# SWEET 16 OPERATING SYSTEM



## EXTERNAL REFERENCE SPECIFICATION

## SUPPLEMENT 3

## - HANDLER LOADER -

SCOTT SCHEIMAN

8-31-82

TABLE OF CONTENTS


1.0 PURPOSE . . . . . 2

1.1 INTRODUCTION . . . . . 2

1.2 CONSUMER PROFILE . . . . . 4

1.3 INTERFACE WITH OTHER PRODUCTS . . . . . 4

1.4 FAMILY OF PRODUCTS . . . . . 5

1.5 DISTRIBUTION . . . . . 5

2.0 APPLICABLE DOCUMENTS . . . . . 5

3.0 REQUIREMENTS . . . . . 6

3.1 INTERFACES . . . . . 6
3.1.1 PHYSICAL REQUIREMENTS . . . . . 7
3.1.2 LOGICAL REQUIREMENTS . . . . . 7
3.1.3 MAN/MACHINE INTERFACE . . . . . 8

3.2 FUNCTIONAL DESCRIPTION . . . . . 8
3.2.1 POWER-ON (COLD START) . . . . . 8
3.2.1.1 PERIPHERAL POLL DURING POWER-ON PROCESSING (TYPE 3) . . . . 9
3.2.1.2 HANDLER LOAD AND RELOCATION DURING POWER-ON PROCESSING. . . 11
3.2.1.3 INITIALIZATION AND LINKING DURING POWER-ON PROCESSING . . . 13
3.2.2 APPLICATION-INITIATED LOAD . . . . . 18
3.2.2.1 APPLICATION-INITIATED OPEN POLL (TYPE 4). . . . . 18
3.2.2.2 LOAD, RELOCATION, INITIALIZATION, USE . . . . . 21
3.2.3 SYSTEM RESET (WARM START) REINITIALIZATION. . . . . 23
3.2.4 SUBROUTINE INTERFACES . . . . . 25

3.3 PERFORMANCE REQUIREMENTS . . . . . 29

3.4 DESIGN REQUIREMENTS . . . . . 29

3.5 PACKAGING REQUIREMENTS . . . . . 29

3.6 SPECIAL REQUIREMENTS . . . . . 29

4.0 APPLICABLE STANDARDS . . . . . 29

## 1.0 PURPOSE

This document describes a set of requirements to achieve the following goals:

- o Standardized polling, loading and linking of device handlers via the serial port during power-on;
- o Standardized polling, loading and linking of device handlers via the serial port after power-on by request of the running application through simple software interfaces.

The objective of this document is to specify the external characteristics of the routines which are to reside in the SWEET16 ROM Operating System to implement these handler loading functions. However, these routines work in a larger environment of procedures residing in the SWEET16 memory. References to these programs and procedures will be made when necessary to clarify the polling, loading, and linkage processes in the context of the whole system.

## 1.1 INTRODUCTION

The operation of loading a peripheral handler is viewed here from three perspectives:

- o Polling--finding out if a peripheral is present, and which one;
- o Loading and relocating the handler via the serial port;
- o Linking the handler into the system.

Two types of polling are specified here. The first (Type 3) occurs during power-on (cold start) processing. This poll makes no assumptions about what peripheral(s) may answer the poll. Each peripheral which answers this poll then has its handler loaded at the low available memory boundary (MEMLO). The poll is repeated until no peripheral answers. See section 3.2.1.

The second type of poll described in this document (Type 4) is used when an application asks the operating system to load a handler. This poll differs from the first type in that information is known about which peripheral is being polled. In a sense this poll is really a form of conditional status request. If the device does not answer it is not there. If it answers, the answer contains status information which is subsequently used to load the handler. See section 3.2.2.

[Note: Three other types of Poll exist, named Types 0, 1, and 2. These three types are the older polls which have been used to automatically load handlers on the A400/A800 computers. These polls are described in reference C. See also the note in section 1.3 regarding compatibility between handlers loaded with this new scheme

and handlers loading under the older schemes.]

Loading of a handler is achieved in a uniform manner regardless of which type of poll has been used. (In fact, the poll need not even precede the handler loading operation.) The loading operation is documented in two pieces: serial port commands (described in this document, section 3.2.1.2) and relocatable format of the handler (described in reference D).

Handler linkage is fairly complex, since it is at this stage that the handler occupies RAM resources of the SWEET16 computer and must coexist with the many other pieces of software in the computer. The linkage must be understood in all of these terms:

- Availability of RAM resources for the handler: RAM is made available for loading a handler and contiguous variables by the OS during power-on initialization at the MEMLO boundary if the peripheral answers Poll Type 3 (section 3.2.1). RAM is made available for loading the handler following a Type 4 Poll by the application (section 3.2.2). No other RAM is made available by the OS. Further RAM allocations may be made via transaction between a loaded handler and the calling application (outside the scope of this document).

- Making RAM use known to the system (MEMLO): The handler size is kept in the handler linkage table; this size is added to MEMLO when the handler is first loaded, and subsequently during the warm-start process. This handler size is set to zero by the operating system if the handler was not loaded at the MEMLO boundary, in order to prevent modification of MEMLO when the handler size is added. See section 3.2.1.3.

