promenade®

MODEL C1

OPERATING INSTRUCTIONS

JASON - RANHEIM

SAN JOSE, CA, USA

Jason-Ranheim Company wishes to acknowledge the leadership role of Commodore Business Machines in making affordable computing products available to the general public. We have always found their products to be well engineered and versatile. Were there no VIC 20 and C64, there would be no PROMENADE® C1.

Introduction

Welcome to the PROMENADE® world.

You have just added a new dimension to your computing capability. Put your information on those marvelous things known as "EPROMS", "EEPROMS", & "EAROMS" and you can have it back in a flash. And even better, you can install your information in your computer's memory as if it were there from birth, so to speak. The programs and data you store on silicon are there for keeps unless you decide to wipe the slate clean.

The Promenade C1 gives you new disk-like capabilities for program and data storage on EPROMs. If your interest is primarily machine language, rather than BASIC, the special PROMOS π , £ and other commands discussed later will be of particular interest. Nonetheless, it will pay to read through the discussion of BASIC commands as an aid to understanding overall operation of your PROMENADE.

Getting Acquainted with the Promenade C1.

Your Promenade installs in the back left corner of the computer in the user's port. It is designed to be easily pushed into or pulled out of engagement with the computer's printed circuit board. The Promenade draws the power it requires directly from the computer.

"ZIF" SOCKET

The socket on top is of the "Zero Insertion Force" (ZIF) type which makes getting EPROMs into and out of your Promenade a snap:

> To install an EPROM in the ZIF socket: flip the lever up, drop the EPROM in and move the lever down toward the enclosure as far as it will go.
>
> To remove an EPROM raise the handle and lift it out.

It is important that the polarity of the EPROM be observed. On every EPROM there is a notch at one end. INSTALL THE EPROM WITH THE NOTCH AT THE LEFT.

With 24 pin EPROM's, the left-most 4 socket spaces are not used. Install these as far to the right as they will go in the ZIF socket.

INDICATOR LIGHTS

The green light labeled PWR indicates that your Promenade is receiving power from your computer.

The red light labeled SKT indicates that the EPROM in your Promenade is energized, and the ZIF socket is live. Installing or removing EPROMs when the red light is on must be avoided whenever possible.

The yellow light labeled PGM does two things:

First, this light comes on whenever information is being programmed into an EPROM.

Second, this light will flash on and off to signal that some error has occurred.

When your computer is first turned on, all three lights are lit. This is normal. The red and yellow lights will go out when you execute a 'Z' command to be discussed later.

PROMOS 1.0 BASIC COMMANDS

All the basic commands your computer uses to file and retrieve data from a peripheral device work with the Promenade C1 as well. The important thing to remember is:

THE DEVICE NUMBER ASSIGNED TO THE PROMENADE IS 16.

Whenever you use one of the commands OPEN, LOAD, SAVE or VERIFY call for device 16 in the command statement. Other basic file commands, PRINT#, GET#, INPUT#, CMD#, refer to the corresponding 'OPEN' statement to get the device number.

SAVE

The format for the PROMOS SAVE statement is:

```
SAVE"<NAME>:<CW>",16,<PMW>
```

NOTE: the carat symbols < and > mean "the value of" in the SAVE statement above and other command statements to come. Do not include these symbols as a part of the command you type on the screen.

Notice two important changes from the regular SAVE format:

1) The program name is followed by a colon and a number called the control word (CW). You must always specify a control word when 'saving' to the Promenade. If you don't have it there, a 'SYNTAX ERROR' will be generated.

The control word has one simple function: It tells PROMOS everything it needs to know about the EPROM you are using at that moment. PROMOS will not guess! No default values will be called up if you omit this vital piece of information.

You get the number you need from the Control Word (CW) table at the back of this booklet.

Make doubly sure you use the right control word. Using an incorrect control word could permanently damage your EPROM when PROMOS tries to execute your command.

