The COMPUTER JOURNAL®

Programming – User Support
Applications

Issue Number 30

$3.00

ISSN # 0748-9331

THE COMPUTER JOURNAL
190 Sullivan Crossroad
Columbia Falls, Montana
59912
406-257-9119

Editor/Publisher
Art Carlson

Art Director
Donna Carlson

Production Assistant
Judie Overbeek

Circulation
Donna Carlson

Contributing Editors
Joe Bartel
Bob Blum
C.Thomas Hilton
Bill Kibler
Frederick B. Maxwell
Jay Sage
Kenneth A. Taschner

***Subscription rates**—$16 one year (6 issues), or $28 two years (12 issues) in the U.S., $22 one year in Canada and Mexico, and $24 (surface) for one year in other countries. All funds must be in US dollars on a US bank.*

Send subscriptions, renewals, or address changes to: The Computer Journal, 190 Sullivan Crossroad, Columbia Falls, Montana, 59912, or The Computer Journal, PO Box 1697, Kalispell, MT 59903.

Address all editorial and advertising inquiries to: The Computer Journal, 190 Sullivan Crossroad, Columbia Falls, MT 59912 phone (406) 257-9119.

The COMPUTER JOURNAL

Features

Issue Number 30

Implementing ZCPR3 IOP Support for Ampro
Featuring NuKey, a Keyboard Re-definition IOP
by Rick Swenton

If you are the proud owner of an Ampro Z80 "Little Board" you are fortunate to have one of the finest complements of hardware and software available for a very modest price. Like any well designed piece of hardware, performance is dependent on well designed software. Ampro, in its infinite wisdom, chose to implement support for Richard Conn's ZCPR3 replacement Console Command Processor (CCP). When they created their Basic Input/Output System (BIOS), they built-in the required routines to initialize the special reserved memory locations needed in a ZCPR3 system. At the time the BIOS was written and subsequently revised, ZCPR3 I/O Packages (IOP) were not widely used. This was because IOPs were not really understood by the masses and IOP software was not readily available. Many users did not feel that they needed the features that an IOP provided such as console capture to disk, redirection of console/printer/modem I/O or operating devices in tandem. Richard Conn's original concept of an IOP was to remove the I/O routines from the BIOS and place them into the IOP segment. In this way, many different IOP's could exist and be loaded on demand. Each would provide you with custom specialized I/O tasks. Because IOP's were not very popular, Ampro chose not to allocate space for an IOP in the reserved memory set-up by the BIOS at cold boot time.

Things chance very fast in the 8-bit world. Suddenly, IOP's are becoming very popular. Echelon, the official source of ZCPR3 and a full line of Z-System products, now has three full-featured IOP's available. If you use Ampro's standard BIOS, you can not run any of them without modifying the BIOS. Users of the Micromint SB-180 were running around left and right using IOP's and rubbing my nose in it. Micromint had chosen to provide IOP support in their BIOS. But then the Ampro had been out in the market before the SB-180.

In this article I will show you how to modify the Ampro BIOS to support IOP's. I will also provide an overview of Echelon's NuKey IOP, which is an advanced function key redefinition program that allows you to redefine your function and cursor keys on the CRT terminal.

Changing the BIOS

I am using Ampro's BIOS version 3.8, but the changes are fairly generic and also apply to the earlier versions although you may have problems finding some of the locations in the older versions. Of course, to make the changes, you will need a copy of the BIOS source code - BIOS38.ASM - or whatever version you have. The BIOS source code is available from AMPRO.

In the current Ampro ZCPR3 implementation, the following areas are initialized by the BIOS (addresses are in Hex, earlier BIOS versions did not implement all of these):

```
FFD0 - ZCPR3 External Stack
FF00 - Multiple Command Line Buffer
FE00 - Environment Descriptor
FDD0 - ZCPR3 External FCB
FD80 - ZCPR3 Message Buffers
FD00 - Shell Stack
FC00 - Named Directory Area
FA00 - Flow Control Package
F200 - Resident Command Package
```

We will locate our IOP segment 1.5k below the RCP:

```
EC00 - Input/Output Package
```

