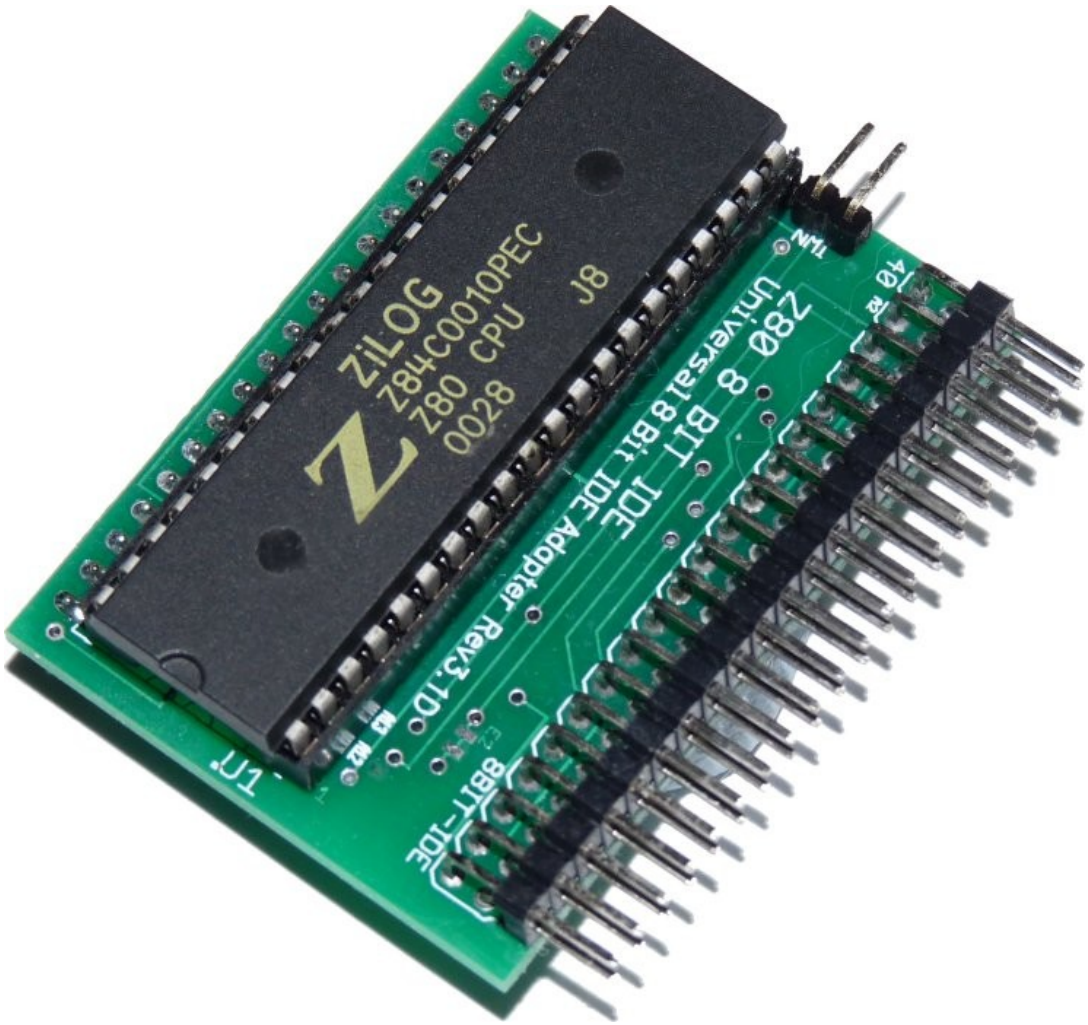
Z80 Universal 8bit IDE Adaptor Revision 3.1



The universal 8bit IDE mass-storage adaptor
For your Sinclair Spectrum home computer

Preface

Thank you for choosing the Spectrum Z80 Universal 8bit IDE adaptor, I hope that you enjoy this product as it has much to offer the Spectrum user, from loading / saving games and applications from mass storage to new developments in streaming video from an IDE device.

This adaptor would not have been possible without the help, knowledge and generosity of Gary Lancaster, Pera Putnik, Miguel Angel Rodriguez Jodar and my fellow forum users of www.worldofspectrum.org/forums - a really friendly community and I hope to see you there.

This is a simple manual to configure and install the Z80 Universal 8bit IDE Adaptor. While this manual is primarily focusing on the Spectrum 128k +2AB (Black) and the Spectrum 128k +3, if you have another model of Spectrum that is not covered in this manual, please see www.amibay.com or www.worldofspectrum.org/forums for more and in depth information.

Thank you for interest in this adapter, I look forward to sharing this project with you.

