Tandy Portable Disk Drive (PDD1 and PDD2) Command Reference

General Reference

- Disks are single sided 3.5 inch standard disks
- PDD1 - 40 tracks 2 sectors 1280 bytes/sector (100K/disk)
- PDD2 - 80 tracks 2 sectors 1280 bytes/sector (200K/disk treated as two banks of 100k each)
- Maximum file size = 64k.
- Maximum number of directory entries (files) is 40 for PDD1 and 80 for PDD2.
- File names are maximum 24 characters (padded with trailing blanks) although Tandy always used 6 for filename and 2 for filetype, with period seperator (i.e. XXXXXX.TT)
- All communications with drive are at 19,200 bps. PDD1 has dip switches so this can drop to 9600bps. PDD2 is auto sensing down to 1200bps (??? - can anyone confirm this?).

Command Format

All commands are in a request/return format (half-duplex)

General request format:

```
preamble type length data checksum
```

General return format:

```
type length data checksum
```

Command Type

(all values are in HEX)

command	request	return
directory ref	00	11 12
open file	01	12
close file	02	12
read file	03	10 12
write file	04	12

delete file	05	12
format disk	06 *	12
drive status	07 *	12
drive condition	0C *	15
rename file	0D	12

* PDD2 treats the disk as two banks of 100k each. All commands except these two must specify the bank number as part of the request. The above commands reference BANK 0. To reference BANK 1 you must add 40 HEX to these values (bit 6 = bank number). E.G. 'open file' becomes 41 for BANK 1.

Request Command details

Type 00 - Directory Reference

```
   2   1  1  24             1           1           1      bytes
 +----+--+--+--------------+----------+------------+------+
 |5a5a|00|1a| filename     |attribute |search form |chksum|
 +----+--+--+--------------+----------+------------+------+
```

Preamble - always 'ZZ'
request - type 00h
length - length of data 1ah (26 decimal)
filename - padded with blanks
attribute - specify 'F' (not used normally)
search form -

- 00h - reference file for open or delete
- 01h - request first directory block
- 02h - request next directory block
- 03h - request previous directory block
- 04h - end directory reference

checksum - see below for calculating

Type 01 - Open file

```
   2   1  1  1     1      bytes
 +----+--+--+------+------+
 |5a5a|01|01| mode |chksum|
 +----+--+--+------+------+
```

Preamble - always 'ZZ'
request - type 01h
length - length of data 01h (1 decimal)
mode -

- 01h - open new file for WRITE
- 02h - open existing file for APPEND

03h - open existing file for READ

checksum - see below for calculating

Type 02 - Close file

```
   2   1  1  1        bytes
+----+--+--+------+
|5a5a|02|00|chksum|
+----+--+--+------+
```

Preamble - always 'ZZ'
request - type 02h
length - length of data 00h (no data field)
checksum - see below for calculating

Type 03 - Read file

```
   2   1  1  1        bytes
+----+--+--+------+
|5a5a|03|00|chksum|
+----+--+--+------+
```

Preamble - always 'ZZ'
request - type 03h
length - length of data 00h (no data field)
checksum - see below for calculating

Type 04 - Write file

```
   2   1  1   1-128  1       bytes
+----+--+---+------+------+
|5a5a|04|01-|data  |chksum|
|    |  | 80|      |      |
+----+--+---+------+------+
```

Preamble - always 'ZZ'
reques - type 04
length - length of data 01h-80h (actual length of data)
checksum - see below for calculating

Type 05 - Delete file

```
   2   1  1  1        bytes
+----+--+--+------+
|5a5a|05|00|chksum|
+----+--+--+------+
```

Preamble - always 'ZZ'
request - type 05h
length - length of data 00h (no data field)
checksum - see below for calculating

Type 06 - Format Disk

```
   2   1  1  1        bytes
+----+--+--+------+
|5a5a|06|00|chksum|
```

```
 +----+--+--+------+
```

Preamble - always 'ZZ'
request - type 06h
length - length of data 00h (no data field)
checksum - see below for calculating

Type 07 - Drive Status

```
    2   1  1  1         bytes
 +----+--+--+------+
 |5a5a|07|00|chksum|
 +----+--+--+------+
```

Preamble - always 'ZZ'
request - type 07h
length - length of data 00h (no data field)
checksum - see below for calculating

Type 0C - Drive Condition

```
    2   1  1  1         bytes
 +----+--+--+------+
 |5a5a|0C|00|chksum|
 +----+--+--+------+
```

Preamble - always 'ZZ'
request - type 0Ch
length - length of data 00h (no data field)
checksum - see below for calculating

Type 0D - Rename file

```
    2   1  1  24     1        1        bytes
 +----+--+--+-------+------+------+
 |5a5a|0D|19|newname|attrib|chksum|
 +----+--+--+-------+------+------+
```