So, the first change to make is to define the address of the IOP. Find the IOP and IOPS equates and change their values as follows:

```
Old Code:

IOP       EQU       00000H       ; Redirectable I/O Package
IOPS      EQU       0            ; size in 128-byte blocks

New Code:

IOP       EQU       0ECOOH       ; Redirectable I/O Package
IOPS      EQU       12           ; size in 128-byte blocks
```

Next, we need to change the BIOS jump table to re-direct certain jumps to the IOP. The jump table is at the start of the BIOS, just after the ORG BIOS statement:

```
Old Code:

          JMP     BOOT          ; Cold start
WBOOTE:   JMP     WBOOT         ; Warm start
          JMP     CONST         ; Console status
          JMP     CONIN         ; Console character in
          JMP     CONNOUT       ; Console character out
          JMP     LIST          ; List character out
          JMP     PUNCH         ; Punch character out
          JMP     READER        ; Reader character in
          JMP     HOME          ; Seek to home position
          JMP     SELDSK        ; Select disk
          JMP     SETTRK        ; Set track number
          JMP     SETSEC        ; Set sector number
          JMP     SETDMA        ; Set DMA address
          JMP     READ          ; Read disk
          JMP     WRITE         ; Write disk
          JMP     LISTST        ; Return list status
          JMP     SECTRAN       ; Sector translate

New Code:

          JMP     BOOT          ; Cold start
WBOOTE:   JMP     WBOOT         ; Warm start
          JMP     IOP+12        ; Console status
          JMP     IOP+15        ; Console character in
          JMP     IOP+18        ; Console character out
          JMP     IOP+21        ; List character out
          JMP     IOP+24        ; Punch character out
          JMP     IOP+27        ; Reader character in
          JMP     HOME          ; Seek to home position
          JMP     SELDSK        ; Select disk
          JMP     SETTRK        ; Set track number
          JMP     SETSEC        ; Set sector number
          JMP     SETDMA        ; Set DMA address
          JMP     READ          ; Read disk
          JMP     WRITE         ; Write disk
          JMP     IOP+30        ; Return list status

          JMP     SECTRAN       ; Sector translate
```

Now find the label HDCODE which is just before the Cold Boot Entry:

```
HDCODE  EQU     $               ; End of hard disk code
```

Between this statement and the Cold Boot code, add the following code:

New Code:

```
;
;       This is the Auxiliary Jump Table for the extended IOP
;
IOPRET:
        JMP     CONST
        JMP     CONIN
        JMP     CONOUT
        JMP     LIST
        JMP     PUNCH
        JMP     READER
        JMP     LISTST
```

Be sure you place this code before the label BOOT: because all code after BOOT: is overlayed by system storage after a Cold Boot. We need to have IOPRET resident all the time.

The next set of changes are to the ZBOOT routine.

Old Code:

```
ZBOOT:  LXI     B,3             ; Set up the ZCPR3 command line
        LXI     H,CMDSET        ; .  pointers
        LXI     D,Z3CL          ;
        DW      LDIR80          ;

        LXI     B,10            ; Move the automatic command to
        LXI     H,AUTOCMD       ; .  the ZCPR3 command line
        LXI     D,Z3CL+3        ;
        DW      LDIR80          ;

        LXI     B,11            ; Move the initial path descriptor
        LXI     H,PATH          ; .  to the proper location
        LXI     D,EXPATH        ;
        DW      LDIR80          ;

        MVI     A,0FFH
        STA     Z3WHL           ; Turn the wheel byte on

        LXI     H,ENV           ; Move the environment to
        LXI     D,Z3ENV         ; the proper location
        LXI     B,ENVEND-ENV
        DW      LDIR80

        ENDIF                   ; ZCPR3 init

        XRA     A               ;
        STA     CDISK           ; Indicate disk 0 selected
        STA     HSTACT          ; Set host buffer inactive
        STA     UNACNT          ; Clear unalloc count
        STA     HSTSID          ; Assume side zero
        JMP     GOCPM           ; Initialize & jump to CP/M

LOGMSG: DB      CR,LF,LF
```

