48K/128K Spectrum

CODE SLICER 2

Cassette Utility Package

"as versatile as your imagination"

Forait Software

CODE SLICER 2

Utility Package

CONTENTS

INTRODUCTION

The quality and technical expertise of recent Spectrum cassette software has led to programmers protecting their creations with sophisticated loading methods. Little or no consideration has been given to the Spectrum owner wishing to transfer their cassette based program to another media such as Microdrive or disc. Another group of Spectrum owners suffering from this unforgivable crime is the inquisitive games player or programmer whose only pleasure in life is to amend his personal copy of *Daley Thompson's Exploding Shorts* to provide 65536 lives thus reducing him to a mass of quivering "hacker" after only 38 hours of continual use.

Code Slicer 2 was written to assist the above 2 groups of deprived humans to achieve their objectives. The Code Slicer 2 package consists of the following programs :-

SIDE 1

SLICER 2 - This routine is really 2 programs in 1.

1) It is able to read ANY section of a Spectrum cassette into ANY store address by skipping past a given number of bytes. This utility will also read tapes created at any of 7 different speeds.

2) This same program will also read a file and count the number of bytes it contains. This routine will also work at any of 7 different speeds and will operate on headerless files.

SIDE 2

CS2 Header - This program will read a standard Spectrum cassette header and list the program/file details to your screen. It will also let you know if the file is headerless and has the ability to remove the "auto-run" facility from Basic programs.

LISTER - This program will enable you to read Basic programs which have been rendered invisible by the programmer. It will also recalculate all numeric constants in the program in case they have been POKEd to confuse the listing.

IMPORTANT NOTE

Your Code Slicer 2 software package consists of some extremely powerful tools and SHOULD NOT be used to breach copyright. Code Slicer 2 is as versatile as your imagination and is intended to further your knowledge and increase your technical ability when dealing with cassette software.

Forait's Software DO NOT condone piracy

FILE STRUCTURE

The majority of commercial software is written in Machine Code. This is a language your Spectrum recognises most easily. It will read programs written in Basic but will have to translate each instruction to Machine Code before it can execute it. This is both time consuming and expensive on storage.

Most programs you buy will consist of several different files on the same cassette. The first one will normally be a small program written in Basic which automatically starts running when loaded. This in turn will load in all subsequent files and start the program when ready. To stop the Basic program from running once loaded, it should be MERGEd and not LOADed. It will then be possible to LIST it to your screen and read the various LOAD commands. This should provide you with details of the files that follow.

When a file is saved from the Spectrum to a cassette tape it is written out in two separate blocks. Block one is known as the *header* and is only 17 bytes long. This contains such information as the name of the file, it's size, where in store it starts and whether it is Basic or Machine Code etc. If it is a Basic program it will also say which line number, if any, it will autorun from. Block two is separated from the header by a one second gap and contains the actual program as described by the header. This will obviously vary in size depending upon it's contents. The first file on a Spectrum cassette must always be in the above format. This is the structure the LOAD "*name*" and LOAD "*name*" CODE commands understand. These "normal" files can be recognised by their sound and screen colours.

A header loading produces a wide, 5 second red/cyan coloured border. This will be followed by a narrow blue/yellow border for approx. half a second. A 1 second gap now follows before the wide, 2 second red/cyan tone leader of the main block. This is immediately followed by a norrow blue/yellow border for the entirety of the block.

It is suggested that you become familiar with the structure of normal files by continually watching and listening to them load. This will enable you to recognise any non-standard methods used in commercial software to provide protection and/or faster loading.

TRICKS of the TRADE

Experienced programmers have developed ways to protect their programs from the vast number of tape copiers available on the market. As explained earlier this makes life difficult for the Spectrum owner wishing to transfer their cassette software to another media. None of the methods adopted are 100% effective against all copiers or manual methods but they do offer a certain level of protection which would probably be sufficient to deter the majority of people from pursuing their devious piracy. Described below are some of the methods used to achieve this protection. It must be stressed that the information given in this manual is only to be used for the purposes of media conversions, self education, "*KNIKLOAD*" conversions etc. For this reason no commercial program names are mentioned and all examples, although based on fact, are just listed to clarify the text.

100% MACHINE CODE

When a Basic program is in your computer it is stored as a series of numbers from 0-255. A SCREEN$ or Machine Code program is also stored in exactly the same way. This means that the entire contents of your Spectrum's RAM can be saved to cassette by use of the SAVE "*name*" CODE 16384,49152 command. When this is done the "System Variables" are also saved to cassette because they occupy an area of RAM between the above addresses.

