a-Series **b**asic **c**ompiler
Date: 01. 09. 2011

abc is a command subset from a BASIC version called „Yabasic“ (at the moment V2.722). Except from the restrictions listed below it is 100% compatible to it, so you can use the original binaries to test your programs under Windows or Linux (downloads and documentation from www.yabasic.de).

Requirements:

- Running abc needs at least 300 kByte of free memory to work smoothly. Parts of this memory are not being released after finishing the program, so restarting abc is faster.

Restrictions:

- No mouse functions (touch coordinates on EOS are mapped to MOUSEX and MOUSEY commands)
- Window functions work differently (no single window on big screen, but mapping of window to LCD possible)
- No PRINT AT
- No COMPILE, no libraries
- No BEEP and BELL, however new X2-based machines have SOUND „name“ command to play sounds
- A/A+ only: abc and JScript work with cooperative multitasking, i.e. a complex JScript command can delay abc commands and vice versa
- The content of a file has priority over abc output to JScript. This way abc can e.g. send „M l lbl;sample“ to JScript. However this means that when a file is executed from card abc output is delayed until the file has been completely read and closed by Jscript!

Import differences to Yabasic PC versions:

- To switch off the ESC command interpretation of JScript you can use POKE „transparent“,0 or 1. However all data which is already in the input buffer (64 kwords on A/A+, variable on EOS) has been filtered. So do not send data with ESC in it before the POKE command has been executed!
- abc works internally with Unicode, so multilingual data processing is no problem for abc programs. abc can also handle chr$(0) within a string which is interpreted as string end in yabasic.
- Programs can be stopped by total CANCEL (pressing CAN more than three seconds on front panel), this can be disabled by ON INTERRUPT command.
- No SYSTEM$() function.

Other restrictions:

- Printing ESC sequences to JScript has no effect

Window-Handling:

abc uses a hidden window which can be (partially) mapped to the front panel LCD. The printer handles the window as a bitmap with 8 bit indexed colours. So each dot can have a value of 0 (black) to 255 (white). During mapping to the LCD, each colour is mapped according to its brightness which is predefined as grayscales, i.e. 128 to 255 gives white pixels, 0 to 127 black pixels. The mapping can be changed with the POKE command to RGB colors which are useful if you want to write the graphic to the card.

- 'OPEN WINDOW width, height' opens the window. Only one is allowed. As this window is stored internally in standard memory, define it only the size you really need. (E.g. a window 100,100 takes 10kByte memory). For the A3-LCD a window of 120 by 32 is sufficient, A+ needs 128x64 and EOS needs 160x255.
- There's only one font (16 dots high), variable width with support of latin, greek, cyrillic, hebrew and arabic scripts. The origin is in the upper left corner of the first character's bounding box. For right-to-left writing countries, the origin is in the upper right corner.

New functions compared to Yabasic:

- POKE „color#",rgb, #=1 to 254, 0 stays always black, 255 stays always white, e.g. POKE „color#15",dec(„ff0000") sets color number 15 to red
- WINDOW TRANSFER TO „name" transfers the window content to a JScript image „name" which can be used e.g. with the I command.
- WINDOW TRANSFER FROM „name" loads the window with a JScript image. If the windows and image size are not identical the result is clipped.
- WINDOW WRITE TO „name" saves the actual window as PNG on the memory card.
- WINDOW READ FROM „name" load a PNG into the actual window. Path names are allowed here.
  The window has to be big enough to hold the image, else loading will fail! Supported formats are:
  * grayscale 1 to 8 bits per pixel
  * paletted images 8 bits per pixel
- JGET$ and JPUT are used to exchange data between JScript and abc. The exchange is synchronized, so you can use abc as a JScript-function. Use always as a pair, else execution of JScript and/or abc can be blocked!
- abc has a command to check for the existence of files or devices: EXISTS(„filename") or EXISTS(„/dev/rawip")

Restrictions compared to Yabasic:

- No CIRCLE command.
- No BITBLT, GETBIT$ and so on.
- WINDOW ORIGIN is not supported, i.e. the origin 0,0 is always in the upper left corner.

