Please do not upload this copyright pdf document to any other website. Breach of copyright may result in a criminal conviction.

This Acrobat document was generated by me, Colin Hinson, from a document held by me. I requested permission to publish this from Texas Instruments (twice) but received no reply. It is presented here (for free) and this pdf version of the document is my copyright in much the same way as a photograph would be. If you believe the document to be under other copyright, please contact me.

The document should have been downloaded from my website https://blunham.com/Radar, or any mirror site named on that site. If you downloaded it from elsewhere, please let me know (particularly if you were charged for it). You can contact me via my Genuki email page: https://www.genuki.org.uk/big/eng/YKS/various?recipient=colin

You may not copy the file for onward transmission of the data nor attempt to make monetary gain by the use of these files. If you want someone else to have a copy of the file, point them at the website. (https://blunham.com/Radar). **Please do not point them at the file itself as it may move or the site may be updated.**

It should be noted that most of the pages are identifiable as having been processed by me.

I put a lot of time into producing these files which is why you are met with this page when you open the file.

In order to generate this file, I need to scan the pages, split the double pages and remove any edge marks such as punch holes, clean up the pages, set the relevant pages to be all the same size and alignment. I then run Omnipage (OCR) to generate the searchable text and then generate the pdf file.

Hopefully after all that, I end up with a presentable file. If you find missing pages, pages in the wrong order, anything else wrong with the file or simply want to make a comment, please drop me a line (see above).

It is my hope that you find the file of use to you personally – I know that I would have liked to have found some of these files years ago – they would have saved me a lot of time !

Colin Hinson
In the village of Blunham, Bedfordshire.

FILE MANAGEMENT SPECIFICATION

FOR THE

TI-99/4 HOME COMPUTER

Consumer Group
Mail Station 5890
2301 N. University
Lubbock, Texas 79414

TEXAS INSTRUMENTS
INCORPORATED

Date: February 25, 1983
Version 2.5

TABLE of CONTENTS

Paragraph Title

SECTION 4 DSR OPERATIONS

SECTION 5 LINKAGE TO BASIC

LIST of TABLES

LIST of FIGURES

SECTION 1

INTRODUCTION

This specification contains a complete description for the File Management System of the TI-99/4 and 4A Home Computer. This description includes information about File structures, File I/O, the Peripheral Access Block, I/O Opcodes and DSR operations.

SECTION 2

I/O HANDLING

The approach used in the TI-99/4 Home Computer File Management System, is that all devices, with the exception of the screen and the keyboard, look the same to an application program. Therefore, when peripherals are added to the computer, the BASIC Interpreter is not affected. Only the peripheral drivers (Device Service Routines or DSRs) have to be added to the software. Physical limitations (like reading data from a thermal printer which is clearly impossible) are determined in the DSR and are returned to the application program as an error condition.

The 99/4 and 4A personal computer File Management System supports two kinds of file organization:

1. Sequential files
2. Relative record files

Both file types use the same supervisor call mechanism, in order to insure a high degree of device independence. Hardware devices (such as a line printer) are accessed as sequential files, except that no file name is appended to the device name.

2.1 File Organization and Use

The following paragraphs discuss the file organization and use for the two file types.

2.1.1 Sequential Files.

The records in sequential files can only be read from, or written to, in sequential order. This is appropriate for printers, modems, cassettes and some disk-based files. The records in sequential files can be either fixed or variable length.

2.1.2 Relative Record Files.

The records in relative files can be read from, or written to, in either sequential order or in random order. Relative record files are also called random access files, since records may be accessed in an arbitrary order. Therefore relative record files can only be supported on diskettes. The records in relative files are always fixed length. This enables the system to compute the actual location of any logical record relative to the beginning of a file.

The records within a relative record file are addressed by a unique record number. To access record X, the value X has to be placed in the appropriate field of the I/O Peripheral Access Block. Each record has a number from zero up to one less than the number of records in the file.