When this cassette is loaded back into the Spectrum it will automatically start running as specified by the contents of the saved program and System Variables. This means there is no way to stop the program to transfer it to Microdrive etc. Some commercial programs are copies of the full Spectrum RAM purely to prevent piracy - not because they use all of the available store.

Some programs may not be a full copy of RAM but they will be sufficiently large to make the loading of the program difficult without running it. Unless a program can be loaded at address 23734 or higher it may present problems to the Spectrum owner without a copy of Code Slicer 2. These files can be identified by the information contained in their header block. Instructions explaining the use of the Code Slicer 2 Header Reader will follow in a later chapter.

TRICKS of the TRADE

HEADERLESS FILES

If all Spectrum files were constructed as described in chapter 2 life would be simple. Unfortunately, or fortunately if you enjoy a challenge, some files on commercial tapes are saved without a header block. The implications of this may not be instantly apparent.If a file has a header block you can, with very little affort, read the information and identify it's start address in store and the length of the file. If the header block is absent you will not be abie to know where the file should be loaded or how long it is. A knowledge of Machine Code would help solve the mystery or Code Slicer 2 can be used to supply 50% of the answer. This type of file can be identified by watching it load. The normal 5 second red/cyan coloured border and half second blue/yellow colours will be missing. The file will start loading directly after the 2 second red/cyan tone leader.

HIGH SPEED LOADERS

High speed loading routines are now becomming a common sight in Spectrum software. There are 2 reasons for their popularity. Firstly, a faster loading method obviously reduces the waiting time while the cassette loads and secondly it provides a means of program protection and helps to reduce piracy.

These files can be identified by their sound and border pattern. The loading noise will be of a higher pitch than normal and the blue/yellow border lines will be narrower and closer together. Further complications may be introduced with high speed loaders. To enable this non-standard speed cassette to load, a special Machine Code routine would have been written to perform the task. In doing so the programmer would have had the opportunity to change the border colours or remove them altogether. It is also unlikely that a header block exists. Any one of these complications or a combination of all methods can prove difficult or impossible to bypass. Code Slicer 2 has the ability to read files created at any of 7 different speeds whether a header block exists or not.

FALSE HEADERS

A file may look and sound perfectly normal whilst loading but will either not perform it's job or crash the Spectrum when loaded manually. This could be due to the use of a false header block. With the aid of a Machine Code routine commercial

programs may read the header block and totally ignore the information regarding start address and length of code. The main block of code will then be loaded into an address different from that stated by the header. This form of protection, although not widely used, can be difficult to recognise until it's too late. Code Slicer 2 does provide assistance to identify this evil deception and is described in a later chapter.

BASIC LOADER

As stated earlier in this manual the majority of commercial programs you purchase will contain a small Basic program as the first file. This program, once listed to the screen, will normally indicate the entire contents of the cassette. There are several ways in which programmers can either stop you reading the Basic program or confuse the listing and make it read differently.

1 - In most cases the Basic program will start to run as soon as it is loaded. This is because it has been SAVEd with the LINE command as described in the Spectrum manual. All you need to do is to type MERGE "" instead of LOAD "". This will allow the program to be loaded, but not run.

2 - A common method to prevent the MERGE command from working is for the programmer to create a program line number greater than the allowable maximum. Although the Spectrum's Basic line checker will not accept this, it can the achieved by the use of the POKE command. When this type of program is MERGEd it will either just "hang up" or crash the computer. Your Code Slicer 2 cassette has a routine which will remove the "auto-run" and allow the program to load but not execute.

3 - Another facility available to the programmer is to render the listing invisible. Although unreadable, it will execute as normal. The simple answer for this protection is to use the LLIST command to route the program to a printer. For those people not fortunate enough to own a printer Code Slicer 2 has a machine code routine which will list invisible programs to your screen.

4 - A simple yet effective method to confuse the Spectrum hacker is to change the Basic listing without affecting the actual statements. This is possible because of the way a line of Basic is stored in the Spectrum RAM. Each numeric constant is held in 2 formats. Firstly it is stored in a display format for the purposes of the Basic listing only, and secondly it is stored in it's 5 byte binary form preceded by the character CHR$ 14. It is the first format that can be altered by use of the POKE comand. This will change the numeric value in the program listing but have no effect upon the execution of the program. Such values as the CLEAR, LOAD and POKE addresses are often changed to make life difficult for the uninitiated Spectrum user. The "Invisible Basic Lister" routine supplied with your Code Slicer 2 software will automatically calculate the correct values and list them to the screen.

