Script language for ersky9x

23-Jan-2019 12:00

Directory structure:

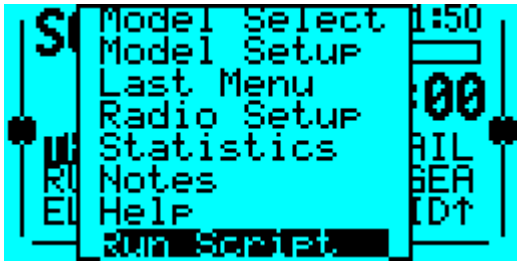
Script filenames are limited to 6 characters and need an extension of ".bas".
/SCRIPTS – put standalone scripts here
/SCRIPTS/TELEMETRY – put scripts that display on the custom telemetry screens here.
/SCRIPTS/MODEL – put "background" scripts here

Script Types:

A model background script is selected in the Model Setup|General menu. It is loaded when the model loads and always runs unless a standalone script is run.

A telemetry script is selected in the custom telemetry configuration. It is loaded when the model loads and always runs unless a standalone script is run. Use the sysflags() function to detect if the custom telemetry screen is currently visible.

A standalone script is run from the main popup, or the "Scripts" menu (STATISTICS menus).



When it runs, it "takes over" the display and stops all other scripts from running. EXIT LONG will terminate the script, when any model specific scripts are re-loaded.

Each script is run every 10mS.

Script Language:

All individual variables are 32-bit, signed integers.
Arrays are either 32-bit, signed integers, or 8-bit unsigned integers.
Arrays must be declared and therefore dimensioned before use.
Arrays may only be one-dimensional.
Numeric constants may be in decimal, octal (0123), hex (0xAB) or binary (0b1010).
Names (commands, variables, labels etc.) are case sensitive.

assignment operator:
One of =, +=, –=, *=, /=, %=

comparision operator:
One of: =, #, <, >, <=, >=, where '#' represents "not equal"

var:
A variable name is alpha-numeric and begins with an alphabetic character.
If the variable is an array, then the array index is enclosed in [] characters.

label:
A label is alpha-numeric, begins with an alphabetic character, is the first item on a line, and ends with a ':'

character. The ':' is optional if the label starts in the first column and has no characters after it on the line.

Strings:
Strings are delimited by “ characters and may include:
\0 – the zero character
\123 – an octal character
\x3F – a hexadecimal character
\” - the “ character
any of \r \n \t \f \b for <cr> <newline> <tab> <formfeed> <backspace>
A string may be indexed by appending an array subscript to it, so the string “Hello”[2] returns the string “llo”. This is useful if you have several sub-strings in one declaration, e.g. "String1\0String2\0String3"[0] gives “String1”, while "String1\0String2\0String3"[8] gives “String2” and "String1\0String2\0String3"[16] gives “String3”.
Strings are limited to a maximum of 254 characters.

array:
syntax:
For a byte array:
array byte <identifier>[<numeric constant>]
For a 32-bit integer array:
array <identifier>[<numeric constant>]
or
array int <identifier>[<numeric constant>]

Language Elements:

let
syntax: let <var> <assignment operator> <expression>
The “let” text is optional, a line beginning with <var> is assumed to be a let statement.

if
syntax:
if <expression> then goto|gosub <label>
or
if <expression> <comparison operator> <expression> then goto|gosub <label>
or
if <expression> <comparison operator> <expression> then statement
or
if <expression>
<statement>
. . .
<statement>
end
or
if <expression>
<statement>
. . .
else
<statement>
. . .
end
or
if <expression>
<statement>
. . .

elseif <expression>
<statement>
. . .
end

You may have many elseif statements, and an else as well,so:
if <expression>
<statement>
. . .
elseif <expression>
<statement>
. . .
elseif <expression>
<statement>
. . .
else
<statement>
. . .
end
is possible.

goto
syntax: goto <label>

gosub
syntax: gosub <label>

return
syntax: return

while
syntax: while <expression>
…..
end

break
syntax: break
Used to break from a while loop. It is really a goto to the end of a while loop.

rem
remark, ignore the line

stop
syntax: stop
The "stop" instruction indicates this run of the script has ended, but the script should be run again.

end
syntax: end

finish
syntax: finish
The "finish" instruction indicated the script is complete and should not run again, indeed any RAM it is using is then available for another script.

const
syntax: const <text_name> <number>
Defines “text_name” as a number.