My best wishes

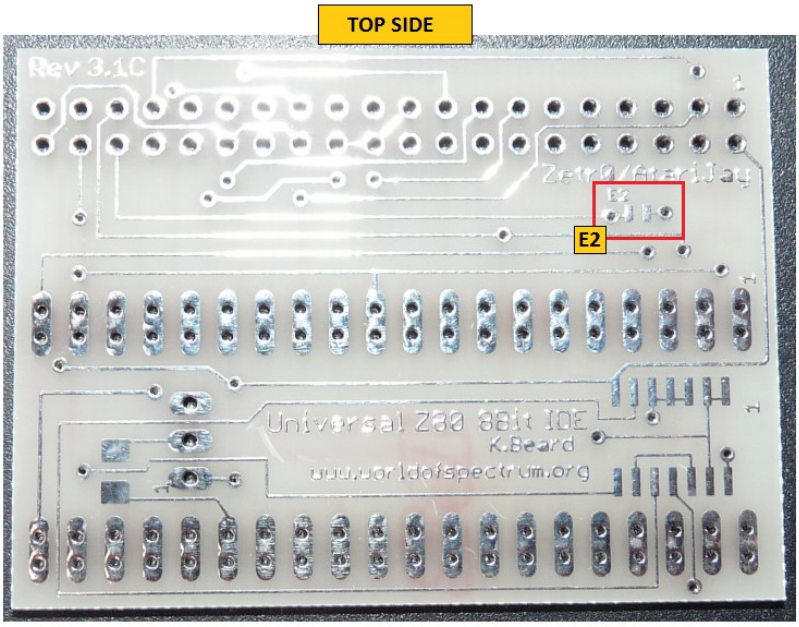
Keith

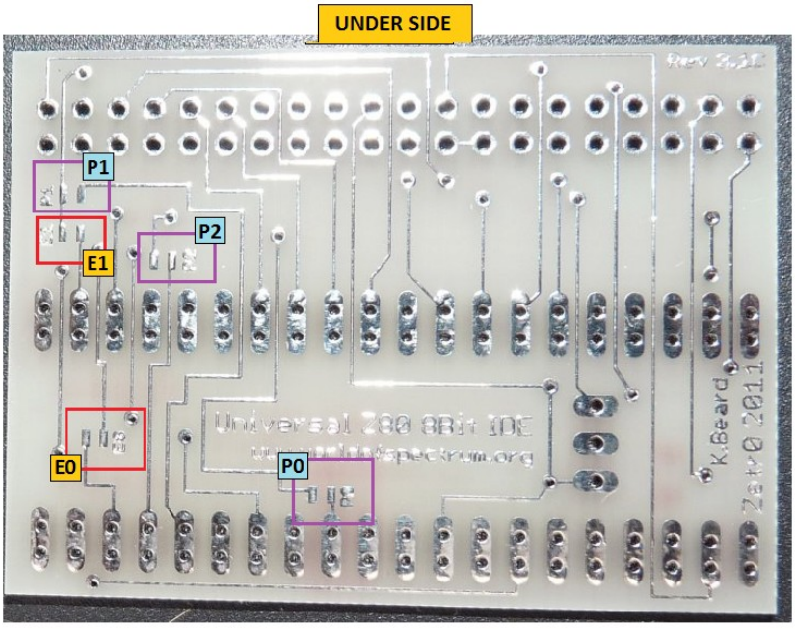
Table of Contents

Z80 Universal 8bit IDE Adaptor Configuration

Configuring the Z80 Universal IDE Adaptor 3.1C and 3.1D

The Z80 UIDE Adaptor Revision 3.1 (C/D) can be configured for either use with the entire range of Sinclair (and later Amstrad) Spectrum home micro computers. The Adaptor has two modes of operation depending on what Spectrum model you have.





CONFIGURATION OF THE REV 3.1C Z80 IDE INTERFACE

Since there are few methods in attributing your Sinclair Spectrum with an IDE interface, this adapter has been designed to take advantage of the two most common types avaialble today.

These types depend on the modle and ROM selection of your intended target system. To select which varient you want the PCB has several solder jumpers which can be bridged.

While your adapter will come to you pre-configured as of you request, this can be changed with a little effort for a different setting if you so wish.

This also opens up the possiblity of other Z80 based computer / console systems at a latter time.

+3E (+2E) ROM Configuration

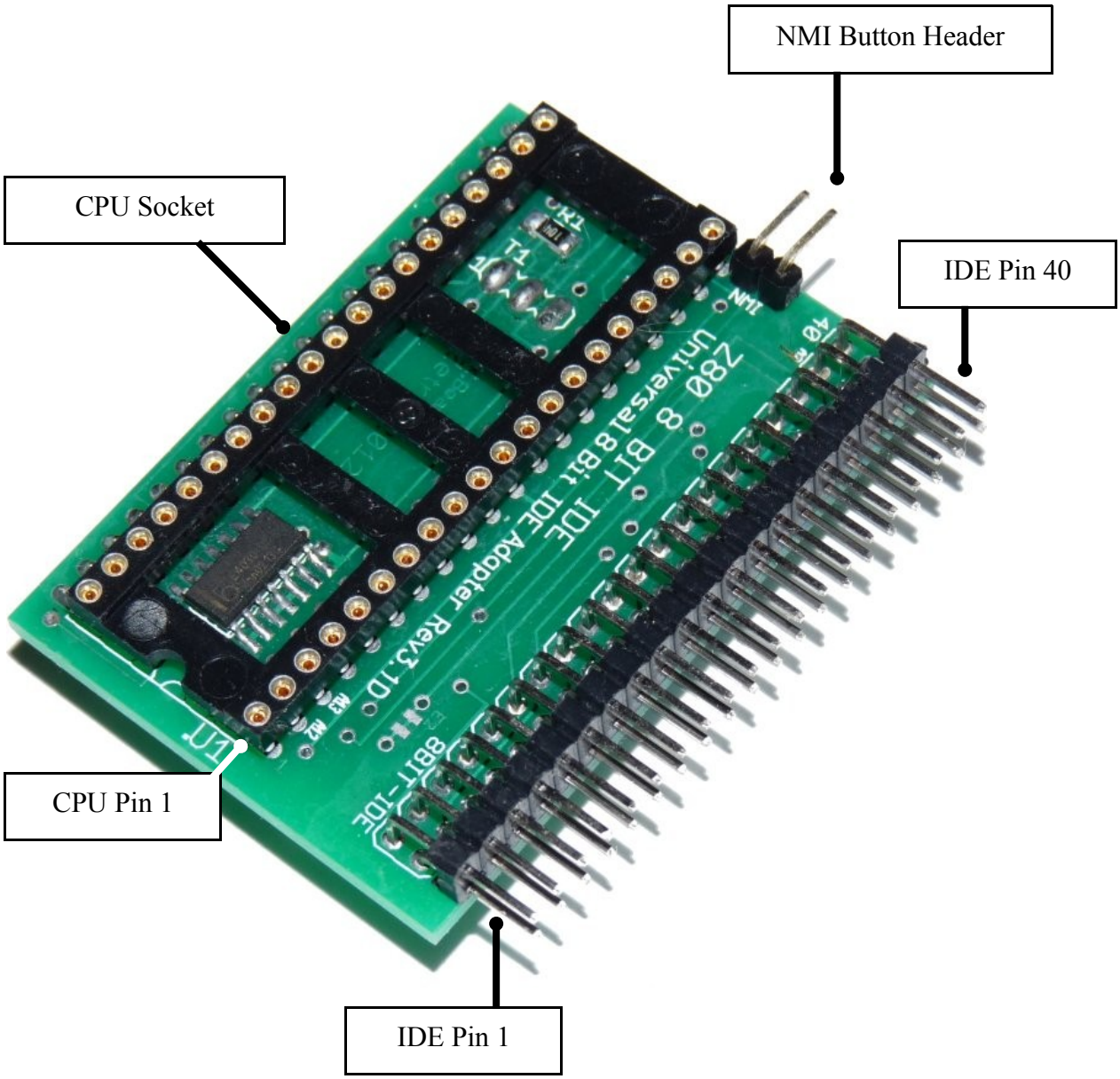
Simply solder bridge E0 and E1 on the under side of the Adapter and then complete this by solder bridging E2 on the top side of the adapter.

PPROM (Pera Putnik's Varient)

Simply solder bridge P0 P1 and P2 from the underside of the Adapter

Revision 3.1D of the Z80 Universal Adaptor has exactly the same solder jumper configuration however are all connected so you will have to cut the connections you do not want. The revision 3.1D also supports the use of a push to connect button to activate the NMI feature of the PPROM.

Z80 Universal 8bit IDE Adaptor Overview Rev 3.1D



***NB**

The Z80 Universal 8bit IDE Adaptor is not static sensitive, however it is quite likely that the replacement ROM(s) will be. Please take appropriate anti-static precautions before installing the device and ROM(s).

