Zprom V1.4 - the Eprom Programmer for Application Cards

Zprom documentation V1.0, published 2. January 1996
V1.1, Description of added Flash Eprom functionality, 28th December 1997
V1.2 Documentation updated and migrated to HTML format.

Before you read any further, the following conventions are used in this text:
[] identifies the SQUARE key
<> identifies the DIAMOND key
<> identifies other system keys, e.g. <MENU>

Contents:

top 1. Introduction

Zprom gives you the ability to produce your own application cards. Even though Cambridge Computer produced a special adaptor for conventional Eprom programmer hardware, they never considered the Z88 as a serious tool for programming Eprom application cards.

Nevertheless its possible, since the hardware was in-built right from the start. Thank you for the File Eprom facility, Cambridge Computer!

Further, Zprom has been designed to take advantage of a modified RAM Card that emulates an Eprom card. This will allow you to easily develope an application card by simply uploading the code into the RAM card. With a simple magnet you can write-enable the card, load your code and remove the magnet to write-protect it again.

You will find the RAM card especially valuable if having tried to develop a 128K card with multiple applications using a stack of Eprom's and an eraser. First of all it's annyoing to wait for the erasure process (when suddenly being faced with no more Eprom's to blow) and the fact that even though the Z88 is a tough little computer the slot connectors gets worn out some day. I've used my faithful Z88 now for 6 years with thousands of insert/remove card actions - the slot still works reliably but I guess some day no more metal is left on that connector!

If you value your Z88, get a modified RAM card to develope your applications and save your slot connectors a hard time. Otherwise, stick to the latest technology, the 1MB Flash Eprom from

Rakewell Ltd.

The latest invention on the Z88 front is the Flash Eproms. They perform as a ordinary Eprom except that they can be erased electronically. Commands have been implemented to handle them as well.

To produce applications on Eprom cards, very simple rules must be obeyed:

1) The Eprom card must be completely erased, i.e. not previously checked as a File Eprom. If so the Z88 has blown a small header at the very top addresses of the card and needs to be erased by UV-light again.

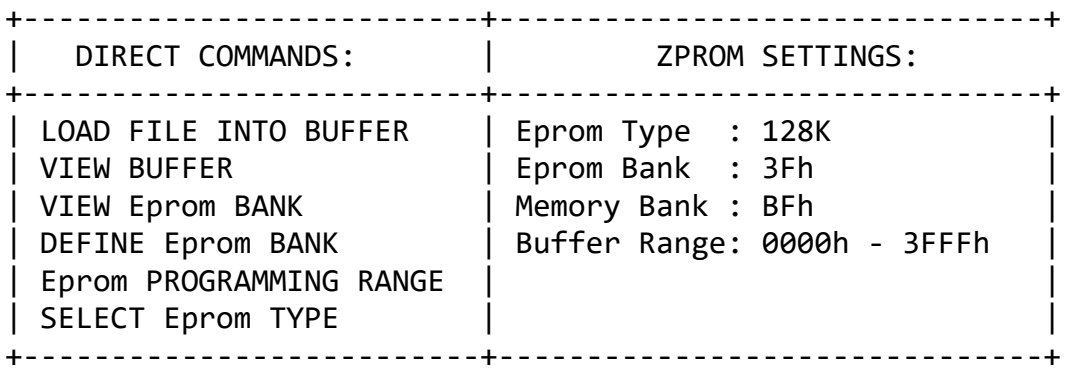
2) The Card ROM header must be blown at the very last stage of the production, otherwise the operating system gets confused and probably crashes the computer.

3) The Z88 must be issued with a soft reset after an Application Card programming session, since the card wasn't properly 'inserted' into the slot.

Zprom executes only on an expanded Z88. This is due to a 16K buffer that is allocated when Zprom is activated. Zprom is designed as a 'bad' application but is a one-instantiation application which causes a minimal hassle for the operating system. An expanded machine requires 128K RAM card or more in slot 1 (or if you have V4 of the operating system, 128K or more in slot 0). The buffer is used for copying Eprom and other memory banks and blowing code into the Eprom.

Zprom is activated using []ZE (or additional Z keys if other applications use []E as hotkey). If Zprom cannot allocate 16K continous RAM, you will be reported with a 'No room' error and excited back to INDEX. You either have filled your RAM with files and application runtime space or don't use an expanded Z88.

When activated successfully, you will be presented with two windows; the left window contains a traditional Z88 command menu with frequently used commands, the right window displays status information of the current selected Eprom type (size), Eprom bank, Memory bank and address range (of a selected bank). The following figure illustrates the start-up screen:



The two windows have been designed to be on top of the previous window information (e.g. windows from suspended applications).

All bank and address information is displayed in hexadecimal notation by Zprom.

The status information displays (from top): the current selected Eprom type, the top bank of an Eprom (relative), the top bank of a memory bank in slot 2 (absolute), and the current programming range inside a bank. The range reflects also the 16K memory buffer.

All Zprom commands fetches these status values as default parameters and pre-types them when a memory parameter is to by entered from the keyboard.

top2. Z88 memory concepts

To fully understand the commands and inner workings of Zprom, we briefly introduce the organisation of the Z88 memory and Eprom bank conventions.

The Z88 are capable of addressing 4MB of memory (24bit address range). This is not possible by the Z80 processor itself - it can only address 64K (16bit address range). The Z88 custom BLINK chip administrates the memory organisation by manipulating the extended 8 address bits outside of the Z80 processor.

To allow the Z80 processor to see the available RAM, all memory is divided into 16K blocks of RAM, identified as banks. Each bank is assigned a unique number, held in a single byte (8bits). Any bank may be bound into the address space.

The Z80 address space is divided into 4 segments each of 16K size. The Z88 hardware has been designed to swap memory banks into each of the 4 segments. However, the processor still only sees 64K of continous memory. To obtain any available (bank) memory, the operating system provides facilities to bind in any of the accessible banks in the computer, identified by their unique ID, into one of the 4 segments of the Z80 address space. The ID may hold a number in the range of 0 to 255, therefore a total of 255*16K (4MB) memory is physically possible to use in the Z88.