Built in functions:

Coordinates (x,y) on the display are measured from the top left (0,0), x across the display and y down the display).

abs
syntax: abs(<expression>)
returns the absloute value of the expression

not
syntax: abs(<expression>)
returns the ones complement of the expression

drawclear
syntax: drawclear()
Clears the display.

drawtext
syntax: drawtext(<expression>, <expression>, “text” [,<expression>] [,<expression>])
drawtext(x, y, text [,attribute][,length])

drawnumber
syntax: drawnumber(<expression>, <expression>, <expression> [,<expression>])
drawnumber(x, y, number [,attribute])

drawline
syntax: drawline(<expression>, <expression>, <expression>, <expression>[,<expression>])
drawline(x1, y1, x2, y2[,colour])

playnumber
syntax: playnumber(<expression>, <expression>, <expression>)
playnumber(number, attribute, units)

getvalue
syntax: getvalue(<expression>|”text”)

Telemetry names:
A1= ,A2= ,RSSI,TSSI,Tim1,Tim2,Alt ,Galt,Gspd,T1= ,T2= ,RPM ,FUEL,Mah1,Mah2,
Cvlt,Batt,Amps,Mah ,Ctot,FasV,AccX,AccY,AccZ,Vspd,Gvr1,Gvr2,Gvr3,Gvr4,Gvr5,Gvr6,
Gvr7,Fwat,RxV ,Hdg ,A3= ,A4= ,SC1 ,SC2 ,SC3 ,SC4 ,SC5 ,SC6 ,SC7 ,SC8 ,RTC ,
TmOK,Aspd,Cel1,Cel2,Cel3,Cel4,Cel5,Cel6,RBv1,RBa1,RBv2,RBa2,RBm1,RBm2,
RBSV,RBST,Cel7,Cel8,Cel9,Cl10,Cl11,Cl12,Cus1,Cus2,Cus3,Cus4,Cus5,Cus6,
LAT,LONG

Control names: (Rud, Ele, Ail, Thr, P1,P2,P3, PPM1-PPM8, CH1-CH24)

Special values of expression (used for debugging and performance testing):
1000-1019, reads physical I/O ports.
1020, Mixer rate.
1021, Idle percentage.

Returns the value requested.

drawpoint
syntax: drawpoint(<expression>, <expression>)
drawpoint(x, y)

drawrectangle
syntax: drawrectangle(<expression>, <expression>, <expression>, <expression>[,<expression>])
drawrectangle(x, y, width, height [,percent])

drawtimer
syntax: drawtimer(<expression>, <expression>, <expression>[, <expression>])
drawtimer(x, y, seconds[, attribute])

idletime
This function will be removed as the value may be obtained using “getvalue(1021)”.
syntax: idletime()
returns the percentage of time for which the idle process is running.

gettime
syntax: gettime([<expression>])
if no parameter then returns the elapsed time in units of 10mS
gettime(0) returns the year (e.g. 2018)
gettime(1) returns the month (1-12)
gettime(2) returns the date (1-31)
gettime(3) returns the hour (0-23)
gettime(4) returns the minute (0-59)
gettime(5) returns the second (0-59)

sysflags
syntax: sysflags()
returns the execution state:
Bit 0 set if display is available
Bit 1 set if running as a standalone script
Bit 2 set if running as a telemetry script
Bit 3 is set to indicate the script is resuming:
To prevent a script from taking over the processor from normal operation, the number of script statements that are executed each time it runs is limited (to 150 currently). If a script reaches a "stop" statement, then it stops executing, the display will be updated if in use, and the script will run from the beginning next time it runs. If a script runs for over 150 statements, then it is paused, the display is NOT updated, and it will continue from that point next time it runs. This is when the bit 3 will be set.

