SANDY

Introduction

This manual describes the facilities and operation of SANDY QL disk interfaces. The following conventions are used throughout:

COMMAND	for parts of commands to be typed as shown
example	for parts of commands shown as examples
description	for descriptive parts of commands.

CONTENTS:

	Page
Introduction	1
Interface Descriptions	2
Fitting instructions	4
Floppy Disk Driver	7
Ram Disk Driver	14
Parallel Printer Port	17
SuperBASIC Extensions	20
Resident Clock	30
Index	31

SuperToolkit II facilities are explained in the separate Toolkit manual.

QRAM / POINTER INTERFACE is described in the QJUMP QRAM manual.

Every effort has been made to ensure that the information given in this manual is as accurate as possible, however, SANDY (UK) PCP LTD do not accept any liability for damage or losses, resulting from the direct or indirect use of this information.

INTERFACE DESCRIPTION

All the interfaces are based on a standard printed circuit board which has a 64 way connector to fit into the QL expansion port and a plastic moulded housing to protect the disk and printer cable connectors positioned on the left of the QL and matching the computer case styling. The boards are correctly sized to slide into the guides inside the QL and provide a completely secure mounting. Only Sandy high technology offers so much power in so neat a package. The printed circuit boards are highly complex fine-line through-hole-plated designs produced on computer controlled high precision machines and electronically tested before components are inserted.

Disk controllers can handle two drives of any type or size from 3" to 5.25" or any mixture of drives.

Parallel ports are Centronics standard with multi-tasking spoolers.

*** SUPERDISK ***

Multi-standard Disk interface
Centronics compatible printer port
ROM Software built in:
Eight static RAM disks
SuperBasic - 31 extra commands
File handling commands
Multi-tasking Spooler controls

*** SUPERQBOARD 0K RAM ***

Multi-standard Disk interface
Centronics compatible printer port
Switch mode transistor power regulation for precise on board supply but drawing only the minimum current to run the IC's which are actually demanding power - this greatly reduces the load on the QL power supply and cuts out wasteful heat build up.
ROM Software built in:
Eight static RAM disks
File handling commands
Multi-tasking Spooler controls
QJUMP SuperToolkit II

*** SUPERQBOARD 512K RAM ***

As above but with 512 Kilobytes of very high speed memory added as a separate close coupled board with built in custom memory management to achieve the fastest access times possible. The memory board is powered from the SUPERQBOARD integral switch mode system and can be plugged into the 0K RAM version to upgrade the memory capacity. Total system memory is 640 Kilobytes.

*** SUPERQBOARD 512K RAM MOUSE VERSION ***

As the 512K SUPERQBOARD but with a built in mouse attachment port.
ROM Software built in:
Disk management commands
Spooler controls
QJUMP SuperToolkit II
QJUMP Pointer interface extensions
DISK or MICRODRIVE based software:
QRAM - QJump real windows/menus multi-tasking environment
Eight static/dynamic RAM disks
CADPAK demonstration graphics program using the mouse system
MOUSE_CDE - a routine to emulate the joystick ports onto a mouse
MOUSE:
High precision 2-button mouse using optical resolvers for long life and great accuracy with silent operation.

*** SUPERQBOARD 0K RAM MOUSE VERSION ***

As above but without the 512K RAM board. Provides the same facilities for QL systems which have internal memory expansion or through connector RAM cards already fitted.

SANDY interfaces are compatible with all versions of the QL including German MGG and USA JSU ROM variants. Although the boards are designed to fit neatly inside the QL expansion port they will operate with any through connector memory card or suitably buffered extension board system. Internally upgraded machines will work correctly although this type of memory expansion executes more slowly than the SuperQBoard memory module.

FITTING INSTRUCTIONS

1. Fitting the interface card

DISCONNECT THE QL FROM ANY OTHER PERIPHERAL AND FROM THE MAINS.

The expansion port is on the far left of your QL. Gently remove the plastic cover by pulling it away from the QL using a screwdriver under the moulded tag. Look into the open end and you may find two small plastic lugs protruding down from the keyboard moulding; remove these with a sharp knife to leave a clear opening. Some German and USA versions have a metal ground contact spring on the left of the opening; this should be removed by pulling it carefully out with a pair of pliers.

Now stand the QL up on its right-hand end with the expansion port facing upwards, grasp the interface by its plastic cover and gently slide it downwards into position; the components on the interface should be to the keyboard side of the QL; there are plastic guides in the QL to locate the board correctly. When the plastic cover is 6mm from the QL casing stop and check that the board is absolutely square to the QL casing in all directions before pushing the board fully home. The plastic should then be flush against the QL casing.

2. Connecting the disk drives

DISCONNECT THE QL AND ANY OTHER PERIPHERAL FROM THE MAINS.

The interface has two ports, a 34 pin socket where you should fit your disk drive ribbon cable connector; this connector will only fit in one way up, with the notch facing upwards, so do not force it.

3. Connecting the printer

The centronics compatible printer port is a 26 pin connector. A suitable ribbon cable can be supplied by your dealer. Once again the connector plugs in one way only, with the notch facing upwards.

4. Testing the Interface

Power-up sequence:

1. Switch on the QL
2. Switch on the disk drives

At this point, the interface will check if there is a floppy disk in the disk drive, or drive one if you have a dual disk drive, and the start up message will appear at the top of the screen with the Sinclair TV/Monitor (F1-F2) prompt at the bottom.

Notice that on machines with extra memory the memory test multi-coloured screen pattern will last longer than for a 128K machine as all the memory is checked and the disk handling routines enabled.

The memory available message is not just a fixed logo but an actual display of the free RAM found and tested by the QL on start up.

At this point you press F1 or F2 accordingly.
If the interface found a disk in the drive during reset, it will check if there is a file called "BOOT". If such a file is found, it will be loaded from disk and RUN.

If there is no floppy disk in before power-up or reset, autoboot will operate from MDV1_ as usual.

Power-down sequence:

1. Switch off disk drives
2. Switch off QL

If the QL and disk drives are connected to one power source they may be safely powered up and down from the one switch.

5. Removing the Interface

If for any reason you need to remove the interface board, you MUST remember to:

DISCONNECT THE QL FROM ANY OTHER PERIPHERAL AND THE MAINS.

Unplug the disk and printer cables, grasp the SuperQBoard by the plastic cover and slide carefully out.