- CIO linkage (handler table entry): See sections 3.2.1.3, 3.2.2.2, and 3.2.4.

- First initialization (immediately upon loading): See sections 3.2.1.3 and 3.2.2.2.

- System reset (warm start) linkage: This is done by chaining handler linkage tables; see section 3.2.1.3.

- Re-initialization (warm start operations): Section 3.2.3.

## 1.2 CONSUMER PROFILE

Does not apply.

## 1.3 INTERFACE WITH OTHER PRODUCTS

Handlers loaded via the procedures described here must interface with CIO and SIO according to references A and B.

[Note: The power-up time loading process described here may interact poorly with handlers loaded by the 400/800-compatible technique. It has not been possible to fully analyze the various load combinations which may occur. However, a partial analysis has been undertaken and the results look promising overall for proper behavior in most practical cases. In order to assure that this is true, a more thorough investigation of the procedures to be used by 400/800-compatible handler loading must be undertaken before any more such devices are designed. Presently the 850 Interface Module is the only such device. It is known that (readily correctable) flaws exist in the 850 handler loading process.]

The power-up processes described here will work with DOS 2 except when an AUTORUN.SYS program fails to return control to the operating system to complete the power-up procedure. In that case, however, it is permissible to program that AUTORUN.SYS file to perform the remaining OS initialization, including the handler polls and loading.

At system reset (warm start) time, each previously loaded handler must be reinitialized. This presents a possible problem if the handler has been modified in any way after it was loaded. If the handler was loaded under BASIC, for example, the user could inadvertently delete the handler (eg., by typing NEW). Under DOS 2, loading of the DUP package swaps memory above the FMS out to MEM.SAV; this happens to be the memory automatically loaded handlers occupy. (DOS 2 DUP traps system reset and reloads MEM.SAV before handlers are reinitialized. However, DUP calls to handlers which are swapped out are fatal.) In any case, it is the responsibility of applications writers and systems software writers to allow for coexistence of automatically loaded handlers. Whenever software does not fully protect against such problems, the users manuals for such software must inform the user of the proper use of the products to avoid problems.

## 1.4 FAMILY OF PRODUCTS

The procedures described here apply to SWEET16-compatible peripherals which require handlers to be loaded into the CPU memory. Major support is given to loading handlers from the peripheral itself. Minor support is given to loading handlers in other ways, eg. from diskette (refer to section 3.2.4).

## 1.5 DISTRIBUTION

Does not apply.

## 2.0 APPLICABLE DOCUMENTS

Ref. A: ATARI PERSONAL COMPUTER SYSTEM OPERATING SYSTEM USER'S MANUAL, number CO16555. Current revision is November 1980.

Refs. B and C: ATARI HOME COMPUTER SYSTEM SERIAL INPUT/OUTPUT INTERFACE USER'S HANDBOOK, PARTS I AND II (software department internal documents).

Ref. D: SWEET16 OS ROM EXTERNAL REQUIREMENTS SPECIFICATION SUPPLEMENT 2 -- RELOCATING LOADER (internal document).

## 3.0 REQUIREMENTS

### 3.1 INTERFACES

#### APPLICATION INTERFACES

When handlers are loaded automatically as a result of power-on poll and loading procedures described here, the lower memory boundary MEMLO shall reflect the size of loaded handlers.

The procedures described here allow an application to load a handler after power-up time into an application-supplied RAM area. A normal OPEN call to CIO specifying a device whose name is not in the handler table shall result in issuing a Type 4 Poll to find the device, or the application may make the Poll unconditional by setting HNDLOD nonzero before the OPEN (section 3.2.2.1). If the device answers the poll, the handler size is in the OS variables DVSTAT through DVSTAT+4; the application will establish a RAM area for the handler and inform the OS by setting DVSTAT through DVSTAT+4 and setting HNDLOD nonzero. Any subsequent I/O call (except CLOSE) to that IOCB shall result in the loading of the handler into the application-supplied area, followed by execution of that I/O command (section 3.2.2.2).

The cassette buffer shall be used by the loading procedure; therefore, the application must not need any data in the cassette buffer when requesting handler loading from the OS.

A procedure to unload handlers (except those loaded at power-on at MEMLO) is described in section 3.2.4. The application sets HNDLOD nonzero prior to issuing a CLOSE command to the handler through CIO. This results in unlinking the handler from the system. Note, however, that implementation of this feature is the responsibility of the handler and is not automatically performed by the OS.

#### SYSTEM INTERFACES

The routines described here interface with peripherals in two basic ways: serial commands, and serial data. Within the context of this specification, the serial commands are the Types 3 and 4 Polling commands, and the handler loading commands (sections 3.2.1.1, 3.2.2.1, and 3.2.1.2). Serial data in this context is the handler being loaded: the block structure for loading a handler is described in section 3.2.1.2; the record structure consists of relocation records described in reference D.

Loaded handlers interface with the I/O portions of the operating system as described in reference A. Loaded handlers also interface with the initialization portions of the OS described primarily here in section 3.2.1.3.