New Code:

```
ZBOOT:  LXI     B,3             ; Set up the ZCPR3 command line
        LXI     H,CMDSET        ; .  pointers
        LXI     D,Z3CL          ;
        DW      LDIR80          ;

        LXI     B,10            ; Move the automatic command to
        LXI     H,AUTOCMD       ; .  the ZCPR3 command line
        LXI     D,Z3CL+3        ;
        DW      LDIR80          ;

        LXI     B,11            ; Move the initial path descriptor
        LXI     H,PATH          ; .  to the proper location
        LXI     D,EXPATH        ;
        DW      LDIR80          ;

        MVI     A,0FFH
        STA     Z3WHL           ; Turn the wheel byte on

        LXI     H,ENV           ; Move the environment to
        LXI     D,Z3ENV         ; the proper location
        LXI     B,ENVEND-ENV
        DW      LDIR80

        LXI     H,IOPENT        ; Move the dummy IOP
        LXI     D,IOP           ; to the proper location
        LXI     B,IOPLEN+2      ; also copy the 2 bytes after IOPEND
        DW      LDIR80
        LXI     H,IOPRET        ; Change the CBOOT vector to point
        SHLD    BIOS+1          ; to the IOPRET table

        ENDIF                   ; ZCPR3 init

        XRA     A               ;
        STA     CDISK           ; Indicate disk 0 selected
        STA     HSTACT          ; Set host buffer inactive
        STA     UNACNT          ; Clear unalloc count
        STA     HSTSID          ; Assume side zero
        JMP     GOCPM           ; Initialize & jump to CP/M

;
;       This is the dummy IOP loaded on cold boot
;
IOPENT:
        JMP     IOPEND          ; return 'not implemented'
        JMP     IOPEND          ; return 'not implemented'
        JMP     IOPEND          ; return 'not implemented'
        JMP     IOPEND          ; return 'not implemented'
        JMP     IOPRET          ; CONST
        JMP     IOPRET+3        ; CONIN
        JMP     IOPRET+6        ; CONOUT
        JMP     IOPRET+9        ; LIST
        JMP     IOPRET+12       ; PUNCH
        JMP     IOPRET+15       ; READER
        JMP     IOPRET+18       ; LISTST
        JMP     IOPEND          ; return 'not implemented'
        JMP     IOPEND          ; return 'not implemented'
        JMP     IOPEND          ; return 'not implemented'
        JMP     IOPEND          ; return 'not implemented'
        JMP     IOPEND          ; return 'not implemented'
        DB      'Z3IOP'
        DB      'DUMMY   '

IOPLEN  EQU     $-IOPENT
IOPEND  EQU     IOP+IOPLEN

        XRA     A               ; this byte is at IOPEND
        RET

LOGMSG: DB      CR,LF,LF
```

One more minor change is to be made. On the last page of the listing, find the two labels RESERVE. The second one will be zero. The first should be set to 0EC00H as shown below:

Old Code:

```
        IF      ZCPR3
RESERVE EQU     0FD00H          ; (0FD00H if ZCPR3)
        ENDIF
        IF      NOT     ZCPR3
RESERVE EQU     00000H          ; (00000H if no ZCPR3)
        ENDIF
```

New Code:

```
        IF      ZCPR3
RESERVE EQU     0EC00H          ; (0EC00H if ZCPR3)
        ENDIF
        IF      NOT     ZCPR3
RESERVE EQU     00000H          ; (00000H if no ZCPR3)
        ENDIF
```

That completes the BIOS changes. You can assemble the BIOS source and check for errors.

IOP Theory

The original concept of an IOP was to remove all console, printer, reader and punch device drivers from the BIOS and place them in the IOP. In this way, separate IOP's could be created for special applications which would otherwise require modifications to the BIOS. For example, you could have an IOP which has data sent to the console port and the modem port at the same time so that two users could be attached to the computer. You could make changes to your BIOS to do this but I am sure you would not want these two devices in parallel all the time. You would then need two separate BIOS's on two separate boot disks. However, with the IOP approach, the console and modem drivers are in the IOP. You could create a standard IOP and a custom IOP with the console and modem in parallel. Each could be loaded on demand thereby transforming standard I/O to a custom one and back again. A side benefit is the decrease in size of the BIOS after the device drivers are removed. You do lose the space savings to the 1.5k IOP reserved memory but you are allowed an unlimited number of 1.5k IOP segments. This adds great versatility to your BIOS.

