SCIENCE AND ENGINEERING RESEARCH COUNCIL
RUTHERFORD AND APPLETON LABORATORIES
COMPUTING DIVISION

SUN/21.1

Starlink Project
Starlink User Note 21.1

Issued by

D.Pearce
28 September 81

A Set of Routines for Handling Non-standard Magnetic Tapes

This note describes a set of routines for performing physical I/O operations on non-VAX-standard magnetic tapes.

The following routines are available:

TIO_OPEN	-	assigns an I/O channel to a device.
TIO_READ	-	reads a physical block.
TIO_WRITE	-	writes a physical block.
TIO_MARK	-	writes an end-of-file mark.
TIO_SKIP	-	skips a given number of tape marks, (in a forward or reverse direction)
TIO_MOVE	-	moves a tape a given number of blocks, (in a forward or reverse direction)
TIO_REWIND	-	repositions a tape to the beginning-of-tape marker.
TIO_CLOSE	-	deassigns an I/O channel from a device.

All of these routines return a status value which indicates whether the operation was successful or not. Currently, this value is the VAX/VMS System Service return status [1], although this may not be the case in future versions. As a consequence, to facilitate VMS-independent validation of the status value, the following LOGICAL functions are provided:

TIO_EOF	-	tests for a physical end-of-file condition.
TIO_ERR	-	tests for an error condition.

A further routine is provided that returns the message text associated with a particular status value:

TIO_GETMSG	-	returns message text.

Before a process can perform any actual I/O operations on a device, an I/O channel must be assigned through TIO_OPEN. The input argument to this routine is the name of the device on which the tape is mounted. Under VMS, this can be either the physical name of the device, such as MTA0, or any pre-assigned logical name - full recursive translation is performed by TIO_OPEN until the physical name is resolved.

Programmers who wish to use these routines must specify the object module library SYS$SYSDISK:[USERLIB]TAPEIO.OLB on the LINK command. e.g. -

```
$ LINK TAPECOPY,SYS$SYSDISK:[USERLIB]TAPEIO/LIBRARY
```

Note, that all tapes to be used by a process must first be mounted using the FOREIGN qualifier. e.g. -

```
$ ALLOCATE MTA0: INPUT
$ ALLOCATE MTA1: OUTPUT
$ MOUNT/FOREIGN  INPUT
$ MOUNT/FOREIGN  OUTPUT
```

Subroutine TIO_OPEN

This routine assigns an I/O channel to a magnetic tape device.

The program specifies the name of the device on which the tape is mounted - TIO_OPEN returns the channel number assigned, which is used as input to the routines which perform the actual I/O operations.

Format of call:

```
TIO_OPEN(DEVICE,IOCHAN,STATUS)
```

Given (argument):

DEVICE:	CHARACTER expression:	Device name.

Returned (argument):

IOCHAN:	INTEGER variable:	I/O channel assigned to device.
STATUS:	INTEGER variable:	Status return value.

This routine must be called for every device to be used by a program before any I/O operations can be performed. For example:

```
CALL TIO_OPEN('INPUT',INCHAN,STATUS)
CALL TIO_OPEN('OUTPUT',OUTCHAN,STATUS)
```

As mentioned previously, under VMS, the device can be accessed through a logical name - full recursive translation is performed by TIO_OPEN to resolve the physical name. Before any translation is attempted, the name is first stripped of leading and trailing blanks and converted to uppercase if necessary.

To ensure that applications software is as VMS-independent as possible, programmers are advised against passing physical device names explicitly to this routine, that is, as CHARACTER constants thus:

```
CALL TIO_OPEN('MTA1:',CHAN,STATUS)
```

Physical (and logical) names can be passed implicitly by using a CHARACTER variable, whose value is determined at run-time. For example:

```
CHARACTER*63 DEVICE

:    :    :
:    :    :

READ (*,'(A)') DEVICE
CALL TIO_OPEN(DEVICE,CHAN,STATUS)
```

Note that this routine does <u>not</u> reposition the tape to the start of the first file and, as a consequence, no assumptions should be made about the current position.

Subroutine TIO_READ

This routine reads data into a specified buffer starting at the next block position on a tape.

The program specifies the maximum number of bytes (MAXLEN) that are to be transferred from the tape block to the buffer (under VMS, this value must lie within the range 14 to 65535 inclusive - a status value of SS$_BADPARAM is returned if not). TIO_READ returns the actual length of the tape block (ACTLEN), which, if less that MAXLEN, indicates the number of bytes read into the buffer. If the block length is greater than the maximum specified, then only MAXLEN bytes will be transferred (note, that the status return will indicate 'successful completion' and not 'buffer overflow'). In both cases, the tape will be positioned to the end of the physical block when the transfer completes.