Three utility subroutines are made available for handlers loaded into the system. They are described in section 3.2.4.

Loaded handlers interface with other parts of the system primarily by the fact of their occupying RAM. Careful study should be made of the effects of loading handlers at MEMLO, particularly when using the DOS 2 product.

### 3.1.1 PHYSICAL REQUIREMENTS

There are no physical requirements.

### 3.1.2 LOGICAL REQUIREMENTS

The routines described here are part of the SWEET16 resident OS and shall use RAM in the OS pages only (pages 0, 2, 3, 4). These variables shall be reserved for OS use, and will either be used exclusively by the OS routines described here or shared with other OS routines (non-overlapped use). Use of these variables by any other routines in the system will produce unpredictable results.

### 3.1.3 MAN/MACHINE INTERFACE

Does not apply.

## 3.2 FUNCTIONAL DESCRIPTION

### 3.2.1 POWER-ON (COLD START)

This section describes the sequence of events taken by the operating system during power-on. This consists of actions which have existed in the 400/800 revision B operating system plus new operations which are the SWEET16 enhancements of this document. Only the degree of detail needed here is given.

This section is an outline of the steps performed. Details of the new operations are given in subsequent sections.

1. The OS shall set the flag WARMST zero (means coldstart);

2. DOSVEC shall be set pointing to default built-in application (logo screen display);

3. MEMLO (02E7 and 02E8 hex) shall be set to 0700 hex;

4. OS resident I/O handlers shall be initialized and have their entry points moved into the handler table (HATABS);

5. Application cartridge ("A") shall be initialized, if present;

6. Cassette or disk boot occurs depending on what is attached to the system.. If no disk or cassette is present, a peripheral may respond to the A800-compatible Type 0 Poll; if such a peripheral answers the poll, it shall be loaded at this time as if it were being loaded from disk;

7. If disk (or Type 0-responding peripheral) boots, operating system shall pass control to DOS initialization at DOSINI. The DOS may optionally perform Type 1 Polling and/or Type 3 Polling and load subsequent peripheral handlers. The DOS may optionally not return control to the Operating System--however, this is not the recommended practice;

8. If DOS initialization returns to operating system, it shall then perform Type 3 Polling and load handlers at the MEMLO boundary and initialize them. Type 3 Polls shall be repeated until no peripheral answers. (Sections 3.2.1.1, 3.2.1.2, 3.2.1.3);

9. The operating system shall complete the system initialization and then start the application cartridge, if present, or run the application loaded at DOSVEC.

### 3.2.1.1 PERIPHERAL POLL DURING POWER-ON PROCESSING (TYPE 3)

Type 3 Polling occurs following DOS loading and initialization. This poll is performed by the operating system; however, the poll may be performed by the initialization portion of the DOS (eg., AUTORUN.SYS as part of DOSINI processing). It is acceptable for the DOS initialization to perform Type 3 Polling and handler loading and subsequently return to the operating system which shall then repeat Type 3 Polling. This works because peripherals will not respond more than once to a Type 3 Poll.

Once only, at cold-start time, the operating system shall send a Type 3 Poll Reset command over the serial port. This command shall be sent during cold-start processing, regardless of whether the cold start is hardware or software initiated (that is, the Vcc/READY line may or may not have fallen to zero prior to the Poll Reset). This command follows the disk and cassette boot operations, if any, and precedes the first Type 3 Poll command. The Poll Reset command shall have the following format:

- Device address of 4F hex (peripherals looking for the Type 3 Poll Reset command may ignore the device address and look only for the poll command '@'; however, the device address will always be 4F hex and the peripheral may check this);
- Command of '@' (40 hex) (peripherals looking for the Poll Reset command will always look for the '@' command);
- Both Aux1 and Aux2 shall be 4F hex to distinguish the Type 3 Poll Reset command;
- Command checksum, which the peripheral checks.

Peripherals which will respond to the Type 3 Poll must also watch the serial port for the Poll Reset. No peripheral will respond to the Poll Reset (no response should be sent to the computer). However, when the Poll Reset command is sensed, each peripheral will re-enable itself to respond to the Type 3 Poll (see below).

The Type 3 Poll shall be a serial port command structured as follows:

- Device address of 4F hex (peripherals looking for Type 3 Poll may ignore the device address and look only for the poll command '@'; however, the device address will always be 4F hex and the peripheral may check this);
- Command is '@' (40 hex) (peripherals looking for this poll will always look for the '@' command);
- Both Aux1 and Aux2 are zero (this distinguishes Type 3 Poll from Type 4);
- Standard command checksum (peripherals check this).

[Note: Command byte "@", 40 hex, 064 dec, is now reserved for Type 3 and Type 4 Poll and must not be used on the serial port for any other purposes, regardless of device address. Similarly, command byte "?", 3F hex, 063 dec, is reserved for Types 1 and 2 poll.]