The modifiers CLEAR and FILL have the following results (shown for the RECT command):
RECT: frame in foreground color
CLEAR RECT: frame in background color
FILL RECT: filled area in foreground color
CLEAR FILL RECT: filled area in background color

PEEK Variables:

„direction“	I	Direction of paper movement: 1 if forward, -1 if backwards and 0 if paper is not moving
„firmware“	S	Returns the firmware version of the machine („e.g. „V3.05 (Sep 13 2006)“)
„freememory“	I	Returns the free main memory (available for abc or Jscript)
„imageheight:name“	I	Returns the height of an image „name“ in dots, 0 if not known
„imagewidth:name“	I	Returns the width of an image „name“ in dots, 0 if not known
„jphase“	I	Phase of JScript-Interpreter: 0 waiting for label definition 1 in process of label definition 2 during printing 3 standby, waiting for new job or new data for old one
„line“	I	Number of the actually printed label
„machine“	S	Returns the type and name of the printer (e.g. „A4+/300“).
„manufacturer“	S	Returns the manufacturer of the machine (e.g. „cab“).
„mlength“	F	Measured length of last label distance (mm), if not known it is 0
„os“	S	Delivers „cab A-Series“ or „EOS“ - only for compatibility with Yabasic
„peelpos“	I	Returns a 1 if the label is in peel-off position.
„peri“	S	Returns name of peripheral (similar to JScript q p command).
„resolution“	F	Resolution of printer in dpi
„rfid_rssi“	I	Returns the signal quality of a detected RFID tag. Range is 0 to 100.
„sec70“	I	Time in unix format - i.e. seconds since Jan 1, 1970.
„serial“	S	Returns the serial number of the PCB.
„slength“	F	Stored label distance (mm), if not known or invalid it is 0 this is effectively the distance of the last defined label before being switched off
„source“	S	Name of last data source: „RS232“, „RS422“, „RS485“, „IEEE1284“, „RAWIP“, „USB“, „FTP“, „LPD“, „ABC“, „SOAP“, „BLUETOOTH“, „UNKNOWN“
„status“	S	State of the printer (same as ESC s answer string)
„ticks“	I	Timer tick since startup of printer in 1/128th seconds, on EOS in 1/1000s
„user“	S	Returns the content of the non-volatile user space (A+/Mach only).
„version“	F	Version of Yabasic
„width“	F	Maximum print width in mm
„winf“	S	Returns the content of the WINF buffer (similar zu ESC i command).
„xinput“	I	Status of the peripheral connector input pin (XSTART)
„xoutput“	I	Reads actual peripheral control bits
„xstatus“	S	Extended state of the printer (same as ESC z answer string, but without CR)
„iobox“	I	Returns the input state of the I/O box on USB. Input data is binary ORed, values ranging from 1 for input 1 to 8 for input 4.

Type: S=string, I=integer, F=float

Keep in mind that PEEKs which return a string need the PEEK$() function, whereas PEEKs which return float or integer need PEEK().

POKE Variables:

„backlight“	I	Controls the backlight of the LCD if „lcd“ is 1. 1 is on, 0 is off, 2 is controlled by JScript (Default).
„bcolor“	I	Sets the background color for abc window operations.
„bypass“	I	Value: 0 or 1. 1 allows data from interfaces to go directly to JScript.
„color#x“	I	Sets the RGB value for color #x. x is valid from 1 to 254. Color 0 (black) and 255 (white) cannot be modified.
„fcolor“	I	Sets the foreground color for abc window operations.
„httpswap“	S	Can be used to swap the normal root directory and the memory card on the webserver. E.g. POKE „httpswap“,“/secret“ moves the applet to /secret/index.htm and /card/index.htm to /index.htm.
„iobox“	I	Sets the output state of the I/O box on USB. Returns error if not available. Output data is binary ORed, values ranging from 1 for output 1 to 8 for output 4.
„key“	I	Puts a character into the key buffer. E.g. POKE „key“,dec(„F001“) simulates pressing the MODE key.
„lcd“	I	Controls the source for the LCD. 0 is standard, JScript content. 1 is the abc window.
„led“	I	Controls the state of the front panel LEDs (if „lcd“ is 1). Bit coded: 1=Cancel 2=Mode (A-Series), Error (M-Series) 4=Feed 8=Pause 16=Arrows (A-Series only) A+/Mach4 and newer machines: 1=Menu 2=Cancel 4=Feed 8=Pause 16=Enter 32=Up arrow 64=Left arrow 128=Right arrow 256=Down arrow EOS: no LEDs available
„ledmask“	I	Masks the LEDs to be lit. Independent of „lcd“-value. Same bit coding as „led“. A 0 masks the respective LED. Not available on EOS.
„lcdx“, „lcdy“	I	Offset for the LCD in the abc window. Works only if the window is bigger than the LCD.
„nice“	I	Sets the multitasking priority of abc vs. JScript. Ranges from 1 (JScript fast) to 20 (abc fast). Default is 10.
„print_with_verify“	I	Controls the usage of a barcode scanner by the printengine of an enabled machine. Set to 1 for the printengine to wait for „scanresult“ after each label.
„read_controls“	I	Value: 0 or 1. 1 allows control characters to pass thru INPUT or INKEY$. All characters are passed to abc, including the character terminating the input line (e.g. CR). (This CR can be removed e.g. with TRIM$.)
„scanresult“	I	Sets the result of the barcode verification scan: 1 Good, apply the label 2 Bad, display error (depending on user decision on front panel reprint will occur or not) 3 Bad, keep label on liner (reprint will occur)