The 8bit ID is split up into two sections; the first 6 bits (0 to 5) identify the relative bank number of each slot, which is from 0 to 63 (3Fh) - the upper two bits identify the slot number (0 to 3). Looking closer at the 6bit relative bank number it is obvious why a maximum of 1MB is possible for each slot: 64*16K is 1MB!

The Bank Identifier:

```
7 6 5 4 3 2 1 0 BIT---------------
S S B B B B B B SIGNIFICANCE
```

S = slot number (BIT 7, 6 gives 4 combinations 00, 01, 10, 11 = 0, 1, 2, 3)
B = relative bank number in slot

top3. Eprom concepts

The operating system (and underlying hardware) uses a simple convention with Eprom bank numbering. The top bank of an Eprom is regarded as the bank with the highest number - in this case 3Fh (00111111b or 63d). The lowest physically possible is regarded as 0. However, no Eprom card has yet been produced that holds 1MB, therefore the lowest bank number in any card would be to count backwards. For a 128K Eprom, 8 banks are available - the lowest bank will then 38H.

The 1MB Flash Eprom Card contains 64 banks - the lowest bank is 00H.

Further, the hardware also sees this. In a slot with 128K Eprom fitted, bank 3FH to 38H identifies the card, then repeated again in 37H to 30H. This system continues right down to the relative bank 0 of the slot.

There is not provided any facilities that may identify an inserted card other than trying to ask the operating system for a :ROM or :RAM device. At hardware low level there probably is, but this is not documented (the operating system knows how!).

Zprom performs Eprom programming on slot 3 whether a card is installed or not. This doesn't affect the hardware if a card is not present. Reading the slot will return random information. Zprom therefore quite quickly recognises an empty slot, since no slot can randomly hold FFh (255d) values in a complete 'empty' bank.

A clean Eprom will only contain bytes of value FFh (255d). Before Zprom performs blowing of data to the Eprom an initial check is performed to assure that the range of programming contain bytes that are prepared to be blown. If Zprom finds the correct value of FFh it assumes the byte is 'clear'.

An Eprom is regarded as 'full' when all bits in all bytes are 0. The blowing process of a byte affect only the bits on the Eprom that are 0. The remaining bits are unaffected - the bits can still be blown to 0. Therefore an Eprom to be programmed with FFh (11111111b) has no effect.

top3.1 Using the right Eprom type

When you are going to blow data on an Eprom it is very important to select the correct type before you start the blowing process with the **<>EPROG** command.

In Zprom several types are available: 32K, 128K (and 256K) and the Intel Flash Eproms. Use 128K type for that Eprom size and 256K. The 32K type is to be used only for 32K Eprom's. Flash Eproms may be used only with the FLASH setting otherwise you might damage the Flash Eproms.

If you by accident are blowing a 32K Eprom with 128K type setup, you might probably get error messages like "incorrectly blown byte at ...". Blowing a 128K Eprom with 32K setup might result in the same problem.

If you succeed in a blowing procedure with the wrong Eprom setup, we don't know exactly how well the bytes have been blown. If they will be "unstable" is impossible to determine. Always make sure that you have chosen the correct setup.

top4. Zprom concepts

The Zprom application replicates the principles of the Z88 memory organisation. All references to the Eprom is through the relative bank number of slot 3. This is because you cannot blow Eprom's in the other slots. As described above, bank 3Fh is regarded as the top bank of the Eprom. All commands related to Eprom manipulation uses the relative bank numbering convention for slot 3 only.

The 16K buffer is used by Zprom to manipulate any bank of the Eprom and may be regarded as a mirror or copy-buffer. For example, when a blowing procedure has started, the contents of the buffer is copied into Eprom (plus enabling hardware facilities that activates Eprom programming) and then verified with the buffer to assure a proper safe programming session.

The current range always reflects both the buffer, the Eprom bank and all other memory bank related commands. The range is defined as a relative address from 0 to 3FFFh (16383d) which is the maximum amount of bytes in a bank.

top5. Zprom commands

All commands available can be found by pressing the <MENU> button, as with all standard applications in the Z88. You will find three topics, each identifying related actions:

CURSOR: All commands related to view and edit memory banks. These commands have only effect when issuing the **<>MV**, **<>ME**, **<>MS**, **<>BV**, **<>BE**, **<>BS**, **<>EV** and **<>ES** commands.

MEMORY: All commands related to manipulate the buffer and modified RAM (pseudo Eprom) card.

Eprom: All commands related to manipulate the Eprom in slot 3 plus a few extra to perform interaction between Eprom and the modified RAM card.

top5.1 The protected commands

Some of the commands in the MEMORY and Eprom menues are protected, i.e. they can only be activated by typing the command sequense. If you try to activate them with <ENTER> a warning beep will be returned and no action taken. The protected commands reflect serious actions that cannot be reversed (undone). To prevent an accidental execution we have chosen to identify them as protected commands. They are:

<>MBCL, Memory Buffer Clear
<>RBCL, Clear RAM Bank
<>RCLC, Clear RAM Card
<>EPROG, Program Eprom
<>FLBE, Erase Flash Eprom
<>CLONE, Program ROM Card

top5.2 Using the direct commands

Some of the commands in Zprom have been copied to the direct command window. Move the cursor bar with either up or down arrow keys and press <ENTER> to activate.

The direct commands have been provided to facilitate easy access of buffer and Eprom file

management.

top 6. Zprom help

All commands contain on-line help text. Using the command menues followed by a press on <HELP> on a particular command entry in one of the topics will immediately display a small help text.

Further, a set of 17 additional help topics have been provided for other guidelines to assure safe and easy usage of Zprom. You obtain this help by just pressing <HELP> when no command menues are active.

top 7. Eprom guidelines

Before going into the description of the various commands of Zprom, a few guidelines is appropriate for handling Eprom's with regard to Zprom.