2) The second difference is the prominent role played by what you call the 'secondary address'. We call this the Program Method Word (PMW) when referring to Promenade operations, because it tells PROMOS how you want the programming to be done. We'll discuss this in detail later on.

The PMW defaults to 0 if left out. This probably won't be the PMW you need. You'll find recommended PMW's in the table at the back of this booklet.

Example:

```
     SAVE"GODZILLA:40",16,6
```

LOAD

The format of the PROMOS LOAD command is:

```
     LOAD"<NAME>:<CW>",16
```

This is exactly like the SAVE format except the PMW isn't required (since you are only reading the EPROM in a LOAD).

Example:

```
LOAD"WORD PROCESSOR:5",16
```

OPEN

The format of the PROMOS OPEN command is:

```
OPEN<FILE NO>,16,<PMW>,"<NAME>:<CW>"
```

Here the FILE NO. is any number you care to use from 1 to 255 which PROMOS will use to keep track of the flow of information into and out of EPROM's in the Promenade.

Example:

```
OPEN7,16,7,"INVENTORY:24"
```

GET#, INPUT#, PRINT#, CMD#, CLOSE

These commands have the same format for PROMOS as in regular basic. You specify the file number, and PROMOS will refer to the information you gave in the corresponding 'OPEN' when carrying out any of these commands.

THE $ COMMAND

An inportant and very useful special PROMOS command is the $, or directory command. The format of $ is:

```
$<CW>
```

When you type the directory command then <c.r.>, a directory of files on the EPROM in the Promenade at that moment will appear on the screen. Not all at once - one line each time you press a key. (Otherwise a long directory would scroll off the top of the screen before you could get it stopped.) Actually, you easily step through the directory by holding down a repeat key such as 'cursor down' or the space bar.

Each directory entry conveys four useful things to know about each file:

The number at the left is the memory location on the

EPROM where that particular file starts. EPROM memory locations start at 0 and run through the maximum capacity of the chip.

The next item is the file name.

The next item is either a "P" for program file or a "D" for data file.

The last item is an asterisk after the P or D of the last entry to indicate whether or not that file has been closed.

Finally, at the bottom of the directory is a number which tells where PROMOS will put your next file should you choose to make one. This number also equals the number of bytes you have used.

Example:

```
$230
```

An important note: The dollar sign of the directory command must be the left-most character on the screen line for the command to be executed.

We'll return to file commands and their use later on.

SPECIAL PROMOS COMMANDS

There are six other special PROMOS commands for your use. We'll discuss these briefly here and in more detail later.

'Z' The 'Z' command is executed by typing a letter Z and a carriage return. It causes the ZIF socket to be zeroed from whatever state it was in. Zeroing the socket sets all socket pins to (essentially) zero voltage. PROMOS uses this command itself as the final step in executing any of the other commands. It's use is primarily to get you to a zero'ed state if your program 'crashes' and you need to 'Reset' or 'Restore' the computer.

(shifted) 'E' The shifted 'E' command erases a 48016P electrically eraseable EPROM. Typing (shift) 'E' and return sets all data bits to "ones" in 2/10ths of a second.

π The 'π' command transfers data from computer memory to an EPROM on a byte for byte basis. It's format is:

```
     π<Mem Strt>,<Mem End>,<EPROM Strt>,<CW>,<PMW>
```

Mem Strt is the location of the first byte in memory you want to put on the EPROM. Mem End is the location of the last byte to be programmed. EPROM Strt is the starting location on the EPROM where programming is to begin; and CW and PMW are the same as in the basic commands. Notice that if Mem Strt and Mem End have the same value, only one byte will be programmed. Remember that EPROM addresses start at 0 and go to the end of the EPROM. For example, EPROM addresses on a 2764 go from 0 to 8191, on a 27256 from 0 to 32767 and so on.

The 'π' command is the one to use for making duplicates and backup copies of programs and data.

