IIC Master interface. Additional parameters have the following meanings: <ul style="list-style-type: none"> ▪ p1 --> Baud Rate

	<ul style="list-style-type: none"> ▪ p2 --> Slave Address ▪ p3 --> Unused, should be 0 ▪ p4 --> Unused, should be 0
Return codes:	<ul style="list-style-type: none"> • LASTERR will be set to ERR_OK(0) if AUXOPEN succeeds • LASTERR will be ERR_ARGUMENT(4) if one of the arguments is wrong • LASTERR will be ERR_REJECTED(12) if FIFO memory allocation fails

INPUT <VARIABLE>	
Description:	<ul style="list-style-type: none"> • reads variable from various sources • source can be selected with INMODE

PRINT <EXPRESSION>[, ;][<EXPRESSION>][...]	
Description:	<ul style="list-style-type: none"> • expression can either be integer or string • use semi-colon to add more expressions after • can print to various destinations (see OUTMODE)

KEYS	
Description:	<ul style="list-style-type: none"> • Get state of input ports • Currently only bit 0 is valid (External key)

SETLEDS <EXPRESSION>	
Description:	<ul style="list-style-type: none"> • Switches LEDS on or off. • Only lower 6 bits are valid • Bits 0..3 switch the internal LEDS

	<ul style="list-style-type: none"> • Bits 4 and 5 switch external LEDS
--	---

GETCAN <BYTEARRAY>	
Description:	<ul style="list-style-type: none"> • Reads buffered CAN-Frame into specified array • Array must be big enough to hold entire CAN frame, header and time stamps (== 28 Bytes) • CAN Frame payload can be found at positions 12 to 19 of array • To extract other information one can use the CANINFO function (see below)
Example:	<ul style="list-style-type: none"> • A frame without payload was fetched. -> BYTESREAD is then 0 and LASTERR is also 0
Return codes:	<p>This command modifies LASTERR and BYTESREAD</p> <ul style="list-style-type: none"> • LASTERR becomes ERR_OK(0), if a frame has been successfully fetched from FIFO buffer • LASTERR becomes ERR_NO_DATA(8) if FIFO was empty (currently no frames available) • LASTERR becomes ERR_ARGUMENT(4) if GETCAN parameter was wrong (not an array or array is too small) • If LASTERR is ERR_OK, BYTESREAD is set to the number of payload bytes (0..8)

CANINFO (NUM)	
Description:	<p>Extracts information from array that is subject to the most recent GETCAN call</p> <ul style="list-style-type: none"> • 1 --> Gets the Message ID • 2 --> Gets the ID type (0==Standard, 1==Extended) • 3 --> Gets the RTR bit (1==Frame has RTR bit set) • 4 --> Gets number of payload bytes (0..8) • 5 --> Gets the RTC second timestamp since 01/01/2007 • 6 --> Gets the millisecond timestamp • 7 --> Gets the random 32bit header value

Return codes:	<ul style="list-style-type: none"> • This Function returns -1 if either the argument was wrong or GETCAN call is missing. In that case, LASTERR becomes ERR_REJECTED(12) • If everything's fine, LASTERR will return ERR_OK(0)
---------------	--

PUTCAN <BYTEARRAY>	
Description:	<ul style="list-style-type: none"> • Immediately sends CAN frame (without buffering) from byte array (== 16 Bytes) • Array must be big enough to hold entire CAN frame, header and time stamps (== 28 Bytes) • Payload of that CAN frame are the bytes at positions 12 to 19 of array
Return codes:	<ul style="list-style-type: none"> • LASTERR becomes ERR_ARGUMENT(4) if PUTCAN parameter was wrong (not an array or array is too small) • LASTERR becomes ERR_REJECTED(12) if the underlying driver could not send the frame • In this case, try again a little later • LASTERR becomes ERR_OK(0) if frame was successfully transmitted

SETCAN <BYTEARRAY>,<INDEX>,<VALUE>	
Description:	<p>Sets entries into byte array that is intended to be send as CAN frame. Index can be:</p> <ul style="list-style-type: none"> • 1 to set the message ID • 2 to set the ID type (0==Standard, 1==Extended) • 3 to set the RTR bit (1== RTR is set) • 4 to set the number of payload bytes (0...8) • 5 to set the RTC second timestamp (seconds since 01/01/2007) • 6 to set the millisecond timestamp • 7 to set the random 32bit header value
Return codes:	<ul style="list-style-type: none"> • LASTERR will be set to ERR_OK(0) if SETCAN succeeds • LASTERR will be ERR_REJECTED if SETCAN detects a faulty

	parameter
--	-----------