Format of call:

```
TIO_READ(IOCHAN,MAXLEN,BUFFER,ACTLEN,STATUS)
```

Given (argument):

IOCHAN:	INTEGER expression:	I/O channel assigned to device.
MAXLEN:	INTEGER expression:	Maximum length of transfer in bytes.

Returned (argument):

BUFFER:	array:	Destination buffer.
ACTLEN:	INTEGER variable:	Actual length of block in bytes.
STATUS:	INTEGER variable:	Status return value.

The destination buffer can be an array of any 'type' appropriate to the format of the data being processed. For character-type data, a BYTE (or LOGICAL*1) array can be used. Under VMS FORTRAN, a CHARACTER variable (or array) can also be specified, but must be passed to TIO_READ explicitly 'by reference' [3], for example:-

```
CHARACTER*80  CARD

:   :   :
:   :   :

CARD=' '
CALL TIO_READ(IOCHAN,80,%ref(CARD),ACTLEN,STATUS)
```

Numeric data, such as 16 or 32-bit integers, can be read into INTEGER*2 or INTEGER*4 arrays, but users should note that the byte order may need to be reversed when reading tapes produced on non-DEC equipment.

Subroutine TIO_WRITE

This routine writes data from a specified buffer starting at the next block position on a tape.

The program specifies the number of bytes that are to be transferred from the buffer to the tape (under VMS, this value must lie within the range 14 to 65535 inclusive - a status value of SS$_BADPARAM is returned if not).

Format of call:

```
TIO_WRITE(IOCHAN,BUFFER,LENGTH,STATUS)
```

Given (argument):

IOCHAN:	INTEGER expression:	I/O channel assigned to device.
BUFFER:	array:	Source buffer.
LENGTH:	INTEGER expression:	Length of transfer, in bytes.

Returned (argument):

STATUS:	INTEGER variable:	Status return value.

The source buffer can be an array of any 'type' appropriate to the format of the data being written. Under VMS FORTRAN, a CHARACTER variable (or array) can be specified, but must be passed to TIO_WRITE explicitly 'by reference' (see TIO_READ).

Subroutine TIO_MARK

This routine writes an end-of-file mark at the current position on the tape.

An error condition occurs if the tape is currently positioned past the end-of-tape marker or if the EOT region is entered as a result of the operation.

Format of call:

TIO_MARK(IOCHAN,STATUS)

Given (argument):

IOCHAN:	INTEGER expression:	I/O channel assigned to device.

Returned (argument):

STATUS:	INTEGER variable:	Status return value.

Subroutine TIO_SKIP

This routine skips past a specified number of tape marks in either a forward or reverse direction.

The number of tape marks to be skipped is passed as a signed integer value - if a positive count is specified, the tape moves forward; if a negative count is specified, the tape moves in reverse.

Only tape marks (when the tape moves in either direction) and the beginning-of-tape marker (when the tape moves in reverse) are counted during a skip operation. The BOT marker terminates a reverse skip operation, whereas an EOT marker has no effect in either direction.

Format of call:

```
TIO_SKIP(IOCHAN,NTM,STATUS)
```

Given (argument):

IOCHAN:	INTEGER expression:	I/O channel assigned to device.
NTM:	INTEGER expression:	Number of tape-marks to skip.

Returned (argument):

STATUS:	INTEGER variable:	Status return value.

Note that a forward skip operation leaves the tape positioned just after a tape mark, whereas a reverse skip operation leaves the tape positioned just before a tape mark, that is, at the end of a file (unless the BOT marker is encountered).

Note also, that consecutive tape marks are treated as such and are not handled as 'logical end-of-volume' which would terminate a skip operation in the forward direction.

Subroutine TIO_MOVE

This routine moves a tape past a specified number of physical blocks in either a forward or reverse direction.

The number of blocks to be skipped over is passed as a signed integer value - if a positive count is specified, the tape moves forward; if a negative count is specified, the tape moves in reverse.

The operation is terminated by a tape mark when the tape moves in either direction, by the beginning-of-tape marker when the tape moves in reverse, and by the end-of-tape marker when the tape moves forwards.

Format of call:

TIO_MOVE(IOCHAN,NPB,STATUS)

Given (argument):

IOCHAN:	INTEGER expression:	I/O channel assigned to device.
NPB:	INTEGER expression:	Number of physical blocks that tape is to be moved.

Returned (argument):

STATUS:	INTEGER variable:	Status return value.

Note that if the operation is terminated by a tape mark, an end-of-file status is returned to the calling program. If the movement is in the forwards direction, then the operation will leave the tape positioned just after the tape mark, whereas movement in the reverse direction leaves the tape positioned just before the tape mark, that is, at the end of a file (unless the BOT marker is encountered).

Note also, that consecutive tape marks are treated as such and are not handled as 'logical end-of-volume' which would terminate a skip operation in the forward direction.