Examples:

```
     π8192,16383,0,5,7

     π33168,40959,400,48,11
```

'£' The '£' command is the converse of the 'π' command. It causes computer memory to be filled with data from an EPROM. The format of '£' is:

```
     £<Mem Strt>,<Mem End>,<EPROM Strt>,<CW>
```

Note that the format is identical to 'π' except that PMW is left off. Again, since you are reading the EPROM , the PMW is immaterial. Mem Strt, Mem End, EPROM Strt, and CW have exactly the same meaning as in 'π'.

Be careful not to execute a £ command which over-writes other important data in your computer. For example:

£0,2047,0,40 will over-write page zero and cause your computer to 'hang'.

Example:

```
     £16384,24575,0,48
```

(shift) 'S' The shifted 'S' command takes data from a specified section of memory and generates a program file using a name you give it. The program file will appear in a directory just like a basic file. The shifted 'S' actually appears on your screen as a "♡" in the standard CBM character set, or as upper case S in the alternate set. The format of the (shift) S command is:

```
     (sh)S<NAME>:<CW>,<Mem Strt>,<Mem End>,<PMW>
```

Mem Strt, Mem End, CW, PMW, all have the same meaning as in prior commands. Notice that quotation marks are not used to enclose the file name, colon and CW. This considerably reduces the hassle of getting the command right on the screen. Second, notice there is no EPROM address given. This is because PROMOS will park this program in the next available space on the EPROM, right after your other program and data files.

Notice that the program you have just filed on the EPROM can be any chunk of memory at all. This is the command you'll use, for example, when developing pieces of machine language, a subroutine perhaps, or an entire ML program for later review and modification. The name you give it may be the label you use in an assembly source code listing.

Example:

```
     (shift)SNMI HANDLER:5,4370,4621,7
```

(shift) 'L' The converse of the (shift) S command is the shift 'L' command. This is the command you use to retrieve that file from your EPROM. The format of the (shift) 'L' command is very simple:

```
     (shift)L<NAME>:<CW>
```

There are no memory references here at all. That is because PROMOS puts your program saved with (shift) S back exactly where it came from. The (shift) S command, in other words, produces a non-relocateable file.

Example:

```
     (shift)LSUBROUTINE3:224
```

There is a capsule look at the commands special to PROMOS.

One important thing to remember: in making any of the special PROMOS commands, the first character of the command must be in the left-most position on the screen line. It won't work anywhere else.

A CLOSER LOOK AT PROMENADE FILES

In order to get the most from your Promenade, you should take a harder look at Promenade files, their structure and use.

File Storage on the EPROM

The structure of a Promenade file in an EPROM memory is in many ways similar to the structure of a basic program file. A basic statement stored in your computer's memory has a short 'header' followed by program data. Each basic statement can be viewed as a miniature file which your computer can quickly leaf through to locate a given data record. This is what happens when you type "LIST 100" for example. Basic quickly skips through the program and finds file 100 and prints it on the screen. This is possible because each basic statement is "linked" to the next by a two-byte pointer in its header. The statement number can be thought of as a file name.

In similar fashion, a Promenade file consists of individual files linked together by a two-byte pointer. PROMOS can therefore quickly step through the memory contents of a Promenade file chain and locate any particular file.

This scheme makes maximum use of each byte on the EPROM. Instead of a directory file at the beginning (which would not be big enough to suit some people or be so big as to be wasteful) the file chain is 'directorized' automatically. Let's look at a PROMENADE FILE header:

```
 FD   LINK     UTIL     NAME
| 0 | 1 | 2 | 3 | 4 | 16 BYTES | DATA - - - - - -
```

At the beginning of each file header is a file designator (FD). This byte tells PROMOS important things about the file.

Next comes a two-byte pointer to the next file. Next, are two utility bytes which may contain, for example, the "saved from" address of a program file.

Next comes 16 bytes of file name. If not all used, these bytes are left in their virgin 'all-ones' state, so that they can be altered later if need be.