Note: if your QL expansion port has never been used before the plasic guides and connector pins may be quite tight - do not force the interface in; gently wriggle the board while pushing lightly in. If your QL has a really tight casing slacken the case securing screws near the expansion port and re-tighten them after the interface is installed.

MOUSE VERSION: the 9-pin connector from the mouse should not be inserted or removed while the QL is powered up. There is no standard for the pin-out arrangement on mouse systems, do not plug in any other mouse or use the SANDY mouse on any other machine without first checking that the connections are the same. Serious damage may occur if the wrong pins are connected.

MAINTENANCE: no maintenance is required and there are no user serviceable components on the interface. When handling the unit take normal care to avoid static discharge (ie. hold the board only by the plastic case and do not leave it lying on a non-conductive surface such as a man-made fibre carpet)

DIAGRAM 1 - SANDY QL DISK INTERFACE

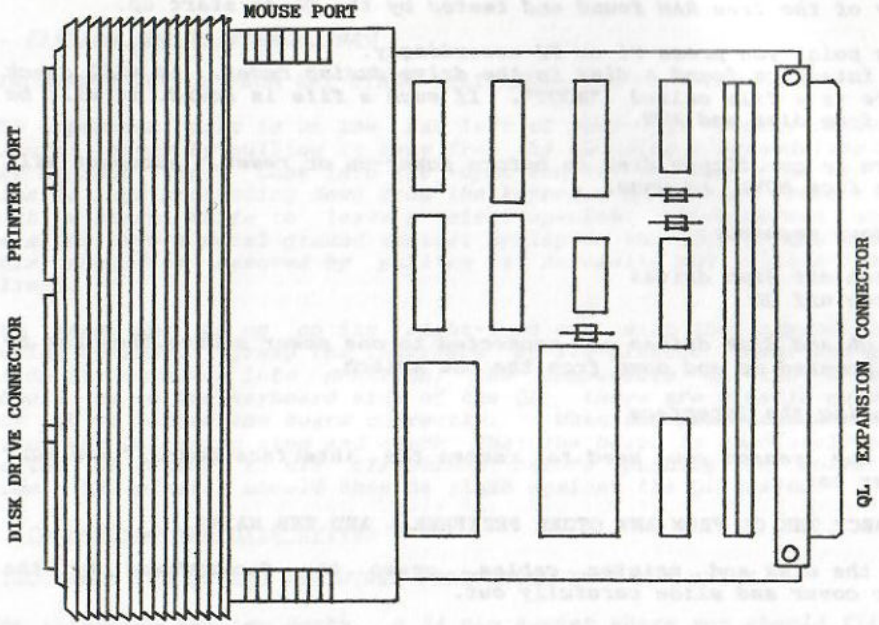

DIAGRAM 2 - PIN CONNECTIONS

PRINTER PORT

13 ●●●●●●●●●●●●● 01
26 ●●●●●●●●●●●●● 14

PIN	CODE
01	STROBE
02	DATA 1
03	DATA 2
04	DATA 3
05	DATA 4
06	DATA 5
07	DATA 6
08	DATA 7
09	DATA 8
10	ACKNLG
11	BUSY
12	NC
13	NC
14/24	GND
26	NC

DISK DRIVE PORT

33 ●●●●●●●●●●●●●●●●● 01
34 ●●●●●●●●●●●●●●●●● 02

PIN	CODE
02	NC
04	NC
06	NC
08	INDEX
10	DRIVE 1
12	DRIVE 2
14	NC
16	MOTOR ON
18	DIRECTION
20	STEP
22	WRITE DATA
24	WRITE GATE
26	TRACK 00
28	WRITE PROTECT
30	READ DATA
32	SIDE 1
34	NC
01/33	GND

QJUMP Floppy Disk Driver

Beginners Start Here

The QL computer is delivered with two 'mass storage' devices: the microdrives. These devices have the same function as floppy disks, being designed for the permanent storage of programs and data. Other devices which behave in the same way as microdrives (such as floppy or hard disks) may be added to the QL 'transparently'. This means that the computer will ensure that a program doesn't need to know where or how its data is stored. A microdrive looks the same to a program as a floppy disk. This 'device independence' is a built in characteristic of the QDOS operating system.

The simplest way of using a floppy disk system on the QL is to copy all programs and data to floppy disks, then make the QL use the disks when it previously used the microdrives. To do this, add the command 'FLP_USE MDV' to all BOOT files, or type this command at the start of a session on the QL. The effect of this command is to make the floppy disks pretend to be rather large and fast microdrives.

For example, a modified BOOT file for executing the PSION program Quill could look like:

```
100 FLP_USE mdv
110 CLOSE #1: CLOSE #2
120 EXEC_W mdv1_quill
....
....
```

This is useful for running the occasional program which has been transferred from microdrive but it is just as easy to use the floppy disks without changing the name and this also gives access to the microdrives using the normal MDV device identifier. All the filing system commands described in the 'Microdrives' section of the QL Reference Guide Concepts section will work with floppy disks, provided the device name is 'FLP' instead of 'MDV':

FORMAT flp1_diskname	formats a new floppy disk in drive 1
*DIR flp1_*	lists directory of floppy disk 1
SAVE flp1_myprog	save the current SuperBASIC program as 'myprog' in floppy disk 1
OPEN_NEW #3,flp2_data	creates and opens new file 'data' in floppy disk 2
COPY mdv1_x TO flp1_x	copies file *x* from microdrive 1 to floppy disk 1

Floppy Disk Compatibility

The QJUMP Floppy Disk Driver software not only provides all the built-in microdrive filing system operations, but also contains all the extended filing system operations provided in the Toolkit for microdrives. This allows the SuperBASIC extensions provided in the QL Toolkit to be used with the floppy disks.

Auto-boot

If there is a disk in drive 1 when the QL is turned on (this may be risky with some makes of floppy disk drive, particularly those with permanently loaded heads) or reset (this should be safe with all drives), then the QL will boot from the disk in drive 1, otherwise the QL will boot from microdrive 1 as usual.

When a 'directory device', such as a floppy disk, is accessed for the first time, QDOS allocates a block of memory which contains all the information it will need about the device. In the case of a floppy disk, the Sinclair standard format requires a block of memory about 1.6 kilobytes long. The auto-boot procedure ensures that if there is no disk in drive 1 when the QL is reset, then the 1.6 kilobytes block for disk drive 1 will not be allocated. Programs that are too large to execute when floppy disks are being used on a 128K machine, should still work from microdrives.