5 - The Spectrum screen display is split into 2 logical sections. By default these consist of the main 22 line section and a small 2 line display at the bottom of the screen. It is possible to alter the number of lines allocated by a simple POKE to the System Variable DF SZ at address 23659. If this address is POKEd with zero the Spectrum is told that there are NO lines in the bottom section of the display. This proves to be a problem because all standard messages expect to be output to the lower screen and will crash the machine if there are no lines available. These messages include the normal "BREAK" and "OK". If a Basic loader program executes a POKE 23659,0 it therefore becomes impossible to break into it to stop it running. The "Header Reader" supplied with Code Slicer 2 can be used to remove the "Auto-Run" facility and allow you to delete the offending line.

As can be seen from the 5 examples above the programmer is able to create havoc with the simple Basic loader program. Just imagine what could be achieved if all 5 forms of protection were used in the same program in conjunction with high speed machine code loaders and headerless files and/or false headers. The next 2 chapters of this manual describe the use of Code Slicer 2 with it's associated utilities and provides working examples and solutions to the above nightmare.

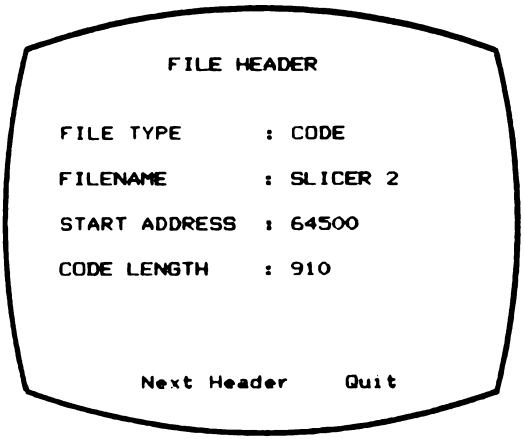
CS2 Header

This is the first program on side 2 of your Code Slicer 2 cassette and is used to read and translate the 17 byte header which precedes all standard Spectrum cassette files.

CS2 Header is written partly in Basic and partly in Machine Code and can be loaded simply by typing :-

LOAD "CS2 Header"

Once loaded, you will be prompted to "Start Tape Now and press any key". When you have pressed a key the message will change to "Loading Header Please Wait". As soon as the header is read by CS2 Header the following display will be shown (for a Machine Code file) :-



Once the above details have been displayed you can stop your cassette recorder. If the file you require is not the first on the tape you can press "N" to search for the next header. If the screen is displaying the correct information, pressing "Q" will stop the program. An explaination of the screen now follows :-

CS2 Header

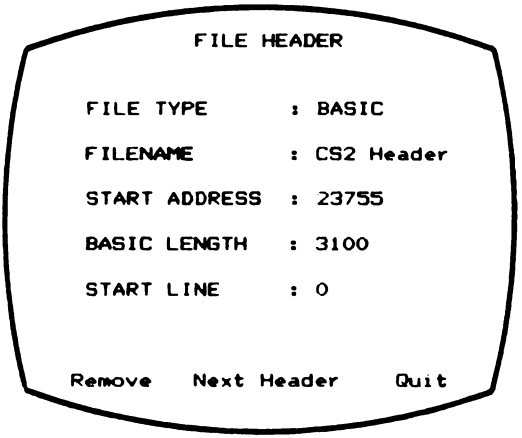
FILE TYPE - This displays the type of file saved to the cassette. It could be either - BASIC, CODE, NUMERIC ARRAY or CHARACTER ARRAY.

FILENAME - This is the name that was given to the file when it was saved. It has a maximum of 10 characters and can include Spectrum Keywords and Control Characters. So as not to confuse the display, CS 2 header will convert any Keywords or Control Characters to a red square when it outputs the name.

START ADDRESS - This is the store address in the Spectrum's RAM where the file was saved from, not necessarily where it will be loaded to.

CODE LENGTH - This is the size of the file.

An example of a Basic header now follows :-



FILE TYPE and **FILENAME** are exactly the same as described under the Machine Code header. The remaining sections are as follows :-

START ADDRESS - For a Basic program this address is the store address the program would load back into in your Spectrum. This may vary depending on what peripherals you have attached. The address of the Basic area in RAM will alter if an Interface 1 is connected and active.