Recently, Joe Wright, a prominent member of the Z-System community proposed an extension to the basic IOP concept. Joe's proposal was to leave the device drivers in the BIOS and to have the IOP act as a "front-end" or "pre-processor" to BIOS device driver calls. With this method, the Jump Table at the beginning of the BIOS was modified to direct console, printer, reader and punch calls to the IOP instead of to their respective routines within the BIOS. When a call is made to the BIOS for one of these I/O routines, the BIOS first passes the call on to the IOP. The IOP will perform some pre-processing and then pass the call on to the originally intended routine in the BIOS. This is how Joe Wright's NuKey works. NuKey is an IOP which performs console keyboard key re-definitions. With NuKey, you could program your function keys to be translated into a string of 1 to 15 characters. Let's say that you use NuKey to translate your CRT's function key F1 to the string "DIR<return>." Let's also say that your CRT sends ESCAPE-A when the F1 key is pressed. When NuKey is running, all console input requests are directed through the NuKey IOP first. NuKey is monitoring all characters coming from your keyboard. Suddenly NuKey sees ESCAPE-A come in. Instead of passing those two characters to the BIOS CONIN routine, NuKey sends the string DIR<return> to the BIOS and your screen displays a disk directory because you pressed the F1 function key.

For any IOP to pass the intercepted call back to the BIOS, there has to be an internal jump table placed somewhere in the BIOS. The IOP will know the starting address of this table and just add multiples of three to this address to get where it wants to go.

During the cold boot, the BIOS must initialize the IOP area with a "DUMMY" IOP. This is needed because the Jump Table at the start of the BIOS was modified to pass some calls to the IOP. There had better be something there in the IOP area to handle these calls. At cold boot, we load a primitive IOP which simply passes all calls back to the BIOS through the internal jump table we placed in the BIOS.

If we were to load any IOP into its reserved memory, it would overlay the DUMMY IOP. Then the newly loaded IOP would handle all BIOS requests for I/O. If at some time we no longer wish to have this IOP active, the only way to de-activate it is to load another IOP over it. If we do not want any IOP to be active, we load the DUMMY IOP into the IOP reserved memory. With the DUMMY IOP loaded, the BIOS acts like a standard one.

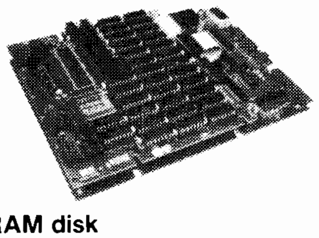
To create a DUMMY.IOP file, boot your system and do not load any IOP so that the BIOS generated DUMMY IOP is resident in the IOP memory area. Invoke DDT or equivalent and enter the following commands:

```
A0>DDT<ret>
-MEC00,F1FF,100        (This moves the DUMMY IOP from EC00 to 100)
-^C                    (Get out of DDT with control-C)
A0>SAVE 6 DUMMY.IOP    (Save 6 pages to the disk file DUMMY.IOP)
```

The above sequence created a copy of the BIOS generated DUMMY IOP which resided in the IOP reserved memory area into a disk file.

It should be noted here that if you make any changes to your BIOS which cause it to change size or change the location of the internal jump table, you will have to create a new copy of the DUMMY IOP into the DUMMY.IOP disk file. Also, any previously created IOP files such as the ones created with NuKey will most probably crash if you attempt to use them on a BIOS which changed size. You will have to use NUKEY.COM to create new ones, which is a very simple procedure.

MDISK



A fast 1 megabyte RAM disk
for your AMPRO Z80 Little Board!

Fast RAM workspace greatly speeds up disk-intensive operations like wordprocessing, database access, and program development.

- 5.75" × 5.25" printed circuit board plugs into your AMPRO Z80 socket.
- Add standard 256K RAM chips for up to 1 megabyte of extended RAM.
- MDISK driver software enables the extended RAM to be used as a solid state disk drive, complete with system track for instant warm boots.
- Driver software supplied as boot-time utility for use with standard AMPRO systems using current BIOS version 3.8. Source code for BIOS driver inserts also included for custom installations.
- Includes extended RAM test utility.
- Requires 5vdc at .60 amp via standard disk drive power connector
- Little Board must be modified to replace the 64k RAM chips with sockets and to add one jumper. Complete instructions included.

