REHDESIGN

CPU280

Software-Manual

1. General

The current versions of the boot loader (V1.02), of the operating system BIOS (V1.02snc) and of the format manager (V1.01) are completely functional, but are not to be considered the final release. A few detail improvements are already scheduled for the next version, in particular concerning the boot loader and the system BIOS. Therefore, please localize and describe any bugs carefully, so they can be fixed in later versions. Also concrete suggestions for improvement are welcome.

The operating system is modular, so modifications can usually be contained in a few modules (in particular for ports and drives). But with a view towards later updates, I recommend that users makes as few changes to the source code as delivered as possible, and keep a protocol of their changes. Additions to the programs should be done as additional modules or include files. Global declarations are contained in two LIB files, separately for the processor (Z280EQU.LIB) and CPU280-board or system (CPU280.LIB). Those two files are used in all BIOS modules.

The software is adapted to the environment of my system, which means it supports the functions of the CPU280 board, and additional interface boards. From the drivers one can see how those additional boards are used; sometimes other boards can be supported by just making a few changes in those drivers. Support for optional boards should always be implemented by conditional assembly, so that the option can be activated / deactivated by setting a flag. Currently, not all optional boards are handled this way yet, that will be taken care of with the next version of the system.

In case of questions about the current status of any feature, the first step should be to look at the corresponding history-file. Such files currently exist for the loader, the system-BIOS and the format manager.

2. Boot loader

The boot loader consists of three modules: One is the basis for cold starting (initalization and RAM test) and the actual booting, the other two contain disk-I/O subroutines and the setup.

After a reset, the first step is to determine the processor clock frequency (in multiples of 614 kHz), using the real time clock. After that the console port is initialized (according to the values stored in the setup; the default is 8+n+1 bits, 9600 baud, no handshake) and a message is output to the console port. Next follows the RAM test, which examines the whole addressable memory for writeability (nondestructive). The available memory capacity is determined in the same step. If there were no errors in the RAM test, then the boot loader is copied into RAM and started there. Further and more detailed hardware diagnostics are planned, but not implemented yet.

During these operations the three LEDs display the current status. The hardware reset enables all three LEDs. Immediately after reset the first one is turned off. The second one is turned off after successfully measuring the clock frequency, and the third one after the RAM test. If the battery in the real-time clock is empty, the second LED stays on while the third is turned off.

Usually (that is if the boot process is not interrupted), the complete operating system is booted from EPROM, using the settings from the NVRAM. If the checksum of the EPROM has changed since the last boot (i.e. the system has been updated), or if the DEL key is hit during the RAM test, then the setup program is entered before booting. All the settings in the NVRAM can be changed in the setup program. Those include among others which drives are connected and what drive types they are, the configuration of both serial ports, the boot drive and the drive search chain.

Hitting ESC during the RAM test will cause the system to boot from floppy instead of from EPROM, with the boot drive specified in the setup.

In the part of the CPU280's generous memory not used by the system itself, the boot loader will create a RAM disk. The RAM disk gets a directory label. If the boot loader finds the label undamaged after a reset, it will not erase the RAM disk. If it finds minor modifications to the label, it will prompt the user whether the RAM disk shall be cleared. Only if it finds massive damage to the label it will automatically erase the RAM disk.

Additionally, the boot loader will recognize a few external pseudo-drives, RAM or EPROM-boards, which are used by cp/m just like drives. Currently supported are the c't 1MB RAM-disk and the c't 256kB solid-state-floppy. Their capacity is encoded through jumper settings. How exactly they are driven can be gleaned from the driver source, or from the author.

Since different console ports can be used, and since no RAM may be used for the first messages from the boot loader and initialization code (the RAM has not been tested yet), the I/O functions required (initialization and character input and output) are implemented as macros in their own include file (LDRIO.LIB). As delivered, the macros use the MPU-internal serial port. If you want to drive other serial ports, you have to provide your own macros, using only the registers explicitly allowed in the source code.

3. Use of the setup program

One module of the boot loader is the setup program. It modifies various settings of the CPU280. The setup program is designed hierarchically, and currently offers three areas to be setup:

1. Drives	Declares which floppy disk drives are connected.
2. Interfaces	All settings of the two serial ports on the CPU280, plus the device assignment after reset.
3. Others	Anything which didn't fit into 1. or 2.

The setup program is always used through single key strokes; usually numeric input (one digit selects from the options offered). Most menu items are self-explanatory, so the following will only explain some less clear points.