Records in a relative record file can also be accessed in a sequential way, by specifying the first record in the sequence only. The Supervisor then automatically updates the record number each time after a record has been read.

2.2 File Characteristics

2.2.1 Terminology.

A file consists of a collection of data groupings called logical records. These records do not necessarily correspond with the physical divisions of the data in the file (like a sector on a disk). Thus, there are two types of records:

1. Logical records - The data grouping of a file as seen by an application program.

2. Physical records - The buffers physically transferred between memory and medium.

File I/O is done on a logical record basis. Manipulation of physical records is handled by the DSR.

When a file is created, its characteristics must be defined. Most of these characteristics cannot be changed later in the file's existence. The logical record size is an attribute which must be specified. For relative record files this size must be

exact. For sequential files the specification indicates an upper limit for the record size. In case a zero is specified for either filetype, the DSR must select a default for the record size. The physical record size for any medium is specified within the DSR and is implementation dependent.

2.2.2 File Type Attribute.

The file type attribute specifies the format in which the data in the file is represented. The two file types are:

1. DISPLAY - Displayable or printable character strings. Each data record corresponds to one print line.

2. INTERNAL - Data in INTERNAL machine format.

The file type attribute is internal to the application program. It is merely stored and passed on by the DSR as a distinction between two data types, without affecting the actual data stored.

2.2.3 Mode of Operation.

A file is opened for a specific mode of operation, specified in the OPEN I/O call. The four modes of operation are:

1. INPUT - The contents of the file may be read, but may not be altered.

2. OUTPUT - The file is being created. Its contents may be written but not read.

3. UPDATE - The contents of the file may both be written and read. Note that this mode of operation will generally only be supported by random access file structured devices or non-file structured devices.

4. APPEND - New data may be added at the end of the file, but the contents of the file may not be read.

Each DSR decides whether or not a specific mode for an I/O operation can be accepted by the corresponding device. For example, the TI Thermal Printer can only be opened in OUTPUT mode.

2.2.4 Temporary Files.

In the subsets of TI standard BASIC used for the 99/4 and 4A Home Computer, the file-life attribute is not implemented. Therefore the File Management System does not support temporary files, so all files are permanent by definition.

SECTION 3

IMPLEMENTATION

As mentioned in section 2, the DSRs should present a uniform interface between the File Management System and the peripherals. This section will give implementational details on this interface. Some remarks are being made on a possible implementation of the file system for random access devices. However, no details are given for such an implementation.

3.1 Peripheral Access Block Definition

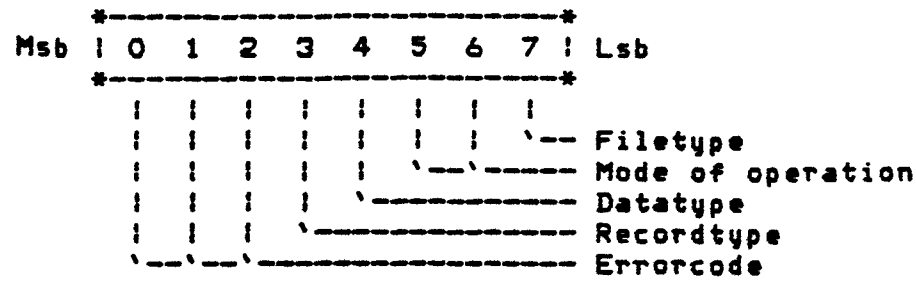
All DSRs are accessed through a Peripheral Access Block (PAB). The definition for these PABs is the same for every peripheral. The only difference between peripherals, as seen by any application program, is that some peripherals will not support every option provided for in the PAB.

All PABs are physically located in VDP RAM. They are created before the OPEN call, and are not to be released until the I/O has been closed for that device or file.

Figure 3-1 shows the layout of a PAB. The PAB has a variable length, depending upon the length of the file descriptor. The meaning of the data within the PAB is explained below.