settelitem
syntax: sysflags(“text”, <expression>)
Sets the specified telemetry item (text is a telemetry name). Note that only actual telemetry items may be set, SC1-8, Gvr1-7, and radio speific items like battery voltage and timers may not be set.

strtoarray
strtoarray(<arrayReference>,"text") OR
strtoarray(<arrayReference>,<arrayReference>)
initialises a byte array from a string or copies a string from one byte array to another.
Returns number of bytes copied including the null terminator.

getswitch

getswitch("name")
gets the current state (on or off) of a switch, physical or logical
getswitch("AIL") returns the state of the AIL switch as 0 or 1 (9X radios)
getswitch("SCv") returns the state of the SCv as 0 or 1 (FrSky radios)

setswitch
setswitch("name",<expression>)
sets a (unused) logical switch to off (expression = 0) or on (expression != 0), as long as the switch function is defined as "----"

playfile
syntax: playfile("fname")
plays the file "fname.wav" from the /voice/user directory

sportTelemetrySend
syntax: sportTelemetrySend(<expression>, <expression>, <expression>, <expression>|<byteArray>)
sportTelemetrySend(PhyId, Command, AppId, data)
"data" may be either a variable, or a byte array.

sportTelemetryReceive
syntax: sportTelemetryReceive(<variable>, <variable>, <variable>, <variable>)
sportTelemetryReceive(PhyId, Command, AppId, data)

getrawvalue
syntax: getrawvalue(<expression>|"text")
Does the same as getvalue, but, where appropriate, instead of returning a percentage value it returns the actual value. For example, for a stick input, a value between -1024 and +1024 is returned instead of -100 t +100.

killevents
syntax: killevents(event)
Prevents any further events for the value "event" from occuring.

bitfield
syntax: bitfield(value, start, width)
returns part of value starting at bit "start" and "width" bits. Bits are counted from the least significant bit.

power
syntax: power(value, exponent)
Returns "value" raised to the power of "exponent". The exponent is limited to a maximum value of 20.

crossfirereceive
syntax: crossfirereceive(length, command, data)
"data" must be a byte array of sufficient size to hold the complete crossfire packet
If a packet is available, then "length", "command" and "data" are filled with the packet and the function returns 1. If no packet is available, or the "data" array is too small, the function returns 0.

crossfiresend
syntax: crossfiresend(command[, length, data])
"data" must be a byte array
If command is 0xFF, then this returns 1 if a packet may be sent or 0 if not.
Returns 1 if the packet was queued for transmission and 0 if the queue is busy.
The length is the length of the data, not including the command byte.

sysstrtoarray
syntax: sysstrtoarray(array, type)
Fetches a system string to a byte array.
array is a byte array.
type is 0 for the current model name and 1 for the radio owner name.

popup
syntax: popup(option_list, mask, width)
Returns 0 while nothing selected, 1 to 16 if an item selected and 99 if EXIT is pressed to cancel the popup.
The option_list is a string with each option separated by a null ('\0') character, e.g."Opt 1\0Opt 2\0Opt 3\0Opt 4"
The mask is a bitfield of 16 bits that indicates which of the options are to be displayed in the popup, with the least significant bit indicating the first option e.g. a mask of 13 (0x0D, 0b00001101), would cause the above list to display Opt1, Opt3 and Opt4. Note the return value always returns the exact position in the list of a selected item, so with a mask of 13, only values 1, 3 or 4 will be returned.

```
if init = 0
 init = 1
end
drawclear()
drawtext(20, 16, "Hello", 0)
rem Pressing MENU starts the popup
if Event = EVT_MENU_BREAK
rem But only if it isn't already running
 if pop = 0
  pop = 1
  rxres = 0
rem Setting Event to 0 removes the EVT_MENU_BREAK event so the popup doesn't “see” it
  Event = 0
 end
end
rem Test if the popup is running
if pop
 result = popup( "Opt 1\0Opt 2\0Opt 3\0Opt 4", 0x0D, 6)
rem If anything non-zero is returned, terminate the popup
 if result
  rxres = result
  pop = 0
 end
end
```

serialreceive
syntax: serialreceive(length, data)
“data” is a byte array
This function receives serial data from either bluetooth or COM2. Which depends on the setings of Btfunction as “Script” or “COM2 Function” as “Script”. If both are set as “Script” then bluetooth is used. If data is available in the serial fifo, then up to “length” bytes are read into the “data” array and the function returns the number of bytes read. If no data is available, or the “data” array is too small, the function returns 0.