Microdrive Emulation

The standard driver includes a command FLP_USE to change the name of the floppy disk driver.

```
FLP_USE mdv     or FLP_USE 'mdv'
```

makes floppy disks call themselves 'mdv', so that all subsequent commands to the microdrives will use the floppy disk instead. Thus the commands:

```
FLP_USE mdv
....
....
OPEN #3,mdv1_myfile
```

will actually open the file 'myfile' on floppy disk 1, rather than trying to open a file on microdrive 1.

Any three letters may be used as a new device name.

```
FLP_USE flp
```

will reset the driver to its normal state. After typing this the floppy disks are once again called 'flp' and the microdrives are used as normal.

Floppy Disk Options

There are three parameters of the floppy disks which are available to the user as options.

The security level is selectable to allow a user to choose higher speed of access at a cost of reduced immunity to erroneous disk swapping. There are three levels of disk security, the lowest level still being at least as secure as some other disk based operating systems like MSDOS and CP/M.

A user may specify the time taken for the disk drive motor to reach full speed. This can vary between different makes of drive.

A user may specify the number of tracks to be formatted on a disk.

These parameters are specified by three separate commands, each of which has only one parameter:

FLP_SEC	security level
FLP_START	start up time
FLP_TRACK	number of tracks

Security

The microdrive filing system is unusual in that, although the data is stored in 'sectors' in just the same way as on a floppy disk, each sector holds information which identifies the cartridge. When a cartridge is changed the filing system will spot any change as soon as it next uses the microdrives although considerable space on the cartridge is taken up by this repeated information.

Standard floppy disk formats do not allow this level of security as it would seriously reduce available storage capacity, all the identifying information is placed in Track 0 Sector 1 of the disk. Clearly, if this were checked every time we wanted to access the disk, then the disk system would be very slow as the head would have to drive to the edge of the disk (track 0) stop; read the information; move back to the data track and continue. Security, in this context, is the extent to which the floppy disk system may be abused by changing disks, while they are in use (though not actually running!), without destroying data stored on the disks.

There are four operations which affect disk security:

[1] Check if the disk has been changed
[2] Flush the slave blocks
[3] Update the map
[4] Update the directory

In these definitions, the term 'the drive has stopped' is usually taken to mean that the drive motor has stopped and no drive select light is visible. On some disk systems - particularly 5.25" drives - if an attempt is made to access a drive which has no disk in place, or has the door open, then the motors continue to run. If a drive select light is still on, then the motors may be stopped by inserting a disk and closing the door. In any case, the interface assumes the drives have stopped if five seconds have passed without a successful disk access.

Security Level 0

The disk is only checked when a file is opened and the drive has stopped since the last time it was checked and there are no files already open on the drive.

The map is only updated after a file is closed (or flushed) when half a second has elapsed without any other disk operation.

At this lowest level of security, confusion and loss of data can be expected if a floppy disk is removed while the motor is running or there are still files open.

Security Level 1

The disk is checked when a file is opened, or data or the map is to be written, and the drive has stopped since the last time it was checked.

The map is only updated after a file is closed (or flushed) when half a second has elapsed since the previous disk operation.

At this level of disk security, disks should only be changed while the motor is stopped. If a disk is changed while there are still files open, then read operations will be confused but any write operations will be aborted. This should maintain the integrity of the data on the disk.

Security Level 2

The disk is checked whenever a file is opened or whenever the map or data is to be read from or written to the disk and the drive has stopped since the last time it was checked.

The map and directory are updated and the buffers are flushed immediately after a file is closed, or after an FS.FLUSH call.

This is the default security level and data should be quite secure unless a disk is changed while the motors are actually running.

Security System Errors

There are two error messages which may be written to the screen by the disk system. These are in the form of the disk name followed by the message itself. The first message indicates that an attempt to read or write a sector on the disk has failed:

disk name read/write failed

The second message indicates that a disk has been changed while it is still in use:

disk name files still open

If the floppy disk system attempts to write to a disk which has been changed, then you might get both messages indicating that the attempt to write the data has been aborted, and that files were still open when the disk was changed.

Start Up Time

For good performance the disk system always tries to read data from a disk as soon as possible but different drives take longer or shorter times to accelerate the disk from stationary to steady speed. Reading data before steady speed is achieved may result in corrupt information. Start up time gives a delay each time the disk starts up with the default at 0.3 second which matches most modern drives. The parameter for FLP_START is in 20 millisecond units with a default of 15 to give 300 milliseconds, older drives may require values up to 60, refer to your disk drive manual for the start up time requirement. If the time is too long some drives may actually shut down automatically because they only wait so long for data to be transferred!

The command to set start up time is:

FLP_START 60 or whatever value matches your drive

Number Of Tracks

The QL format for disks stores the number of tracks on the disk itself during formatting. The interface finds out whether the drive is a 40 or 80 track unit when formatting, by attempting to drive the head to track 55. If it succeeds the drive is assumed to be 80 tracks, otherwise it is assumed there are 40 tracks. This internal check may be overwritten, allowing 37 and 75 track drives to be formatted as well as saving possible wear or damage to a 40 track drive when seeking track 55 (somewhere in the middle of the jacket) as not all drives have fixed end stops. The command is:

FLP_TRACK 40 or however many tracks are needed

Remember this only affects formatting; if an 80 track diskette is subsequently inserted the disk interface will know from the format information on the disk itself.

Direct Sector Read/Write

The software includes provision for reading sectors from a disk using direct addressing. To do this a special file is opened on the disk. The name of the file is:

```
FLP1_*Dsd       where 's' is the sector length
                0=128 bytes     1=256 bytes
                2=512 bytes     3=1024

                and 'd' is the density
                S=single (FM)   D=double (MFM)
```

When opening a disk for direct sector read/write from SuperBASIC, the name should be enclosed in quotes or apostrophes.

```
OPEN #3,'flp1_*D2d'
```

Opens a direct access file with sector length of 512 bytes and double density.

When this type of file is open no other file may be opened on the drive. The only IO calls supported for this type of file are IO.FSTRG, IO.SSTRG, IO.POSAB and IO.POSRE; to read or write complete sectors or to set the position. The parameter (D1) to the POSRE call is ignored, but the current position is returned. Reading or writing a sector does not change the file position.

If the attempt to read or write a sector fails, D0 will be returned as a standard error message pointer (read/write failed).

