Published: 1/10/68

Identification

The Interface between the Canonicalizer and the Typewriter Device Strategy Module.
J. F. Ossanna.

Purpose

This document describes the interface to the Canonicalizer (CAN) used by the Typewriter Device Strategy Module (DSM).

General

The data received by the DSM from the DCM consists of device-independent ASCII data in which all non-Multics-defined characters have been removed. Further, all control characters and sequences have been translated by the DCM into corresponding Multics-defined control characters or sequences. The DSM calls the CAN to canonicalize input data only upon reciept of a read call from the user. The CAN processes only enough data to satisfy the read call. Because the CAN must canonicalize between canonicalization delimiters, it may process more data then consumed by the read call, or it may not find enough canonicalization delimiters to completely satisfy the read call. In the first case, the CAN preserves the processed but unread data for later use, and in the second case it returns status indicating the need for more data.

Call Interface

During attach-time initialization the DSM initializes the CAN by making the following call.

```
call canon$init(pibp,cstatus);

dcl pibp ptr,                    /*PIB pointer*/
 cstatus;                        /*call status*/
```

The CAN allocates its PIBE as the second PIBE (the first one is allocated by the DSM) and initializes it. The CAN uses (pibp->pib.dtabp2) as its driving table pointer (see Section BF.2.20). Following successful return from the canon$init call, the DSM informs the CAN of the default tab settings for the particular typewriter by making the following call.

```
call canon$tabs(pibp,htabs,vtabs,cstatus);

dcl htabs bit (144),             /*each bit corresponds to a column
                                   position. 1=tab set*/
 vtab bit (144);                 /* each bit corresponds to a whole line
                                   position. 1=tab set. Last 1 bit
                                   indicates number of lines in page*/
```

The tab settings are obtained from the device profile by the DSM. The canon$tabs call is also made by the DSM upon receipt of a restart call, and as the result of certain status returned by the DCM.

The following call is made whenever the read delimiters or the break delimiters are changed by the receipt of a setdelim call. (Note that break delimiters are canonicalization delimiters.)

```
call canon$setdelim(pibp,rdelim,cdelim,cstatus);

dcl rdelim char(*),              /*read delimiter string*/
 cdelim char (*);                /*canonicalization delimiter string*/
```

The canon$setdelim call is not implemented in the initial IOS. Instead, the "new line" character is used as the only read delimiter, and only the intrinsic canonicalization delimiters (new line, vertical tab, and form feed) are used.

When the DSM receives a read call from the user, the following call is made to the CAN.

```
call canon$input(pibp,source_ptr,source_size,source_offset,devstat,
                 workspace_ptr,workspace_offset,nelem,nelemt,
                 new_offset,cstatus);

dcl source_ptr ptr,              /*pointer to input data buffer*/
 source_offset fixed bin,        /*offset to first character in
                                   input data buffer*/
 source_size fixed bin,          /*size of input buffer*/
 devstat bit (36),               /*device status corresponding to
                                   character in input buffer at
                                   source_offset. Otherwise zero*/
 workspace_ptr ptr,              /*pointer to user's workspace*/
 workspace_offset fixed bin,     /*offset in user's workspace*/
 nelem fixed bin,                /*number of elements required by user*/
 nelemt fixed bin,               /*number of elements returned to user,
                                   at time of return from CAN*/
 new_offset fixed bin;           /*offset to first character in input
                                   buffer not used by CAN, usually used
                                   later as new source_offset*/
```

The internal breakdown of the device status bit string is indicated by the following declaration.

```
dcl 1 devstat_map,               /*structure defining the bits in devstat*/
 2 vert_pos bit (9),             /*number of half-lines*/
 2 hor_pos bit (8),              /*number of columns*/
 2 red_black bit (1),            /*red-black indicator, 1=red*/
 2 undefined bit(18);
```

The CAN processes data beginning at source_offset in the input buffer pointed to by source_ptr. Only enough data to satisfy the effective read call described by the four arguments,

workspace_ptr, workspace_offset, nelem, and nelemt, is processed. The minimum amount of data that can be processed is that up to the first canonicalization delimiter. If the read call requires less data, the CAN preserves the canonicalized remainder of the line in a string allocated in pibp->pib.ioarea; a relative pointer to the string is kept in the CAN's PIBE. Upon receipt of the next canon$input call, this previously-unread data is the first to be utilized. If the CAN cannot find enough canonicalization delimiters to satisfy the read call, a bit is turned on in cstatus; nelemt gives the amount of data that was returned. The DSM then calls the CAN again with a new supply of data. The CAN has kept the previously processed part line and picks up where it left off.

Following a canon$input call, the CAN may have preserved either of two kinds of residual data; (1) a canonicalized but unread portion of a line, or (2) a partly-processed but uncanonicalized fraction of a line. In either case, such preserved data may be discarded by the DSM by the use of the following call.

```
call canon$reset(pibp,cstatus);
```

When repeated canon$input calls are made to satisfy one read call, it is the responsibility of the DSM to properly adjust workspace_offset and nelem when necessary. Similarly, the actual nelemt returned to the user must be computed by the DSM.

Device Status

The device status, devstat, is constructed by the DSM from information contained in the hardware- and call-oriented status subfields in the status bit string returned by the DCM for every read call to it. This mechanism is necessary to account for changes in device status wrought by intervening write calls and other disturbances.