CamelForth

CamelForth/Z80

Version 1.2 - 16 Apr 1995

CamelForth/Z80 is an ANSI compliant Forth system for the Zilog Z80 microprocessor. It includes the Forth kernel, interpreter, and compiler. This is a **beta test** version. which means that, although I have tested the bulk of this code for correct functioning, and have fixed several bugs, you may discover new bugs. I'd appreciate hearing of any such, via the contact link on this web site.

Download CamelForth/Z80

System Requirements

As distributed, CamelForth will assemble to run under CP/M 2.x. It determines the highest available RAM location from CP/M, and places its data areas (stacks, user area, etc.) immediately below that. At least 8K of free space is required in the TPA (Transient Program Area).

The CamelForth program resides in the bottom of the CP/M program area (100h), and any user definitions are added immediately after. CP/M's default command buffer at 80h is used for the Terminal Input Buffer.

Licensing

Running the Program

To start CamelForth under CP/M, type the command

```
CAMEL80  ...any Forth commands...
```

CamelForth will execute the rest of the CP/M command line as a Forth statement, and then enter the Forth interpreter. To return to CP/M, use the Forth command

```
BYE
```

Note that CamelForth is **case sensitive**, and all Forth words are in **UPPER CASE**.

Recompiling the Kernel

This program was written using the Z80MR macro assembler under CP/M. Z80MR is a freeware assembler, available from several CP/M archives. Assemble the CamelForth source files with the commands

Main Menu
- Home
- Forum
- Downloads
- Documentation
  - 1802
  - 6809
  - 8051
  - 8052/C8051F
  - 8086
  - MSP430
  - **Z80**
- Links
- Members
- Contact Us
- RSS Feeds
- Bugtracker3
- Search

Quick Downloads
8051
8086
Z80
6809
MSP430
8052/C8051F

Contributed:
1802 v1.1
8051
MSP430 (eZ430-RF2500T)
MSP430 FRAM (MSP-EXP430FR5739)

Offsite downloads:
8051/CC1110 (as31)
MSP430G2553 (4e4th)
DS80C400
Z180
Z88
Z80, ZNEO, Z180, Z380/Z382, and eZ80 Acclaim!

```
z80mr camel80
load camel80
```

Z80MR produces an Intel hex file **camel80.hex**, and LOAD generates the file **camel80.com**. (Note: do ***not*** use the version of Z80MR that directly outputs a .COM file; that version of the assembler has bugs.) For embedded applications you probably can skip the LOAD, since most PROM programmers, PROM emulators, and debug programs will accept Intel hex files.

If you don't have CP/M, you can use the MYZ80 emulator on an IBM PC, or you can rewrite the source code for your Z80 macro assembler. (A version for Zilog's Z80 Macro Cross Assembler, ZMADOS, is available from Douglas Beattie Jr. at http://www2.whidbey.net/~beattidp/)

Program Development

Disk I/O is not yet supported under CP/M. However, CamelForth v1.2 will accept commands from a CP/M SUBMIT file using the XSUB utility. The SUBMIT file should contain the commands

```
XSUB
CAMEL80
...Forth source code...
```

This will run CamelForth/80 under XSUB, which will feed the rest of the file to CamelForth as terminal input. You can automatically return to CP/M by putting the CamelForth **BYE** command in the file. Then you can save the modified CamelForth image with the CP/M command

```
SAVE nn CAMELNEW.COM
```

'nn' is the decimal number of pages occupied by the CamelForth dictionary. You can determine this value while in CamelForth with the statement

```
DECIMAL HERE 0 256 UM/MOD NIP .
```

Unfortunately, at the moment there's no way to totally automate this as part of the SUBMIT file. And I'm reluctant to add SAVE to CamelForth when CP/M has a perfectly good SAVE command.