CS2 Header

BASIC LENGTH - This is the size of the Basic program including any variables which were saved with it.

START LINE - This is the line Number which the Basic program will "Auto-Run" from if it was saved with the **LINE** command. As described in "Tricks of the Trade" it is possible to save a program with "Auto-Run" which is impossible to **MERGE**. CS2 Header has the ability to remove the "Auto-Run" facility allowing you to load the program as normal without it executing.

If the header shows the file to be a Basic program with "Auto-Run" it will display a **Remove** option at the bottom of the screen. Pressing the "R" key will activate the "Tape Header Creator" program which will ask you to :-

Place blank cassette into
recorder and press any key
to save new Header

As soon as a key is pressed a new 17 byte header will be saved out to your blank cassette retaining all the original information with the exception of the "Auto-Run" facility. The header details will now be redisplayed showing "**No Auto Start**"

To make use of this new header it only requires you to type **LOAD ""** and load the new header. When loaded, start the original tape **AFTER** the header and it will load the Basic program but not run it. It will then be possible to LIST, LLIST or amend the program which would have previously been impossible.

NOTE - If you load a file into CS2 Header which is headerless it will produce a flashing "**HEADERLESS**" message in the FILE TYPE section.

LISTER

This is the second program on side 2 of your Code Slicer 2 cassette and is used to make invisible Basic programs visible. The Machine Code routine is loaded into a high store address and the Basic program is automatically loaded at address 32768. To load and run the program type :-

```
CLEAR 32767
LOAD "LISTER" CODE
RANDOMIZE USR 64000
```

You will then be presented with a blue screen and a flashing border. At this point you must load in the Basic program including it's header. Once the header has been accepted it will print the name of the file in the centre of the screen :-

Loading Program - THINGY

If the LISTER program encounters any Keywords or Control Characters in the Filename it will print the Spectrum Copyright symbol in the corresponding position.

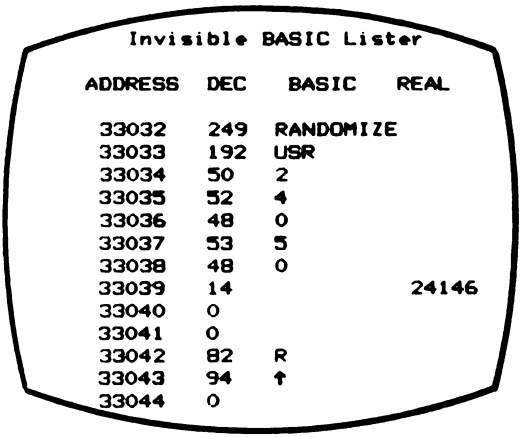
2 other possible messages could be output at this time :-

File is not BASIC
or
File is Headerless

Both these messages are obvious in meaning and assume you have loaded the wrong file.

Assuming you received the first of the 3 messages the Basic program will automatically be loaded into store and the screen will start to display the invisible program as follows :-

LISTER



As you can see from the above example the program is not displayed across your screen as normal, but scrolls up the screen. The four columns are now described :-

ADDRESS - This is the current address of the Basic program byte as loaded by LISTER. It is not the address the Basic program would normally occupy whilst running.

DEC - This is the decimal value held at the stated store address.

BASIC - This is the equivalent Spectrum Keyword or Token translated from the decimal value above. (codes 0-31 are not printed)

REAL - Chapter 3 of this manual explained how a Basic program could be POKEd to produce a listing which was different to the actual program. This column of the LISTER display automatically calculates every 5 byte positive, numeric constant and prints it opposite it's preceding CHR$14. By checking this number against those printed vertically you will be able to read the correct value within the program. Each time a value is calculated by LISTER a BEEP is output to warn you of the possible deception.

The program listing will continually scroll up the screen until you press the SPACE key. When this is done the screen will be frozen and the message **Continue or Quit** will be issued. press "C" to continue or "Q" to end the program.

SLICER 2

This program is on side 1 of your Code Slicer 2 cassette and has 3 main functions as described below :-

1) It can count the number of bytes in a cassette file even if it has no header.

2) It can read cassettes which have been saved at any of 7 different speeds.

3) It can read any section of a cassette file into any store address by skipping over a pre-defined number of bytes.