Peripherals which are to respond to the Type 3 Poll will keep a count of SIO retries of this command over the serial port. Each such peripheral is assigned a unique "slot" or retry on which it is to answer. The peripheral must keep a running count of the number of Type 3 Poll retries following the last serial command which was NOT a Type 3 Poll (even if that non-poll command was addressed to another peripheral). In other words, each time a non-poll command is seen the peripheral will reset its retry counter. Also, the peripheral must not respond to a Type 3 Poll if it has already done so since the last Type 3 Poll Reset command was sensed by the peripheral.

When it is the peripheral's turn (per retry count) to respond to the poll, the peripheral will ACK the command and then return a standard serial input data frame containing the following four data bytes, which shall be interpreted as follows:

1. Low byte of handler size (in bytes; must be EVEN);

2. High byte of handler size;

3. Device serial I/O address to be used for loading;

4. Peripheral revision number.

The SIO call initiating the Type 3 Poll shall place these result bytes in OS variables DVSTAT (02EA hex) through DVSTAT+3 (02ED hex). Successful return from SIO from the poll shall indicate to the OS that the peripheral has a handler to be loaded. Unsuccessful return indicates there is no peripheral answering the poll and the polling operation is not repeated.

Various errors may occur during the loading or initialization of a handler (see sections 3.2.1.2 and 3.2.1.3). In some of these cases, it is unclear whether a non-Type 3 Poll command has been sent over the serial bus. A non-Type 3 Poll command is necessary in order to insure that the peripherals, which count retries of the Type 3 Poll command, do not get out of synchronization with the computer. Accordingly, when such errors occur during cold-start processing, the operating system shall send a Type 3 Null Poll command over the serial bus before sending the Type 3 Poll for the next polled peripheral. The Null Poll shall have the same form as the Poll Reset (above), except that Aux1 and Aux2 shall have the value 4E hex. No peripheral should treat the Null Poll in any special way, except that it should not be confused with any other of the Type 3 Poll commands. No peripheral should respond to the Null Poll in any way. The Null Poll command is really just a place holder--effectively a serial bus "NOP".

### 3.2.1.2 HANDLER LOAD AND RELOCATION DURING POWER-UP PROCESSING

[Note: The loading procedure described here is also used to load handlers when the loading is application-specified after power-on has completed. The only differences are where in RAM the handler is loaded, and handling of loading errors. Accordingly, this single section deals with both loading operations. The major point of view is toward loading at power-on time to the MEMLO boundary; differences for the application-initiated load are noted. See section 3.2.2 for more on application-initiated loading.]

After a peripheral responds to the Type 3 Poll, the OS shall then compare the sum of MEMLO (02E7 and 02E8 hex) and the size of the handler to be loaded (DVSTAT and DVSTAT+1, 02EA and 02EB hex) to MEMTOP (02E5 and 02E6 hex) to determine that there is room to load the handler. If there is insufficient room, the handler shall not be loaded, and the OS shall issue a Null Poll command (3.2.1.1) and proceed with further Type 3 Polling (3.2.1 step 9).

Otherwise the peripheral handler is loaded, starting at MEMLO and proceeding until the load is completed. (Note: the load address may also be application specified; see section 3.2.2.) The loading operation shall be achieved using the Operating System's relocator (reference D). An appropriate call to the relocator shall be made, specifing all parameters needed:

- o Loading address. This shall be either a copy of MEMLO, or the application-supplied load address (section 3.2.2.2). Before handing this value to the relocator, the OS Type 3 Poll process shall insure that it is even-valued by adding one if it is found to be odd;

- o Zero-page loading address. The handler will not load into page zero. This address shall be set to 80 hex;

- o Address of get-byte subroutine described below.

The get-byte subroutine supplied to the relocator shall call on SIO to get the handler relocatable object records from the peripheral and then pass them a byte at a time to the relocator. The records shall be read from the peripheral in numbered blocks of 128 bytes each, numbering starting at 0 and going as high as needed (255 max). The cassette buffer shall be used for storing each block as it is being fed to the relocator. The final block may be unfilled; get-bytes will stop from the relocator when the End record is processed, so the remaining portion of that block shall be ignored.

Serial port load commands shall be as follows:

- o Device address taken from Poll response;

- o Load command "&" (26 hex, 038 decimal);

- Aux1 = block number to be loaded;

- Aux2 = undefined (must be ignored by peripheral);

- Appropriate checksum.

If the peripheral is asked to supply a block whose number is out of range, it will either NAK or not respond (preferable action is no response). The reader will then pass error status to the relocator which will pass the error on to the caller. At power-on, the caller is the OS Type 3 Poll routine, which shall respond to the error by ignoring this peripheral and continuing polling for other peripherals (Null Poll, section 3.2.1.1, then 3.2.1 step 9). When loading is being called by an application, the IOCB shall be closed and error 133 (Device Not Open) shall be returned to the application.

During cold-start processing, the OS shall ignore all parameters returned by the relocator when relocation completes except the error status. All relocating loader errors shall produce the results of the preceding paragraph.