The position is a composite of the required sector, side and track:

```
sector number + side * 256 + track * 65536
```

To ensure compatability with string IO the length specified in the SSTRG and FSTRG calls may be one of three values:

sector length	the complete sector is read or written
2	returns the sector length (IO.FSTRG) or ignored (IO.SSTRG)
2 + sector len	returns the sector length followed by the sector (IO.FSTRG) or skips the first two bytes, and writes the rest to the sector (IO.SSTRG)

This variety enables sectors to be written and read in SuperBASIC using the normal string IO in the QL Toolkit, as well as by assembler programs. For example, sector 1 of side 1 on track 2 may be read into the string a$ using the following command:

```
GET #n\1+256+2*65536, a$
```


When using the direct sector read/write calls for a 40 track disk in an 80 track drive, the track number should be doubled. Seek errors will not be detected. If a read/write error is returned from a direct sector read/write call, then it will be safest to make another call to read from track zero. Calls to read or write to track zero will cause a 'restore' rather than a seek, and will thus reset the drive to a known state.

Disk Drive Specifications

It is a requirement that disk drives used with this version of the disk driver should be set to have the motor on when given a 'motor on' signal and there is a disk in the drive. Drives which turn the motor off when the drive is not selected will not give reliable service.

The disk driver will automatically adjust itself to use any mixture of disk drives, 40 or 80 track, single or double sided. In addition, it will adjust itself to use slow step rate drives. Disks need not have been formatted and written on the same specification drive as a drive being used to read them.

Compatability chart

	DISK FORMAT			
DRIVE TYPE	40T	40T+40T	80T	80T+80T
40T	C	?	X	X
40T+40T	C	C	X	X
80T	R	?	C	?
80T+80T	R	R	C	C

C = compatible
X = incompatible
R = compatible for read only
? = incompatible but may be detected detected correctly on some drives

Single Sided Formatting

It is possible to force the disk driver to format a disk as single sided on a double sided drive by making the 11th character (it is invisible) of the medium name an asterisk: e.g.

```
FORMAT 'flp1_disk_name *'
```

Quotes or apostrophes around the entire parameter are necessary.

QJUMP 'RAM DISK' Driver

The term RAM DISK is a misnomer. It is used to describe a 'virtual' device, one that exists only within the memory of the QL, which looks and behaves like a very fast disk drive. It is so fast because being virtual, there nothing mechanical to move around to get information in and out. It is, in reality, a reserved area of the QL's memory. This means that the memory taken by a ram disk is no longer available to a program. It also means that all its contents will be lost if the QL is reset or turned off.

Up to 8 RAM disks can be formatted at the same time and may be of any size, subject to the limitations of available memory. The normal usage of a ram disk would be to copy all working files from a floppy disk or microdrive into a ram disk, rename the RAM disk as 'mdv' to pretend the data is really on microdrives, execute the programs, then at the end of a session rename the RAM disk as 'RAM' before copying all its contents onto microdrive.

It is just as easy to use a RAM disk directly without changing its name. All the filing system commands described under the 'Microdrives' heading in the concepts section of the QL User Guide will work with RAM disks, provided the filenames start with 'RAM' instead of 'MDV'.

`FORMAT ram2_200`	creates new RAM disk 2, see below
`DIR ram1_`	directory listing of RAM disk 1
`SAVE ram1_myprog`	save the current SuperBASIC program as 'myproc' in RAM disk 1
`OPEN_NEW #3,ram2_data`	creates and opens a new file 'data' in RAM disk 2
`COPY mdv1_x to ram1_x`	copies file x from microdrive 1 to RAM disk 1

RAM Disk Creation

There are two forms of RAM disk for the QL; in one form the space for the files in the RAM disk is allocated dynamically using any spare memory in the QL. Unfortunately, this scheme, although very simple to implement using QDOS, does not work in conjunction with Psion programs or any other programs which automatically use all the spare memory themselves. However, the second form of RAM disk has its own memory allocation routines which operate within a pre-defined area of the QL's memory these are Static RAM disks. Dynamic RAM disks are only used in the Mouse Version interfaces where QRAM has special techniques to limit the memory taken by programs such as the Psion suite.

A RAM disk is created by formatting it : the size, in sectors, is given in place of the usual medium name.

```
FORMAT ram2_80
```

removes the old RAM disk 2, if there was one, and sets up a new RAM disk of 80 sectors. A RAM disk may be removed by formatting it to zero sectors, or with no number at all:

```
FORMAT ram1_    or    FORMAT ram1_0
```

The RAM disk number should be between 1 and 8, inclusive, while the number of sectors is limited only by the amount of memory available.

It is important to note that each sector is the same size as a microdrive sector, which is 512 bytes (or characters) long. If you wish to, for example, load a 16 kilobyte program into a RAM disk the RAM disk must be formatted to at least 32 sectors.

Microdrive Emulation

The standard driver includes a SuperBASIC procedure RAM_USE to change the name of the RAM disk driver.

```
RAM_USE mdv     or     RAM_USE 'mdv'
```

resets the name of the RAM disk driver to 'mdv', so that all subsequent microdrive commands will operate the RAM disk instead:

```
RAM_USE mdv
....
....
OPEN #3, mdv1_myfile
```

will actually open the file 'myfile' on RAM disk 1, rather than trying to open a file on microdrive 1.

Any three letters may be used as a new device name.

```
RAM_USE ram
```

will reset the driver and microdrives to their normal state.

Heap Fragmentation

The primary storage mechanism in the QL for permanent or semi-permanent memory allocations is the 'heap'. Allocating space in a heap, and then re-allocating this space as a different size, inevitably causes holes to be left within the heap. This reduces the amount of memory available to either SuperBASIC or executable programs.

The RAM disk driver has precautions to prevent the possibility of heap fragmentation, but it is preferable to consider a RAM disk to be a permanent feature until the QL is reset.