In the "interfaces" menu one can not only set the parameters of the serial ports, but also the device assignment after booting. These are the assignments which can also be shown or changed by the cp/m+ command DEVICE. The setup program uses codes for the device, which are identical to the number of the device in the device-table, beginning with 0. Therefore a zero represents CRT1, 1 for CRT2, and 5 the parallel port SPOOL (using my system configuration). If you enter the wrong values here, the system will not output anything after the message "62K TPA"!

The menu item "memory size" in the "others" menu is only for test purposes. The actual memory size is set to the amount of memory actually found.

The selection of "regular time" or "daylight saving time" does not directly influence the actual time. Instead it sets a flag in the real-time clock, which in the case of daylight saving time will jump from 1:59AM to 3:00AM on the first Sunday in April, and equivalently switch back on the last Sunday in October. Since the current version of the software does not set the weekday register, the switch can happen on any other weekday. Since the switchover dates do not agree with the ones used in Europe (to switch from MEZ to MESZ), this setting isn't very useful in Europe. Therefore it is safer to leave this setting on 'regular time', and change the clock by one hour when changing to or from daylight saving time (just like any other clock).

To enter the drive search chain, simply enter the four drive letters (without colons between them). If you want to enter less than four drives, end by entering "x".

The drive search chain (and the temporary drive, and the search order) are automatically set whenever the system is started. That saves having to call SETDEF seperately each time.

The "boot drive", which can be set in the "others" menu, should not be confused with the cp/m default drive (which can only be set by GENCPM), which is the drive the user is logged into after starting the system. The "boot drive" just defines from which drive the system file CPM3.SYS shall be loaded when booting from floppy (for example to test new versions).

4. AutoFormat system

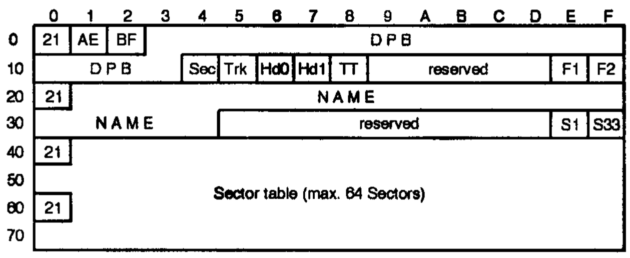
Both the boot loader and the system BIOS contain an Auto-Format function, which means they adapt to the format of the floppy currently in the drive. The floppies contain a 128-byte parameter block (format parameter block, FPB) on track 0 sector 1, which completely describes the physical and logical disk format.

If a disk change occurs (which is sensed by the system BIOS), then the next disk access will first attempt to read this parameter block. If an error occurs when reading it, or if this floppy does not contain a valid parameter block, the parameters of the floppy most recently used will be retained. Using the format manager, the system can be set up for foreign formats which don't contain a parameter block.

The default format is determined in the disk module of the loader BIOS or in the table module of the system BIOS. It has to describe the largest possible format, since the size of the directory buffers and allocation vector are determined by GENCPM when generating a new system from the disk parameters. As delivered, the default format is set to a very large placeholder, a floppy with 880 blocks, 512 directory entries and 1024 byte long sectors (that means one can only boot from floppies with a parameter block).

Currently the AutoFormat processing does not check whether the buffers are large enough for the format described in the parameter block; erraneously generating a system with buffers too small for the largest floppy ever encountered will invariable lead to the system crashing.

The layout of the parameter block (FPB) is as follows:



The following abbreviations are used:

21,AE,BF	cp/m compatible FPB marker and ID
DPB	standard cp/m-plus disk parameter block
Sec,Trk	Number of physical sectors and tracks
Hd0,Hd1	Logical head number on front and back side
TT	Track translation flag (see below)
F1,F2	Bit-masked flags, see below
NAME	Format name, 20 characters ASCII
S1,S33	Sector numbers 1 and 33 (missing from table)
Sector table	Table with logical sector numbers (skew table)

The track translation flag is defined like this (f=front side, b=back side):

0	= No track translation (mode in the bit-flag is valid)
1	= f0..fmax, b0..bmax
2	= fmax/2..fmax, fmax/2-1..f0, bmax/2..bmax, bmax/2-1..b0
3	= f0, f1, b0, b1, f2, b2, f3, b3, ..., fmax, bmax
4	= f0, b0, f1, b1, .., fmax, bmax (Half tracks with consecutive sector numbers, for example Kaypro)

Bit-mapped flag F1:

Bit 7,6	: Density (00 = FM, 01 = MFM, 11 = HD)
Bit 5,4	: reserved
Bit 3	: Data inverted
Bit 2	: Multi-Sector-I/O allowed
Bit 1,0	: Overflow (00 = none (single sided), 01 = sector, 10 = track, 11 = half track)

The second bit-mapped flag (F2) is currently reserved.