An standard UV light Eprom may be written (blown) only once. If it needs to be re-written an erasure of the Eprom is necessary. A new Eprom chip is erased properly the first time using UV-light about 20 minutes. An Eprom having been used several hundred times or more needs at least 2 times 20 minutes, preferably 3 times. With old Eprom you may experience Eprom's that perform irregular if not erased properly.

Erasing UV light Eprom's three times in a row is a slow process, especially when you need them to perform application programming tests. We recommend a good stack of Eprom cards (5 or 6 at the least) during software testing. However, work is performed much more efficiently with a modified RAM card or the Flash Eprom...

When you want to produce an application Eprom, the chip must NOT have been initialized by the FILER (e.g. by issuing a CATALOGUE Eprom command). You will also get the Eprom initialized automatically when performing a soft reset and a "fresh" Eprom inserted into slot 3. The initialization process blows a small header (or watermark) in the top bank (3Fh) that identifies it as a file Eprom.

An application Eprom also uses a special header in the same location as the File Eprom. Therefore be aware of what you do when inserting a clean, empty Eprom card into slot 3.

top 8. Using the buffer commands

The 16K buffer is the key to program your Eprom's. Further, using the buffer maximize the speed of blowing code onto the Eprom. Since all the code is contained in the buffer, no I/O overhead is involved in performing the programming and verification of the chip. In fact, the Eprom programming on the Z88 is performed just as fast as conventional hardware programmers on stationary computers.

top8.1 Defining the default range

The range affects all commands that manipulate bank data (editing viewing, copying and programming).

You can change the default range by either using the direct command menu or activating the **<>ER** (Define Bank Range) command. You will be prompted to enter the start and end range address. In both cases the current values are pre-typed for convenience.

It might be necessary to change the default range if you're going to copy a whole bank into the buffer and the current range only covers a part of a bank.

top8.2 Loading files into the buffer

You load data into the buffer using the **<>MBL** command. You will be requested to enter the start (range) address of the data to be loaded. The end (range) address is resolved automatically when the file has been loaded into the buffer. Any previous data in the new buffer range will have been overwritten. Please note that you can load a maximum of 16K into the buffer.

The default start and end range will also have changed according to the loading address of the file.

top8.3 Saving the buffer to a file

You can save the contents of the buffer to a file using the **<>MBS** command. You will be requested to enter the start and end range address of the buffer. Both start and end range addresses are the current range settings pre-typed for convenience. Please note that altering start and end range addresses will not affect the default buffer range settings.

You need to use this command if you're going to copy an application Eprom as 16K bank files.

top8.4 Editing or viewing the buffer

It might be necessary to examine the buffer contents, or even modify a few byte before it is blowed to Eprom or saved to a file. This facility is available through the **<>MV** and **<>ME** commands. To just view the buffer contents in the current range, use the **<>MV** command. To edit the buffer contents, use the **<>ME** command.

When one of the two commands have been selected (through command menues or typing the command sequense) you will be asked to enter the start address of memory you want to access in the buffer. The default start range is pre-typed in the menu. When that is selected, the view/edit window is displayed and you're ready to look. The window are split into two sections - the first displays the current memory contents using the hexadecimal notation, the second section display the ASCII version (printable characters). All non-printable characters (in principle bytes with ASCII values less than 32 and larger than 127) are displayed using the '.' symbol. Both sections displays the same memory, just in different formats.

Here is an example which displays the top 192 bytes of the danish Z88 ROM:

```
3F58h 20 43 6C 69 76 65 20 44 61 76 65 20  Clive Dave
3F64h 45 72 69 63 20 46 65 6C 69 63 69 74 Eric Felicit
3F70h 79 5E 32 20 47 72 61 68 61 6D 20 4A y<>2 Graham J
3F7Ch 69 6D 20 4A 6F 68 6E 20 4D 61 72 6B im John Mark
3F88h 20 4D 61 74 74 68 65 77 5E 32 20 50  Matthew<>2 P
3F94h 61 75 6C 20 50 65 74 65 72 20 52 69 aul Peter Ri
3FA0h 63 68 61 72 64 5E 33 20 54 69 6D 20 chard<>3 Tim
3FACh 57 69 6E 67 73 20 5A 65 65 26 4B 65 Wings Zee&Ke
3FB8h 73 73 6E 61 20 7D 0D 00 00 00 00 00 ssna }......
3FC4h 00 00 06 FE 07 13 08 4E 05 41 50 50 .......N.APP
3FD0h 4C 00 FF 00 00 00 00 00 00 00 00 00 L...........
3FDCh 00 00 00 00 00 00 00 00 00 00 00 00 ............
3FE8h 00 00 00 00 00 00 00 00 00 00 00 00 ............
3FF4h 00 00 00 00 54 43 4C 81 08 00 4F 5A ....TCL...OZ
```

A small window is displayed to the far right of the screen which shows the available cursor movement commands.

A cursor is displayed at the first byte of the selected range. The following cursor movements are available:

Previous byte, Next byte: <LeftCursor>,< RightCursor>Previous line, Next line: <UpCursor>,< DownCursor>Previous Screen, Next Screen: <SHIFT><UpCursor>,< SHIFT><DownCursor>Top Bank, Bottom Bank: <><UpCursor>,<>< DownCursor>

The default cursor is placed in the Hexadecimal section. You may toggle the cursor to the ASCII section by pressing <TAB>. This feature is very useful when you stand at a non-printable byte and want to know the value of that byte - just press< TAB> and the cursor is moved to the hexadecimal version of the byte.

When you move the cursor beyond the buffer boundaries, you are automatically wrapped to the opposite end of the buffer, e.g. moving beyond the top begins at the start of the buffer.