Byte	Bit	Meaning
0	All	I/O opcode - Contains opcode for the current I/O-call. A description of the valid opcodes will be given in section 3.2.
1	All	Flag/Status - File-type, mode of operation and data-type is stored in this byte. The meaning of the bits within this flagbyte is:

Byte	Bit	Meaning



1	0-2	Errorcode - These three bits indicate, in combination with the I/O opcode, the error type that has occurred (0 = no error). See section 3.4 for error codes.
	3	Recordtype - Indicates type of record used. 0 = Fixed length records 1 = Variable length records
	4	Datatype - Indicates type of data stored in the file. 0 = DISPLAY type data 1 = INTERNAL type data
	5,6	Mode of operation - Indicates operation mode file has been opened for. 00 = UPDATE 01 = OUTPUT 10 = INPUT 11 = APPEND
	7	Filetype - Indicates file-type. 0 = Sequential file 1 = Relative record file
2,3	All	Data buffer address - Address of the data buffer in VDP RAM the data has to be written to or read from.
4	All	Logical record length - Indicates the logical record length for fixed length records, or the maximum length for a variable length record (see flagbyte).
5	All	Character count - Number of characters to be transferred for a WRITE opcode, or the number of bytes actually read for a READ opcode.

Byte	Bit	Meaning
6,7	All	Record number - Only required if the file opened is of the relative record type. Indicates the record number the current I/O operation is to be performed upon (this limits the range of record-numbers to 0 - 32767). The highest bit will be ignored by the DSR.
8	All	Screen offset - Offset of the screen characters in respect to their normal ASCII value. (Normally >60 while a BASIC is running, >00 otherwise.) This byte is used by cassettes, disks and the RS232.
9	All	Name length - Length of the file descriptor following the PAB.
10+	All	File descriptor - The device name and, if required, the filename and options. The length of this descriptor is given in byte 9.

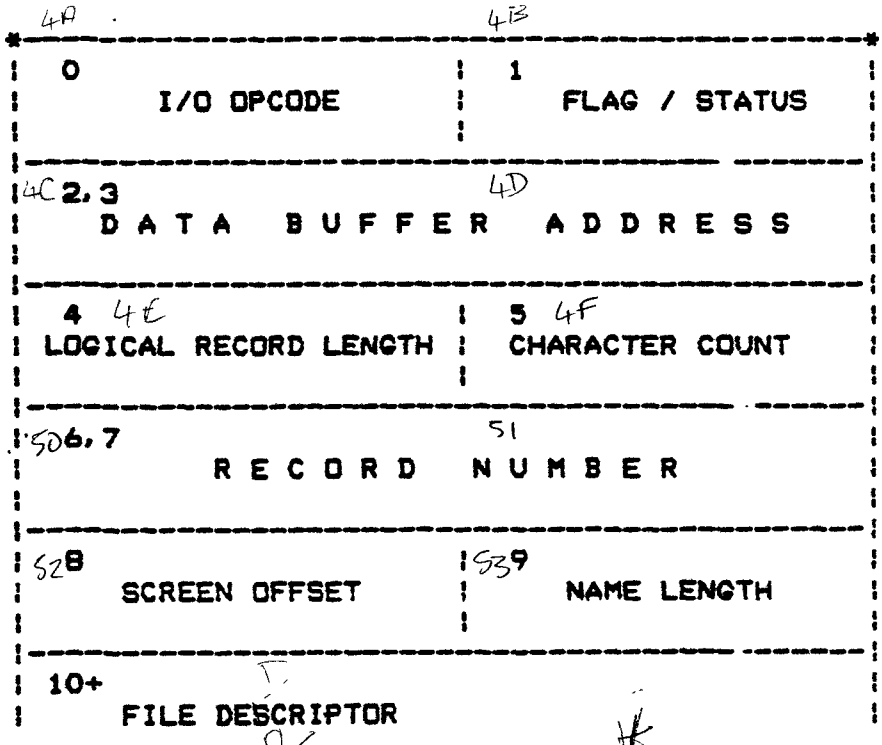


Figure 3-1 PAB Layout

3.2 I/O Opcodes

This paragraph describes the valid opcodes that can be used in a PAB. Valid opcodes are shown in Figure 3-2.