Examples

The following example will copy selected files from the microdrive cartridge to the RAM disk as well as copying Quill (v2.3, other versions may require a different size of RAM disk.

```
100 PRINT #0,'Put QUILL in MDV1 and press a key'
110 PAUSE
120 FORMAT ram1_150
130 COPY mdv1_quill,ram1_quill
140 COPY mdv1_quil_hob,ram1_quil_hob
150 COPY mdv1_printer_dat,ram1_printer_dat
160 :
170 print #0,'Put data cartridge in MDV2 and press a key'
180 PAUSE
190 FORMAT ram2_200
200 OPEN_NEW #3,ram2_file_list : REMark  make list of files
210 DIR #3,mdv2_
220 CLOSE #3
230 OPEN_IN #3,ram2_file_list
240 INPUT #3,a$,a$ :              REMark  skip heading
250 REPeat files
260   IF EOF(#3) : EXIT files
270   INPUT #3,file$
280   INPUT #0,'Copy '&file$&' to RAM disk? ';ans$
290   IF 'y' INSTR ans$: COPY 'mdv2_'&file$ to 'ram2_'&file$
300 END REPeat files
310 CLOSE #3
320 DELETE ram2_file_list
330 :
340 RAM_USE mdv :                 REMark  all across from mdv
```

This program copies files back to microdrive at the end of a session.

```
100 RAM_USE ram                   REMark reset RAM name
110 :
120 PRINT #0,'Put the data cartridge in MDV2 and press a key'
130 PAUSE
140 OPEN_NEW #3,ram2_file_list:  REMark  make list of files
150 DIR #3,ram2_
160 CLOSE #3
170 OPEN_IN #3,ram2_file_list
180 INPUT #3,a$,a$ :              REMark  skip heading
190 REPeat  files
200   IF EOF(#3): EXIT files
210   INPUT #3,file$
220   if file$='file_list': NEXT files
230   DELETE 'mdv2_'&file$
240   COPY 'ram2_'&file$ TO 'mdv2_'&file$
250 END REPeat files
260 CLOSE #3
270 DELETE ram2_file_list
```


Parallel Printer Port

All devices which connect your QL to the outside world are identified to QDOS by a name. The serial ports, marked SER1 and SER2, are identified by the name SER. The new parallel printer port on your SANDY disk interface is called PAR.

Most programs which send output to a printer are written to use the serial port SER1. The simplest way of changing over to using the parallel port is to include the command 'PAR_USE ser' which may be included in a BOOT file or entered directly at the keyboard. The command changes the name of the Parallel Port to SER and cancels the name of the serial ports - any output sent to the serial ports will go to the parallel printer port instead.

For example, a modified BOOT program to load the PSION program QUILL could look like:

```
100 PAR_USE ser
110 CLEAR:WINDOW 512,256,0,0
120 AT 2,4:PRINT "LOADING QUILL"
130 AT 4,13:PRINT "VERSION ";x.xx
140 AT 6,6:PRINT "copyright 1984 PSION LTD"
150 AT 8,12:PRINT "word processor"
160 CLOSE #1:CLOSE #2:WINDOW #0,400,20,35,215
170 EXEC_W MDV1_QUILL
180 OPEN #1,scr:OPEN #2,scr
....
....
```

The Psion programs, and many others, can be configured to change the name of the printer device. If this is to be done, it is only necessary to set the name of the printer to PAR in the Psion and other programs which have internal configuration for the various types of printers. Some other programs may require the name of the printer device to be set to PARC for use with most daisywheel printers.

It is possible to make the parallel printer port use a large buffer within the QL to make printing more efficient. Printers are slow devices; to avoid waiting for the printer to finish, particularly with long documents, it is advisable to use a buffer. The buffer is an area of memory set aside to store files waiting to be printed. Having a large buffer means that the whole print file can be stored and you can proceed with your task while the printing is handled in the background by the printer driver, which automatically handles the print file in the buffer. Several print files can be sent to a buffer without corruption, being queued by the printer driver until the printer is free.

This technique will not work on 128K QLs with the Psion programs which tend to grab all the QL's available memory for themselves, and it is most effective to use with additional memory.

More Power To Your Printer

The QJUMP driver for the parallel printer port provides several advantages over the SER drivers provided with the QL.

Large buffers may be specified to allow efficient printer spooling

A form feed may be created automatically when the channel is closed

While only one channel may be open to the parallel printer port at a time, many complete print files may be held pending without tying up the port

A channel is opened to the parallel printer in exactly the same way as a channel is opened to one of the QL's serial ports. The only differences lie in the name that is used, and the options which are accepted as part of that name.

The specification of the device name is:

PARcf_nk	the 'c' flag is used if a carraige return <CR> is required as the newline character
	the 'f' flag is used if a form feed is required when the channel is closed
	_n is the buffer size in bytes (up to 32767) unless...
	the 'k' flag is used to specify the buffer size in kilobytes

For example:

PAR	parallel printer port with default buffer
PARf_3k	... with form feed at the end of a file and a 3 kilobytes buffer, resets the default buffer length to 3 kilobytes
PARc_400	... with <CR> in place of <LF> as newline and a 400 byte buffer, resets the default length to 400 bytes

The initial default buffer length is 128 bytes.

SER Emulation

There is an additional SuperBASIC command to make it possible to get the benefits of the parallel printer port without any need to change existing programs that use one of the QL's SER ports for printer output. This is the PAR_USE command.

```
PAR_USE SER
```

will make the parallel printer port recognise the device name 'SER' in addition to the device name PAR. The normal SER options of port number, parity, handshaking, and protocols are ignored, while the PAR options are recognised. In particular the default buffer length, set when a PAR channel is opened with a specified buffer length, is used for the pseudo-SER device.

For example:

```
OPEN #3,PAR_10K: CLOSE #3: REMark   reset default buffer
PAR_USE SER
....
....
OPEN #3,SER1C
```

will open a channel to the parallel printer port with a 10 kilobyte buffer, and <CR> in place of <LF>, in the same way as the command:

```
OPEN #3,PARC_10K
```

This facility is disabled by re-specifying the PAR_USE to exclude the name SER. This will return the serial ports to their normal usage:

```
PAR_USE PAR
```

Multiple Buffering

To illustrate the multiple buffer capability of the printer driver, connect a printer and set it 'off line', then type in and run the following program:

```
100 FOR try=1 TO 4
110   OPEN #3,PARf_2K
120   FOR lno=1 TO 50: PRINT #3,'Line ';lno;' of try ';try
130 END FOR try
140 CLOSE #3
```

When the printer is turned 'on line' the printer should print each try in turn, showing the way in which the multiple outputs are queued, rather than confused, inside the parallel port driver.