THE FILE DESIGNATOR BYTE

When a file (either program or data) is opened at a particular location, bit 7 of the FD is 'written down' to zero. This tells PROMOS if it needs to know that yes, a file has been opened here. When the file is later closed, bit 6 of the FD is also written down. Bits 5,4 and 3 are not used by PROMOS 1.0 but may be used in later versions. Bit 2 is a "write protect" bit. If this bit is zero, PROMOS will not attempt to write to that file but can read from it.

Bit 1 pertains to program files and tells PROMOS whether the file is relocateable or not. A "0" here means the file is not relocateable and PROMOS will put it back where it came from or not at all. Bit 0 is the program/data file indicator bit. A "0" here shows the file to be a program file.

THE NEXT-FILE LINK

Bytes 1 & 2 of the header are the EPROM address of the next file, given in lo-byte/hi-byte order. This link is written when the file is closed. An unclosed file has 255 ($FF) in these two places. An unclosed file is consequently a barrier to the addition of any more files to the EPROM since PROMOS has no way of knowing where the new file should be put. PROMOS will refuse to OPEN another file with an unclosed file at the end of the file chain. It is therefore important to remember to close all files!

FILE NAMES

Proper file names can contain any data whatever (machine code for example). When searching for a given file name, all characters must match for the file to be 'found' unless there is an asterisk at the end of the sought-for character group. All characters after the asterisk are disregarded.

Thus:

```
     LOAD"AB*:5",16
```

will cause the first program file whose name begins with AB to be loaded from a 2764.

```
     LOAD"*:5",16
```

will load the first program file regardless of name.

SOME THINGS TO KNOW

The next-file pointer is written on the header the first time the file is closed. PROMOS will not attempt to rewrite it on subsequent closings.

Promenade files are always opened for both read and write. There is no distinction between 'read' and 'write' files. The exception is 'write protected' files as discussed earlier. In advanced filing systems, advantage can be taken of this to alter or add to a file. Relative files are particularly easy to set up with this system.

PROMOS when receiving a 'SAVE' command will first search the EPROM for a name match, then finding none, will proceed to 'SAVE' the file. If an existing file shows a name match, the 'SAVE' will not take place.

PROMOS when receiving a 'OPEN' command will first search for a name match. If it finds one, that file will be 'OPEN'ed. If PROMOS does not find a name match, it will open a new file if possible.

One doesn't have to have all the files open at a particular moment on a single type of EPROM. Up to 10 files can be open at any time; and these ten can each be opened on a different EPROM type if that were desired. PROMOS will keep things straight as to what goes with what.

PROMOS maintains a file data pointer for each open Promenade file. This pointer runs from -1 ($FFFF) to 32767 ($7FFF). This pointer is a "pre-incremented pointer"; that is, it is left pointing to the last byte accessed and is incremented prior to accessing the next data byte. The current value of the file data pointer is kept in the PROMOS file table.

THE PROMOS FILE TABLE

When each file is opened, data concerning the file is stored away in a PROMOS file table. This table corresponds to basic's table of file numbers, device numbers and secondary addresses. PROMOS uses this information and keeps other information as follows:

Each file table contains:

CW, FD, NFL, NFH, FSL, FSH, FPL, FPH

NFL and H are the lo and hi byte pointers to the next file taken from the header.

FSL and H are the lo and hi byte pointers to the start of this file.

FPL and H are the lo and hi byte of the file data pointer for this file.

CW is the Control Word for this file. This table is maintained in locations $02A7 thru $02F6.

PROMOS 1.0 does not have any direct commands for file data pointer manipulation. Indirect methods must be used if one wishes to set the pointer to a given value. "GET#'s" may be used to advance the pointer forward. Closing and re-opening the file will reset the pointer to the beginning of data.

The simplest and best way to set the pointer is to go into the table and poke the desired value prior to the data access required.