When you have selected the **<>ME** command you may enter hexadecimal values at the cursor or type characters directly from the keyboard when you have selected the ASCII section. When in the hexadecimal section simply type your hex-code. Only legal hex codes are allowed ('0'-'9','A'-'F'). Case conversion is automatically performed so you may type your hex code both in upper or lower case. The **<>MV** command ignores input - you will only be allowed to execute cursor commands.

The cursor will automatically move to the next byte when a hex or ASCII byte has been typed.

top8.4.1 Using the buffer as a file editor

The buffer is very useful for examining files, and on occasion modify them on a byte-by-byte basis. In the Assembler Workbench context it is obvious that you may use the buffer as a file editor for patching machine code intructions.

In other situations it may be needed to examine the file contents and correct a few control ASCII

characters.

Please remember that you can only load files that are not larger than 16K.

When using the buffer as a file editor it's not necessary to load it at a specified address (otherwise load it where appropriate). Use the **<>MBL** to load a file into the buffer. Then you may examine the file - you can just view it with the **<>MV** command or for editing purposes, the **<>ME** command.

When you have altered the file (buffer), just save it back to the filing system using the **<>MBS** command. If you like, give your file a different name to reflect the modifications.

Conventional file editors use a sector-size buffer for editing, simply to avoid accidental modifications. When an editing is completed, the sector-buffer is written back to the file. Using Zprom as a file editor is the same principle, using a buffer for the complete file!

top8.4.2 Using the buffer as a file merge utility

The buffer may also be used as a file merge utility. It might be necessary to combine several machine code routines (that don't use absolute addressing). Since you are going to merge files, start to load your files into the start of the buffer. The range will automatically reflect the total size of your files during the 'appending' of the files. Simply use the **<>MBL** commands for your files - take note on the end range parameter and add 1 for the start range address of the next file in line. When completed, simply save the buffer contents from start range 0 to the current end range address, using the **<>MBS** command.

Please remember that you have a 16K limit to merge files.

top8.5 Searching in the buffer

In many situations you cannot identify the position of text or executable code that needs to be investigated (or edited). You can search for ASCII strings or hexadecimal sequenses in the buffer with the **<>MS** command.

When the command is activated, you will be requested to enter the start search address. The current start range address has been pre-typed for convenience.

When the search address has been entered you will be requested to enter your search string. You enter an ASCII string by preceeding it with the ' symbol followed by your string, e.g. 'this is a test. Spaces are allowed. Please note that control characters cannot be entered in ASCII search strings. Searching of ASCII string are performed as entered - ie. case-dependent searching.

The default input is hexadecimal sequense. Simply type your hex codes without any preceeding symbols. All hex codes must follow one another without any spaces, e.g. ff80aac090.

The search begins from the start address and forward. When the buffer end boundary is reached, the search continues until the start address.

When a match is found, you are automatically entered the view memory mode which displays the start of the found match and forward. You can use the normal cursor movement as in the view/edit commands.

top8.6 Clearing (resetting) the buffer

Having used the buffer in several occasions using various locations and having overwritten different parts of the buffer, it tends to get messy. Not that it matters, but if you would like to get a clean buffer filled with FFh (255d) byte values then this is the command to use.

This might be handy if you are going to blow a bank containing different codesections and need to emulate the empty gaps (which aren't used) on the Eprom. Clear (reset) the buffer and then start loading the code sections.

You can only activate the **<>MBCL** command by typing its sequense - using the MEMORY menu will not activate it (since it's a protected command).

top9. Using the Eprom commands

All Eprom related commands use the default range, Eprom type and Eprom bank settings, as seen in the status window. Further, the Eprom is defined to be in slot 3. Therefore selecting Eprom bank 3Fh (63d) will select bank FFh (255d) behind the scenes (top bank of slot 3). This is not reflected in the Eprom bank setting, but necessary since the blowing hardware is connected to port 3 only. The Eprom bank setting uses relative bank numbering, with 3Fh as the top bank. The relative numbering scheme is also very useful, since all operating system data structures uses this too. It makes it therefore very easy to find the actual information on application cards with regard to the pointers used in the data structures; just select the relative bank with the **<>EB** command as defined in the pointer and view the bank with **<>EV** using the offset of the pointer. You can read a more detailed explanation of the Z88 memory hardware in section 2.

top9.1 Defining the Eprom type

This setting is very important and is used when blowing data on your Eprom (activating the **<>EPROG** command). It defines whether you want to blow **32K** or **128K**, **256K** UV Eprom's and finally the new Intel Flash Eprom Pack (**FLASH**). When Zprom is activated the first time, the **128K** Eprom setting is chosen.

It is very important to have selected the proper Eprom type before blowing, otherwise you might end up with lots of peculiar error messages of bytes not being properly blown on your Eprom. For more information on Eprom types, read section 3.

To select another Eprom type, activate the **<>ET** command or use the direct command 'Select Eprom Type' in the front window. You will be prompted to enter an Eprom type in a small command window.

To select the available Eprom types you can use **<>J** repeatedly. This will insert the parameter into

the command line for you. Four types are currently available; **32K**, **128K**, **256K** and **FLASH**. **32K** represents the 32K Eprom type, **128K** and **256K** represents the 128K, 256K Eprom types. **FLASH** represents the 256K, 512K or 1MB Intel Flash Eprom types (available from Rakwell). If you don't want to pre-select a type with **<>J**, just enter the type directly from the keyboard.

To install a new default Eprom type setting, just press< ENTER> after your selection. The Eprom type will be displayed in the status window.

top9.2 Defining the default Eprom bank

This setting is used to identify which bank is to be blown with the contents of the buffer when you activate the **<>EPROG** command. The default Eprom bank is displayed in the status window. Please note the relative bank numbering, even though slot 3 is actually used by Zprom.

You can change the default Eprom bank with the **<>EB** command or using the direct command 'Select Eprom Bank' in the front window. You will be prompted to enter
a relative bank number of slot 3. The current bank number is pre-typed for convenience.