No check will be made that the handler is actually relocated properly. Some errors will be detected by the relocator; however it is the responsibility of the peripheral designer to create a proper device handler. The handler must occupy contiguous RAM, starting at the load address. No restriction is placed on the use of this RAM area (it may be code, variables, or data) except that the linkage conventions (section 3.2.1.3) must be followed. When loaded under applications request, the size of the area allocated for the handler can be larger than the minimum required, and the handler may make use of this extra RAM as needed (see section 3.2.2.2). When loaded at MEMLO during power-up, the handler will specify its RAM needs (section 3.2.1.3 and section 3.2.3).

[Note: a "bug" exists in the 6502 processor where a JMP indirect instruction will fail if the two-byte indirect pointer is relocated across a page boundary. This may be avoided by placing all indirect pointers on even addresses; since loading always occurs on even boundaries, the pointer will never cross a page boundary.]

### 3.2.1.3 INITIALIZATION AND LINKING DURING POWER-UP PROCESSING

[Note: The handler initialization and linking procedures during power-up processing are very similar to those during warm-start reinitialization and application-initiated handler loading. Therefore, this section serves for all processes. It is written in terms of the power-up sequence, with occasional test conditions for the warm-start variations. The major differences in this procedure between power-up and warm-start are described in section 3.2.3. Application-initiated load is described in 3.2.2.2.]

Once loaded, a handler will be linked into the system in three ways:

- The handler's RAM usage will be declared;
- The handler's name and linkage table address will be entered into the handler table;
- The handler's linkage table will be entered into a linked-list of known loaded handlers for System Reset (warm start reinitialization).

The handler will have a linkage table at its load address. This table contains the following:

| OFFSET | CONTENTS |
|---|---|
| 0 - 14 | Standard handler entry vectors (reference A): |
| | OPEN vector; |
| | CLOSE vector; |
| | GETBYTE vector; |
| | PUTBYTE vector; |
| | GETSTAT vector; |
| | SPECIAL vector; |
| | initialization code JMP; |
| 15 | Linkage table checksum; |
| 16 - 17 | Handler size in bytes to add to MEMLO. |
| 18 - 19 | Handler linkage table chain forward pointer; |
| 20 - 21 | Zero (reserved for future expansion). |

Byte 15 (checksum) is calculated such that the wrap-around-carry sum of bytes 0 through 17 is FF hex (one's-complement negative zero); it is used by the operating system to check the integrity of the linkage table during system reset (warm start) reinitialization. Since bytes 0-17 may vary depending on load address the checksum will be calculated after the handler is loaded. Bytes 18-19 point to the handler linkage table loaded next. If this is the last handler loaded, this forward pointer is null (zero).

The initialization process for a newly loaded handler immediately follows its loading:

[Note: All steps of this process are performed by a subroutine which is normally used as part of the OS process of linking new handlers into the system. This subroutine can be called by other system routines; the calling sequence is discussed in section 3.2.4. As used during power-up loading of handlers following Type 3 Polling, the MEMLO parameter used in step 4 shall be set on, indicating that the handler's size is to be added to MEMLO.]

1. The OS shall add the new handler linkage table at the end of the linkage table chain. This is done by starting at the head of the chain, CHLINK (in the OS database) and following the pointers until a null (zero) pointer is found. For each linkage table in the chain (except the last), the checksum is checked to verify the integrity of the linkage table; checksum failure results in failure to initialize the newly loaded handler, and the rest of this initialization procedure is bypassed. No error is reported out of the OS during coldstart (in this case, polling continues with a Null Poll, followed by Type 3 Poll, 3.2.1 step 9). In the case of a non-OS caller, the error shall be indicated to the caller by returning with carry bit set. If the checksums are OK, the address of the new linkage table, which is the load address of the handler, is placed in the null pointer which was at the end of the chain. Then the pointer in the new linkage table is nulled (zeroed);

2. The OS loader shall then JSR to the handler initialization code;

2a. The handler will initialize itself, optionally utilizing the handler table entry subroutine in the resident OS (section 3.2.4). Errors occurring in the linking process will produce linking failure (discussed below). The handler will initialize itself as follows:

2b. Call the OS-resident handler table entry subroutine to add a handler entry for this new handler;

2c. Optionally establish the linkage table handler size. The handler size could simply have been loaded into the linkage table at relocation time, in which case the handler initialization procedure now takes no further action. Alternatively, the handler can calculate the size and insert the result during this first initialization. The handler will calculate this size only once, and supply the result to the operating system in the linkage table at this point during power-up initialization. The handler will not modify these bytes in the linkage table at any subsequent time. The OS flag WARMST can be used to distinguish power-on initialization from subsequent warm-start reinitialization. The handler size need not be returned to the OS if WARMST is nonzero. If the handler calculates its RAM needs, it is responsible for

insuring that the resulting addition to MEMLO does not exceed MEMTOP. Also, it is the handler's responsibility to ensure that the size set by the handler is even-valued. It is safe i the calculated size does not exceed the size reported by the Type 3 Poll (section 3.2.1.1);

2d. Return with Carry bit clear if there was no init error; otherwise, return with carry set;

(Note: the handler init need not save any 6502 registers.)