SLICER 2 - Byte Count

SLICER 2 resides at address 64500 and must therefore be loaded by typing :-

```
CLEAR 64499
LOAD "SLICER 2" CODE
```

It now becomes a simple matter of typing **RANDOMIZE USR 64500** to execute the program. When this is entered the following message will prompt you to supply the speed of the cassette file :-

Enter Speed (1,2,3,4,5,6,7)

The numbers 1 to 7 correspond to the following loading speeds :-

OPTION NUMBER	BAUD RATE	SPEED FACTOR	PATCH ADDRESS	PATCH VALUE
1	1500	1	65276	22
2	2000	1.3	65281	17
3	2500	1.6	65286	14
4	3000	2	65291	11
5	3500	2.3	65296	9
6	4000	2.6	65301	8
7	4500	3	65306	6

The last 2 columns on the chart are listed to enable you to "fine tune" the loading speed of each cassette. This may only be necessary on the faster speeds and will give you the flexibility to cope with a wide variety of tapes and cassette recorders. This facility will be described in greater detail in a later chapter.

Once you have typed a number from 1 to 7 and pressed ENTER you will hear a BEEP as acceptance of your input. The "Enter Speed" message will now change to :-

"Y" = Header "N" = Headerless

If the cassette you are reading has a header, reply "Y", if it is headerless, reply "N". This message will now change to a flashing "Counting Bytes" message and is the time to start loading your cassette.

If the tape has a header it will appear to be read in the normal manner but will be lost. After this the file will be read by the program and produce a black & white border instead of the familiar blue/yellow colours as described in chapter 2. A headerless file will start loading with the black & white border. On completion the size of the file will be displayed in the centre of the screen :-

File Length = XXXXX

This process can be repeated as many times as necessary without reloading the SLICER 2 software.

Suggestions

As stated in chapter 3 a file may be preceded by a "false header". It may be possible to identify this by counting the size of the file as previously described and checking this against the size from the CS2 Header program on side 2 of the Code Slicer 2 cassette. This will show whether the size of the file actually matches the header information. Unfortunately there is no way that Code Slicer 2 or any utilitiy can automatically tell you the start address of a headerless file. If, however, during the normal loading of the software it starts from the SCREEN$ you will know that it's start address is 16384.

SLICER 2

SLICER 2 - Code Slicer

This is the main routine on the cassette and provides the powerful facility of splitting a large file into smaller sections for security back ups, Microdrive transfers, disc transfers, *KNIKLOAD* conversions, general hacking etc.

This program is called from the same code as "Byte Count". Follow the loading instructions as previously described but use a suitably low address in the CLEAR statement to allow for your file to be loaded as well as the Code Slicer 2 software.

The syntax for calling this routine may at first seem a little unusual but will soon become a familiar addition to your commonly used commands :-

RANDOMIZE *SKIP* and *START=LENGTH*+USR 64567

Where *SKIP* = No. of bytes to be skipped from start of file

START = Start address that file will be loaded

LENGTH = No. of bytes that will be loaded

NOTE - RANDOMIZE AND = USR are all Spectrum keywords

As an example you may choose to load the first 30000 bytes of a file into store at address 26000. After typing CLEAR 25999 you would enter the line :-

RANDOMIZE 0 AND 26000=30000+USR 64567

1) The **CLEAR** command would ensure that store is reserved above RAM TOP to accommodate your file.

2) **RANDOMIZE** 0 skips zero bytes - ie it starts to load your file from the beginning.

3) **AND** 26000 tells the machine code routine to load your file starting at address 26000.

4) =30000 instructs the program to load 30000 bytes into store.

5) +**USR** 64567 is necessary to execute the program.

Once the above line has been entered it will be vetted for accuracy. The following Spectrum error messages may be output if an incorrect entry has been made :-

MESSAGE	REASON
Q Parameter Error	The Syntax of the line was incorrect. Check that all 3 values were entered including the correct keyword separators.
Integer out of range	This message is output if any of the values entered fall outside of the allowed range. 1 - The ***SKIP*** value must be an integer in the range 0 to 32768 2 - The ***LENGTH*** value must be an integer in the range 1 to 64499 3 - The total of ***START*** & ***LENGTH*** must not exceed 64499

Assuming your command line sneaks past the above errors you will be asked to enter the speed of the file exactly as described in the "Byte Count" instructions. If you are uncertain as to the speed you will have to estimate it from the border pattern and pitch of loading (see File Structure). If the file has a SCREEN$ you can easily test it by loading it at address 16384 for 6912 bytes and view it's success.