You install the new default Eprom bank number with <ENTER>. The bank number will be displayed in the status window.

top9.3 Checking the Eprom bank range

Although Zprom performs an internal check of the current range in the current Eprom bank before starting to blow data on the Eprom (the actions of the **<>EPROG** command) you might perform the check your self. An empty Eprom ready to be blown contains only bytes of value FFh (255d or 11111111b), ie. all bits of the Eprom is 1.

You perform a check of the current range of the current Eprom bank with the
<>EPCK command, which immediately performs the checking. If all bytes in the range are FFh then you will be reported with a "NOT USED" message, otherwise the first byte found not to be empty is reported as "Eprom already used at

top9.4 Blowing the buffer into the current Eprom bank

This command is the very core or purpose of Zprom - to actually blow data from the buffer of the current range into the equal range in the current bank of the Eprom installed in slot 3.

You execute this action with the **<>EPROG** command using the current Eprom Type Setting. You cannot execute it from the Eprom command menu because it is a protected command which needs to be explicitly typed to be executed.

Before issuing this command make sure that you have selected the proper Eprom type, otherwise you might end up with an error message saying that a particular byte wasn't able to be programmed by Zprom. The Eprom should be in place as well...

There is three phases of the blowing process:

1. The current range of the current Eprom bank is checked to contain only byte values of FFh (255d or 11111111b). If not, the command is aborted with an error message of the first address that didn't held the correct byte. All empty Eproms (whether UV light or Flash) contains FFh in all bytes.

2. The actual blowing begins. The screen is turned off (only for conventional UV Eprom's - Flash Eproms are being blown with the screen turned on). Then the contents of the buffer in the current range is blown to the Eprom. All bytes are usually blown the first time. 'Lazy' bytes are re-programmed if they weren't blown the first time. Re-programming is attempted max. 75 times of a byte according to the hardware information in the Developers' Notes. If the any byte of the Eprom didn't blow correctly, an error message is reported and the command is aborted.

3. Although an implicit action, the verification and over-programming of the Eprom range contents is performed during programming. Please note that Flash Eproms are not being overprogrammed (all necessary programming is performed by the Flash Eprom Chip).

The overall process takes approx. 30 seconds for a full 16K on UV Eprom's and about 2 seconds for Flash Eproms. When completed, you will be reported with a "Eprom at range programmed successfully".

When blowing operating system data structures to the Eprom (creating external applications) you must be very careful. Please read section 9.9 for more details.

top9.5 Verifying the current Eprom bank with buffer

The verification process is performed automatically during Eprom programming. However, it might be useful if you have programmed several sub sections of the Eprom in different stages and want to make sure that all sections are present on the Eprom in the complete bank range (0000h - 3FFFh).

You perform a verification of the current range of the buffer with the current range of the current Eprom bank with the **<>EPVF** command.

top9.6 Reading the current Eprom bank into the buffer

Zprom also facilitates Eprom reading with the **<>EPRD** command. As always, the reading is performed by copying the current Eprom bank in the current range to the buffer of the equivalent range.

Pay attention to the range settings before reading Eprom contents. This is particularly important if you want to copy a whole Eprom bank into the buffer.

Reading Eprom's is necessary if you want to make internal copies of application cards. Set the range to 0000h - 3FFFh read all the appropriate banks of the card. For each bank, issue the **<>MBS** command to save the bank to the Z88 filing system. To distinguish each bank, name the files with an appropriate name and use the file name extension to identify the bank number, e.g. '3f' for the top bank of the card.

When completed with all banks, make a backup on a stationary computer using the EazyLink popdown []L (default no conversion, no translation during file transfer). It should then be safe to use the Eprom card for other purposes. Please, don't misuse this command for software piracy.

top9.7 Viewing the current Eprom bank

You can easily view the contents of the current bank (and range) by executing the **<>EV** command or selecting 'View Eprom Bank' in the front window. As with the **<>MV** command you will be prompted to enter the start address to be viewed. The start range address is pre-typed for convenience.

Viewing an Eprom is always performed in slot 3. You can even view the contents of an empty Eprom slot. Try moving back and forth in a single page - you'll notice that the hardware generates random bytes in an empty slot when reading it.

Please refer to section 8.4 for a description of available cursor movement facilities.

Editing of an Eprom is not possible. You can only blow information into it using the **<>EPROG** command. Therefore an "Edit Eprom" command is redundant.

top9.8 Searching in the current Eprom bank

Seaching on an Eprom is possible using the **<>ES** command. As with the **<>MS** command you will be prompted to enter a search address.

Searching is performed in the current Eprom bank (in slot 3) only. Therefore, when the search reaches the top of the bank, it continues at the bottom of the bank and finishes at the start search address.

You may enter an ASCII search string (case-dependent search) or hexadecimal byte sequense. Please refer to section 8.5 on how to enter the search parametes.

top9.9 Application Card Guidelines

When you blow an application ROM header (at the top of the Eprom), the Z88 immediately becomes aware of this and performs an initial installation of the external application(s). Make sure that all necessary data structures are in place before blowing the ROM header, otherwise the system reads random information and probably crashes the Z88 when the in-appropriate ROM header was blown.

However, the external applications have not been properly installed. You can identify this by looking in the 'Applications' list - the new external appliation(s) haven't got their hotkeys yet. Remember that you didn't physically insert the card, but just put a ROM header on the card.

You can create instantiations of the partially installed appliations by just hitting <ENTER> on the

name in the INDEX applications list, but we don't know if there is any side-effects. Please perform a proper installation before running the applications.

To perform a proper insertion of the new application card it is necessary to perform one of the following methods:

1. Activate the INDEX. Open the flap, remove the card and close the flap. The system removes the partial installation of the external applications. Then re-open the flap and insert the card again. This time the external applications will be installed properly into the system.