A CLOSER LOOK AT PMW

The Program Method Word tells PROMOS how to proceed with the prog ramming of a given EPROM. YOU choose:

Actually, bits 0 thru 3 control the programming method, b its 4 & 5 have other uses as described later.

If you look at the PMW table, you'll see listed a 'standard' method and one or more recommended intelligent methods.

Let's look at these in some detail:

Standard Programming:

In standard programming, the programming pulse applied to the EPROM is of constant duration, usually 50 milliseconds. All EPROMs are checked at the factory to be sure they will program successfully with the specified pulse duration (with normal tolerances on the pulse width). PROMOS selects standard programming with the pulse durations listed for the PMW values from 0-3:

PMW	Pulse Duration (milliseconds)
0	6
1	12
2	24
3	48

For example, to program a 2764 the standard way, you choose PMW=3. (about 7 minutes will be taken to program 8192 bytes).

Intelligent Methods for Programming

Standard programming assumes the worst for every byte on an EPROM. In fact, most bytes will not require nearly as long to program as the "worst-case" byte. Intelligent methods take advantage of this fact to reduce over-all programming time by, in effect, testing the EPROM as the process is going on. It works like this:

> The address and data are set up and a short programming pulse is applied. The EPROM is read back and data compared with that applied. If the data doesn't 'verify', then another pulse is applied, the EPROM read back, and so on. Programming time is accumulated. At some point, the data will have "stuck" and a positive verification will be obtained.
>
> At this point another, longer pulse is applied to insure some margin is achieved. Intel calls this the 'over-program' pulse. We call it the 'margin pulse'. The duration of this pulse is a multiple of the sum of the durations of all preceding pulses. This is based on the simple idea that the longer it took to get verification, the longer it will take to achieve a given margin. After this the data is verified once again, the programmer then moves on to the next address.

All this sounds terribly involved and it is. But its just the kind of thing your computer is good at, and PROMOS directs the process at lightning speed.

The three intelligent methods you can choose differ in a few respects.

> Intelligent #1 developed by Jason-Ranheim Co. uses test pulses of increasing duration. This method starts always with a .1 ms pulse and doubles the duration with each failure to verify. The margin pulse is of a duration equal to the sum of all preceding pulses.
>
> Intelligent #2 is compatib le with Intel recommendations for programming the 27256. The margin pulse is 3x the duration of all preceding pulses. If after a total of 25 ms. of short pulses, the EPROM has not verified, then nothing further is done and a failure is reported.
>
> Intelligent #3 is compatible with Intel recommendations for programming their 2764's and 27128's. The margin pulse is 4 times the duration of the preceding pulses. If after a total of 15 ms. of short pulses, a verify has not been achieved, then a 60 ms. pulse is applied and another attempt to verify is made. If the data is good, then programming moves on. If not, the EPROM has failed. This method in effect, reverts to standard programming (where success is pretty certain) after a certain point.

Note that in all intelligent methods, the data programmed into each location is verified twice.

PMW's from 4 to 15 have the meanings given in the following table:

PMW	INTL METHOD	TEST PULSE	MAX PERMITTED PROG TIME*
4	#1	Variable	12 ms
5	#1	"	25 ms
6	#1	"	50 ms
7	#1	"	100 ms
8	#2	.25 ms	75 ms
9	#2	.5 ms	75 ms
10	#2	1.0 ms	75 ms
11	#2	2.0 ms	75 ms
12	#3	.25 ms	100 ms
13	#3	.5 ms	100 ms
14	#3	1.0 ms	100 ms
15	#3	2.0 ms	100 ms

* This is the max allowable time that can be accumulated at a given location before failure to program is reported.

A CLOSER LOOK AT PROMENADE OPERATION

Using the commands: All the data entered in PROMOS 1.0 commands is in decimal form. No hexadecimal inputs are permitted. Later versions of PROMOS are planned which will accept inputs in Hex form as well.

All the basic commands can be used within a basic program. INPUT# and GET# must be used in a program.