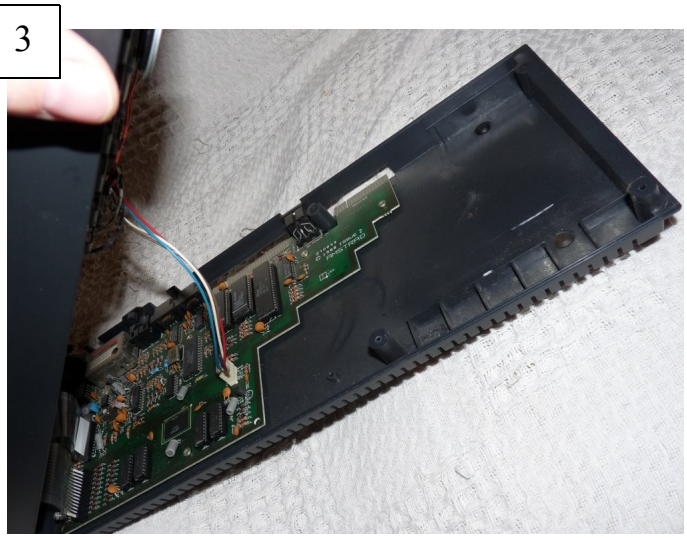
1

Underside of a Sinclair Spectrum 128k +2A/B (Black)

2

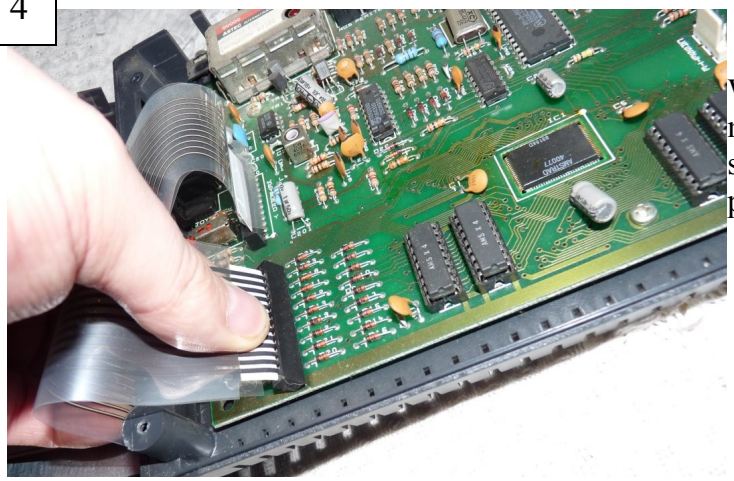
Be cautious when undoing screws as sudden torque can damage the plastic housing.

3



When removing the upper chassis, please remember to disconnect the TAPE mechanism power and communication lead from the header.

4

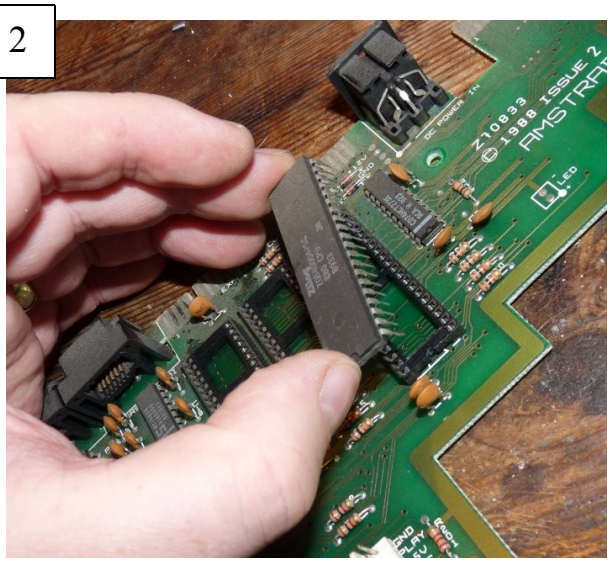


When removing the keyboard membrane tails from the connector, you should only need to use finger pinch pressure.

1

Sinclair Spectrum 128k +2A/B (Black) Motherboard

2



Removing both ROM chips and Z80 CPU

Unless you are an experienced technician I would humbly suggest using a "Chip Puller" IC removal tool to removed both ROM and Z80 CPU Chips.

The can be sourced at any local Maplin or electronics store.

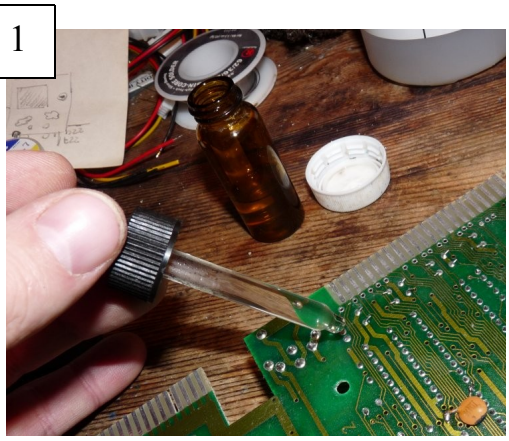
3

Both ROM chips and Z80 CPU Removed

Now is a great opportunity to clean the card-edge expansion sockets as well as the CPU and connectors. A good cloth and IPA (isopropyl alcohol) solution is recommended to remove dirt and grime build up.

Sinclair Spectrum 128k +2A/B (Black)

1

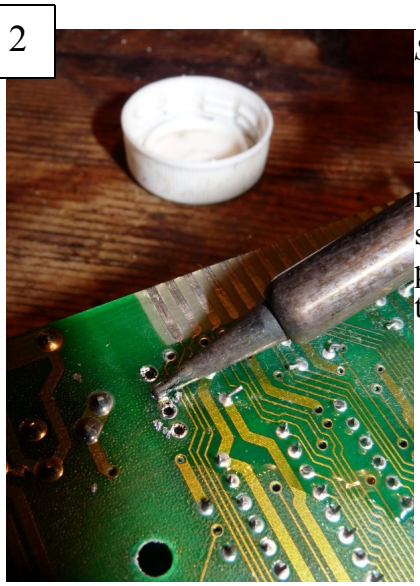


Using No Clean – Non Corrosive Flux

This allows for better heating of the aged solder to help with melting it and removing it from the though hole.

On the underneath of the motherboard next to the power connecter is a small row of 4 holes, of which the middle two are connected.