If the screen image looks corrupt with the wrong attributes (colours) you will need to reload it using a different speed. If after experimenting with different speeds the picture still appears slightly wrong it can be "fine tuned" by the use of the POKE command. The speed table on page 12 shows the "Patch Address" associated with each speed. By poking a number *slightly* above or below the "Patch Value" listed you will be able to adjust the loading speed between the 7 pre-set values.

Once the speed has been entered you will be prompted to reply to the following message :-

"Y" = Header "N" = Headerless

This should be answered in exactly the same way as described for "Byte Count". A message in the bottom left-hand corner of the screen will now ask you to "**Start Tape**".

In the current example the file would be read into store at address 26000 for 30000 bytes. The routine will stop at this point although the file will still contain more data. It is now that you could use any of the standard cassette, Microdrive or disc commands to save this section of the file to another media.

Up until now the Code Slicer routine does not appear revolutionary. With a bit of guess work (and a Tape Loading Error message) you could achieve the same results by loading the whole file and switching the cassette recorder off before the end. The second part of the example, however, shows the powerful facilities available with this software.

Let us now load the second part of the fictitious cassette. We will use the same starting address but will need to skip over the first 30000 bytes which have already been loaded and processed. Assuming this file is the maximum size of 49152 we need to load 19152 bytes this time. The command line required is :-

```
RANDOMIZE 30000 AND 26000=19152+USR 64567
```

The familiar speed and header messages will still be output and your cassette should be loaded from the start.

If your file contained a header it will be read into store and promptly lost. The file will now enter the main Slicer 2 routine and will look very different. The border will remain white whilst Slicer 2 counts each byte on the tape until it reaches 30001. Until now all this section of the cassette has gone down the Slicer 2 plug hole. Byte 30001 for 19152 bytes will now be read in at address 26000 with the return of a black/white border. Once complete, this section of the file can then be copied out to whatever media you choose, for whatever reason you require.

With 2 simple passes through the file you have been able to read in and copy out a large file which would have been impossible previously.

Suggestions

1) Use the above method to make security tape copies of your valuable cassette software. All you will need is a simple Basic loader program first which will look something like this :-

```
10 CLEAR XXXXX
20 LOAD "BLOCK 2" CODE
30 LOAD "BLOCK 1" CODE
40 RANDOMIZE USR XXXXX
```

it is necessary to load the first block of code last because it will probably contain the System Variables. If it was loaded first it may try to start the program automatically and crash because the second block was missing.

2) Use the above theory to make Microdrive or disc transfers. It will obviously be necessary to substitute the load syntax to the appropriate format in the Basic loader. Further details will be given in a later chapter for *problem* conversions.

3) Programs could be re-saved without a SCREEN$. This would obviously save loading time if you consider it was a problem. For blocks of code starting with a SCREEN$ you would only need to type the following line to remove it :-

```
RANDOMIZE 6912 AND 23296=XXXXX+USR 64567
```

The file could then be saved out to a new cassette and become your working copy.

ADVANCED CODE

This section of the manual is more suitable for people with a knowledge of Z80 Machine Code but is not intended to frighten away the non-programmers amongst you. No matter how far you delve into the depths of commercial Spectrum programs the detail in this chapter will be required sooner or later. A piece of software known as a disassembler would also be useful to convert the decimal values held in Machine Code programs into their respective Z80 assembler mnemonics.

HEADERLESS FILES

Headerless files should not be thought of as a threat to your activities, merely a challenge. Before any headerless file can be loaded into store the Spectrum must be told where it will be loaded and how large it is.

There are areas within the Z80 microprocessor inside your Spectrum called registers. These single registers can each hold a number up to 255 or can be paired together to hold 65536. It is selected registers that hold the start address and length of code for headerless files. A couple of other Machine Code instructions will also be required and a simple CALL to the Spectrums's ROM will perform the load. The example below would load a headerless SCREEN$ into store :-

Assembler Code	Decimal Values	Comments
LD IX,16384	221 33 0 64	Load the IX register pair with the start address
LD DE,6912	17 0 27	Load the DE register pair with the length of code
SCF	55	Set Carry Flag - this is necessary to perform a load
LD A,255	62 255	Load the A register with 255 - code is not Basic
CALL 1366	205 86 5	Call the loading routine in the Spectrum's ROM
RET	201	Return to Basic

ADVANCED CODE

Assembler Code - These are the Z80 assembler mnemonics which would be output by a disassembler or input to an assembler.