Opcode	Meaning
00	OPEN
01	CLOSE
02	READ
03	WRITE
04	RESTORE/REWIND
05	LOAD
06	SAVE
07	DELETE
08	SCRATCH RECORD
09	STATUS

Figure 3-2 I/O Opcodes

The following describes the general actions invoked by an I/O-call with each of the I/O-opcodes. Each I/O-call returns any error-codes in the Flag/Status byte of the PAB.

3.2.1 OPEN.

The OPEN operation should be performed before any data-transfer operation except those performed with LOAD or SAVE. The file remains open until a CLOSE operation is performed. The mode of operation for which the file is to be opened must be indicated in byte 1 (Flag/Status) of the PAB. In case this mode is UPDATE, APPEND or INPUT, and a record length of zero is given in byte 4 (Logical Record Length), the assigned record length (which depends on the peripheral) is returned in byte 4. If a non-zero record length is given, it is used after being checked for correctness with the given peripheral. For OUTPUT, the record length can be specified, or a default can be used by specifying record length zero.

For any device, an OPEN operation must be performed before any other I/O operation. The DSR need only check the record length and I/O mode on an OPEN. Changing I/O modes after an OPEN may cause unpredictable results.

3.2.2 CLOSE.

The CLOSE operation closes the file. It informs the DSR that the current I/O sequence to that DSR has been completed. If

the file or device was opened in OUTPUT or APPEND mode, an End Of File (EOF) record is written to the device or file before deallocating the PAB. After the CLOSE operation, the space allocated for the PAB may be used for other purposes. As long as a PAB is connected to an active device, the contents of that PAB must be preserved.

3.2.3 READ.

The READ operation reads a record from the selected device and copies the bytes into the buffer specified in bytes 2 and 3 (Data Buffer Address) of the PAB. The size of the buffer is specified in byte 4 (Logical Record Length) of the PAB. The actual number of bytes stored is specified in byte 5 (Character Count) of the PAB. If the length of the input record exceeds the buffer size, the remaining characters are discarded.

3.2.4 WRITE.

The WRITE operation writes a record from the buffer specified in bytes 2 and 3 (Data Buffer Address) of the PAB to the specified device. The number of bytes to be written is specified in byte 5 (Character Count) of the PAB.

3.2.5 RESTORE/REWIND.

The RESTORE/REWIND operation repositions the file READ/WRITE pointer either to the beginning of the file, or, in the case of a relative record file, to the record specified in bytes 6 and 7 (Record Number) of the PAB. This operation can only be used if the file was opened in INPUT or UPDATE mode. For relative record files, a RESTORE can be simulated in any I/O mode by specifying the record at which the file is to be positioned in bytes 6 and 7 (Record Number) of the PAB. The next I/O operation then automatically uses the indicated record.

3.2.6 LOAD.

The LOAD operation loads an entire memory image of a file from an external device or file into VDP RAM. All the control information the application program needs should be concatenated to the program image. Since no intermediary buffers are used, the LOAD operation requires as much buffer in VDP RAM as the file occupies on the diskette or other device. The entire memory image is dumped starting at the specified location.

The LOAD operation is a stand alone operation, i.e. the LOAD operation is used without a previous OPEN operation.

For this operation, the PAB needs to contain only the following information:

Byte 0 : I/O opcode.

Bytes 2,3 : Start address of the VDP RAM memory dump area.

Bytes 6,7 : Maximum number of bytes to be loaded.

Byte 9 : Name length.

Bytes 10+ : File descriptor information.