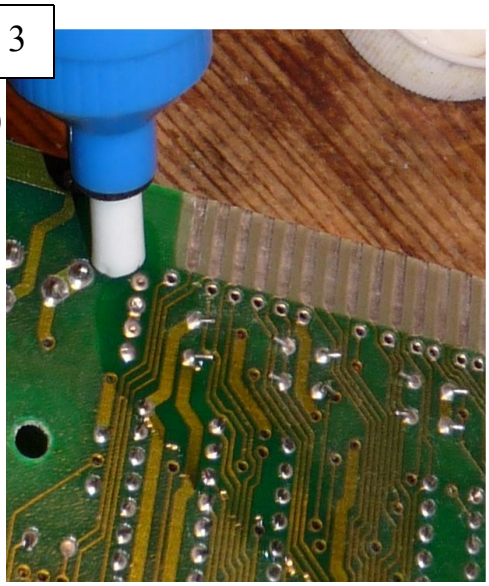
2



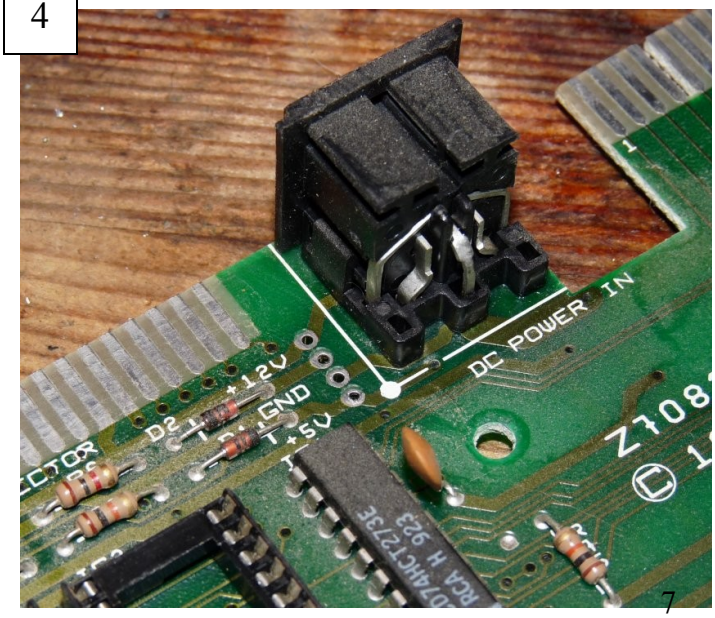
Solder Removal

Using a soldering iron (set to 280 – 310c) heat the solder until it melts and then using a solder sucker (spring loaded vacuum pump) suck up the solder from the holes.

3



4



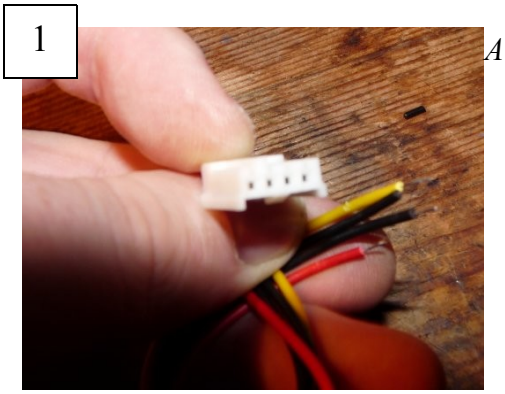
Power header revealed

After removing the solder you will notice that that there is a provision for a +12v and +5v power rail.

Here we will add our 4 pin Molex to provide power to our device.

Sinclair Spectrum 128k +2A/B (Black)

1



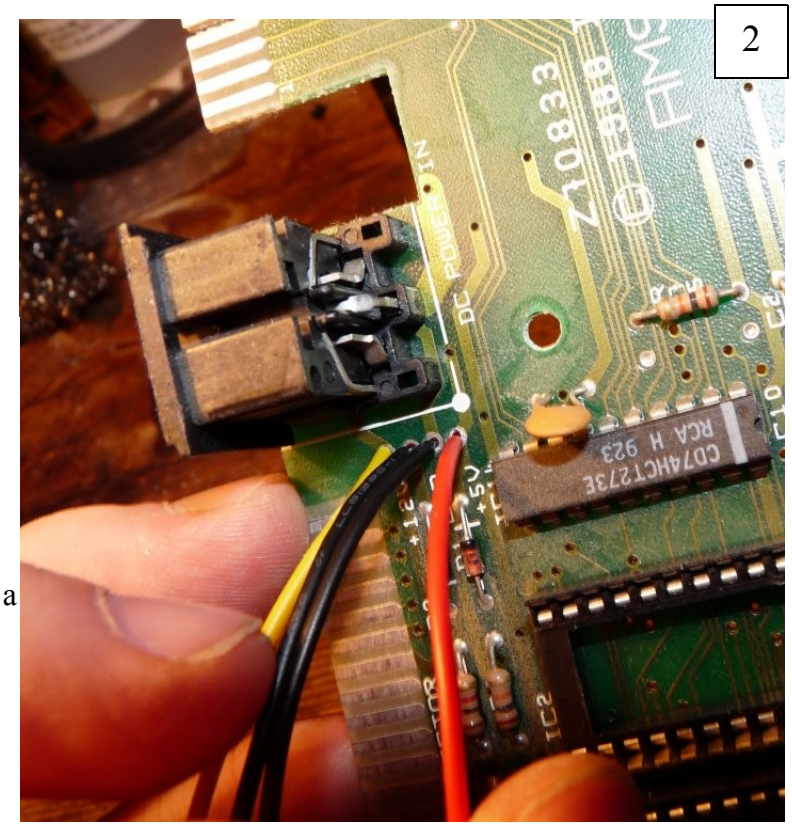
A floppy drive Molex connector with solder 'tinned' ends

Installing the Power Connector

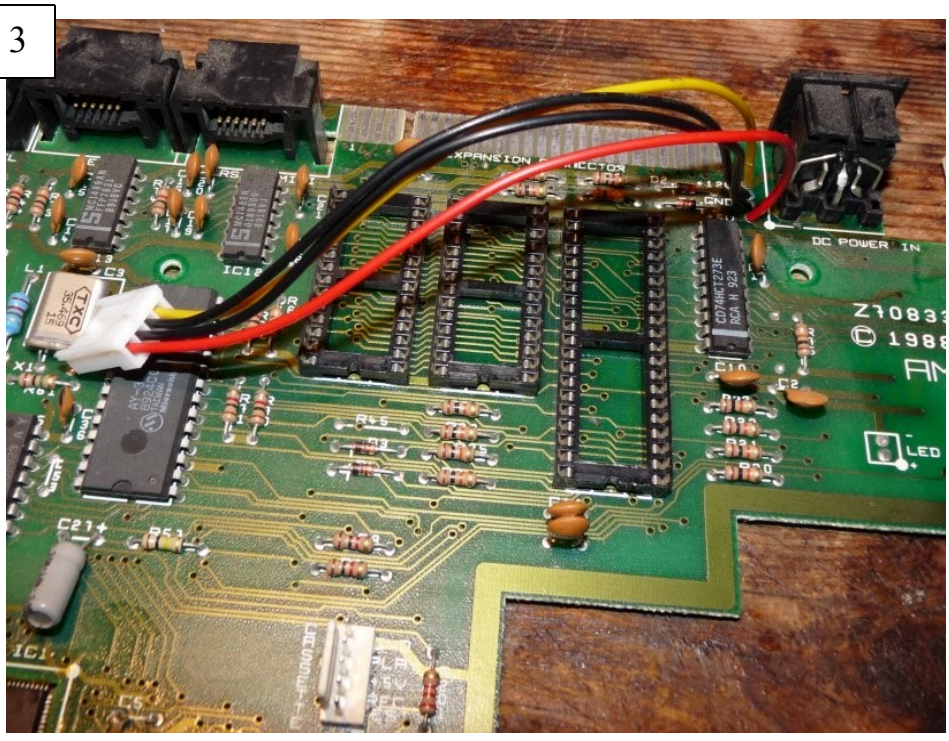
The yellow wire is installed into the +12 hole and the red wire is installed into the +5v hole. The two black wires are installed into the two central holes.

It is generally easier to solder one wire at a time, however like before, use flux if the solder is difficult to melt.