		4 Bad, put label in recycle position (if hardware available, reprint will occur) 5 Bad, put label on product (reprint will occur) 3+8 Bad, keep label on liner (no reprint) 4+8 Bad, put label in recycle position (if hardware available, no reprint) 5+8 Bad, put label on product (no reprint)
„syserror“	S	Puts the first character of the string into the error message puffer. Allowed characters are the same as in the ESC s response.
„transparent“	I	Value: 0 or 1. 1 switches off ESC-command interpretation
„user“	S	Writes a value into the non-volatile user space (A+/Mach only). Max. 31 UTF-8 characters allowed.
„wakeup“	I	Wakes the printer resp. prevents it from falling asleep.
„widget“	S	Puts text into abc debug widget. Up to four characters printable (only digits and upper case letters). (Only available on A+/Mach4 machines.)
„winf“	S	Writes a value into the „winf“-Buffer.
„xoutput“	I	Status of the peripheral connector control bits (output) Note: you have to set the peripheral mask to 0 (x m command) before!
„xstart“	I	Triggers the print of a label (analog to start input signal) on supported hardware (e.g. Hermes+).

Type: S=string, I=integer, F=float

Streams:

Filename	Direction/Bit	Description
„/dev/rs232:baud,handshake,parity,stopbits“		
	I/O,8	baud[1]: 1200-230400, handshake: ---,RTS/CTS,XON/XOFF, parity: N,E,O, Stopbits: 1,2
„/dev/ieee1284“	I/O,8	bidirectional parallel interface
„/dev/rs422:baud,handshake,parity,stopbits“		
	I/O,8[1]	rs-422 interface, baud: 1200-230400, handshake: ---,XON/XOFF parity: N,E,O, Stopbits: 1,2
„/dev/rs485:baud,parity,stopbits“		
	I/O,8	rs-485 interface, baud: 1200-230400 parity: N,E,O, Stopbits: 1,2
„/dev/usb“	I/O,8	USB-Client
„/dev/rawip“	I/O,8	raw-IP interface
„/dev/lpr“	I,8	lpr server
„/dev/panel“	I,16	input from front panel keys, key values are $F001 Mode $F002 Formfeed $F003 Cancel $F004 Pause $F090 Cancel longer than 3 seconds $F100 OK-Key (A+ and Mach4) $F101 OK longer than 2 seconds (A+ and Mach4)
„/dev/keyboard“	I,16	input from external keyboard *There are too many keycode to list them here - please use the program listed in the sample section of this document.*
„/dev/jscript“	I,16	JScript-Interpreter - needed for reading back answers
„/card/filename.ext“	I/O*,8/16	file from memory card
„/iffs/name.ext“	I,8/16	file from internal memory
„mailto:address“	O,8	Writes an email to the specified address. An SMTP-Server address and a return address has to be set in the setup! The subject is the first line printed into the stream.
„sql:ip,port“	I/O,16	Database Connector, always Unicode You have to open two streams, one for reading, one for writing. After printing the SQL query, you have to input the result, even if you don't need them, e.g. after INSERT. The query is sent at the moment to do the first INPUT on the reading stream.

* no random writing within a file, only append or overwriting, according to the filename extension the files are automatically sorted into the appropriate directories (i.e. /images, /labels, /fonts and /misc) on the card

[1] note: on A3 setting the baudrate on RS-422 sets the RS-232 baudrate too and vice versa!

Modes:

„r“, „w“, „a“	read, write and append (file reading and writing automatically transforms Unicode to ASCII and vice versa according to selected codepage, reading a Unicode or ASCII file is automatically detected)
„rb“, „wb“, „ab“	read, write and append without transforming (file reading and writing uses only low-byte of e.g. string)
„wu“, „au“	write and append using Unicode

Notes:

– Some streams like „/dev/panel“ are always Unicode-streams. Using ‘b’ or ‘u’ modifiers can

have strange effects!