Decimal Values - These are the equivalent decimal numbers of the Z80 assembler code. By lowering RAM TOP by use of the CLEAR 59999 command and poking the 14 decimal numbers into store from address 60000 onwards it would be possible to execute the above Machine Code routine by typing :-

RANDOMIZE USR 60000

As you can see each assembler mnemonic is made up of 1 or more decimal values. It would be possible to PEEK the store addresses of a headerless loader routine looking for the above code but this would be time consuming and prone to error. The use of a disassembler is widely recommended.

Where a high speed headerless block of code is loaded the above routine would look slightly different. Instead of calling the Spectrum's ROM routine at address 1366 it would use it's own special load routine. Remember, SLICER 2 can load any of 7 different speed files, all you must do is supply it with the start address and length of code - happy hunting !!!

MICRODRIVE TRANSFERS

Page 17 mentions *problem* conversions. These may be experienced when a Microdrive Channel is opened and makes it impossible to load a block of code low enough in store. This will occur if the CLEAR command is 24610 or lower. Your Code Slicer 2 software in conjunction with the powerful Z80 Machine Code instruction LDIR will allow you to accomplish this transfer. The LDIR instruction is known as a "Repeating Block Load with increment". In practise it will move a predefine block of store from one address to another in 1 instruction, extremely quickly.

If the Machine Code program you want to transfer from tape to Microdrive is small enough to be loaded into a higher address first, the LDIR instruction can be used to relocate it to it's correct address before running. If the code is too large you will need to use SLICER 2 to split it into 2 sections first, then use LDIR to move the lower one into the correct store address. Examples now follow .....

Example 1

Our example cassette based program contains the following sections of code :-

Basic Loader
SCREEN$
CODE 24500,32768

There would be no problem transfering the first 2 sections of code to Microdrive, but the last one would overwrite the Channel Information and/or the Microdrive Maps when it started to reload. This is how we cope with the problem .....

1) Load the main block of code from cassette into a higher store address such as 30000. This could be achieved with a simple LOAD "" CODE command if the file was saved at normal speed with a header, or you may need to use SLICER 2 if the file was headerless or saved at a higher speed.

2) Write the following Machine Code routine to perform the relocation of the code :-

Assembler Code	Decimal Values	Comments
LD HL,30000	33 48 117	Load the HL resister pair with the start address
LD DE,24500	17 180 95	Load the DE register pair with the destination address
LD BC,32768	01 00 128	Load the BC resister pair with the length of code to be moved
LDIR	237 176	Perform the actual move
JP 24500	195 180 95	Jump to the start of the Machine Code program equivalent to Basic RANDOMIZE USR XXXXX

The above 14 bytes of Machine Code should either be assembled or POKEd into store directly in front of the main block of code and the whole section saved to Microdrive.

IE - If the main 32768 byte block was loaded at address 30000 in store you should enter the above routine at 29986. Then save the whole section to Microdrive by typing :-

```
SAVE *"M";1;"sample" CODE 29986,32782
```

The Basic loader program would have to be amended to execute the relocator code and not the program direct. This is achieved by substituting the address 24500 for 29986 in the RANDOMIZE USR statement. Examples of the cassette and Microdrive Basic loader programs now follow to clarify the above explaination :-

```
                10 CLEAR XXXXX
                20 LOAD "pic" CODE 16384,6912
CASSETTE        30 LOAD "sample" CODE 24500,32768
                40 RANDOMIZE USR 24500
```

```
                10 CLEAR XXXXX
                20 LOAD *"M";1;"pic" CODE 16384,6912
MICRODRIVE      30 LOAD *"M";1;"sample" CODE 29986,32782
                40 RANDOMIZE USR 29986
```

Once the SCREEN$ and the main code are loaded line 40 would be exectuted. This would perform the block move and place the code where it should have been in the first place. In doing so it will overwrite the Channel Information and/or the Microdrive Maps. This will have no effect at this stage because the Microdrive is no longer required now the program has started.

The above example obviously assumes the program being transfered does not fill your Spectrum's RAM entirely. If it did you would have to use the next example to perform the transfer.