2



3



Power lead Installed

Here is the finished work, a shiny new power lead for use with your device.

Please note that the Spectrum +2AB (Black) Power Supply cannot provide more than 200ma (milliamps) on the +12v. If you attach a device that uses more than that it will damage the Power Supply. I suggest only to use +5v devices.

Sinclair Spectrum 128k +2A/B (Black)

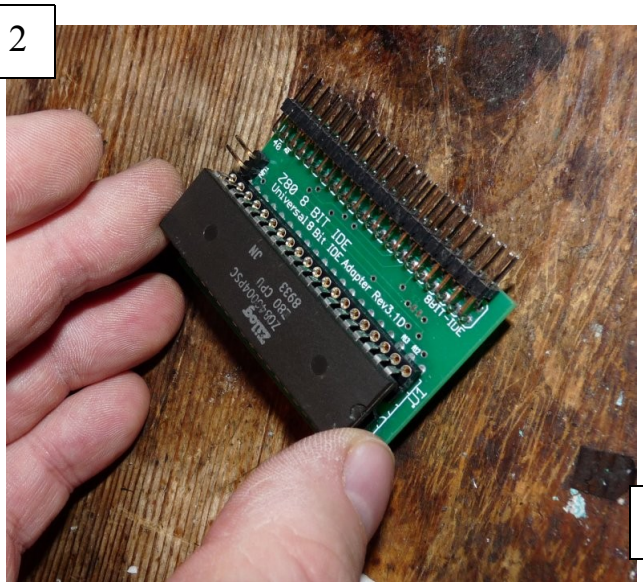
1

Z80 Universal 8bit IDE (Full Kit)

This kit includes everything you need to add mass storage to your spectrum

1x Z80 IDE Adaptor
1x +3e ROM set [*a]
1x 15cm IDE Custom Cable
1x CF to IDE Card
1x Compact Flash Card
1x 4pin Power Molex

2



Installing the Z80 CPU into the IDE Adaptor

The adaptor is marked for Pin 1 of the CPU, (in the show image it is in the bottom right corner).

Correct orientation is paramount or you will likely damage your Spectrum and the Z80 CPU.

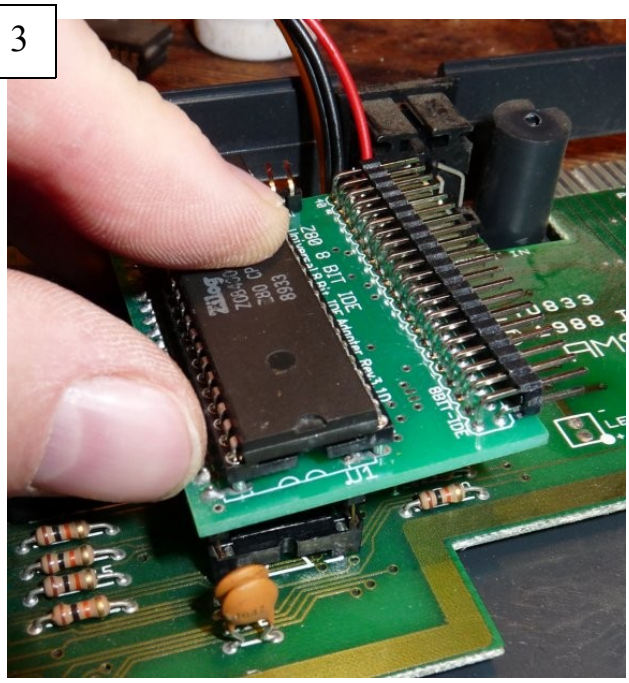
Installing the Adaptor on the motherboard

The adaptor installs into the CPU socket of the Spectrum, please not the orientation is the same as the CPU – In the adjacent image, Pin 1 of the CPU is bottom right.

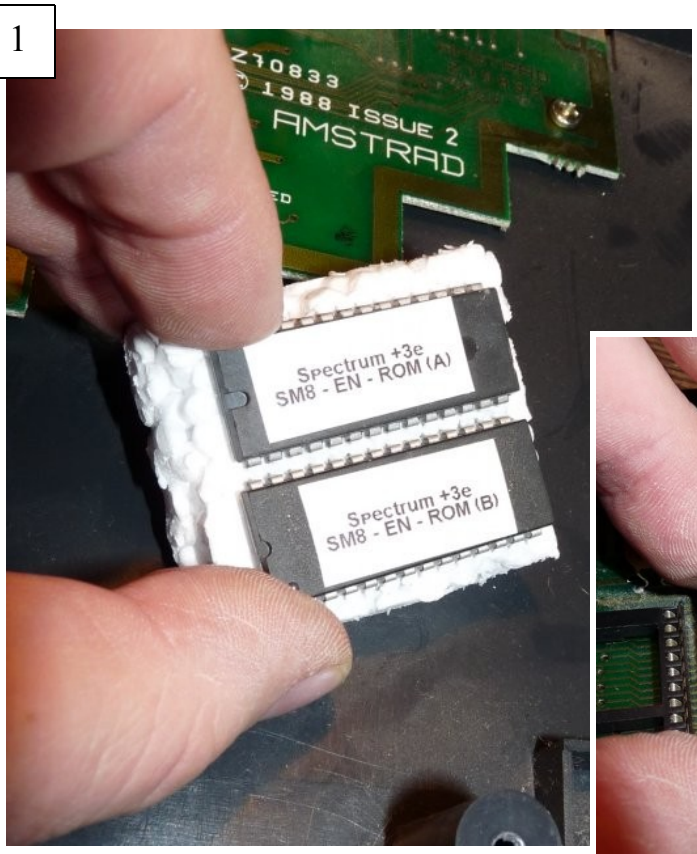
Correct orientation is paramount, to ensure that you do not damage you Spectrum or the CPU.

The socket maybe stiff, however I humbly suggest that you ensure no bent pins and press slowly into the socket – checking as you go.