serialsend
syntax: serialsend(length, data)
“data” is a byte array

This function sends serial data to either bluetooth or COM2. Which depends on the setings of Btfunction as “Script” or “COM2 Function” as “Script”. If both are set as “Script” then bluetooth is used.
If length is 0, then this returns 1 if a data may be sent or 0 if not.
Length may be a maximum of 32.
If length is not zero, then “length” bytes are sent out over the serial.

The following file related functions are only available to “StandAlone” scripts.

directory
syntax: directory(path, extension)
This sets up the list of files from the specified directory path with the given extension.

fileselect
syntax: fileselect(filepath, size)
This allows you to select a file from the list created using the directory() function. If you select a file, then the size of the file is placed in the variable you supply (called size here).
Returns 0 = no user input, 1 = file selected, 2 = exit with no selection, 3 = file “tagged”.

fopen
syntax: fopen(filename, mode)
This opens the file with the (full path) of filename. mode is 0 for read and 1 for write. Opening a file for write will clear any existing data in an existing file.
Note that you may only have a single file open at a time.
Returns 0 = not opened, 1 = opened OK.

fread
syntax: fread(length, buffer, number_read)
Reads length bytes from the open file, storing them in buffer, and setting number_read to the number of bytes actually read.
Returns 0 = read failed, 1 = read OK.

fwrite
syntax: fwrite(length, buffer, number_written)
Writes length bytes from buffer to the open file, and sets number_written to the number of bytes actually written.
Returns 0 = write failed, 1 = write OK.

fclose
syntax: fclose()
Closes an open file.

bytemove
syntax: bytemove(destination, source, length)
Copies bytes from one byte array to another.
Returns number of bytes moved.

alert
syntax: alert(“text” [,<expression>])
Displays either an “ALERT” or a “MESSAGE” with the specified text. If the expression is not present or is zero, then the display type is “ALERT”. If the expression is not zero, then the display type is “MESSAGE”.

Constants:
For display:
LEFT – Display numbers left justified (the default is right justified).

PREC1 – Display number with 1 decimal place.
PREC2 – Display number with 2 decimal places.
DBLSIZE – Display using double size text.
INVERS – Display highlighted.
BLINK – Display with highlight flashing.
LCD_W – Display width in pixels.
LCD_H – Display height in pixels.

For Event:
EVT_MENU_BREAK
EVT_MENU_LONG
EVT_EXIT_BREAK
EVT_UP_BREAK
EVT_DOWN_BREAK
EVT_UP_FIRST
EVT_DOWN_FIRST
EVT_UP_REPT
EVT_DOWN_REPT
EVT_LEFT_FIRST
EVT_RIGHT_FIRST
EVT_BTN_BREAK – Rotary encoder button.
EVT_BTN_LONG – Rotary encoder button.

Error Numbers:
1 – Duplicate label
2 – Syntax (line index)
3 – Syntax
4 – Too many variables
5 – Missing ')'
6 – Divide by 0
7 – Missing THEN
8 – Return without gosub
9 – Invalid function name
10 – Too large
11 – Exceed dimension size
12 – Too many nested calls
13 – Break without while
Error numbers are returned with 100 added if detected at run time.

getvalue() numeric parameters:
120 P4 or SR
121 P5
122 P6

Writing for different radios:
Some radios have buttons for navigation, while others use an encoder. Rotary movement of the encoder is translated into up and down events, but only as a “FIRST” event, not ea “BREAK” event. Unless a script has a specific requirement to use EVT_UP_BREAK or EVT_DOWN_BREAK, using EVT_UP_FIRST or EVT_DOWN_FIRST maintains compatibility with an encoder. A radio that has a “PAGE” button (e.g. FrSky X7 and X9E) that button maps to the LEFT button.