All the special PROMOS commands $,Z,(sh)E,(sh)L,(sh)S,π & £ are direct commands only; and cannot b e used within a basic program.

All commands (except of course Z and (sh)E) can take variable data as parameter input.

For example, one can type on the screen: .

```
ML=8x4096:MH=ML+179:CW=5:PMW=4:ES=6x256+23
```

Then:

```
π ML,MH,ES,CW,PMW (C.R)
```

Similarly:

```
     100 A$=STR$(I)+"DATA"+":"+STR$(CW)
     110 N=8:PM=4+x:POS=16
     120 OPENN,POS,PM,A$
     etc.....is perfectly legitimate and very useful.
```

Machine Language and the Promenade

PROMOS 1.0 is coded in machine language for speed. Excellent results can be obtained when PROMOS is comb ined with user written machine language routines for performing various input/output tasks. For this purpose the use of the computers 'Kernal' routines is strongly recommended. All of these pertaining to I/0 to device 16 is implemented.

Using Promos with a Monitor

PROMOS 1.0 is designed to work with most monitors. All the special commands should work with your monitor. Much development work has been done with HESMON & HESMON 64, and except for question marks appearing here and there, PROMOS is quite compatible.

Some words of caution: If you generate a syntax error by improperly entering a parameter, you may well find yourself in BASIC. If this happens, SYS to a zero byte to 'break' back to the monitor. If you must reset, remove your EPROM first.

Another caution: When using HESMON or HESMON 64, do not scroll one of the special Promenade commands. Scrolling a PROMOS command will likely cause it to execute! Wipe the PROMOS command off before continuing your work with HESMON.

When using HESMON or HESMON 64, exiting to BASIC disables PROMOS. To re-enable, do SYS668.

Error Messages and Status

PROMOS has been liberally provided with error indications to keep you apprised of any problem. The yellow LED is particularly useful in this regard. Its flashing will alert you to an error condition generated by an operation involving the Promenade.

NON-RELOCATEABLE PROGRAM FILES

To produce a non-relocateable program file, add 16 (decimal) to the PMW you choose. (In other words, set bit 4). To load a non-relocateable program file, use any non-zero PMW tacked on to the basic load command.

Example:

```
SAVE "GAME:40",16,21 generates a non-relocateable
file named "GAME" on a 2716.  To load "GAME"

LOAD"GAME:40",16,1
```

Remember, all (sh)S files are non-relocateable.

WRITE PROTECTING FILES

To "write-protect" a file, add 32 to the PMW you choose (set bit 5). Then, on closure the write-protect bit in the File Designator will be cleared, preventing any further writing to that file.

GETTING STARTED

To get operating with your Promenade, load PROMOS into your computer from tape or disk. Then "RUN" it. PROMOS locates itself in the top area of RAM where it waits for commands involving the Promenade. If you wish to disengage PROMOS, do:

```
SYS64789 (C64)

     or

SYS64850 (VIC 20)
```

To re-engage PROMOS again without re'LOAD'ing and re'RUN'ning it. Do:

```
SYS668 (C64 or VIC 20)
```

From a monitor, you need a little program:

```
ADDR:  JSR$029C
       BRK
```

TYPE G(ADDR) and PROMOS should be there again.

NOTE: On power-up, all three indicator lights come on. This is normal. Before inserting an EPROM in the ZIF socket, do a "Z" command. This turns off the red and yellow lights and prepares the Promenade for operation.

Saving EPROM Data on Disk

Very often one wishes to read a ROM or EPROM and save the information on disk or cassette as a named program file. This is most easily done using a monitor program. All monitors have an 'S' (SAVE) command which is just what is needed. The following example will illustrate the procedure. We assume that a 2716 is to be read and saved on the 1541 disk drive as a file called "2716 DATA". We will demonstrate using the HESMON monitor.