3



1

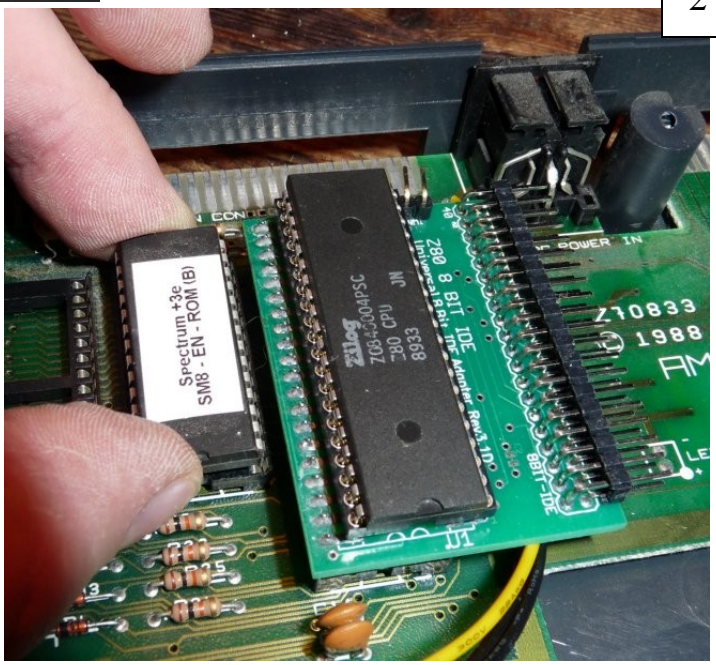


The Spectrum +3e ROM

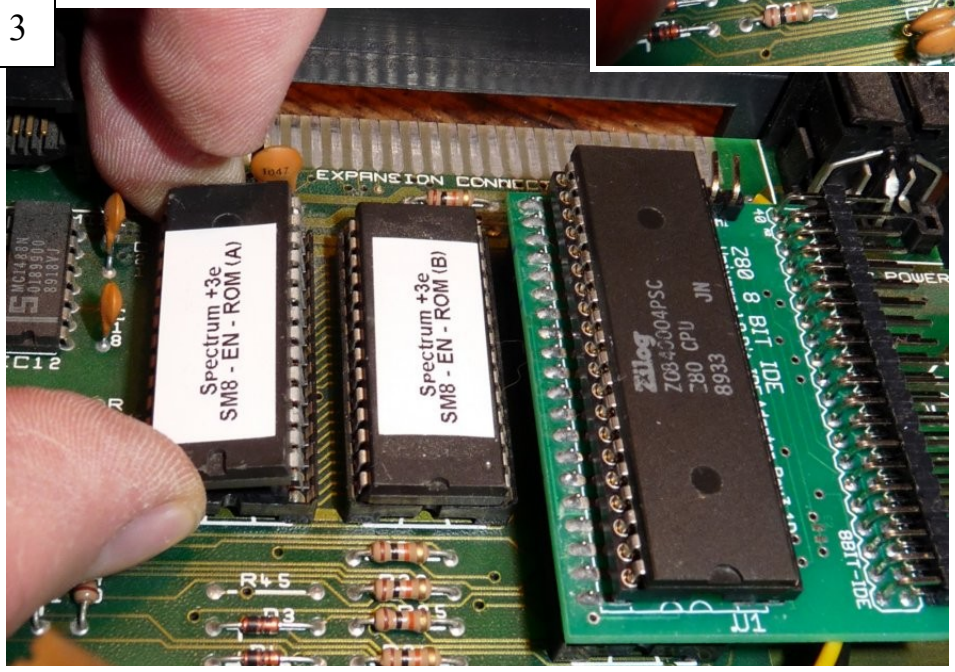
These are the heart of the upgrade – providing many bug fixes over the original ROM as well as support for 8 bit IDE.

The +3e ROM comes as two programmed IC's

2



3



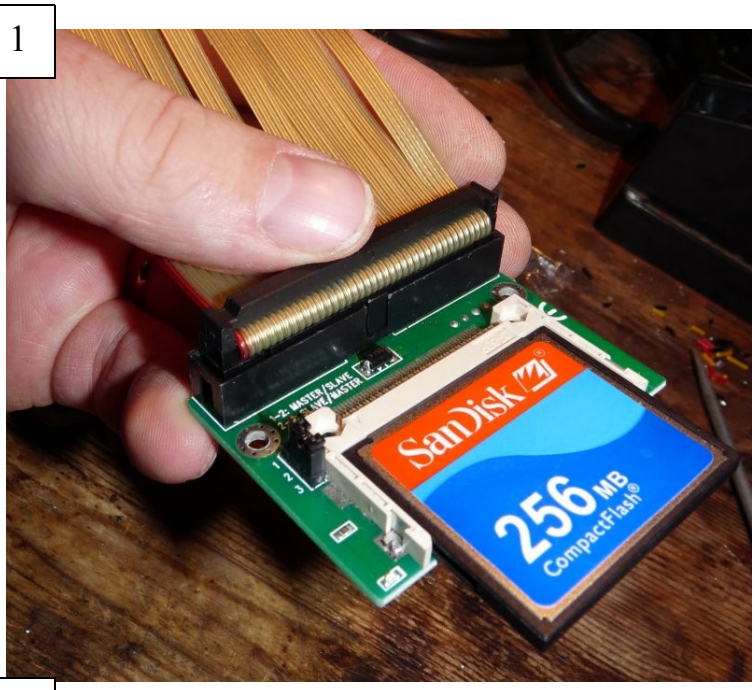
Installing +3e ROM(s)

The programmed +3e ROM's are labelled 'A' and 'B' for ease of installing.

ROM 'B' is closest to the IDE adapter and ROM 'A' closest to IC12 as depicted in the picture adjacent.

Sinclair Spectrum 128k +2A/B (Black)

1



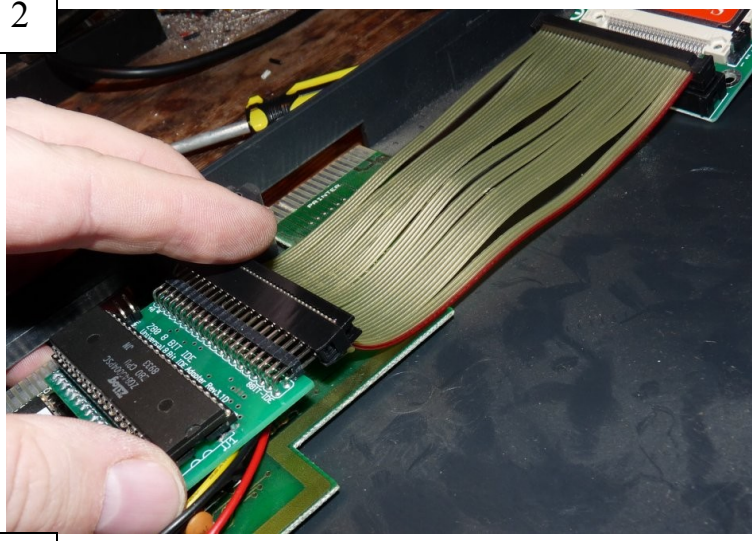
Connecting the Device to IDE

If you got the full kit, then you would have been provided with a custom 15cm IDE cable with a Device end and a Host end.

The Device end of the cable has a cable lock over the female connector. The Host end of the IDE cable does not.

In the picture example I have connected a Compact Flash to IDE adapter.

2

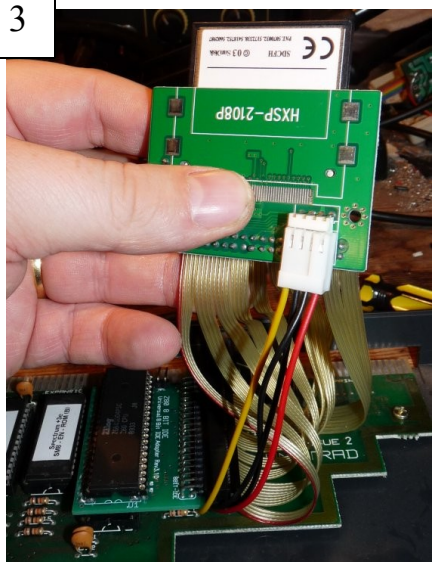


Connecting the Host to IDE

The Host is the IDE adapter, you will notice that the Host end of the provided IDE cable does not have a cable lock. This is to denote which should be connected for ease of use.