3. If the handler initialized unsuccessfully (Carry returned set the new handler linkage table shall be removed from the linkage table chain using the routine described in section 3.2.4 for that purpose, and the handler installation is terminated. In this case, none of the following steps is performed; no error indication is given out of the OS during coldstart, and polling continues with a Poll Reset followed b further Type 3 Polling. In the case of a non-OS call to this initialization process, the error shall be returned to the caller by returning with the carry bit set.

4. If the handler initialization was successful (Carry returned clear) the OS shall then check the parameter to see what mode of initialization is being performed, to determine whether or not the handler size should be added to MEMLO. If the parameter is set, then the handler size should be added to MEMLO. If the parameter is not set, the handler size should not be added to MEMLO. In the latter case, the handler size entry in the handler linkage table shall be cleared to zero;

5. The handler size is added from the handler linkage table to MEMLO (02E7 and 02E8 hex);

6. The linkage table checksum shall be calculated and inserted into the table. This is done by first zeroing the checksum; then calculating the checksum of the first 18 bytes of the table; then storing the one's complement of the resulting sum as the calculated checksum of the linkage table.

In step 2, above, the handler may interrogate the system flag WARMST to determine the time of initialization. WARMST (0008 hex) shall be zeroed by the OS at the beginning of power-on processing. Unless modified by other code in the system, WARMST remains zero until the [SYSTEM.RESET] key is pressed, when it is set to FF (hex). Should this be unacceptable to the handler initialization, the handler should keep an internal variable to keep track of which initialization is occurring.

Handler table overflow error is a possibility in step 2b, above. The handler will return with Carry set to indicate initialization failure, unless it performs some reasonable error recovery.

Note: The above procedure uses DVSTAT+2 and DVSTAT+3 (02EC and 02ED hex).

### 3.2.2 APPLICATION-INITIATED LOAD

Most of the loading and initialization processes of an application-initiated load are identical to those used for power-up load. Those differences between the two (MEMLO handling) which affect the handler are discussed in section 3.2.1.3. The major difference lies in the polling processes used.

#### 3.2.2.1 APPLICATION-INITIATED OPEN POLL (TYPE 4)

When an application calls CIO to perform an open, the following shall occur:

1. The OS flag HNDLOD (02E9 hex) shall be interrogated to determine whether the application desires a Type 4 Poll for the device being opened. HNDLOD=zero means conditional poll (step 3); anything else means unconditional poll (step 2);

   [Note: the operating system shall set HNDLOD zero at power-on and system reset. If the application does not modify HNDLOD, conditional poll will always be selected by any OPEN.]

2. If unconditional poll is selected, a Type 4 Poll (see below) occurs. If no peripheral answers, step 7 is performed. If a peripheral answers, its 4-byte answer is returned by CIO to the application in DVSTAT through DVSTAT+3 (02EA through 02ED hex) (proceed to step 6);

3. If conditional poll is specified, CIO checks for the device in the handler table. If an entry is found, the handler already exists and normal open processing continues. Proceed to step 5;

4. If conditional poll is specified and no handler entry is found, a Type 4 Poll is issued. Everything proceeds from here as in step 2;

5. If no poll was issued, this fact is flagged to the calling application by setting DVSTAT and DVSTAT+1 (02EA and 02EB hex) to zero. I/O status returned indicates either successful OPEN, or open failure for any of the standard set of possible reasons;

6. If a poll was issued and successful, the IOCB is "provisionally" opened. This includes all normal CIO OPEN processing, but includes none of the handler open processing since the handler is not loaded at this time. The IOCB is marked "provisionally" open in the following ways:

   o The handler table pointer ICHID is set to 7F (hex);

- The put address ICPTL, ICPTH is set pointing to the OS-resident application loader routine;

- ICAX3 contains the device name for the handler loader table;

- ICAX4 contains the device serial address for loading.

Normal status (01) is returned following a provisional open, and DVSTAT through DVSTAT+4 (02EA through 02ED hex) contain information needed by the application to provide RAM for the handler load which will follow (see below);

7. If a poll was issued and no device answered, the IOCB is not opened and error 130, Non-existent Device, is returned.

The OS flag HNDLOD (02E9 hex) shall be set to zero each time CIO returns to the application, regardless of what call was made or the results of the call.

The Type 4 Poll shall be a serial port command structured as follows:

- o Device address of 4F hex (peripherals looking for Type 4 Poll may ignore the device address and look only for the poll comman '@'; however, the device address will always be 4F hex and the peripheral may check this);
- o Command is '@' (40 hex) (peripherals looking for this poll will always look for the '@' command);
- o Aux1 contains the device name, which is an ATASCII upper-case letter (range 41 hex through 5A hex) (the peripheral must be assigned that device name in order to legally answer the poll);
- o Aux2 contains the device number, which is an ATASCII digit (range ATASCII 1 through 9, 31 hex through 39 hex) (the peripheral may optionally use this information in deciding whether or not to answer the poll);
- o Standard command checksum (peripheral checks this).

This poll differs from the Type 3 Poll in that the device name and number is included in the poll. Therefore the peripheral need not count retries of the Type 4 Poll and should answer the Poll as soon as the Poll command is recognized. There is no limitation on answering the Type 4 Poll; the peripheral should answer its Type 4 Poll each time the computer issues this Poll.

