1 Z80-Reto-cpmlib

This guide is intended provide some commentary that can be read along with the source code to showcase the various components of the library and how they can be used in your own applications.

You should read the comments in the header files. They are updated during development and will be the most accurate.

1.1 What Is Z80-Retro-cpmlib?

Z80-Retro-cpmlib is a C library that you can link to in your applications for use with the FUZIX-Compiler-Kit. The library is specifically targeted at the Z80-Retro! Single Board Computer by John Winans.

Some standard libraries are implimented where they can be easily backed by the CP/M 2.2 BDOS function calls. Additionally the library includes functions for working with the TMS9918 and Atari style joystick ports on the VDP daughter board designed for the Z80-Retro!.

For instructions on how to install the compiler and this library, see the `./BUILD.md` documentation.

1.2 Headers

The headers are all located in the projects "include" directory. You just need to `#include` the ones you need and make sure to link to the `cpmlib.a` library.

As the code is split out into multiple translation units, your resulting binary should include almost no wasted code.

1.3 Usage

You can use the by including the headers you need, calling the functions in your code and finally compilling and linking. See: "Listing: 1.3 - Example Makefile" on page 2

The process is something like this:

Compile fcc -O2 -mz80 -Iinclude -I /opt/fcc/lib/z80/include -c -o main.o main.c

Link ldz80 -b C0x100 -o main.bin crt0.o main.o libcpm.a

Truncate dd if=main.bin of=main.com skip=1 bs=256

The linker documentation is very minimal because it's a very minimal linker. The `‘-b‘` switch tells the linker to output a binary file without relocatable code. The -C0x100 tells the linker to begin the `‘code‘` segment at 0x100 which is the beginning of the TPA for CP/M.

Because the linker always starts filling code from 0x0000 we need to remove the first 256 bytes using the `‘dd‘` command.

```
TOP=.
CC=/opt/fcc/bin/fcc
AS=/opt/fcc/bin/asz80
LD=/opt/fcc/bin/ldz80

CFLAGS=-O2 -mz80 -I $(TOP)/../include -I /opt/fcc/lib/z80/include
LDFLAGS=-b -C0x100
LIBS=\
     $(TOP)/../cpmlib.a \
     /opt/fcc/lib/z80/libz80.a \
     /opt/fcc/lib/z80/libc.a

CRT=$(TOP)/../crt0.o

all: clean malloc.com fileio.com testtms.com copy

malloc.bin: malloc.o
  $(LD) $(LDFLAGS) -o $@ $(CRT) $^ $(LIBS)

malloc.com: malloc.bin
  dd if=$^ of=$@ skip=1 bs=256

fileio.bin: fileio.o
  $(LD) $(LDFLAGS) -o $@ $(CRT) $^ $(LIBS)

fileio.com: fileio.bin
  dd if=$^ of=$@ skip=1 bs=256

testtms.bin: testtms.o
  $(LD) $(LDFLAGS) -o $@ $(CRT) $^ $(LIBS)

testtms.com: testtms.bin
  dd if=$^ of=$@ skip=1 bs=256


clean:
  rm -fv malloc.bin malloc.com fileio.bin fileio.com
  find . -name "*.o" -exec rm -fv {} \;
```

Listing 1: Example Makefile

This example does not show the actual FCC commands explicitly. Make is automating that step for us.

2 CP/M

The CP/M header provides C wrappers for almost all the CP/M BDOS function calls.

3 malloc, free

4 string

5 fcntl

6 TMS99xx