3.2.7 SAVE.

SAVE is the complementary operation for LOAD. It writes memory images from VDP RAM to a peripheral. The SAVE operation is used without a previous OPEN operation. It copies the entire memory image from the buffer in VDP RAM to the diskette or other device. All necessary control information should be linked to the memory image, so that the information plus program image use one contiguous memory area. Again, only a small part of the PAB is used. The PAB contains:

Byte 0 : I/O opcode.

Bytes 2,3 : Start address of the VDP RAM memory area.

Bytes 6,7 : Number of bytes to be saved.

Byte 9 : Name length.

Bytes 10+ : File descriptor information.

3.2.8 DELETE.

The DELETE operation deletes the specified file from the specified peripheral. This operation also performs a CLOSE.

3.2.9 SCRATCH RECORD.

The SCRATCH RECORD operation removes the record specified in bytes 6 and 7 (Record Number) of the PAB from the specified relative record file. This operation causes an error for peripherals opened as sequential files. No device currently supports this operation.

3.2.10 STATUS.

The STATUS operation is used for obtaining information about a file. This information can be examined at any time, although bits 6 and 7 only have meaning if a file has been opened.

To indicate the current status of the file, byte 8 (SCREEN OFFSET) is used. Upon the DSR-call, byte 8 should contain the usual screen characters base address. The DSR can only use this byte, and is guaranteed not to destroy any other entry in the PAB.

The meaning of the bits within byte 8 after return from the DSR is shown in the following table.

Table 3-1 Meaning of byte 8 after return from DSR

Bit	Information
0	If this bit is set, the requested file doesn't exist. If reset, the file does exist. On some devices, such as a printer, this bit is never set since any file could exist.
1	PROTECT flag. If set, the file is protected against modifications. If reset, the file is not protected.
2	Reserved for future use. Fixed to zero in the current peripherals.
3	Data type. If set, the data type is binary (INTERNAL). If reset, the data type is ASCII (DISPLAY) or file is program file.
4	Filetype. If set, the file is a program file. If reset, the file is a data file.
5	Record type. If set, the record type is VARIABLE length. If reset, the record type is FIXED length.
6	Physical end of file. If set, no more data can be written, since the physical limits of the device have been reached. Generally this means an end of medium has been detected on the device.
7	Logical end of file. If set, the file is at the end of its previously created contents. This is usually the case if the file has been opened for APPEND mode. Depending upon the mode of operation for which the file has been opened, data can still be written to the file (APPEND, OUTPUT or UPDATE mode), however, a "read" operation will cause an ATTEMPT TO READ PAST EOF error to occur.

Bits 0 - 5 have meaning even if the file is not open. Bits 6 and 7 only have meaning for files that are currently open, otherwise a zero should be indicated in these two bits.

3.3 Directory Handling

The GROM cartridge containing the DSR for a device that supports files, shall also contain a CATALOG program, which can be used to list the current contents of the medium.

3.4 Error Codes

The File Management System supports a number of error codes. Errors are indicated in bits 0 thru 2 of byte 1 (Flag/Status) of the PAB. The following table shows the possible error codes and their meanings.

Table 3-2 Error Codes and Meanings

Error Code	Meaning
0	BAD DEVICE NAME The device indicated is not in the system.
1	DEVICE WRITE PROTECTED
2	BAD OPEN ATTRIBUTE One or more of the given OPEN attributes are illegal or do not match the file's actual characteristics. This could be: * File type * Record length * I/O mode * File organization
3	ILLEGAL OPERATION Either an issued I/O command was not supported, or a conflict with the OPEN mode has occurred.
4	OUT OF TABLE/BUFFER SPACE The amount of space left on the device is insufficient for the requested operation.
5	ATTEMPT TO READ PAST EOF This error may also be given for non-existing records in a relative record file.
6	DEVICE ERROR Covers all hard device errors, such as parity and bad medium errors.