- Writing to an interface (e.g. /dev/rs232) will fail if the printer cannot send the data. There's a timeout of 10 seconds.
- Opening an interface as file stops ESC interpretation on this device.
- abc has an additional command called FLUSH which enables you to clear the input puffer of /dev-streams in read mode (e.g. FLUSH #1 when 1 ist /dev/rawip). FLUSH #0 clears standard input.
- abc has an additional command to erase files: ERASE „name“
- on EOS, /dev/keyboard works only if a window is opened and displayed, some keycodes have changed compared to old printers

Communication with Web Browsers:

A-Series printers have a web server which is usually used for administration, but can also be used to access data like images or HTML pages from the card. So it is only logical to seek a way to transmit data from the browser *to* the printer. This is normally done by CGI scripts using forms. We do it the same way :-) You can however not define CGI scripts your own, but we provide a way to get form data into your abc program:

HTML
You simply define a form in your HTML page which uses get_form.cgi as ACTION. Example:

```
<form action="/get_form.cgi" method="post">
<input type="hidden" name="nextpage" value="thanks.htm">
<input type="text" name="example">
<input type="submit" value="Send data">
</form>
```

This form lets the user enter some data in a text field called „example". After clicking the „Send data" button, the form content is sent from the browser to the web server and parsed there. Then the extracted data is put into the input buffer which can be read by abc or directly by JScript. There are two special field names available:

- nextpage — this defines the name of the html page which is loaded after sending the form. Default is index.htm.
- jscript — Can be used to send a JScript command before the data. So you can e.g. send a „M l lbl" command before the data of the form.

A more complex example showing most of the possibilities of the CGI interface is the „cinema ticket" program.

Examples

Small program to print a 100mm long ruler with 1mm markings:

```
; Test label for ruler
J
S l1;0,0,68,71,104
G 0,10,0;L:100,.1
<ABC>
FOR X=0 TO 100
      IF MOD(X,10) = 0 THEN
            PRINT "G ",X,",10,270;L:4,.1"
      ELSE
            PRINT "G ",X,",10,270;L:2,.1"
      END IF
NEXT X
END
</ABC>
A 1
```

Small program to print a text in a circle:

```
; Test label for rotated text
J
S l1;0,0,68,71,104
<ABC>
A$="Rotated text with Euro sign: "+CHR$(DEC("20AC"))+" "
N=LEN(A$)
D=360/N
FOR I=1 TO N
      W=((I-1)*D)/180*PI
      X=50-25*COS(W)
      Y=30-25*SIN(W)
      R=90-(I-1)*D
      IF R<0 THEN
            R = R + 360
      ENDIF
      PRINT "T ",X,",",Y,",",R,",3,6,b;",MID$(A$,I,1)
NEXT I
PRINT "T 0,30,0,3,5;[J:c100]",date$
PRINT "T 0,38,0,3,5;[J:c100]",time$
END
</ABC>
A 1
```

Small program to show usage of local and static variables. Uses ASCII dump mode to show what happens:

```
a
<ABC>
for a=1 to 4:stars():next a
sub stars()
    static a$
    local b$
    a$=a$+"*"
    b$=b$+"*"
    print "; ",a$," ",b$
end sub
</ABC>
```

Small program to show ON GOSUB. Uses ASCII dump mode to show what happens:

```
a
<ABC>
for number=0 to 6
      on number+1 gosub sorry,one,two,three,four,five,sorry
next number
end
label sorry:print "; Sorry, can't convert ",number:return
label one:print "; 1=one":return
label two:print "; 2=two":return
label three:print "; 3=three":return
label four:print "; 4=four":return
label five:print "; 5=five":return
</ABC>
```

Small program to show READ,DATA and RESTORE. Uses ASCII dump mode to show what happens:

```
a
<ABC>
restore names

read maxnum
dim names$(maxnum)
for a=1 to maxnum:read names$(a):next a
for number=0 to 10
      if (number>=1 and number<=maxnum) then
            print "; ",number,"=",names$(number)
      else
            print "; Sorry, can't convert ",number
      endif
next number
error "Program finished"

label names
data 9,"one","two","three","four","five","six"
data "seven","eight","nine"
</ABC>
```

Small program for measuring the distance between two label edges:

```
<ABC>
DO
      REM read measured length
      dy=PEEK("mlength")
      IF dy>0 BREAK
      PRINT "f"
      WAIT 0.25
      REM wait until standing again REPEAT
      UNTIL (PEEK("direction")=0)
LOOP
PRINT "J"
PRINT "S l1;0,0,",dy-2,",",dy,",100"
PRINT "T 0,10,0,3,5;Measured distance: ",dy,"mm"
PRINT "A 1"
</ABC>
```