The peripheral response to a Type 4 Poll is the same as for the Type 3 Poll (section 3.2.1.1). The four response bytes shall be placed in DVSTAT through DVSTAT+3 (02EA through 02ED hex).

### 3.2.2.2 LOAD, RELOCATION, INITIALIZATION, USE

Following a "provisional" open the application must check the DVSTAT bytes to determine the need to allocate an area for the handler which is to be loaded. The application must set aside an area, on an even address, at least as large as the handler size specified in DVSTAT and DVSTAT+1 (02EA and 02EB hex). Then the application must place the address of this area in DVSTAT+2 and DVSTAT+3 (02EC and 02ED hex) and the length of the area in DVSTAT and DVSTAT+1 (02EA and 02EB hex). (The application may allocate the minimum area by leaving DVSTAT and DVSTAT+1 alone.) If the even starting boundary cannot be assured by the application, it must allocate one more byte than it reports in DVSTAT/DVSTAT+1. The application signals the completion of these steps by setting the flag HNDLOD (02E9 hex) nonzero.

The handler load shall occur automatically when the application calls CIO to perform any I/O operation except CLOSE via the "provisionally" open IOCB, when HNDLOD is nonzero (the CLOSE command shall simply close the IOCB without loading the handler). The steps taken by CIO shall be as follows:

1. The IOCB is checked to see is it is provisionally open. If it is not, normal I/O processing continues;

2. If the IOCB is provisionally open, the flag HNDLOD is checked. If the flag is zero, error 130, Non-existent Device, is returned;

3. If the IOCB is provisionally open and HNDLOD is nonzero, the handler is loaded (using the procedure of section 3.2.1.2) and linked (using the procedure of section 3.2.1.3). Prior to the load, the load address in DVSTAT+2 & DVSTAT+3 is forced even. The initialization process is called with the MEMLO parameter off, indicating that the handler size is not added to MEMLO;

4. If the loading or initialization fails, the IOCB is closed and error 130, Non-existent Device, is returned;

5. If the loading and initialization succeeds, the IOCB is modified to indicate it is properly opened:

   - Handler ID, ICHID, is set to point to the proper handler table entry. If the entry is not found, error 130, Non-existent Device, is returned, and the IOCB is closed;

   - Normal CIO OPEN processing is performed, thus filling the IOCB properly, including the put address ICPTL, ICPTH which is set to point to the handler put-byte entry. Additionally, the handler OPEN entry point is called by CIO.

6. CIO completes processing of the I/O command originally called by the application.

[Note: it is extremely important that the application not modify the handler once it has been loaded. Users of high-level languages such as BASIC or PASCAL must remain aware of how the language environment, particularly the language memory useage, may affect the handler. DOS 2 users must be aware that the DUP overlays memory whicl could contain I/O handlers. [SYSTEM.RESET] "uses" loaded handlers vi the process of reinitialization; therefore, system reset processing could fail if any loaded handlers have been modified. Also note that unpredictable results will occur should the handler be loaded more than once by an application.]

### 3.2.3 SYSTEM RESET (WARM START) REINITIALIZATION

This section describes the sequence of events taken by the operating system during system reset (warm start) reinitialization. This consists of actions which have existed in the 400/800 revision B operating system plus new operations which are the SWEET16 enhancements being described in this document. Only that degree of detail needed here is included.

1. The OS shall set the warm start flag WARMST (0008 hex) to FF hex;

2. Certain variables in the OS database are cleared to zero. RAM outside the OS database is left untouched. In particular, the handler table and all IOCB's shall be zeroed;

3. MEMLO (02E7 and 02E8 hex) shall be set to 0700 hex;

4. OS resident handlers shall be initialized and entered into the handler table;

5. The application cartridge "A" shall be initialized, if present;

6. Cassette or disk initialization shall occur (CASINI or DOSINI). At this time, the DOS updates MEMLO by adding its size, and any handlers within the DOS are initialized and entered into the handler table;

7. Upon return from the cassette-booted or disk-booted reinitialization, the operating system shall reinitialize all handlers which have been loaded into RAM. The procedure is described in detail below;

8. The OS shall start the cartridge or jump through DOSVEC.

To perform the initialization of loaded handlers (step 8 above), the operating system shall proceed as follows:

1. The internal pointer CHLINK is checked to see if any handlers have been loaded. This pointer is null (zero) if there are no loaded handlers, or it points to the linkage table of the first such handler;

2. If a loaded handler exists, its linkage table checksum is calculated and checked. If the sum is not two's-complement zero, the handler has been destroyed and this portion of the OS initialization terminates (no error is reported);

3. If the linkage table checksum is OK, the handler is re-linked and re-initialized according to the procedure of steps 2 through 6 of section 3.2.1.3; the MEMLO parameter is set on so that the handler size will be added to MEMLO;

4. If an error occurs while re-initializing the handler, this portion of OS initialization is terminated (no error is reported);

5. The forward pointer for the handler linkage table chain in this handler's linkage table is checked. If it is null (zero), this phase of initialization is complete. If it points to another handler, steps 2 through 5 are repeated for each handler in the chain.