7 FILE ERROR
Covers all file-related errors like: program/data file mismatch, non-existing file opened for INPUT mode, etc.

An attempt is made to use these error-codes consistently for all 99/4 peripherals. If any deviation from the given codes is necessary, this should be clearly noted in the device's Software Functional Specification.

NOTE

Error code 0 usually indicates that no error has occurred. However, an error code of 0 with the COND bit (bit 2) set in the STATUS byte at address >837C indicates a bad device name.

SECTION 4

DSR OPERATIONS

This section describes how a variety of DSRs should react on the different I/O calls. It also discusses detailed software operations descriptions such as available registers and memory.

4.1 DSR Actions and Reactions

In the Professional/Home Computer File Management System, several assumptions are made about the way in which DSRs should react on conditions like errors, special I/O modes, defaults etc. This section is intended to explain the reactions of a DSR on these conditions.

4.1.1 Error conditions.

4.1.1.1 Non-existing DSRs.

If a non-existing DSR is called by an application program, the File Management System will automatically return with the COND bit set. In this case, no DSR has actually been called, so the error-code will show no errors.

The DSR search mechanism of the File Management System takes care of searching for the requested DSR. It tries to match the file descriptor to the DSR entries in the system. The matching algorithm matches up to the end of the descriptor, or up to the first period ("."), whichever comes first. This enables the application program to add special information for the DSR in the file descriptor, like: filename, BAUD-rate, print-width, character set, etc.

4.1.1.2 DSR-detected errors.

DSR-detected error (see section 3.4) should be indicated in the flag-byte of the PAB. It is the application program's responsibility to clear this flag-byte before every I/O-call, and check it after the I/O-call. This type of errors is NOT indicated with the COND bit.

The DSR may provide additional information about the error-type in the I/O opcode byte, although it is good practice not to destroy the least significant 4 bits of this byte, since they specify the I/O call.

At no time should the DSR use bits 0-4 of the flagbyte for error-indication, since these bits might contain vital system information about the file/device.

4.1.2 Special I/O modes.

To enable the application program to use special device-dependent functions, the File Management System DSR search algorithm only uses a well-defined part of the file descriptor for its search (see section 4.1.1). The remainder of the descriptor may be used to indicate special device-related functions such as BAUD-rate, print-width, etc. It is advisable for a DSR to ignore descriptor parts it doesn't recognize, so that the same application program might be used for different devices. In the latter case it would handle specific device-dependent functions only if the device used was capable of performing them, and it would ignore them if the DSR wouldn't recognize them.

An example of a special I/O mode descriptor could be:

```
RS232.BAUDRATE=1200.DATABITS=7.CHECKPARITY.PARITY=ODD
```

4.1.3 Default Handling.

Sometimes, especially if a file is opened for UPDATE, INPUT or APPEND, it is useful to provide a default value for the record length. In the above cases, the application program will usually use the value specified on file creation. Therefore, if the application program does not provide a value for the record length, the DSR should provide this value for it. In case the application program does provide a record length, the DSR should check this value against the value given on record creation. An error should be indicated in case of incompatibilities between stored and provided record length.

The DSR handles the default value, if no value is given, for the record length. TI BASIC supplies default values for other information. These default values are used only if no values are specified. The following shows these defaults.

Table 4-1 TI BASIC Default values

Possibilities	Default
Sequential or relative	Sequential
UPDATE, OUTPUT, INPUT or APPEND	UPDATE
DISPLAY or INTERNAL	DISPLAY
Fixed or variable length	Fixed, if relative and Variable, if sequential
Logical record length	Depends on the peripheral

4.2 Memory Requirements

Because of the limited amount of register memory, 256 bytes of RAM, the register usage for DSRs has to be restricted to the following registers/memory, to avoid interference with application programs:

Registers R0 - R10 of the calling workspace.

If the DSR is called in a non-interrupt driven mode, i.e. through a standard DSR-entry, memory locations >DA through >DF are available.