MDISK (0k RAM supplied), including complete manual and software disk, only $149 plus $5 shipping and handling. California residents add 6% sales tax. Checks, COD, MO accepted.

n/SYSTEMS

21460 Bear Creek Road, Los Gatos, CA 95030

The NuKey.IOP

Now that you have modified your BIOS to support IOP's, you can get on with the task of making your terminal's keyboard a friendly place to work. If you are like most computer users, you regularly run several keyboard intensive programs such as word processors, spreadsheets, database managers and even Z-System (ZCPR3). If your applications are like most applications, the keyboard and special function key definitions are different between applications and even when they work, you wish they could be the way YOU wanted them. You also wish that the definitions would remain the same from program to program. Enter NuKey.

Earlier, I mentioned that Joe Wright was a prominent member of the Z-System community. Joe is much more than that. Joe is an experienced Z-System "Power Programmer." When ZCPR3 first came out, we were all very impressed. Joe Wright's Z-System programs are so tightly integrated into the "Z" environment and command full control of the "Z" features that he renews our excitement all over again. NuKey is no exception.

The Basics

NuKey is a keychanger IOP. A keychanger, sometimes called a keyboard redefinition program, can be described as a key translator. When the keychanger is running, it places itself between the keyboard data and the data's destination, usually the system BIOS. Without the keychanger, the keyboard data is processed by the BIOS. With the keychanger, the keychanger intercepts the keyboard data and examines it. It looks to see if it should pass this data along to the BIOS without modifying it or to translate the data into some pre-determined different data. Normally, the characters such as the letters of the alphabet, numerals and punctuation will not be altered. But you may decide to have the keychanger translate your keyboard's special function keys into strings of characters to perform system commands. For example, your F1 through F6 keys could be programmed to perform the functions DIR<ret>, LDIR<ret>, VFILER<ret>, VMENU<ret>, NULU<ret> and EDIT<ret> respectively. Notice that the carriage return shown as "<ret>" was also programmed into the string sent out by the function keys. You could have also programmed your F1 through F4 keys to generate the control functions ^E, ^X, ^S, and ^D for Wordstar compatible cursor movement. Under NuKey, you can have as many different keychanger IOP's as you desire. For example, my SYS.IOP system keychanger translates my keyboard's "ERASE" key into the string "CLS" which is the Z-System command to clear my screen. When I run my word processor, I load WP.IOP which translates by keyboard's "ERASE" key into ^G (control-G) which is the Wordstar compatible erase character function. When I use the modem program, the MODEM.IOP allows me to send my name and password to sign-on to a remote computer with the touch of two keys.

To make things confusing, NuKey is more than a keychanger. NuKey is a "string changer." This means that NuKey can not only translate single character keys into a string of one to 500 characters but it can also translate keys which themselves generate from one to 15 characters into another string of one to 500 characters. Those of us who have CRT terminals which generate more than one character when the function and cursor keys are pressed can really appreciate this feature of NuKey. Almost any character or string of characters generated by a keyboard can be translated into another string of characters by NuKey.

Another very powerful feature of NuKey is the "extend" key. The extend key can almost be thought of as a special keyboard "shift" key. Using the extend key, you can assign a translation to any key on your keyboard, but the translation will only take place if the extend key was pressed first. For example, you can use NuKey to allow a control key, such as control-S, to be passed unchanged but be translated to some other character or string only if the extend key was pressed before the control-S was pressed. This allows you to keep any or all keyboard keys unchanged yet still have a complete set of definitions assigned to them. The translation to the special definitions will only take place if the extend key is typed first.

The Details

NuKey requires a Z80 compatible microprocessor. This includes the 64180 and the NSC-800. NuKey also requires that you be running ZCPR3 and that your BIOS supports the implementation of IOP's as described in the Echelon publication "ZCPR3 and IOPs."If you followed my BIOS modifications in the first part of this article, then your BIOS does conform to this requirement.

NuKey also works with Echelon's "Z-Com" auto-install Z-System package as well as their bootable Z-System disks for Kaypro and Morrow, and Micromint's SB-180.

As distributed, NuKey is in COM file form. NUKEY.COM can create an unlimited number of personally customized IOP's for your every application.