This program demonstrates the differences for file handling (connect the memory card/USB drive to your PC and use a hex editor to see the different results):

```
<ABC>
a$="Hello "+CHR$(DEC("20AC"))
OPEN 1,"test.dat","w"
PRINT #1 a$
CLOSE 1
OPEN 1,"testu.dat","wu"
PRINT #1 a$
CLOSE 1
OPEN 1,"testb.dat","wb"
PRINT #1 a$
CLOSE 1
</ABC>
```

This program does also writing using files but on the RS-232:

```
<ABC>
a$="Hello "+CHR$(DEC("20AC"))
OPEN 1,"/DEV/RS232:57600,RTS/CTS","w"
PRINT #1 a$,chr$(13);
FOR i=1 TO 10
PRINT #1 i,chr$(13);
NEXT i
CLOSE 1
</ABC>
```

This demonstrates the file path and name handling of abc (it is necessary to have test.dat on the card, e.g. from the last demo program):

```
<ABC>
PRINT "a"
PRINT "; test.dat: ",exists("test.dat")
PRINT "; test.dat: ",exists("TEST.DAT")
PRINT "; test.dat: ",exists("/card/misc/test.dat")
PRINT "; test.dat: ",exists("/CARD/TEST.dat")
PRINT "; test2.dat: ",exists("test2.dat")
</ABC>
```

If you want to know the dimensions of an image try this:

```
a
<ABC>
print "M l img;sample"
wait 1
b=0
h=0
DO
b=PEEK("imagewidth:SAMPLE")
h=PEEK("imageheight:SAMPLE")
IF b>0 AND h>0 BREAK
LOOP
PRINT "; Width: ",b
PRINT "; Height: ",h
PRINT "; Free memory: ",PEEK("freememory")
</ABC>
```

Simple program to show the capture of interface data, parsing it, extracting the data and sending it forward to the JScript interpreter:

```
J
S l1;0,0,68,71,104
T:t1;20,10,0,3,8;
T:t2;20,20,0,3,8;
T:t3;40,40,0,3,8;
<ABC>
label start
line input a$
if left$(a$,15)="194300301480070" then
      print "R t2;",mid$(a$,16)
endif
if left$(a$,15)="194300300580172" then
      print "R t3;",mid$(a$,16)
endif
if left$(a$,15)="194300301970073" then
      print "R t1;",mid$(a$,16)
endif
if a$="Q0001" then
      print "A 1"
endif
goto start
</ABC>
```

This is the original Datamax DPL data stream sent from Easylabel:

```
M3000
<STX>d
<STX>e
<STX>f260
<STX>O0220
<STX>V0
<STX>L
D11
PA
SA
H10
z
194300301480070Rot
19430030058017248
194300301970073Bernd
W
Q0001
E
<STX>L
D11
PA
SA
H10
z
194300301480070gelb
19430030058017248
194300301970073Bertha
W
Q0001
E
```

Program to read keyboard codes:

```
<ABC>
OPEN 1,"/dev/keyboard","r"
OPEN WINDOW 120,32
POKE "lcd",1
DO
 DO
  x=PEEK(#1)
  IF x<>-1 BREAK
 LOOP
 CLEAR WINDOW
 TEXT 0,0,"Last character:"
 TEXT 0,16,"$"+hex$(x)+" = "+chr$(x)
LOOP
CLOSE WINDOW
</ABC>
```

Program to show use of JGET$ and JPUT:

```
<ABC>
POKE "bypass",1
DO
      DO
            a$=JGET$
            IF a$<>"" BREAK
      LOOP
      b$="<"+a$+">"
      JPUT b$
LOOP
</ABC>
J
S e;0,0,30,32,100
H 100,0,T
T:text;10,10,0,3,5;[SER:1]
T 10,20,0,3,5;[ABC:text]
A 5
```

Program to show readback of JScript-Commands and the FLUSH command:

```
<ABC>
OPEN 1,"/dev/jscript","r"
OPEN 2,"/dev/rs232","w"
PRINT "qm"
LINE INPUT #1 a$
PRINT #2 a$
CLOSE 2
CLOSE 1
rem FLUSH #0
PRINT "f"
</ABC>
Here is text which would normally trigger protocol error.
It is deleted by FLUSH #0, so the PRINT "f" can work without problems.
```

Program to show how to „press“ a key using a program:

```
; Label does an endless loop which is terminated by pressing "total Cancel"
<ABC>
x=0
DO
 IF x=0 THEN
  x=1
```

```
  POKE "key",dec("F090")
 ENDIF
LOOP
</ABC>
```