5. System generation

Both for generating the boot loader and for the system BIOS, the use of make-files is very helpful, to automate the sometimes complex program calls.

We use the make utility of Wolfgang Muees, which runs under cp/m+, uses the regular file date stamps, and is freeware. The program itself and the required make files are included with the system software. Therefore, after changes to the source code, one only has to start make.

Since there is no freely available real Z280 assembler, we use another free utility. Axel 'Fifi' Zinser wrote the PRE280 program, which searches assembler source file for Z280 instructions, and replaces them by DB/DW pseudo-ops. This allows use of a regular Z80 assembler (suggestion: SLR180) even with real Z280 mnemonics in the source (no macros).

For linking the system BIOS one can only use the original DRI LINK program. It is the only linker, which correctly generates SPR-code (system page relocatable). The system BIOS has to be in SPR format, so that GENCPM can merge it with the BDOS. LINK is a part of cp/m+, but also runs under cp/m-2.2.

When configuring the system, GENCPM uses the values from the control file GENCPM.DAT. This file contains information about available memory, hash and cache setting, and similar questions. In case of major changes to the operating system (for example adding a hard disk) one should run through GENCPM *manually* and check and maybe change the settings. A few recommendations: For all available physical disk drives, reserve enough directory buffers for the complete directory of the normal format.

Drives which are not connected get no directory buffer (set to share the buffer of another drive). Preudo-drives (for example RAM- or EPROM-disks) usually only require one directory buffer. The handling of data buffers is different. To have a shared disk cache, it is advisable to make the buffer for drive A: as large as possible, and have all other drives share it.

To generate a bootable EPROM, the system file CPM3.SYS, which has been generated by GENCPM, has to be merged with the boot loader LOADER.COM and the command processor CCP.COM. This is handled by the program GENEPR, which creates a file SYSTEM.EPR. This file then has to be split into two parts by the SPLIT16 utility (EVN for even, ODD for odd), which can then be programmed into EPROMs.

6. Error handling

If an error occurs during a disk operation (time out, read or write error, CRC error), an error message similar to MS-DOS is output, and the system waits for user input. The user can select Abort, Retry and Ignore by entering A, I or R. Entering a ^C here acts like 'Abort', but also passes that character to the program.

In the current version of the system (V1.02snc) there is a problem, that after a time-out when accessing a drive, all further accesses to the other drives are impossible and generate error messages. This can be cured by entering ^C from the cp/m+ command line (logs out all the drives). The cause of the problem is known and will be fixed in the next release.

In case of CPU errors (for example divide-by-zero traps, or privileged commands in user mode), an error message is output, and the command processor is restarted. Also an NMI-signal (Non Maskable Interrupt) will restart the CCP, but without error message.

7. Format manager

The format manager is used to manage the various formats in the AutoFormat system. It is also designed modularly, but programmed in Turbo-Pascal. It uses menus, so no detail manual is necessary.

All the information about formats is stored in the file FORMAT.DAT. When the program is started, it looks for this file, first on the current drive, then it follows the cp/m+ 'drive search chain'. However, this works only if the data file is located in the current user area.

The handling of format definitions is restricted to a senseful minimum. Formats can not be reordered or deleted randomly; one can only add new formats or change existing ones. This is actually desirable, since one tends to remember the numbers of frequently used formats, and those numbers shouldn't change. For tests when trying to read foreign floppies, one should always use the same format number, to prevent accumulating too much garbage in the file.

The format manager can print a formatted listing of all known formats. It requires about 120 character per line, so you have to set your printer to print narrow characters. Since printer control sequences are not standardized, the format manager does not do that by itself.

To select a format for a disk drive which contains a floppy without a valid FPB, one can either use the menu item "Process foreign format", or one can use command line parameters. For the latter method, specify the number of the desired format and the drive letter, for example 'FORMAT 20 C'. In this case the format manager will exit immediately after executing this change.

8. Standard formats

The recommended standard formats (Reh CP/M-3 V2.1, Reh 5.25" HD and Reh 3.5" HD) are designed for maximum use of the capacity of each media, and by simplicity. A sector size of 1 kB is always used, the back side is used via sector overflow, and the skew is 1 (physical) and 2 (logical). There are no reserved tracks (offset tracks).