2. Activate the INDEX and perform a soft reset with the **<>PURGE** command.

top10. Using the modified RAM card commands

The task of continously blowing Eprom cards for each modification of data or code of your applications is a serious attack on your Z88 hardware. Firstly, your slot 3 edge connector gets heavily used resulting in bad connections some time in the future. Secondly, your Eprom cards will need more and more time to get erased properly. Thirdly, the time taken to test the new software IS LONG (inserting, removing, erasing and blowing Eprom cards).

This tedious task created the need for something better. The Developers' Notes V2 handed the solution: Cambridge Computer wrote about a developers' package containing a modified RAM card, that would enable you to use it as an Eprom card and to upload data using a simple file utility. They never finished the package - just a few months after the release of Developers' Notes they moved to Scotland and dumped all Z88 relates affaires.

top10.1 The modification

We made the RAM card with the easiest usage interface as possible. And it was in fact a very simple setup.

The RAM card has its Write-Enable pin controlled by a read-switch. By simply putting a magnet close to the switch, it will write-enable the RAM card. With no magnet added the RAM card is write-protected by default and behaves like an Eprom card. Removing the RAM card from the Z88 will erase its contents.

However, you need to have the right tools to open and modify the RAM card. The plastic casing is strong around the four clamps that hold the card which makes it difficult to remove the circuitry plate. If you break the clamps, a strong glue is needed to hold it together when re-assemling the card. A strong magnet is recommended too. The read-switch is positioned as close as possible to the front of the RAM card. Remember that the magnet needs to 'open' the read- switch through both the plastic casing of the RAM card and the flap (when the card has been inserted into a slot).

We use a professional engineer to make the modification. And buy a strong magnet that attaches properly.

top10.2 Inserting, removing the modified RAM card

The RAM card is treated as any normal card when it is to be inserted. Open the flap, and put it

preferably in slot 2, or slot 1 if you have an expanded Z88 installed with 128K RAM or more of internal RAM and V4 of the operating system. Further, avoid inserting it in slot 3 since it draws more current from the batteries (no problem if connected to a power supply).

Since the RAM card always will be empty, you can insert the card without being in INDEX. The contents of the RAM card differs from an empty Eprom. If you look with the **<>BV** command on e.g. the top bank of the RAM card you will notice a pattern of two bytes containing FFh followed by the next two bytes containing 0. This pattern is regular and changes order for each 256th byte. We don't know why the hardware does this. But it has effect on the operating system.

You can change the contents similar to an empty Eprom by issuing the **<>RCLC** command (filling the card with bytes of FFh).

It is a different matter when applications are on the modified RAM card and you want to remove it. As with all application cards it is necessary to use INDEX to remove them. As usual, don't run suspended activities from the modified RAM card. When you first remove it, the contents are lost. You will then have to perform an external soft reset of the Z88 to stop it squeeking.

top10.3 Uploading code into the current RAM bank

Uploading code into the modified RAM with the **<>RBW** command is in principle similar to blowing data on an Eprom.

When activated, you will be prompted for an ABSOLUTE bank number, ie. BFh as the top bank in slot 2. The current memory bank is pre-typed for convenience. When you have acknowledged with <ENTER>, the bank number will be the new default memory bank. Then the slot containing the bank is examined not to hold a RAM device (:RAM.x) since it would be fatal to upload your code into a place where the filing system keeps your vital data. When all seems OK the contents of the current range in the buffer is uploaded into the equivalent range in the memory bank of the modified RAM card.

If you haven't put the magnet in front of the modified RAM card, then you'll be faced with a "RAM card is write-protected" error.

It cannot be written enough: Be careful when uploading data structures on your RAM card when the system is aware of your external applications. Especially be careful when uploading the application ROM header - first upload all data structures (DOR's, menu & help topics, etc.) and related data (help text and executable code), then at the very end upload the application ROM header. AND, remember to install your new application card properly with **<>PURGE** in INDEX. Please read section 10.8 for more information.

top10.4 Clear the current RAM bank

With the **<>RBCL** you may reset the contents of a memory bank in the modified RAM card to bytes of value FFh (similar to an empty Eprom). Please note that the **<>RBCL** command is protected - hence you can only activate it by typing the command sequense.

When activated, you will be prompted for an ABSOLUTE bank number, ie. BFh as the top bank in slot 2. The current memory bank is pre-typed for convenience.

When you have acknowledged with <ENTER>, the bank number will be the new default memory bank. Then the slot containing the bank is examined not to hold a RAM device (:RAM.x) since it would be fatal to reset memory that is occupied by the filing system. When all seems OK the specified bank will be reset.

Remember to put the magnet in front of the modified RAM card, or you will be faced with a "RAM card is write-protected" error message.

Attention: Be careful what you reset when the system is aware of external application on the modified RAM card. Deleting system data structures after the applications have been installed would be fatal to the operating system if you try to either create or call the applications on your RAM card. A system crash is most likely to occur.

top10.5 Clear RAM card

Using the **<>RCLC** you may reset the complete modified RAM card to bytes of value FFh (similar to an empty Eprom). Please note that the **<>RCLC** command is protected - hence you can only activate it by typing the command sequense.

When activated, you will be prompted for the slot number containing the card. The default slot number 02 is pre-typed for convenience.

When you have acknowledged with <ENTER>, the slot containing the bank is examined not to hold a RAM device (:RAM.x) since it would be fatal to reset memory that is occupied by the filing system. When all seems OK the specified RAM card will be reset.

Remember to put the magnet in front of the modified RAM card, or you will be faced with a "RAM card is write-protected" error message.

If you have previously installed applications (with application ROM header and related data structures) on your modified RAM card, then it is very important to IMMEDIATELY perform an external soft reset - DON'T CALL INDEX, THE SYSTEM WILL GET VERY CONFUSED AND PROBABLY CRASH THE Z88.

You could just as well activate INDEX and then remove the card physically. This would clear the memory of the card and the system will remove any application information properly out of the system.