Subroutine TIO_REWIND

This routine repositions a tape to the beginning-of-tape (BOT) marker.

Under VMS, the tape is rewound to its load point and the device is left online. This has the same effect as the DCL command: SET MAGTAPE/REWIND.

Format of call:

```
TIO_REWIND(IOCHAN,STATUS)
```

Given (argument):

IOCHAN:	INTEGER expression:	I/O channel assigned to device.

Returned (argument):

STATUS:	INTEGER variable:	Status return value.

This routine will generally be used after a call to TIO_OPEN to ensure that the tape is positioned correctly for subsequent I/O operations.

Subroutine TIO_CLOSE

This routine de-assigns an I/O channel from a magnetic tape device.

No further I/O operations can be performed on the device, unless TIO_OPEN is called again.

Format of call:

```
TIO_CLOSE(IOCHAN,STATUS)
```

Given (argument):

IOCHAN:	INTEGER expression:	I/O channel assigned to device.

Returned (argument):

STATUS:	INTEGER variable:	Status return value.

Subroutine TIO_GETMSG

This routine returns the message text associated with a given status value.

Currently, the text is obtained from the VMS system message file, although this may not be the case in future versions.

Format of call:

```
TIO_GETMSG(STATUS,MSGBUF,MSGLEN)
```

Given (argument):

STATUS:	INTEGER expression:	Status value.

Returned (argument):

MSGBUF:	CHARACTER variable:	Buffer to receive message text.
MSGLEN:	INTEGER variable:	Length of message text.

This routine will generally be used in conjuction with TIO_ERR. For example:

```
    CHARACTER*72 MSG

    :    :    :
    :    :    :

    CALL TIO_OPEN('TAPE',IOCHAN,STATUS)
    IF (TIO_ERR(STATUS)) THEN
       CALL TIO_GETMSG(STATUS,MSG,LEN)
       WRITE (*,'(X,A)') MSG(:LEN)
       STOP  '*** PROGRAM ABORTED ***'
    ENDIF
```

Note that if the user-specified buffer is too small to accommodate the complete message text, the string will be truncated from the right.

Function TIO_EOF

This function determines whether the status return value from a previous I/O operation indicates an end-of-file condition.

Under VMS, this is accomplished by comparing the value with the appropriate System Service return status (SS$_ENDOFFILE).

Format of call:

TIO_EOF(STATUS)

Given (argument):

STATUS:	INTEGER expression:	Status value.

Returned (function value):

TIO_EOF:	LOGICAL variable:	Set to TRUE if status indicates an end-of-file condition.

Users should note that an end-of-file condition is not regarded as successful completion of an I/O operation and would, therefore, be treated as an error by TIO_ERR. As a consequence, programs should test for this condition prior to any error checking. For example:

```
    IF (TIO_EOF(STATUS)) THEN

       :    :    :

    ELSEIF (TIO_ERR(STATUS)) THEN

       :    :    :

    ENDIF
```

Function TIO_ERR

This function determines whether the status return value from a previous I/O operation indicates an error condition.

Under VMS, this is accomplished by examining the low-order bit of the status value - if the bit is set, then the operation completed successfully and TIO_ERR is set to FALSE. Non-successful completion is indicated by this bit being clear and TIO_ERR is set to TRUE.

Format of call:

TIO_ERR(STATUS)

Given (argument):

STATUS:	INTEGER expression:	Status value.

Returned (function value):

TIO_ERR:	LOGICAL variable:	Set to TRUE if status indicates an error condition.

Users can determine the actual cause of the error by calling TIO_GETMSG.

References

[1] "VAX/VMS System Services Reference Manual"
Digital Equipment Corporation.

[2] "VAX/VMS I/O User's Guide"
Digital Equipment Corporation.

[3] "VAX-11 FORTRAN User's Guide"
Digital Equipment Corporation.

COVERNOTE FOR SUN/21.1

Users should note that these routines are intended as replacements for those described in Starlink System Note 7 and, as such, should be used in preference. Existing software which include calls to SSN7 routines should be modified accordingly as the library may be withdrawn at some future date.

Although the most significant difference between the two sets of routines is one of naming, there are also some less obvious changes to argument lists.

(1) MTOPEN is renamed TIO_OPEN.

(2) MTREAD is renamed TIO_READ and two arguments transposed.

(3) MTWRIT is renamed TIO_WRITE.

(4) MTWTM is renamed TIO_MARK.

(5) MTSKIP is renamed TIO_SKIP.

(6) MTBACK is replaced by TIO_MOVE which provides block skipping in either direction.

(7) MTREW is renamed TIO_REWIND.

(8) MTEOF is renamed TIO_EOF.

(9) MTEOT is withdrawn.

(A) MTERR is renamed TIO_ERR and loses one argument.