SUPERBASIC EXTENSIONS

For SUPERQBOARDS the ToolKit II extensions are linked by the command:

TK2_EXT <enter>

For SUPERDISK interfaces the following extensions add extra commands or improve existing commands - for a full list of extensions type:

EXTRAS <enter>

SPL SPL_USE - File Spooler

The SPL command sets up a job to copy a file. Only the source needs to be given, the destination may be defaulted. The source file has the default set up by the DATA_USE command. As supplied, the default destination is SER. The SuperBASIC interpreter will continue after the Job has been set up, with the file being copied as a background task. SPL differs from COPY not only in that it operates as a job in the background, but also in its handling of file headers.

The COPY procedure copies both the file and the file header; to copy a file to the printer, the variation COPY_N is used to copy without the header. SPL will, however, not copy the header from an ordinary data file, but it will copy the header of a file which is one of the special types (e.g. executable program file). Furthermore, when using SPL to copy from file to file, if the destination already exists it will be overwritten. The command syntax is:

SPL source_file or

SPL source_file TO destination

The source and destination files may be given as names, or as a SuperBASIC channel number (e.g. 3).

The default set by the DATA_USE command is used to find the source file, and there is a special command, SPL_USE, to set up the default destination. The default destination device or directory may be up to 32 characters long.

SPL_USE device_name or

SPL_USE directory_name_

A device_name does not end in '_'. A directory_name must end in '_'.

If the SPL command is given only with one parameter (the source filename) the output file (or device) will be derived from the current default set by SPL_USE as follows:

1. directory_name & source_filename or
2. device_name

If the SPL command is given with two parameters, the output file (or device) will be derived as follows:

1. destination_filename or
2. directory_name & destination_filename

SPL will often be used to copy files in the background, but it can be used as a true spooler when used with the default output device. In this case if the output device is in use, the SPL job will suspend itself until the device is available.

SPL examples

SPL myfile	using the supplied defaults this will spool FLP2_myfile to SER
SPL flp1_demo_myfile TO ser2	the file FLP1_demo_myfile will be spooled to SER2
DATA_USE flp2_demo SPL_USE ser2 .... SPL myfile	this will also spool the file FLP2_demo_myfile to SER2
SPL mdv2_myfile,mdv1_myfile	does the obvious
SPL_USE mdv1_ .... SPL myfile	using the supplied DATA_USE default, this will also spool FLP2_myfile to MDV1_myfile
SPL myfile TO #3	will spool myfile to the file or device already opened as #3

JOBS AJOB SPJOB RJOB - JOB CONTROL

As QDOS is a multi-tasking operating system, it is possible to have, at one time, in the QL a number of competing or co-operating jobs. Jobs compete for resources in line with their priority, or co-operate using pipes or shared memory to communicate. The basic attributes of a job are its priority and its position within the tree of jobs (ownership). A job is identified by two numbers: one is the job number which is an index into the table of jobs, and the other is a tag which is used to identify a particular job so that it cannot be confused with a previous job occupying the same position in the job table. Within QDOS the two numbers are combined into the job ID which is job number + tag*65536. For these job control routines, where job_id is a parameter of one of the job control routines, it may be given either as a single number (the job ID, as returned from OJOB or NXJOB of the QL Toolkit) or as a pair of numbers (job number, job tag). Thus the single parameter 65538 (2+1*65536) is equivalent to the two parameters 2,1.

JOBS is a command to list all the jobs running on the QL at the time. If there are more jobs in the machine than can be listed to the output window then the procedure will freeze the screen (CTRL F5) when it is full. The procedure may fail if a job is removed from the QL while the procedure is listing them. The following information is returned for each job:

job number and the job tag
job owner's job number
flag 'S' if the job is suspended
job priority
job (or program) name

The syntax of the JOBS command is:

```
JOBS                    list jobs to window #1
JOBS #channel           list jobs to a given channel
```

There are also three procedures for controlling jobs in the QL:

```
AJOB job_id,priority        activates a job
SPJOB job_id,priority       sets a job's priority
RJOB job_id,error_code      removes a job from the QL
```

If there is a job waiting for the completion of a job removed by RJOB, it will be released with D0 set to error_code, e.g.

```
RJOB 2,1,0              remove job 2 (tag 1) with no error
```

GET BGET PUT BPUT FPOS - Direct Access Files

In QDOS, files appear as a continuous stream of bytes. On directory devices, such as disk drives or microdrives, the file pointer can be set to any position within a file. This provides direct access to any data stored in the file. Access implies both read and, if the file is not open for read only, write access (OPEN_IN FROM SUPERBASIC/IO.SHARE IN QDOS). Parts of a file as small as a byte may be read from, or written to any position within a file. QDOS does not impose any fixed record structure upon files - applications may provide these if they wish.

Procedures are provided for accessing single bytes, integers, floating point numbers and strings. There is also a function to find the current file position.

To keep files tidy there is a command to truncate a file, when information at the end of a file is no longer required, and a command to flush the file buffers.

A direct access input or output (I/O) command specifies the I/O channel, a pointer to the position in the file for the I/O operation to start and a list of items to be input or output:

```
command #channel\position,items
```


It is usual (although not essential - the default is #3) to give a channel number for the direct I/O commands. If no pointer is given, the routines will read or write from the current file position, otherwise the file position is set before processing the list of I/O items; if the pointer is a floating point variable rather than an expression, then, when all the items have been read from or written to the file, the pointer is updated to the new current file position. If no items are given then nothing is written to or read from the file. This can be used to position a file for use by other commands (eg. INPUT for formatted input).

Byte Input/Output (I/O)

```
BGET #channel\position,items     get bytes from a file
BPUT #channel\position,items     put bytes onto a file
```

BGET gets 0 or more bytes from the channel. BPUT puts 0 or more bytes to a channel. For BGET, each item must be a floating point or integer variable; for each variable a byte is fetched from the channel. For BPUT, each item must evaluate to an integer between 0 and 255; for each item a byte is sent to the output channel.

For example the statements:

```
abcd=2.6
zz%=243
BPUT #3,abcd+1,'12',zz%
```

will put the byte values 4,12 and 243 after the current file position on the file open on channel#3.

Provided no attempt is made to set a file position, the direct I/O routines can be used to send unformatted data to devices which are not part of the filing system. If, for example, a channel is opened to an EPSON/SINCLAIR compatible printer (channel #3) then the printer may be put into condensed underline mode by either:

```
BPUT #3,15,27,45,1                    or
PRINT #3,chr$(15);chr$(27);'-';chr$(1);
```

BPUT is much more compact.