Either end of the cable can be connected, however please note the orientation of Pin 1 on the IDE Host (in the image adjacent it bottom left)

3



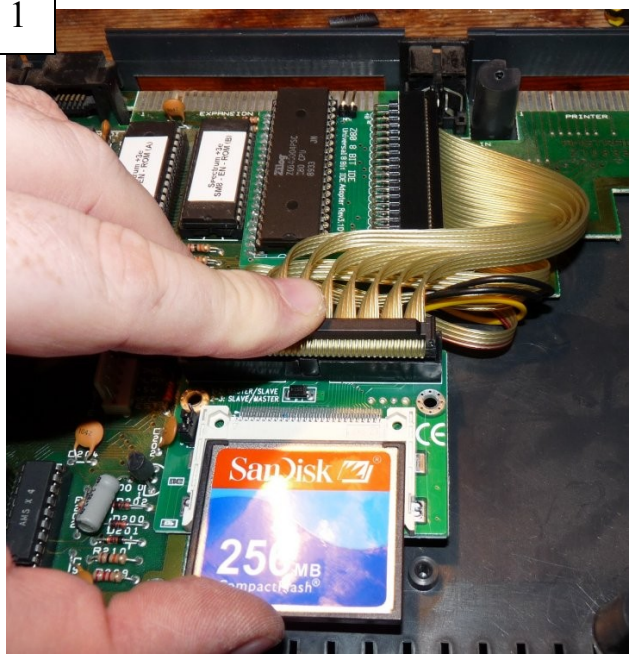
Connecting Power to the Device

In this instance we are using a Compact Flash to IDE adaptor. Thee adaptors only require +5 volts to operate.

As you can see in the picture adjacent I have fed the power cable under the IDE adaptor as well as pleated the IDE cable to ensure a cable tidy area.

Sinclair Spectrum 128k +2A/B (Black)

1



Fitting the Device to IDE

I decided to make this an internal modification as I prefer a stock looking model; however you could mount a device like this anywhere.

Please note that the 128k +2A/B space is a little limited as the cassette mechanism does take up a large area of space.

Also if you wanted to the Z80 Universal Z80 8bit IDE Adaptor can have up to two devices – thus one could have a permanent internal device as well as a removable external one. Your imagination is the limit here.

2

Fitting the Device

I used hot glue to mount the adapter in place, this is a very strong bond and only through intention will this break. Please note that I pleated the IDE cable to ensure a soft twist was possible – allowing me to install the device slightly below the host.

When starting up for the first time, here is what you will need to know.

FORTMATTING THE DEVICE

When you first turn on your spectrum you will need to format the IDE device you do this by using the FORMAT command

Synopsis:

FORMAT TO <unit number>,<number of partition>

This command formats a specified device with a maximum number of partitions available

Example:

```
FORMAT TO 0, 256
```

This formats IDE device '0' with up to 256 partitions. A top tip would be on deciding how many partitions you will need when formatting your device; divide the total number of megabytes in half and then divide that result by 8; i.e.

A 512MB Hard disk device would be 256MB / 8 = 32 Partitions, so we would write

```
FORMAT TO 0, 32
```

We use this method as the more partitions that are setup, the longer the CPU requires to read them.

PARTITIONING THE DEVICE

After formatting your device you will need to create partitions to store the data, games and utilities you want, for this you will need the NEW DATA command.

Synopsis:

NEW DATA "[unit number] > [partition name]", <size in megabytes>

This command creates a partition from the IDE unit's partition table, it then sets a size in megabytes on the partition table. Maximum size for a partition is 16Mbytes and the minimum is 1Mbyte

Example:

```
NEW DATA "0>GAMES_01",16
```

This creates a partition called GAMES_01 on IDE device 0 with 16MB of space. You are allowed up to 16 characters for a Partition name, while you are allowed spaces I would humbly suggest to use either a hyphenate symbol or use an underscore character.

Once a partition is created it takes a one away from the partition pool, so if you had stated 32 partitions (after formatting) and creating a new partition you will have 30 partitions as one is reserved as a SYSTEM partition

VIEWING IDE DEVICE(s) / PARTITION(s)

After you have created a few partitions you might want to review what is available, you can do this by using the CAT TAB command

Synopsis:

CAT TAB

This command lists the connected devices, partitions and available free storage memory and partitions available.

Example:

```
CAT TAB
```

DELETING PARTITION(s)

If you have made an error or simply do not want a partition any more you can delete this by using the MOVE and BIN commands, please note that all data in the partition will be lost!

Synopsis:

MOVE "[unit number]>[partition name]" BIN

This command will query if you are sure you want to delete, upon pressing 'Y' for yes it will then proceed and delete the given partition from the devices partition table..

Example:

```
MOVE "0>TEMP_GAMES" BIN
```

This deletes (removes) "TEMP_GAMES" partition from IDE Device '0'

RENAMING PARTITION(s)

If you wish to rename a partition you can use the MOVE and FOR commands

Synopsis:

MOVE "[unit number]>[partition name]" FOR "[partition name]"

This command will rename the partition on the given IDE device number for the new partition name

Example:

```
MOVE "0>GAMES_2" FOR "GAMES_ADVENTURE"
```

This renames the "GAMES_2" partition on IDE Device '0' (master) to "GAMES_ADVENTURE"

MOUNTING PARTITION(s)

After you have created a few partitions you will need to mount them so you can use DOS commands on them, you can do this by using the MOVE and IN commands

Synopsis:

MOVE "[XDPB]" IN "[unit number]>[partition name]"

This command sets an XDPB resource (a mount entry) so that the system can use DOS commands on the partition like you would with a standard DISK.

Example:

```
MOVE "C:" IN "0>GAMES_ADVENTURE"
```

This creates a mount point "C:" that you can now use with DISK commands such as LOAD, SAVE, COPY and ERASE. All DISK commands that are specified to "C:" will effect the "GAMES_ADVENTURE" partition on Unit 0 (master) of the IDE.

Example:

```
LOAD "C:ATICATAC.BAS"
```

This will load the file "ATICATAC.BAS" where "C:" points to, in this case it is IDE 0, partition "GAMES_ADVENTURE"

You can have up to two XDPB's at a time, when you want to use another partition you will need to assign another mount (XDPB) entry.

DISMOUNTING PARTITION(s)

As you can only have two mount points at anyone time, you will need to shuffle their use as you work with your spectrum environment, you can do this by using the MOVE and OUT commands

Synopsis:

MOVE "[XDPB]" OUT

This command removes an XDPB resource (a mount entry) so that the system can use / allocate this later.

Example:

```
MOVE "C:" OUT
```

This removes the mount point "C:" so that you can use this XDPB resource again on another partition.