### 3.2.4 SUBROUTINE INTERFACES

Three subroutines are added to aid the initialization process for loaded handlers. The first searches the handler table for an empty slot and makes the entry for the handler. The second follows the handler linkage table chain to remove a handler from the chain. The third performs initialization processing for a loaded handler.

All three routines are called via JSR to the appropriate entry vectors (below). All parameters are passed through the machine registers.

The entry addresses for these routines shall be as follows:

| | |
|---|---|
| E489 hex | Handler Entry Routine |
| E48C hex | Handler Linkage Removal Routine |
| E48F hex | Handler Initialization Routine |

Parameters for the HANDLER ENTRY ROUTINE are as follows:

X: Handler name;

A: High byte of linkage table start address;

Y: Low byte of linkage table start address.

This routine shall search the handler table from start to the first empty slot. If no empty slot is found (the table is full), error status is returned to the handler (see below). If a duplicate handler name is found, a different error is returned (also see below). If neither of these error occurs, the handler entry is inserted into the table at the first empty slot.

If the entry was successful made, the Carry bit shall be cleared on return to the handler.

If the handler table is full, error return shall be indicated by setting the carry bit. This error is distinguished from the duplicate-entry error by also setting the Negative bit. The registers shall be undefined when this return is made. The handler should not proceed with initialization; see section 3.2.1.3.

If there is a duplicate handler name in the table, the condition shall be indicated to the calling handler by returning with Carry set and Negative clear. In this case the A and Y registers shall be returned to the handler unchanged from the call, and the X register shall be an offset, relative to the first byte of the handler table, pointing to the second byte of the 3-byte table entry where the matching device name was found. The handler has the choice of discontinuing initialization, replacing the older handler entry, or chaining itself in (replacing the old entry but saving it in order to call the older handler whenever an I/O call belongs to the older handler).

The HANDLER LINKAGE REMOVAL subroutine requires the following parameters:

A: High byte of address of handler linkage table;

Y: Low byte of address of handler linkage table.

This subroutine shall search the handler linkage table chain for the linkage table having the address passed in A and Y. The linkage table checksums shall be computed and checked along the way to verify the integrity of the chain. When the proper linkage table is found, the handler size is checked to determine whether or not the handler was loaded at MEMLO. If the handler size is nonzero, the handler was loaded during power up at MEMLO, and it is illegal to remove it. In this case, the subroutine shall return with the Carry set. Otherwise the linkage table shall be removed from the chain by copying its forward chain pointer contents into the forward chain pointer of its predecessor in the chain.

If the chain search terminates either by finding the end of the chain (null pointer) or a bad linkage table, no action is taken and the Carry bit is returned set to indicate the error. Carry is cleared to indicate that the table was found and removed. The other registers are undefined upon return.

This subroutine is supplied to allow an application to request removal of a previously loaded handler when it is no longer needed or when the RAM must be reclaimed. It is suggested that the handler CLOSE routine check the flag HNDLOD (02E9 hex); it may be set nonzero by the application before CLOSE to indicate that the application wishes the handler unloaded. The handler is responsible for removing itself when unloading is requested: the handler table entry should be deleted, and the linkage table must be removed from the chain. The IOCB byte ICHID may be used to find the handler table entry, and this subroutine is used to remove the link from the chain. [Note: The OS variable COLDST shall be interrogated by this routine to determine when the caller is the operating system itself at cold start time. In this case, the handler shall be unlinked even though it is loaded at MEMLO.]

Note that the handler must NOT remove itself if it has been loaded at MEMLO. This is the reason that this subroutine checks the handler size for application-loaded handlers. If the handler receives error status from this subroutine, it should NOT remove itself from the system (except it is still permissible to remove the handler table entry).

Handler table removal is done by zeroing the device name byte in the handler table.

An INITIALIZATION subroutine entry point is included in the OS to provide the handler initialization function to be easily performed when handlers are loaded by a non-OS routine, for example by AUTORUN.SYS.

The INITIALIZATION subroutine performs all the tasks (steps 1-6) for initialization described in section 3.2.1.3. This routine requires the following parameters:

A: High byte of address of handler linkage table;

Y: Low byte of address of handler linkage table.

In addition, the Carry bit must be set by the caller to indicate whether the handler size should be added to MEMLO: Carry set on means the subroutine shall allow the adding of the handler size to MEMLO. Carry clear means the handler size shall be zeroed, thus suppressing its addition to MEMLO.

This subroutine shall return to its caller with Carry set if linking error occurred (and the linking is not performed). Carry shall be clear if linking was successful.

## 3.3 PERFORMANCE REQUIREMENTS

There are no performance requirements.

## 3.4 DESIGN REQUIREMENTS

Does not apply.

## 3.5 PACKAGING REQUIREMENTS

There are no packaging requirements.

## 3.6 SPECIAL REQUIREMENTS

There are no special requirements.

## 4.0 APPLICABLE STANDARDS

There are no applicable standards.