Standalone Use

CamelForth can be easily assembled for a standalone or embedded Z80. About 6K of PROM and 640 bytes of RAM are used by CamelForth, plus whatever additional PROM and RAM is needed by your program. You will probably need to provide the Z80 reset vector, e.g.

```
org 0
jp reset
```

You must also add any required hardware initialization, and the Forth words **KEY KEY?** and **EMIT** for your hardware. You should modify the 'reset' routine to use an equate for end of RAM, e.g.

```
reset:  ld hl,ramend  ; end of available memory (EM)
        dec h         ; EM-100h
        ld sp,hl      ;      = top of param stack
        inc h         ; EM
         etc.
```

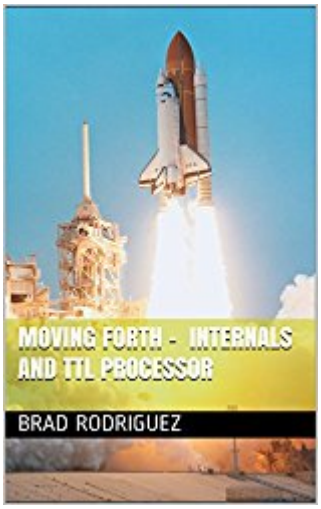
If you are putting CamelForth in PROM, but want to have a Forth dictionary in RAM (so you can add new definitions), you'll have to change the 'enddict' equate (at the end of **camel80.azm**) to the your starting RAM address. Do NOT change the 'lastword' equate.

The Terminal Input Buffer must be moved to a new location in RAM. The usual CamelForth usage is 80h bytes below the user area. TIB can be redefined as

```
;X tib     -- a-addr     Terminal Input Buffer
;  HEX -80 USER TIB      below user area
    head TIB,3,TIB,douser
        dw -80h
```

You should also delete the line

Moving Forth: The e-Book



"Moving Forth," the series of articles that led to CamelForth, is now available as an Amazon Kindle e-Book. All thanks to Juergen Pintaske and his Forth Bookshelf project!

Links

navigator

- view links frontpage

Recent Additions

- AmForth for the AVR8
  A small, open-source, ANS compatible Forth for AVR8/ATMega microcontrollers.
- naken_asm Assembler
  A multi-CPU assembler for MSP430, ARM, AVR, dsPIC, 65xx, Z80, and others (more on the way). This is the preferred tool for MSP430 CamelForth.

Latest Forum Posts

- Posted by Brad R
  Hi Barney, if you've downloaded the 8052 CamelFort[more ...]
  16 Nov : 05:56

- Posted by barney
  Hi Brad and everybody, hope you are all well and c[more ...]
  15 Nov : 23:49

- Posted by barney
  Thanks Brad, and also for moving Forth inspiring
  17 Oct : 06:47

- Posted by Brad R
  Hi Barney, try here:

```
DW LIT,80h,COUNT,INTERPRET
```

from the routine **COLD**. This line causes the CP/M command "tail" to be executed as a Forth command...inapplicable in a standalone system.

Embedded Development

There are **two ways** to write embedded programs in CamelForth:

1. If you have CamelForth running on an embedded Z80, you can download Forth code directly to CamelForth. This lets you type new words from the keyboard, test them as they are defined, and re-define them to make changes. Or you can edit an ASCII text file, and use a program such as Procomm to send this file over the serial port to your Z80. It can take a few seconds to compile each line, so be sure to leave plenty of delay after the line. (I'm working on handshaking to improve this.) Also be sure that no line exceeds 80 characters.

2. If you you want to burn your program into PROM, you can add your code to the file CAMEL80.ASM. (I recommend creating a separate file and using the *INCLUDE directive.) This requires you to convert your Forth code to assembler code. To show how this is done, every high-level Forth word in the file is shown with its equivalent Forth code in a comment. Be especially careful with control structures (IF..ELSE..THEN, BEGIN..UNTIL, DO..LOOP, and the like), and with the Forth word headers. Reassemble CAMEL80.AZM and burn a PROM (or download to a PROM emulator or debug monitor), then test. This is a much slower process, and is best saved for the final stage when you have a tested & debugged program that you want to put in PROM.

CamelForth/Z80 is not (yet) equipped for cross-development.

Application Notes

Report on the FM/MOD bug corrected in the 8051 version.

Related Links

http://www.camelforth.com/pag[more ...]
16 Oct : 07:24

Posted by barney
Hi brad hope you are well, can you point me in the[more ...]
15 Oct : 02:47

Posted by Brad R
Hi Philip, thanks for the update!
19 Jun : 05:43

Posted by Phillip Eaton
Hi Brad,Thanks for the message and sorry for takin[more ...]
11 Jun : 09:03

Posted by Brad R
Hi Phil, yes, I check in here every few days.I got[more ...]
13 Jan : 04:10

Posted by Phillip Eaton
Hi Brad, hopefully you're still monitoring this fo[more ...]
11 Jan : 10:28

Posted by Brad R
I thought I had written something up about this, b[more ...]
15 May : 07:06