Example 2

In our second example we will assume the cassette based program is an entire copy of the Spectrum's RAM - ie. it is one section of code :-

```
CODE 16384,49152
```

To effect a Microdrive transfer the following steps would be necessary :-

1) Use SLICER 2 to remove the first 6912 bytes (the SCREEN$) and save it to a temporary tape.

2) Use SLICER 2 to remove the next 4000 bytes after the SCREEN$ and save it to a temporary tape.

3) Use SLICER 2 to skip past 10912 bytes (6912+4000) and save the remaining section of code to a temporary tape.

You now have the whole program split into 3 sections. Sections 1 & 3 will load back from Microdrive with no problems because they occupy the screen display area and a high(ish) area in store. Section 2, however, will need to be loaded at addresses 23296-27296 which would overwrite the System Variables, Channel Information and Microdrive Maps. The only safe place for this code is to temporarily put it in the screen display at address 17000(ish). A small relocator routine, as previously described for example 1, could then be used to push it into the correct area of store directly before running.

The only problem with the above solution is that the original SCREEN$ will become corrupt when the 4000 byte (+14 byte relocator) is loaded into the screen area. This, however, is the ONLY option open to you if you require large Spectrum programs on Microdrive. Using the above addresses the example relocator code and Basic loaders should like this :-

Relocator

```
LD HL,17000
LD DE,23296
LD BC,4000
LDIR
RET
```

Basic Loader

```
10 CLEAR XXXXX
20 LOAD *"M";1;"pic" SCREEN$
30 LOAD *"M";1;"sect3" CODE 27296,38240
40 LOAD *"M";1;"sect2" CODE 16986,4014
```

The above examples will enable you to convert most, if not all your cassette based Spectrum software to Microdrive. The theory will also prove successful for disc transfers. Before leaving this subject - one word of warning. The Stack held within your Spectrum's RAM does not like getting overwritten. It's position in store is dependent upon the address used in the CLEAR statement. When loading sections of code into different store addresses make certain the CLEAR address is below the lowest address you are loading to. In some cases you may be lucky - in most you will regret not taking the advise.

SPECTRUM 128K DETAILS

The Spectrum 128K still runs the Z80 CPU chip which is known as an 8 bit processor. Due to limitations within their architecture all 8 bit processors are restricted to address a maximum of 64K of memory. Considering the Spectrum uses a 16K ROM this only leaves 48K of RAM available at one time. The 128K model uses a system known as "bank switching" to swop "pages" of 16K in and out of the *memory map* under program control.

The Spectrum 128K consists of 2 pages of 16K ROM and 8 pages of 16K RAM. Only 1 of the 2 ROMS can be active at addresses 0-16383 at any one time. Any of the RAM pages 0-7 can be mapped to addresses 49152-65536. In addition, RAM page 2 is mapped to addresses 32768-49151 and RAM page 5 is held at 16384-32767. It is therefore possible to have 2 different pages of store mapped into different address spaces ??

Unfortunately it is not possible to perform this bank switching from Basic programming - it must be achieved by the use of Machine Code. Sinclair have designated the I/O address FD to perform the bank switching :-

ADDRESS	DATA BIT	COMMENT
7FFDh	D2-D0	RAM Page selection
	D3	SCREEN$ selection
	D4	ROM section (0=128K ROM 1=48K ROM)
	D5	48K mode

ADVANCED CODE

If you intend to delve into 128K programs you must first know what to look for in a cassette loader routine. This will be a short section of code which loads before any bank switching can be performed. The following piece of assembly code would be repeated somewhere in the loader program for each 16K page required, with only the second instruction altered to specify a different page number.

Assembly Code	Comments
LD BC,#7FFD	Load BC register pair with I/O address
LD A,#13	Load A register with data to perform bank switching
OUT (C),A	Perform the switch
LD IX,#C000	Load IX register pair with start address
LD DE,#4000	Load DE register pair with length of code
LD A,#FF	Load the A register with 255 - code is not Basic
SCF	Set Carry Flag - this is necessary to perform a load
CALL #0556	Call the loading routine in the Spectrum's ROM
RET	Return to Basic

Unlike the previous assembly code examples, this one was listed with hexadecimal addresses. Most disassemblers on the market will produce listings as above and you should become familiar with the format after a little exposure !!!

Although this introduction to the Spectrum 128K has been brief it should provide you with the necessary knowledge to tackle any of the present cassette loading systems available.

COPYRIGHT

Foraits Software

32, Gladstone Road
Hockley
Essex
SS5 4BT
ENGLAND

Tel : (0702) 201368