NUKEY.COM is a ZCPR3 utility but does not have to be "installed" on most systems. The owners manual does state that NUKEY.COM may lock-up some systems. If this happens on your system, just install NUKEY.COM like you do most other ZCPR3 utilities with the Z3INS program.
When you invoke NUKEY, select "M" from the Main Menu to create a new IOP. When you are asked for a filename, press RETURN and the default file NUKEY.IOP will be created.

NUKEY.IOP is a ZCPR3 segment loaded with the LDR utility. Just type "LDR NUKEY.IOP". Now you need to select your DEFine and EXTend keys.

Defining Your Keys

When you are asked to select your DEFine key you should select a key which is not frequently used. Remember that whatever key you select will no longer be available for use under any other program except NuKey. A good choice for this character is the back-slash "\" or the tilde "~". I prefer the tilde because on my keyboard it requires the shift key to be pressed at the same time. This minimizes accidental pressing of the DEFine key.

Next, you are asked to select the EXTend key. Here again you should select a key which is not frequently used. However, whatever key you select will remain available to your other software.

After you select your DEFine and EXTend keys, NuKey is ready to run. All "virgin" copies of NUKEY.IOP (or whatever you name them) created with NUKEY.COM will ask you to select these two keys the very first time the IOP is run. After they are selected, they are stored within the IOP and are never asked for again. If you need to change them, you must create a new "virgin" copy of the IOP and start from scratch.

To assign new definitions to the keys, press the DEFine key. If the EXTend function will be used with this key, press the EXTend key next. Then, press the key you wish to re-define. Now, enter the character or characters that you wish this key to generate. The DEL, RUB or backspace keys can be used to correct your typing mistakes while entering the characters. If you want a carriage return at the end of the character string, you must enter it explicitly. It will appear as ^M on the screen. Press the DEFine key again to end the definition.

To delete an old definition, press the DEFine key. If the EXTend function was used with this key, press the EXTend key next. Now enter the character whose definition you wish to delete. The DEL or RUB keys will delete that definition.

To display the currently assigned definitions, press the DEFine key twice. A list of definitions will appear on the screen.

The assigning of new definitions, deleting of old definitions and viewing of current definitions can be performed at any time, even while you are running an application such as a word processor or spreadsheet. There is a minor drawback to this, and I personally feel it is very minor. In order to see the definitions, you have to display them on the screen. The application program does not know that NuKey is writing information to the screen and NuKey may be writing over some data being displayed by your application. NuKey is only writing over the data displayed on the screen and not over the data in the application program's buffers. This means that if you view, add or delete NuKey definitions while you are running an application program, you may have to make your application program re-write the screen after you are done with NuKey.

Saving Definitions to Disk

There are two ways to save NuKey definitions to disk. You can invoke NUKEY.COM by just typing NUKEY<return>. This will bring-up NuKey in the menu mode. Simply follow the menu choices. The alternate method is to invoke NUKEY.COM and pass the parameters from the command line. Type NUKEY S NUKEY.IOP<return> to save the current NuKey definitions into the file called NUKEY.IOP. As with any ZCPR3 utility, the command NUKEY // will display a short help screen.

Restoring Definitions From Disk

The ZCPR3 loader program LDR.COM is used to load previously saved IOP files into memory. Just type LDR NUKEY.IOP<return>.

Deactivating NuKey Completely

As I stated in the first part of this article, in order to deactivate an IOP, a new IOP must be loaded on top of the old one. If you do not want any IOP to be running, you need to load a "dummy" IOP. This "dummy" IOP is just a series of JUMP instructions which are loaded into the IOP memory segment. They make the IOP look just like it did right after cold boot when your BIOS initialized the IOP area. You can create a DUMMY.IOP file using a debugger like DDT.

Immediately after a cold boot and before any IOP is loaded, the IOP memory segment has a dummy IOP in place. Use the ZCPR3 SHOW utility to determine the address of your IOP. (If you are using an AMPRO BIOS modified as shown in this article, then that address is EC00H.) Using the debugger, enter the following commands:

```
A0>DDT<ret>
-MEC00 F1FF 100
-G0
A0>SAVE 6 DUMMY.IOP
```

This will move the 1.5K dummy IOP image residing in memory between EC00H and F1FFH down to 100H through 6FFH. After exiting the debugger six 256 byte pages are saved to the filename DUMMY.IOP.