1- Power-up.
2-Do the 'XC' command, exiting to BASIC.
3-LOAD and RUN PROMOS.
4-Do the 'Z' command.
5-Install the 2716 in the Promenade ZIF socket.
6-Read the EPROM:

```
    £8192,10239,0,40
```

7-Get back to HESMON by doing SYS8.
8-Save the data on disk:

```
    S"2716 DATA" 08 2000 2800
```

To LOAD this data back from disk and program a fresh 2716:

Do steps 1 thru 4 above.
5-Do SYS8.
6-LOAD the data from disk:

```
    L"2716 DATA" 08       (Data is loaded into $2000
                          thru $27FF automatically. The
                          disk file is non-relocateable.)
```

7-Burn the fresh 2716:

```
    π8192,10239,0,40,7
```

That's all there is to it.

Note: many 24 pin ROMs are compatible pin for pin with the MCM68764. These can be read using control word 48. Some newer 28 pin types are compatible with the 2764, 27128, and 27256 and can be read using the appropriate CW.

A WORD ABOUT DATA RETENTION

Much research has been done on the subject of EPROM data retention and long term reliability. This work shows that data is held for a great many years, provided the EPROM is protected from direct sunlight or fluorescent room light. To insure against slow erasure from those sources, cover the window with a piece of masking tape, an adhesive label or something similar.

EPROM data stored using the intelligent methods provided by the Promenade is reliably held. Even so, we rank the intelligent methods in order of assurance from least to greatest as - #1, #2, #3. The greater the duration of the margin pulse, the greater assurance we have that data will be retained over the long term.

PROMOS and the 1541 DOS WEDGE

Because of common zero page usage, PROMOS and the DOS WEDGE are incompatible. You can use the DOS WEDGE with PROMOS installed, but prior to a PROMOS 'SAVE', 'OPEN', π or any operation which programs data into an EPROM, the DOS WEDGE must be disabled by typing

```
@Q <CR>.
```

CHANGES FROM PROMOS 1.0 TO PROMOS 1.1

A. Z command done automatically when PROMOS 1.1 enabled.

B. Code change permits programming 27512.

C. Code change to improve timing margin programming MCM68764.

D. Last byte bug in programming 27256 corrected.

All other functions of PROMOS 1.1 are identical to PROMOS 1.0.

PROMENADE C1

TABLE OF CONTROL AND PROGRAM METHOD WORDS

EPROM	CONTROL	PROGRAM METHOD WORDS			
TYPE	WORD	STD	I#1	I#2	I#3
2758	40	3	7	10	14
2516	40	3	7	10	14
2716	40	3	7	10	14
2532	24	3	7	10	14
2732	224	3	7	10	14
2732A	225	3	7	10	14
2564	20	3	7	10	14
2764	5	3	7	10	14
2764A	6	NR	7	10	14
27128	5	3	6	10	14
27128A	6	NR	6	10	14
27256	230	NR	6	10	NR
68764	48	NR	6	11	15
68766	48	NR	6	11	15
2815	57	3	6	11	15
2816	57	3	7	NR	NR
X2804A	57	1	NR	NR	NR
X2816A	57	1	NR	NR	NR
48016	40	2	5	9	13
5133	5	3	7	NR	15
5143	5	3	7	NR	15
5213	57	2	4	9	13
52B13	57	1	NR	NR	NR

NR-- NOT RECOMMENDED

NOTE: EPROMS OF THE SAME TYPE BUT FROM DIFFERENT MANUFACTURERS WILL RESPOND DIFFERENTLY TO INTELLIGENT PROGRAMMING. PROGRAMMING TIMES MAY VARY BY A FACTOR OF FIVE OR MORE.

JASON-RANHEIM COMPANY UPDATES THIS LIST PERIODICALLY. FOR A COPY OF THE LATEST REVISION, SEND A STAMPED, SELF ADDRESSED ENVELOPE TO:

JASON-RANHEIM COMPANY
580 PARROTT ST.
SAN JOSE, CALIFORNIA 95112