The only reason for this command is to preserve the slot connectors, since you don't have to remove the card to reset it.

top10.6 Viewing and editing a RAM bank

The functionality of these two commands is identical to the **<>MV** and **<>ME** commands. The only difference is that you have to specify an absolute memory bank, ie. BFh for the top bank of slot 2.

The **<>BV** allowes you to view any memory bank in the Z88; the ROM and all available slots.

However, the primary reason for this command is to examine the contents of the modified RAM card.

The **<>BE** command gives you the ability to modify any byte of RAM in your Z88. This could be very dangerous if you select a bank in a RAM device or other system related memory. This command was primarily designed to be used for the modified RAM card.

Remember to put the magnet in front of the modified RAM card, or you will not be able to modify the individual bytes even though you actually type them in - the bytes just get back to their original value.

Please refer to section 8.4 to get information of the cursor commands available in **<>BV** and **<>BE**.

top10.7 Searching in a RAM bank

You can search information in all available banks of the Z88 memory with the **<>BS** command. However, searching is limited to an individual bank boundary (identical to **<>MS** and **<>ES** commands). The primary purpose of this command is to use it on the modified RAM card.

When activated, you will be prompted to enter the absolute bank number where searching will be performed. The current memory bank will be pre-typed at the command line for convenience. Then you will be prompted to enter the start search address (offset) of the bank.

Please refer to section 8.5 for information of the search parameter options.

top10.8 Operating system precautions and the modified RAM card

When using the modified RAM card you have to be careful about any editing you perform on the contents of the card when applications are registered by the operating system.

When the card contains no system application data structurres nor ROM header then uploading (with **<>RBW)** and resetting the card (with **<>RBCL** or **<>RCLC**) will have no effect on the operating system. The very moment an application ROM header has been placed on the card you have to be careful.

top10.8.1 Application Data structures

First of all don't alter any pointers to application DOR's or ROM headers using the **<>BE** (Edit Memory Bank) command. The system uses interrupts to monitor application data structures therefore if you have altered one byte of a pointer, the system is already totally confused. If you are going to change pointers, then upload the complete data structure - but before doing that make sure that the information which the pointers in the new data structure points at is already present on the card. This rule applies to all kind of application data structures: DOR's, help topics, menu topics, command topics, etc.

top10.8.2 Executable code and suspended activities

It is equally dangerous to alter the executable code of an instantiation of the application that is currently running (suspended). The system keeps track of PC counter and registers of the suspended application. If you change the executable code, it might begin executing code that is not there anymore when that application is re-entered. If you're going to upload a new version of executable code then make sure that no suspended activities of that application has been created.

top 10.8.3 Installing external applications from the modified RAM card

As with Eprom's it is necessary to install the external applications properly when an application ROM header has been uploaded to the RAM card the first time (your're probably only doing this once). The system has been aware of the applications on the card but is not properly 'inserted' as performed normallywith opening the flap and stuffing the card into one of the slots.

First of all remove the magnet (if still on the RAM card). Simply activate INDEX, then perform a soft reset with the **<>PURGE** command. You could off cource perform an external soft reset, but it's easier from INDEX. The operating system then has a proper installation of your new external application(s) on the modified RAM card.

top 10.8.4 Soft resetting and the modified RAM card

Remember to remove the magnet from the modified RAM card when you perform a soft reset, otherwise the system sees it as a conventional RAM card during the reset phase. All your application data will be lost and you will have to physically remove the modified RAM card then perform a soft reset, and finally insert the modified RAM card again. This is necessary to reclaim the RAM card for pseudo Eprom purposes and letting the system know that you have removed a conventional RAM card.

top 11. Using the Flash Eprom commands

The new Flash Eprom (256K, 512K or 1MB size) introduces a whole new possibility for developing software for the Z88. First of all the mind-blowing 1MB would be able to contain all the current available Z88 software. Next, it can be electronicallly erased which means that the old 20-40 minute UV light erasing is history!

Now, you insert the Flash Eprom in slot 3, erase the whole card or a 64K block and just re-blow your updated software, ready to be tested!

A few extra commands have been implemted in Zprom to support the Flash Eproms.

top 11.1 Getting information about the inserted Flash Eprom

Using the **<>FLI** command sequense makes a quick poll of the slot 3 and returns information about which Flash Eprom type is available. You will be presented with a small window displaying the internal chip name, it's physical size in K and how many 64K blocks that are available.

top 11.2 Erasing the Flash Eprom

The revolution about the Flash Eprom Card with regard to the Z88 is it's ability to get erased electronically. The core of the process is actually erasing 64K entities of the card. This is a new concept to the Z88 world where everything related to memory has been designed in 16K banks.

All the Intel 8 bit Flash Memory chips are designed around a 64K block architecture. The smallest member is the 256K chip ("28F020") having 4x64K blocks, then followed by the 512K chip ("28F004S5") having 8x64K blocks, and the two 1MB chips ("28F008SA" and "28F008S5") having 16x64K memory blocks.

Block numbers are defined from 0 to n-1 (n = total number of blocks on chip).

Zprom has been implemented with the **<>FLBE** (Flash Eprom Block Erase - part of the EPROM menu) to erase the card. The Flash Eprom Card can only be erased in slot 3 (due to the special hardware of that slot).

Upon activating the command you are prompted to erase the complete card. Pressing Y (for Yes) followed by <ENTER> will erase all available blocks on the card. Pressing N (for No) followed by <ENTER> will display another window where
a block number must be specified to be erased.

Erasing a 64K block takes less than a second.

top11.2.1 Precautions with erasing blocks and resident ROM applications

If you have previously "installed" applications (with application ROM header and related data structures) on your Flash Eprom Card, then it is very important to IMMEDIATELY perform an external soft reset - DON'T ACTIVATE THE INDEX, THE SYSTEM WILL GET VERY CONFUSED AND PROBABLY CRASH THE Z88.