PERMANENT PARTITION(s)

Constantly adding mount points (XDPB's) to partitions on every boot would be a really painful experience; because of this you can make a partition mount point semi permanent that will survive a power off. To achieve this you use the "ASN" command / token

Synopsis:

MOVE "[XDPB]" IN "[unit number]>[partition name]" ASN

This command sets an XDPB resource (a mount entry) so that the system can use DOS commands on the partition like you would with a standard DISK and makes it permanent saving this information onto the SYSTEM partition.

Example:

```
MOVE "C:" IN "0>GAMES_ADVENTURE" ASN
```

This creates a mount point "C:" that you can now use with DISK commands such as LOAD, SAVE, COPY and ERASE. All DISK commands that are specified to "C:" will effect the "GAMES_ADVENTURE" partition on Unit 0 (master) of the IDE, this has been made permanent so on a reboot or power cycle "C:" will always be mounted as above.

DISMOUNTING PERMANENT PARTITION(s)

At times you may need to reuse an XDPB resource that is semi permanent, to do this you this you will use the "ASN" and "OUT" command / tokens

Synopsis:

MOVE "[XDPB]" OUT ASN

This command removes an XDPB resource (a mount entry) as well as its information stored on the SYSTEM partition.

Example:

```
MOVE "C:" IN OUT ASN
```

This action allows the XDPB resource to be reused and assigned to another partition. A point to note is that you can also dismount drive "A" and or "B" however you cannot make these permanent dismounts nor can you re-use these as an XDPB resource.

LOADING DATA FROM PARTITION(s)

Before you can **load** data from a partition you will need you assign that partition a mount point (XDPB) so that DOS commands can work on it. Please see page 15 for more details. Once you have a mount point, we will use "C:" for the following example.

Synopsis:

LOAD "[XDPB][filename]"

This command loads a file name from the given XDPB resource (mount entry). LOAD is the same as the standard disk LOAD. You may add modifiers like SCREEN$ or CODE

Examplse:

```
LOAD "C:ATICATAK.BAS"
LOAD "C:Pic-01" SCREEN$
LOAD "C:DATABLOK.C" CODE
```

At this point you should notice that a file name is limited to 8 characters long with a '.' and then a 3 character extension. You can use / derive any extension you like or not have one at all, the choice is yours.

SAVING DATA TO PARTITION(s)

Before you can **save** data from a partition you will need you assign that partition a mount point (XDPB) so that DOS commands can work on it. Please see page 15 for more details. Once you have a mount point, we will use "C:" for the following example.

Synopsis:

SAVE "[XDPB][filename]"

This command **saves** a file name to the given XDPB resource (mount entry). SAVE is the same as the standard disk SAVE. You may add modifiers like SCREEN$ or CODE

Examplse:

```
SAVE "C:MYPROG.BAS"
SAVE "C:Pic-01" SCREEN$
SAVE "C:GAMELOAD." LINE 10
SAVE "C:DATABLOK.C" 32762,4096
```

At this point you should notice that a file name is limited to 8 characters long with a '.' and then a 3 character extension. You can use / derive any extension you like or not have one at all, the choice is yours.

LISTING PARTITION DATA

Before you can view data from a partition you will need you assign that partition a mount point (XDPB) so that DOS commands can work on it. Please see page 15 for more details. Once you have a mount point, we will use "C:" for the following example you can use the CAT command to list a CATALOG view of the stored data.

Synopsis:

CAT "[XDPB]"

This command prints to the screen the filenames stored in the partition that the XDPB resource (mount entry) points to.

Examplse:

```
CAT "C:"
CAT "A:"
```

LISTING MOUNTED DEVICES

Another feature of the CAT command is to list for the use the current XDPB (DOS Mounting) entries that are in use on the system, you do this by using the command CAT and ASN

Synopsis:

CAT ASN

This command prints to the screen a list of current devices (XDPB)'s and their paths. This is a very useful function to discover what resource is in use and where it is pointing too.

Examplse:

```
CAT ASN
A: 2>FLOPPY DEVICE 0
B: 3>FLOPPY DEVICE 1
C: 0>GAMES_01
D: 0>USERDATA
M: 4>RAMDISK
```

The above example shoes that XDPB's "C:" and "D:" are in use and point to two different partitions on the IDE device 0 (master).

PARTITION(s) AND FOCUS

You will need you assign that partition a mount point (XDPB) so that DOS commands can work. Please see page 15 for more details. We will use "C:" as out mount point for the following example; Setting a focus to a partition a BOOT using the VERIFY command.

Synopsis:

VERIFY "[XDPB]"

This command set the current XDPB as the focus of all DOS commands. This means that all DOS operations are defaulted to the current XDPB.

Examplse:

```
VERIFY "C:"
```

After typing this command one can now use standard DOS commands like LOAD, SAVE, ERASE without the need to include an XDPB in the command

Examplse:

```
LOAD "MYPROG.BAS"
CAT
```

As you can see this reduces the time in typing expressions as well as allowing for a dynamic loading and saving that can be altered by the user in a program or by the program itself.

BOOT PARTITION(s)

Before you can set a partition as a bootable you will need you assign that partition a mount point (XDPB) so that DOS commands can work on it. Please see page 15 for more details. Once you have a mount point, we will use "C:" for the following example you can make any partition a BOOT partition with the use of the VERIFY and ASN commands.

Synopsis:

VERIFY "[XDPB]" ASN

This command set the current XDPB as the focus of all DOS commands and makes this permanent. This means that all DOS operations are defaulted to the current XDPB.

Examplse:

```
VERIFY "C:" ASN
```

After typing this command one can now use standard DOS commands like LOAD, SAVE, ERASE without the need to include an XDPB in the command. On boot, selecting the LOADER will search for a file called "DISK." This can be any executable file.

Usually the BOOT focus is set to "A:" Floppy Device, however this can be set to any current XDPB including "M:" RAM Disk

Using the Z80 Universal 8bit IDE Adaptor

TESTING THE IDE ADAPTOR AND IDE DEVICE

Once you have setup a partition on your IDE device and assigned a mount point to it, please see page 15 for more information, we shall assume "C:" in this matter, now we can test the device and the adaptor with a very simple program.

Program:

```
10   FOR F = 0 TO 7
11   PRINT ; INK F ; PAPER 7 – F ; "HELLO WORLD"
12   BEEP 0.01,F+6
13   NEXT F
14   GOTO 10
9998 STOP
9999 CLS : PRINT "SAVING PROGRAM." : SAVE "C:MYPROG.BAS" LINE
     1 : CLS : PRINT "PROGRAM SAVED." : STOP
```

Once you have typed this simple listing out then type -

```
RUN 9999
```

This will save the program to the IDE device, once this is done you can type

```
RUN
```

To see the program execute, once you have seen this reset the computer and load this program in by typing

Example:

```
LOAD "C:MYPROG.BAS"
```

This program will automatically run from disk and repeat until either reset or the program is stopped by breaking into. When you are ready press **break** and **space** keys and once you have entered the program listing type

```
RUN 9999
```

This will save the program again, however this time, as there is already a program with the name "MYPROG.BAS" it will create a backup of the file, renaming it "MYPROG.BAK" and then save the current program as "MYPROG.BAS"