If you perform any changes to your BIOS which will change the location of the internal BIOS jump table, you will have to create new copies of your NUKEY.IOPs and the DUMMY.IOP. If you attempt to load a previously created IOP after BIOS changes which caused the BIOS to change size, the IOP will almost certainly cause the system to hang.

Customizing NuKey

NuKey has three built-in features which can be customized by the user. For nearly all applications, NuKey will run "right out-of-the-box" and customization will not be necessary. But for those times you need them, having the ability to change these three parameters can be a life saver.

The first parameter is called FUNDEL. Some terminals transmit their function keys as fast as their current baud rate will permit while other terminals deliberately insert a time delay between the function key characters. By changing the value of FUNDEL, NuKey can accommodate inter-character delays from 0 to 254 milliseconds. The default setting is 35 milliseconds.

The second parameter is called CTLDEL. This parameter is used to tell NuKey how many false console status "deceptions" to provide. This feature is used to "fake-out" certain application programs. Let's say you define a particular function key to send a complete command string and filename to your word processor. As soon as the word processor loads, it flushes keyboard input and some of your commands or some characters from the filename will disappear. Setting the CTLDEL parameter to some number other than zero will cause NuKey to respond "false" to a console status request after sending a carriage return so that the word processor will "think" that it has successfully flushed the keyboard buffer and no characters from the re-defined function key will be lost. The default setting is 8 deceptions. The range is from 0 to 254.

The third and final parameter is called KEYDEL. Some applications programs flush the keyboard after every character. Just as the above CTLDEL function provides for false console status after a carriage return, KEYDEL provides for false console status between individual characters. The default setting is 2 deceptions and can be changed from 0 to 254.

I personally have experimented with these values but can honestly say that the default values worked just fine on my Heath H19 and H29 terminals.

In Conclusion

Many Ampro Little Board computer owners have approached me with their dilemma of the BIOS not supporting IOP segments. Some have purchased one or more of the three Echelon IOP products only to find out that they can not be run on their systems. With the information provided in this article, a person experienced in assembly language programming and BIOS integration (MOVCPM, SYSGEN, DDT, etc.) should have no trouble implementing IOP support for their Ampro BIOS. The required source code to the BIOS is is available from Ampro Computers for a modest charge.

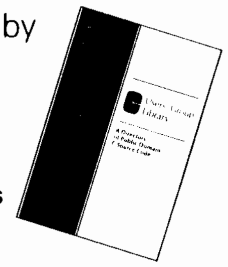
C Users' Group

Over 115 volumes of Public Domain C source code, including: editors, compilers, communication packages, text formatters, UNIX-like tools, etc., available in over 100 formats.

NEW! $10

CUG Directory of Public Domain C source code.

200+ pages of file by file descriptions and index.



Write or call for more details

The C Users' Group

Box 97 McPherson, KS 67460 (316) 241-1065

• Z Best Sellers •

Z80 Turbo Modula-2 (1 disk) $89.95
The best high-level language development system for your Z80-compatible computer. Created by a famous language developer. High performance, with many advanced features; includes editor, compiler, linker, 552 page manual, and more.

Z-COM (5 disks) $119.00
Easy auto-installation complete Z-System for virtually any Z80 computer presently running CP/M 2.2. In minutes you can be running ZCPR3 and ZRDOS on your machine, enjoying the vast benefits. Includes 80+ utility programs and **ZCPR3: The Manual**.

Z-Tools (4 disks) $169.00
A bundle of software tools individually priced at $260 total. Includes the ZAS Macro Assembler, ZDM debuggers, REVAS4 disassembler, and ITOZ/ZTOI source code converters. HD64180 support.

PUBLIC ZRDOS (1 disk) $59.50
If you have acquired ZCPR3 for your Z80-compatible system and want to upgrade to full Z-System, all you need is ZRDOS. ZRDOS features elimination of control-C after disk change, public directories, faster execution than CP/M, archive status for easy backup, and more!

DSD (1 disk) $129.95
The premier debugger for your 8080, Z80, or HD64180 systems. Full screen, with windows for RAM, code listing, registers, and stack. We feature ZCPR3 versions of this professional debugger.

Quick Task (3 disks) $249.00
Z80/HD64180 multitasking realtime executive for embedded computer applications. Full source code, no run time fees, site license for development. Comparable to systems from $2000 to $40,000! Request our free Q-T Demonstration Program.