Unformatted Input/Output (I/O)

It is possible to put or get values in their internal form. The PRINT and INPUT statements of SuperBASIC handle formatted I/O, whereas the direct I/O routines GET and PUT handle unformatted I/O. For example, if the value 1.5 is PRINTed, the byte values 49 ('1') 46 ('.') 53 ('5') are sent to the output channel.Internally 1.5 is represented by 6 bytes (as are all floating point numbers) with values 08 01 60 00 00 00 (hex) if the value is PUT; these 6 bytes are sent to the output channel.

The internal form of an integer is 2 bytes (most significant byte first). A floating point number is a 2 byte exponent to base 2 (offset by HEX 81F), followed by a 4 byte mantissa, normalised so that the most significant bits (bits 31 and 30) are different. The internal form of a string is a 2 byte positive integer, holding the number of characters in the string, followed by the characters.

```
GET #channel\position,items     get internal format data from a file

PUT #channel\position,items     put internal format data onto a file
```

GET gets data in internal format from the channel.
PUT puts puts data in internal format into the channel.

For GET each item must be an integer, floating point or string variable Each item should match the type of the next data item from the channel.

For PUT the type of data put into the channel is the type of the item in the parameter list.

```
fpoint=54
.....
zz%=42:salary=78000:name$='Smith'
PUT #3\fpoint,zz%,salary,name$
```

position the file open on #3 to the 54th byte
put 2 bytes (integer 42)
put 6 bytes (floating point 78000)
put 5 characters ('Smith')
fpoint will be set to 69 (54+2+6+2+5)

For variables or array elements the type is self evident, while for expressions there are some tricks which can be used to force the type:

```
.... +0              will force floating point type
.... &''             will force string type
.... ||0             will force integer type

xyz$='ab258.z'
....
PUT #3\37,xyz$(3 to 5)||0
```

position the file open on #3 to the 37th byte
*put 2 bytes (integer 258) - (value 1 and 2, ie. 1*256+2)*

File Position

There is one function to assist in direct access I/O:
FPOS returns the current file position for a channel. Syntax is:

```
FPOS (#channel)          Find the file position
```

For example:

```
PUT #4\102,value1,value2
ptr=FPOS(#4)
```

will set 'ptr' to 114 (102+6+6).

The file pointer can be set by the commands BGET, BPUT, GET or PUT with no items to be got or put. If an attempt is made to put the file pointer beyond the end of the file, the file pointer will be set to the end of the file and no error will be returned. Note that setting the file pointer does not mean that the required part of the file is actually in a buffer, but that the required part of the file is being fetched. In this way, it is possible for an application to control prefetch of parts of a file where the device driver is capable of prefetching.

FLEN FTYP FDAT - File Enquiry Functions

There are three functions to extract information from the header of a file. Note that in current versions of the Microdrive handler, the header is only updated on an FS.HEADS call or on closing the file, the QJUMP Floppy Disk Driver also updates the header on a call to flush the disk buffers. This means that the file length read from the header will usually be the file length as it was when the file was opened.

If a file is being extended, the file length can be found by using the FPOS function to find the current file position. If necessary the file pointer can be set to the end of the file by the command:

```
GET #n\99999
```

```
FLEN(#n)          returns the file length

FTYP(#n)          returns the file type (0=normal, 1=EXEC)

FDAT(#n)          returns the data space for EXEC files

OPEN #3,mdv1_fred
PRINT FLEN(#3)    Prints the length of file fred on mdv1_
```

FOPEN FOP_IN FOP_NEW FOP_OVER FOP_DIR - File Open Functions

This is a set of functions for opening files which differ from the OPEN procedures in ROM in two ways:

[1] If a file system error occurs ('not found' or 'already exists') these functions return the error code and continue.

[2] The functions use the DATA_USE directory default.

FOPEN (#channel,name)	open for read/write
FOP_IN (#channel,name)	open for read only
FOP_NEW (#channel,name)	open a new file
FOP_OVER (#channel,name)	open a new file or overwrite old file
FOP_DIR (#channel,name)	open a directory

Directory enquiries may be read using GET to get information. Each entry is 64 bytes long, the length of the file is at the start of the entry, there is a standard string starting at the 14th byte of the entry giving the filename and the update date is a long integer starting at the 56th byte.

A file may be opened for read only with an optional extension using the following code:

```
ferr=FOP_IN(#3,name$&'_ASM')      : REMark try to open _ASM file
If ferr=-7: ferr=FOP_IN(#3,name$): REMark ERR.NF, try without _ASM
```

VIEW - Examining a file

VIEW is a procedure intended to allow a file to be examined in a window in the QL display:

VIEW name	view a file (in #1): lines are truncated to fit in the window and, when the window is full, CTRL F5 is generated
VIEW #window,name	view a file in a given window with lines truncated to the window's width

The VIEW command uses the DATA_USE directory default, which will be used whenever a specific device is not named.

RENAME and TRUNCATE

The RENAME and TRUNCATE procedures operate on files on floppy disks and RAM disks but only on Microdrives if the SuperToolkit is being used. The standard microdrive driver as a bug which returns a 'bad parameter'

RENAME old name,new name	renames a file using the DATA_USE default for both filenames
TRUNCATE #n	truncates the file open on #n to the current file position

Wild Card Names

A wild card name is a special type of filename where part of the name is treated as a wild card which can be substituted by any string of characters. If the wild card name used is a normal SuperBASIC name, then special characters cannot be used for the wild card: (e.g. myfiles_*_asm would be treated by SuperBASIC as an arithmetic expression and SuperBASIC would attempt to multiply myfiles_ by _asm).

For this reason a simpler scheme is adopted; any missing section of a file name is treated as a wild card. The end of a wild card is implicitly missing. If the wild card name does not include a directory name, the default directory name is added to the start of the name.

In the following example, the default directory is assumed to be FLP2_

Wild Card Name	Full Wild Card Name	Typical matching files
fred	flp2_fred	flp2_fred flp2_freda_list
_fred	flp2_fred	flp2_fred flp2_freda_list flp2_old_fred flp2_old_fred_list
flp1_old_list	flp1_old_list	flp1_old_jo_list flp1_old_fred_list

WDIR - Wild Card Directory Listing

Uses wild card names and the DATA_USE default directory. The output is sent to channel #1 by default; but a channel or implicit channel may be specified; if the output channel is to a window the listing is halted (CTRL F5) when the window is full and waits for a key to be pressed:

WDIR #channel,name	list of files, stop when window full

The channel specification and the name are optional.

