Script language for ersky9x

15-Jul-2018 22:00

Directory structure:

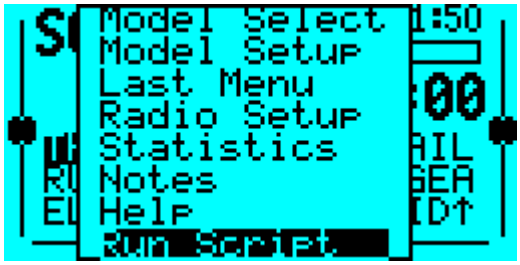
Script filenames are limited to 6 characters and need an extension of ".bas".
/SCRIPTS – put standalone scripts here
/SCRIPTS/TELEMETRY – put scripts that display on the custom telemetry screens here.
/SCRIPTS/MODEL – put "background" scripts here

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>

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

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)

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
syntax: idletime()
returns the percentage of time for which the idle process is running.

gettime
syntax: gettime()
returns the elapsed time in units of 10mS

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")
initialises a byte array from a string

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>)
sportTelemetrySend(PhyId, Command, AppId, data)

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 ix 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
```

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
Error numbers are returned with 100 added if detected at run time.

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