These principles should also be enployed for future standard formats. These formats have high performace while still being easily ported.

The logical format parameters (block size, number of directory entries) can be left to personal taste. Still, the use of standard sizes is preferable. The block size should be 2 kb, to avoid unneccessary blocking losses. For larger capacity media (for example 3.5" HD) block sizes of 4 kB might be reasonable.

9. Plans for improvements

When working with the software release described here, some areas for improvement have already been identified. The following items are planned to be realized in one of the next software versions:

a) Global:

- Extensions can be turned off using conditional assembly. Uses new control file OPTIONS.LIB.
- All strings (setup, format manager, error messages) can be selected from different languages by conditional assembly.

b) Boot loader:

- As complete a hardware test as possible, using clear error messages on the LEDs or in text.
- Maybe allow the CCP to be loaded from disk. Suggestions for how to implement this are welcome.
- Extend the setup by adjustable startup and shutdown time for the floppy disk drives, and by the number of retries for read, write and verify.
- Extend the setup to harddisk parameters.

c) System-BIOS:

- A new resident BDOS can replace the current RESBDOS, giving major improvements. Call the banked system using system call instructions (SC), faster and safer.
- New user functions "Connect To Interrupt" and "Disconnect Interrupt" allow user programs to employ external interrupts from the ECB bus.
- Start a time-out after DI instruction. If user programs block interrupts for too long (for example after 100ms), simply reenable them.
- Maybe change the BIOS-internal XON/XOFF protocol, which currently has some shortcomings. Maybe completely remove it from the BIOS, since hardware handshake is more effective, and the BDOS has a powerful software handshake for the console port.
- For 40-track drives cut the step rate in half (to match their specifications).
- After a FDC-Reset (in case of timeouts when trying to access a drive) move the heads of all drives to track zero, which prevents a time-out of one drive from creating errors on other drives.
- Maybe handle the character I/O-functions by interrupts (but still table drives).
- Automatic setting of the weekday when setting the clock, thus supporting correct d.s. time switching in the US.

d) Format manager:

- Find the data file even if located in user 0 with system attribute set. (This is a bug of turbo-pascal which has to be fixed.)

10. Bugs in the CPU chip

Unfortunately, the Z280-MPU is not at all bug-free. In addition to the problems acknowledged by Zilog, we found bugs, some serious. Usually they don't influence user programs; packages like WordStar, Turbo-Pascal, Macro-80, the SLR tools, ARC and many others run without any problems. However, difficulties can occur in the system software, for example a subroutine, which is perfectly well written and has always worked just fine, fails to function after moving by a few bytes. For that reason, the development of the system software has hit a snag recently. By now it has become obvious that Zilog does not provide support concerning bug reports; so system software which runs stable can only be achieved by trial-and-error.

Coincidentally, the software described above is in the category of software which runs stable.

Here is a short listing of the relevant bugs:

- There may not be a jump instruction (including call/restart) directly following an I/O-write transaction which uses wait states. This is both for the CPU (using OUT instructions) and the DMA (in flowthrough, single-byte and burst modes). Supposedly the problem occurs only for lots of wait states, not for the four wait states the CPU280 uses. The problem can be worked around by adding an IN instruction or four NOPs between the two critical instructions. To check user programs for such critical instruction combinations one can use the OUTJMP program. For the DMA, there can by design not be a workaround (!).
- The DIVUW gives wrong results in certain cases. Such a case is the MSB (bit 15) of the divisor being set. From other sources we hear that there are additional cases where the result is wrong. The signed division using DIVW is

supposedly correct.

- After an I/O-write, the DMA will sometimes not release the bus, but stay in a wait state for as much as 20 µs. This will create errors when writing HD-floppies, caused by not servicing the FDC in time. For this problem there is a work-around, which Stefan Nitschke found by chance, and which is not understood, not even by Zilog. But it works reliably. The software release 1.02snc differes from release 1.02 only by adding this workaround ('Stefan Nitschke Chaos').
- The CPU will give an invalid carry flag after certain operations. An example of such an operation is an arithmetic instruction sandwiched between two EX AF,AF' instructions; no other examples are currently known. The error depends on the location of the offending code, and the location dependance can be influenced by enabling or disabling the cache. This could be related to another case of a routine in the system BIOS failing depending on location, which also involves the carry flag. This is so far the most serious bug, since it can't be localized or identified (at least so far).

This manual was translated by Ralph Becker.