WDIR	list directory to #1
WDIR #channel	list directory to #channel
WDIR \name	list directory to 'name'
WDIR name	list directory 'name' to #1
WDIR #channel,name	list directory 'name' to #channel
WDIR \name1,name2	list directory of 'name2' to 'name1'

For example:

WDIR \ser,_asm	list all _asm files in current directory to SER
WDIR flp1_	list all files on flp1_ in window #1
WDIR #3	list all files in current directory to #3

STAT WSTAT - Drive & File Statistics

There are two commands to print the statistics of the drive holding a current directory, or the data default directory:

```
STAT                    list statistics of current drive
STAT #2,mdv1_           list statistics of mdv1_ to channel #3

WSTAT #channel,name     list file name, length, update date
WSTAT\name1,name2
```

Both the channel number and the name are optional.

WDEL WDEL_F - Wild Card File Deletion

When using WDEL each file name is displayed in the chosen channel and the user is requested to confirm deletion:

```
WDEL                    deletes files from current directory
                        (requesting confirmation)
```

Confirmation is requested by one of four responses:

```
Y  (YES)     delete this file
N  (NO)      do not delete this file
A  (ALL)     delete this and all subsequent matching files
Q  (QUIT)    do not delete any further files and exit command
```

When using WDEL_F take great care - you will not be asked to confirm the deletion!

```
WDEL_F       forced file deletion
```

WCOPY - Wild Card File Copying

Wild card copying is fast and accurate, no more retyping lines because of a minor keying error. The command has several optional forms:

```
WCOPY  source wild name TO destination wild name
WCOPY  source wild name,destination wild name
WCOPY  #channel,source wild name TO destination wild name
WCOPY  #channel,source wild name,destination wild name
```

If no channel is specified the dialogue will be in #1.

Confirmation is requested by one of four responses:

```
Y  (YES)     copy this file
N  (NO)      do not copy this file
A  (ALL)     copy this and all subsequent matching files
Q  (QUIT)    do not copy any further files and exit command
```


If the destination file already exists, the user is prompted:

```
OVERWRITE?
```

Confirmation is again Y,N,A,Q as above.

Wcopy may be used to copy whole directories. The destination name is made up from the actual source file name and the destination wild name. If a missing section of the source wild name is matched by a missing section of the destination wild name, then that part of the actual source file name will be used as the corresponding part of the actual destination name. Otherwise, the actual destination file name is taken from the destination wild name. If there are more sections in the destination wild name than in the source wild name, these extra sections will be inserted after the drive name, and vice versa.

For example, if the default data directory is flp2_ :

```
WCOPY flp1_,flp2_              copy all files from flp1_ to flp2_

WCOPY fred,mog                 would copy
  flp2_fred                 to flp2_mog
  flp2_freda_list              flp2_moga_list

WCOPY _fred,_mog
  flp2_fred                    flp2_mog
  flp2_freda_list              flp2_moga_list
  flp2_old_fred                flp2_old_mog
  flp2_old_freda_list          flp2_old_moga_list

WCOPY _list,old_list
  flp2_jo_list                 flp2_old_jo_list
  flp2_freda_list              flp2_old_freda_list

WCOPY old_list,flp1_list
  flp2_old_jo_list             flp1_jo_list
  flp2_old_freda_list          flp1_freda_list
```

And so on with infinite variations!

EXTRAS - Listing All Extensions

```
EXTRAS                  lists all extra procedures & functions

EXTRAS#channel          lists to a specified channel
```

The SuperBASIC interpreter is extendable. The procedure EXTRAS may be used to list all extra procedures and functions which have been linked into the interpreter. When the window is full press any key to continue the listing.

CLOCK - Resident Clock

There are a number of optional forms for the CLOCK command:

CLOCK	default clock, 2 rows of 10 characters in default position
CLOCK #channel	default clock in defined channel
CLOCK string	user defined clock in default position
CLOCK #channel,string	user defined clock in defined channel

CLOCK is a procedure to set up a resident digital display. If no window is specified the default is the top RHS of the monitor mode default channel 0. This window is 60 X 20 pixels and is only suitable for four colour mode. The CLOCK may be invoked to execute within a window set up by BASIC. In this case the clock job will be removed when the window is closed.

The string is used to define the characters written to the clock window; any character may be written except $ or %. If a dollar sign is found in the string then the next character is checked and:

$d or $D will insert the 3 characters of the day of the week
$m or $M will insert the 3 characters of the month

If a percentage sign is found:

%y or %Y will insert the 2 digit year
%d or %D will insert the 2 digit day of the month
%h or %H will insert the 2 digit hour
%m or %M will insert the 2 digit minute
%s or %S will insert the 2 digit second

The default string is '$d %d $m %h/%m/%s' a newline should be forced by padding out a line with spaces until the right hand margin of the window is reached.

For example:

```
MODE 8
OPEN #6,'scr_156x10a32x16'
INK #6,0: PAPER #6,4
CLOCK #6,'QL time %h:%m'
```

The clock should be set using the normal SDATE command.

INDEX

A: AJOB 22
AUTO-BOOT 8

B: BGET 23
BPUT 23
BUFFER - PRINTER 19

C: CLOCK 30

D: DIRECT ACCESS FILES 22
DIRECT SECTOR ACCESS 12

E: EXTENSIONS 20
EXTRAS 29

F: FDAT 25
FITTING INSTRUCTIONS 4
FLEN 25
FLOPPY DISK DRIVER 7
FLP_SEC 9
FLP_START 9
FLP_TRACK 9
FLP_USE 8
FOP_DIR 26
FOP_IN 26
FOP_NEW 26
FOPEN 26
FOP_OVER 26
FPOS 25
FTYP 25

G: GET 24

H: HEAP FRAGMENTATION 15

I: INDEX 31

J: JOB CONTROL 21
JOBS 22

M: MICRODRIVE EMULATION 8

P: PAR_USE 17
POWER ON SEQUENCE 4
PRINTER PORT 17
PUT 24

R: RAM DISK DRIVER 14
RAM_USE 15
RENAME 15
RJOB 22

S: SECURITY LEVELS 10
SPJOB 22
SPL 2
SPOOLER 20
STAT 28

T: TK2_EXT 20
TRUNCATE 26

V: VIEW 26

W: WCOPY 28
WDEL 28
WDEL_F 28
WDIR 27
WILD CARD 27
WSTAT 28