Preamble - always 'ZZ'
request - type 0D
length - length of data 19h (25 decimal)
newname - new name for the file
attribute- not used (specify 'F')
checksum - see below for calculating

Return Command Details

Type 10 - Read file Return

```
  1  1   0-128        1       bytes
 +--+---+----------+-----+
 |10|00-| file data|cksum|
 |  | 80|          |cksum|
 +--+---+----------+-----+
```

return - type 10
length - length of data 00h-80h (0-128 decimal)

- if length is equal to 80h there may be more data - you must issue another read command
- if length is less than 80h then this is last block

file data - data read from file
checksum - see below for calculating

Type 11 - Directory reference return

```
  1  1   24        1       2    1    1       bytes
 +--+--+--------+------+----+----+-----+
 |11|1c|filename|attrib|size|free|cksum|
 +--+--+--------+------+----+----+-----+
```

return - type 11h
length - length of data 1Ch (28 decimal)
filename - file name - if no name specied is 00H. If at end of directory is 00H.
attribute - not used
size - size of file
free - number of free sectors (multiply by 1280 for bytes)
checksum - see below for calculating

Type 12 - Normal Return

```
  1  1  1       1       bytes
 +--+--+-----+-----+
 |12|01|error|cksum|
 +--+--+-----+-----+
```

return - type 12h
length - length of data 01h
error code-

00 - normal (no error)	10 - file does not exist	11 - file exists
30 - no filename	31 - dir search error	35 - bank error
36 - parameter error	37 - open format mismatch	3f - end of file
40 - no start mark	41 - crc check error in ID	42 - sector length error
44 - format verify error	46 - format interruption	47 - erase offset error
49 - crc check error in data	4a - sector number error	4b - read data timeout
4d - sector number error	50 - disk write protect	5e - un-initilized disk
60 - directory full	61 - disk full	6e - file too long
70 - no disk	71 - disk change error	

checksum - see below for calculating

Type 15 - Drive Condition Return

```
  1  1  1           1       bytes
 +--+--+---------+-----+
 |15|01|condition|cksum|
 +--+--+---------+-----+
```

return - type 15h
length - length of data 01h
condition- bit values

```
       7 6 5 4 3 2 1 0    bit
      +-+-+-+-+-+-+-+-+
  MSB |0|0|0|0|x|x|x|x| LSB
      +-+-+-+-+-+-+-+-+
              | | | |
              | | | +--power (0=normal 1=low)
              | | +----write protect (0=not prot 1=prot)
              | +------disk out (0=disk in 1=disk out)
              +--------disk change status (0=not changed 1=changed)
```

checksum - see below for calculating

Sequence of Events

Get directory	req 00 search form 01 req 00 search form 02 (repeat as needed)
Write file	req 00 search form 00 req 01 mode 01 or 02 req 04 (repeat as needed) req 02
Read file	req 00 search form 00 req 01 mode 03 req 03 (repeat as needed) req 02
Rename file	req 00 search form 00 req 0d
Delete file	req 00 search form 00 req 05

Calculating Checksum

The check sum is "the one's complement of the least significant byte of the number of bytes from the block format through the data block". Most people (me included) don't understand what that involves if you have to calculate it. Fortunately I found an example of how to do this and so I'm passing it on to you.

Checksum=(bytes MOD 256) XOR 255

where bytes = number of bytes including the Request Type, Length and all Data fields (but not including the preamble).

Using this Information

A lot of the above commands can be set up in advance since there is no variable part to calculate. Some commands must have the length and checksum calculate as the data is built but the others don't. Here's how I set up some of the commands in my program:

```
Close$="ZZ"+Chr$(2)+Chr$(0)+Chr$(253)
Dir1$="ZZ"+Chr$(0)+Chr$(26)+Space$(24)+"F"+Chr$(1)+Chr$(158)
Dir2$="ZZ"+Chr$(0)+Chr$(26)+Space$(24)+"F"+Chr$(2)+Chr$(157)
Status$="ZZ"+Chr$(7)+Chr$(0)+Chr$(248)+Chr$(13)
Format$="ZZ"+Chr$(6)+Chr$(0)+Chr$(249)+Chr$(13)
```

```
Erase$="ZZ"+Chr$(5)+Chr$(0)+Chr$(250)
Seek$(1)="ZZ"+Chr$(1)+Chr$(1)+Chr$(1)+Chr$(252)
Seek$(2)="ZZ"+Chr$(1)+Chr$(1)+Chr$(2)+Chr$(251)
Seek$(3)="ZZ"+Chr$(1)+Chr$(1)+Chr$(3)+Chr$(250)
```

The above commands can be sent directly without any calculating to speed up your program.