Z-System OEM inquiries invited.
Visa/Mastercard accepted. Add $4.00 shipping/handling in North America, actual cost elsewhere. Specify disk format

Echelon, Inc.
885 N. San Antonio Road • Los Altos, CA 94022
415/948-3820 (Order line and tech support) Telex 4931646

If you are uncomfortable making these changes yourself, just send me a bootable disk of the system you are currently running. Please indicate your system configuration (hard disk, floppy disks, BIOS options desired). I will return the disk with the new BIOS for IOP support and the DUMMY.IOP file. Please indicate whether the disk is 48 or 96 tpi.

Rick Swenton
19 Allen Street
Bristol, CT 06010

Editor's Note: In view of Rick's very generous offer, it would only be fair to include some money for his time and postage.

Now that my two main systems, the Ampro Little Board and the Heath H89 are running NuKey IOP, I don't know how I have lived without it for so long. Joe Wright's NuKey IOP program really makes your keyboard a pure joy to use!

Until now, Ampro Computer users could not take advantage of the IOP applications that other users were enjoying for some time. Now you can make those changes yourself, raise your level of understanding, and install a great keychanger on your system at the same time.

Exploring the fascinating world of Z-System encourages all kinds of learning activities by its users. The original BIOS authors and systems integrators of the late 70's and early 80's never imagined in their wildest dreams that a home computer hobbyist would be modifying and enhancing the BIOS himself! This kind of activity will probably not take place in the 16-bit world for a long time if at all. Only in our 8-bit CP/M compatible world could we have all the information at our finger tips to do whatever we want, whenever we want. Only in our 8-bit CP/M compatible world do we have support companies like Echelon, expert programmers like Richard Conn and Joe Wright, and the volunteer efforts of the Z-System user community, most especially the ZSIG organizers, Jay Sage, Bruce Morgen, Richard Jacobson and many, many others.

I wish to thank Joe Wright for his assistance in the implementation of the Ampro BIOS IOP support.

The following items are available from:
Echelon, Inc.
885 N. San Antonio Road
Los Altos, CA 94022
(415) 948-3820

ZCPR3: The Manual is the reference manual for ZCPR3 and its utilities, 351 pages, typeset, bound book. $24.00

Z-System Users' Guide is a "how-to" step-by-step tutorial and an excellent complement to "The Manual" above. Loose-leaf, 95 pages. $14.95

ZCPR3 and IOPs is the standard reference manual for implementing IOPs and operating the associated utilities. 50 page loose-leaf manual. $9.95

Three IOP programs are available:

NuKey is an advanced function key generator. $39.95

Input/Output Recorder (I/OR) redirects input/output to/from console-printer-disk file. (ZRDOS Plus required) $39.95

Print Spooler (B/Printer) background single file print spooler. (ZRDOS Plus required) $39.95

ALL THREE IOP Programs above are available for $89.95

(ZRDOS Plus, an improved replacement BDOS is $59.50 additional)

Prices are subject to change.

Sources for more information on ZCPR3 IOP's:

ECHELON, INC.
885 N. San Antonio Road
Los Altos, CA 94022
(415) 948-3820

Echelon publishes a booklet called 'ZCPR3 and IOPs' by Richard Conn. ($9.95) It is a 50 page loose-leaf bound manual which explains BIOS implementation of IOP support. This manual is a MUST for anyone seriously considering writing their own IOP's or customizing their BIOS. It is also a MUST for anyone serious about understanding IOP's.

Source for BIOS.ASM for the Little Board:

Ampro Computers
67 East Evelyn Ave.
Mountain View, CA 94041
(415) 962-0230

Ampro Computers offers a product identified as "1AS" which is called "Z80 BIOS and UTILITIES Source Code." I have a fairly old price list from 1985 which shows this product's cost as $79.00. It includes the BIOS.ASM file as well as the assembly language listings of all the Ampro Little Board Utilities.

Registered Trademarks: ZCPR3, I/OR, B/Printer, Z-System, NuKey, Echelon; Little Board, Ampro Computers; CP/M, DDT, Digital Research; Z80, Zilog; H89, H19, H29, Heath/Zenith. ■