A standard scratch area of 36 bytes, at locations >4A through >6D, has been assigned for DSR usage.

The base address for CPU memory in the 99/4 Home Computer is >8300. To allow for future changes in this base address, it will be derived from the given value of the workspace pointer. This means the loss of one of the workspace registers for this purpose.

4.3 GPL Interface to DSRs

The GPL interpreter interfaces to DSRs through the monitor. The GPL program that wants to access a DSR, has to use the following GPL CALL sequence:

```
CALL >10
DATA 8
```

This will cause the monitor to start searching for a DSR with the same name as the string pointed to by CPU location >56. This search routine will stop comparing names at the end of the

given string, or at the first imbedded period, whichever comes first.

On the DSR side, CPU location >56 is left pointing at the first character behind the DSR name, i.e. a period or an end of string. CPU location >54 contains the DSR name length (1 word). To get the start address of the PAB in VDP RAM, the following formula has to be computed:

CPU(>56) - CPU(>54) - >0A

The result will point at the I/O OPCODE entry in the PAB.

It is up to the DSR to check for any switches in the name. For this purpose the length of the PAB name string has been given in the PAB. Comparing this length against the length given in CPU location >54 will show if the user specified more than just the DSR name.

SECTION 5

LINKAGE TO BASIC

This section describes the way the BASIC versions of the Professional Computer and the Home Computer are linked to the File Management System.

This section also describes how to access PABs from GPL subroutines that are callable from BASIC and assume a PAB link-structure has been set up by BASIC.

5.1 BASIC PAB modifications

Aside from the control information contained within the PAB, as already discussed in section 3, BASIC adds four (4) more bytes to the top of the PAB for specific BASIC related control information. The new PAB structure within BASIC is drawn in Figure 5-1.