Before erasing the Flash Eprom, make sure that no applications are being executed by OZ on the card. Activating those apps after an erasure will only execute FFh instructions (RST 38H) - in other words a crash of your Z88.

top11.3 Programming the Flash Eprom

As always, this is done using the **<>EPROG** command, previously installed with the **FLASH** setting using the **<>ET** (Eprom Type) command.
Please refer to section 9.1 and 9.4 for complete description of functionality.

top11.3.1 Precautions with blowing software and OZ data structures

It cannot be written enough: Be careful when blowing data structures on your Flash Eprom when the system is aware of your external applications. Especially be careful when blowing the application ROM header - first blow all data structures (DOR's, menu & help topics, etc.) and related data (help text and executable code), then at the very end blow the application ROM header. AND, remember to install your new application card properly with **<>PURGE** in INDEX.

top12. Cloning Application Cards

Zprom includes facilities to clone a complete Eprom application card. However, to be able to use it, you need to have an extra slot free, and that's impossible on a standard Z88 since you will have a conventional RAM card in slot 1, probably the modified RAM card (pseudo Eprom) in slot 2 and the Zprom application in slot 3 (the assembler workbench eprom). Since Zprom needs an expanded machine, slot 1 is used for extended RAM which otherwise could have been used to get a free slot.

It would be possible to have a single slot free with a Z88 upgraded to V4 of the operating system and 128K RAM or more in slot 0 (internal):

Slot 1 would be fitted with the Workbench ROM (with Zprom), slot 2 would contain the modified RAM card and slot 3 free for a card to be cloned or produced.

top12.1 Copying a ROM Card

Insert the ROM card to be copied in slot 3. For V2.2 ROM owner's: Remember to have INDEX active when you insert the card. Apply the magnet in front of modified RAM card.

Execute the **<>COPY** command. You will be asked to enter the slot number of the card to be copied. Slot 3 is always the default. Just press <ENTER> if your card is present in slot 3. Then you will be asked to enter the slot number of the modified RAM card. The default is always slot 2. When you have confirmed with< ENTER> the copying process begins. It will take approx. 3 seconds to copy a 128K Eprom application card. You will be reported with the "ROM Card is successfully copied" message when the process is completed.

Please note that the command checks the presence for a source ROM card and that you haven't tried to copy your ROM card into a true RAM card (:RAM.x device). It will report immediately if it finds such errors.

You must also pay attention to the contents of the modified RAM card. If it already contains ROM applications other than those you're going to copy, you will cause serious damage to the operating system. Just imagine if you have suspended activities of that code resident in your system which is going to get overwritten by the copy process of the new ROM card! Always remove the modified RAM card from the slot, close the flap, wait a few seconds, and insert it back
- the system has removed any applications properly and the card is cleared automatically (there is battery backup on RAM Cards). If you don't want to physically remove the modified RAM card, follow the procedure of the **<>RCLC** (Clear RAM Card) command.

When the card has been successfully copied, remove the magnet, activate INDEX and issue a **<>PURGE** (soft reset) command. This is because the new ROM card has not yet been properly installed (usually performed by inserting a physical card into one of the slots). However, the system is already aware of the new ROM card.

The soft reset will install both the copied ROM card and original ROM card properly into the system. If you look in the INDEX you will notice that the second set of the same applications appears with a leading Z in the application hotkeys.

You can now safely remove the original ROM card (probably inserted in slot 3) during the active INDEX popdown.

top 12.2 Cloning a ROM Card

This feature is an automatical process of copying all banks from a source application Eprom (or the modified RAM card) and to blow them on an empty Eprom card in slot 3. The **<>CLONE** command may be regarded as the pendant of the **<>COPY** command, that is to first copy a ROM card into the modified RAM card, then blowing it back into a new card.

The **<>CLONE** command is ideal for producing ROM cards quickly and efficiently.

You will be asked to enter the slot number of the ROM card to be cloned. Slot 2 is always the default. First it is checked that you actually are going to clone
a ROM card, then if no RAM device nor ROM device is found in slot 3, the blowing process begins, starting from the bottom bank and ending with the top bank of the source card. The order of blowing is important - remember that the system gets aware of application as soon its sees the application ROM header on the new card.

When the command has finished you will have a new ROM card partially inserted into your Z88 because the operating system has recognized it. When your're going to remove the Eprom card, it is necessary that you have activated the INDEX first, otherwise the system will remind you with its the horrifying squeeking.

The time to process an entire 128K Eprom application card may vary, but is estimated to about 6 minutes. Cloning a ROM card to a Flash Eprom is very fast
- about 4 seconds per 16K.

Due to the relatively long time the machine is not reading the keyboard while blowing data, it will probably timeout when the command has completed - this will shut down the Z88 and it seems to the user that something has gone wrong. If you awake the Z88 with both <SHIFT> keys you will be presented with a "successful" message when the command completed (or an error message if something went wrong). You can prevent this by setting the timeout parameter in the Panel to 0.

top 13. Batch programming of files using the CLI

The CLI is very useful when you have to blow lots of different files on an Eprom.

You only need to activate []+K to record your file programming once (assuming all went well, ie. no programming errors). When the programming session is over, de-activate the CLI with []-K. Copy the system file ":RAM.-/K.sgn" to your preferred RAM device. Load the CLI file into PipeDream (as plain text), insert
a new line at top of the file and add the **[]ZE** hotkey. Save the file back (as plain text) and it's ready to be executed from BBC BASIC or the FILER.

When you are creating the CLI with []+K remember to explicitly select the Eprom type. Even though it may not be necessary at the time, it may be important when you execute the CLI file another day. You might have changed the Eprom type and when executing the CLI, the wrong Eprom type selection may be active and your programming session will probably fail.

Executing a CLI file reflects a successful programming session of the Eprom. If an error occurs

during CLI execution, it will continue to be executed since there is no supervising error logic built into the CLI system. We have prevented such accidental executions by forcing you to press <ESC> at an error message. This key is not used in Zprom programming sessions therefore when Zprom encounters an error messag