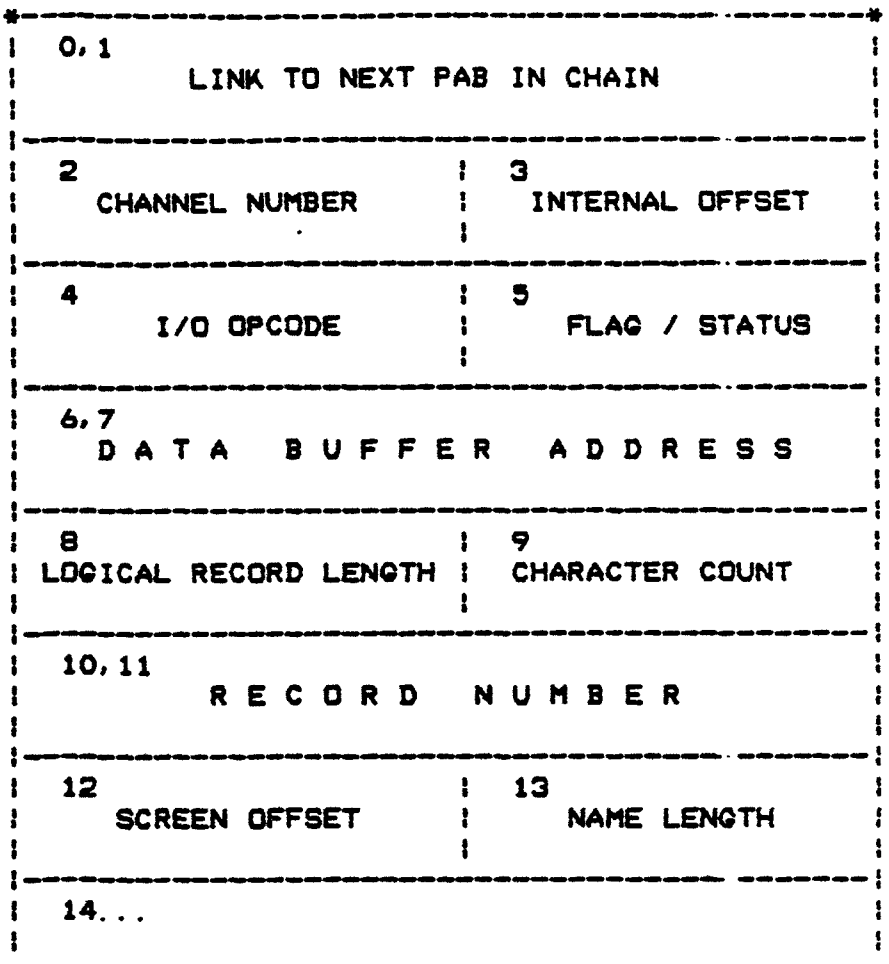


Figure 5-1 Modified PAB Layout

The additional four bytes contain additional control information BASIC needs for its internal PAB linkage structure. The PABs within the BASIC control structure, form a simple linked lists, which means each PAB has a pointer to the next PAB in the system. The last PAB in the list has a zero ("0") link, indicating that it is at the end of the list.

The first 2 bytes in the BASIC PABs contain the link mentioned above, whereas the next two bytes control additional information. Byte 2 contains the actual BASIC channel or file number (1-255); byte 3 contains the offset of the current data-pointer within the data-block.

The offset indicated in byte 3 of the BASIC PAB, indicates the position of the current data pointer within the data buffer given in bytes 6 and 7. If byte 3 equals zero, the current data buffer is "blank", i.e. if we're in "read" mode, a new buffer has to be read in before any further processing; in "write" mode, the entire buffer is still available for data-storage.

If byte 3 is non-zero, it contains an "offset" within the data buffer. Added to the start address of the data buffer, it will give the actual address of the first data-byte to be read or written. This is only the case if we have pending PRINT operations (i.e the most recent PRINT ended on ";" or "," or pending INPUT operations) the most recent INPUT ended on a ",". In all other cases, byte 3 will be zero.

5.2 BASIC PAB Linkage

As mentioned already, BASIC utilizes a simple linked structure for the management of its PABs. Each PAB contains a link to the next PAB in the chain. In order to access the chain, we need to have a link to the first PAB in that chain. This link is given in CPU location >3C for GPL programs, which is equal to location >833C in 9900 assembly language. A graphical representation of how three PABs could be linked in TI BASIC is:

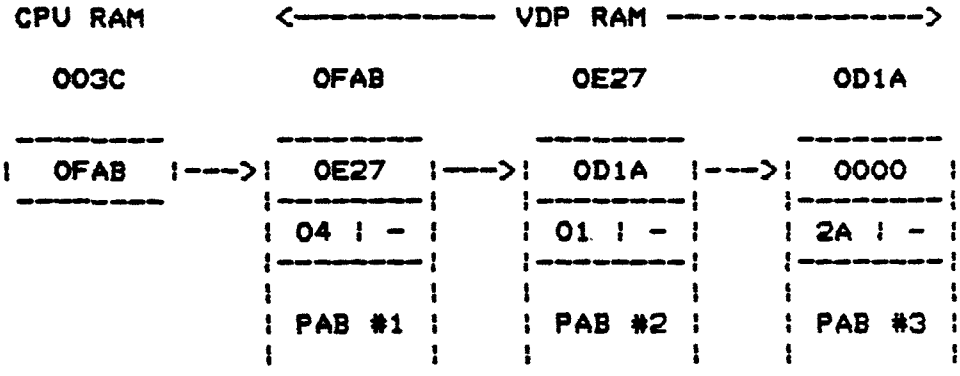


Figure 5-2 BASIC Link Structure

Note that all PABs are located in VDP memory, whereas the initial link to them is located in CPU memory. In addition, although the PABs will usually be allocated in lexical order, depending upon the program, they can be allocated in any arbitrary order. Therefore, the fact that in some applications they happen to be in lexical order according to channel numbers should never be used.