SWEET 16 OPERATING SYSTEM

EXTERNAL REFERENCE SPECIFICATION

SUPPLEMENT 2

-Relocating Loader-

By Amy Chen

3-29-82

REV A (9-11-82)

Functional Requirement

1.0 FUNCTIONAL DESCRIPTION

The Relocating Loader shall reside in the Sweet 16 O.S. Rom acting as a system subroutine.

The Relocating Loader shall relocate object code.

Since the Loader will reside in the OS ROM and since there is limited space available in the OS ROM, the Loader will not perform extensive error checking on the user object file. The Loader shall assume that the user object file is correctly formated as per this specification. Incorrecly formatted object files will produce undefined results. (Note: this implies that programs which produce object files must be extensively tested to insure that their output format is correct to this specification.)

2.0 LOADER INTERFACE

2.1 Interface Overview

The Loader user will set up a parameter block and execute a 6502 JSR to the Loader. The Loader shall relocate code based on the parameters. When the Loader has finished its task, it shall return to the user with a condition code and certain parameters.

One of the parameters passed to the loader will be GETBYTE address which shall be used by the loader to obtain data bytes for relocation.

The following is a schematic overview of the Loader process.

```
           +-------------------+                +---------------------+
+---->-----| user main program |                | get byte subroutine |
|          +-------------------+                +---------------------+
|     +--->----->----|                                |           |
|     |              |                                |    +-----------+
|     |       +--------------+                        |    | byte in A |
|  +------+   | need a byte  |------->--------->------+    | register  |
|  |relo- |   +--------------+                             +-----------+
|  |cation|          |                                           |
|  +------+          |                                           |
|     |       +??????????????+                                   |
|     |  NO   |    error ?   |-----------<------------<-----------+
|     +--<----|              |
|             +??????????????+
|                    | Yes
|         +--------------------+
|         | sets return call   |
|         | parameter block    |
|         +--------------------+
|                    |
|                    |
+----<-------<-------+
```

2.2 User-Loader Parameter Block

The user sets up a parameter block before initiating a relocating execution. The Loader shall accept the following 5 byte parameter block:

```
                +-----------------------------+    Byte  0
    low byte    |                             |
                +          GBYTEADR           +
    high byte   |                             |
                +-----------------------------+          2
    low byte    |                             |
                +          LOADADR            +
    high byte   |                             |
                +-----------------------------+          4
    1 byte      |          ZLOADADR           |
                +-----------------------------+          5
```

This block shall be interpreted as follows:

GBYTEADR (get byte address): The 2 byte address of the entry point of the Get Byte subroutine provided by the user.

LOADADR (non-zero page loading address): The 2 byte memory base address where all relocatable non-zero page machine instructions and data shall be relocated.

ZLOADADR (zero page loading address): The 1 byte zero page base address where all the relocatable zero page machine instructions and data shall be relocated.

The User-Loader Parameter Block shall be located at $2CF in the OS RAM data base.

2.3 User GETBYTE Subroutine

The user supplied GETBYTE subroutine address shall be called via the 6502 JSR instruction by the Loader to get data bytes and control information for the loader to process. Each time the loader calls GETBYTE the user will return a value in the 6502 A register. The loader shall check the condition of the 6502 carry (C) flag upon returned from the user GETBYTE subroutine. If the C flag is true (one) the Loader shall exit, otherwise the Loader shall process the new data byte. The value returned shall be interpreted by the loader as the next data or control byte.

2.4 Loader Return Condition Codes

When the Loader has finished its task, it shall return to the user via the 6502 RTS Instruction. Upon return to the user the contents of the 6502 Y register shall contain:

$01 (HEX) - indicating no errors. The LOADER-To-User parameter block contains valid information.

----- or -----

$9C (HEX) - Incomplete execution has occurred. This condition shall be returned when GETBYTE returns with the carry set.

$9D (HEX) - A memory insufficient error has occured. The return parameter block has no valid information.

The N flag shall be set according to the content of the Y register.

2.5 Loader-user Parameter Block upon successful operation return to the user, the Loader shall set up a Loader-User parameter block. The User will accept the following 5 bytes parameter block:

```
            +--------------------------+
low byte    |                          |    Byte  0
            +          RUNADR          +
high byte   |                          |
            +--------------------------+          2
low byte    |                          |
            +          HIUSED          +
high byte   |                          |          4
            +--------------------------+
1 byte      |         ZHIUSED          |          5
            +--------------------------+
```

RUNADR (execution entry point): This shall be the absolute entry point produced by the Loader from the information in the END record. If the RUNADR value is zero, then the End record did not indicate a run address. If the RUNADR is non-zero, the user may transfer to this address to start program execution.

HIUSED (highest non-zero page used): Shall contain the address of the next available memory location. (The highest address plus one used by the Loader).

ZHIUSED (highest zero page used): Shall contain the address of the next available zero page address.

The Loader-User parameter block shall be located at $2C9 in the OS RAM data base.

3.0 Record Structure

The relocatable object file consists of a sequence of one or more object segments. An object segment consists of a single Text record followed by one or more Information records.

The Loader shall treat the information passed via GETBYTE as object segments. The text is a sequence of machine instructions and data. The relocation Information shall determine which words or bytes in the Text will be relocated.

The Loader shall read the object Text into the allocated space computed from the load address provided by the User. The Loader shall then modify the contents of the object text based on the data contained in the relocation information. If there is to be no Text modification, then no relocation Information is required. Text records shall immediately proceed the relocation Information records. One can think of a Text record and its following Information records as one segment. There shall be no limit on the number of segments or the sequence in which the segments are arranged. An object file shall begin with a Text record and shall end with an End Information record. The Loader shall exit immediately after the END Information record is processed.

4.0 RECORD FORMAT DEFINITION

The records format for the object Text records and relocation Information records shall be interpreted as one of the 12 record types illustrated below:

4.1 Text Records

4.1.1 Text Record Types

0. Non-zero Page Relocatable Text

   Non-zero page object code that requires relocation.

1. Absolute text.
   Object code which does not require relocation.

2. Zero Page Relocatable Text

   Zero-page object code that requires relocations.

4.1.2 Text Record Format

```
+-------------------------+---------------//------+
| Type  |  Length  | Relative addr. |  Object Text |
|  ID   |          |       or       |              |
|       |          | Absolute addr. |              |
+-------------------------+---------------//------+

                       low & high

   1        1            2                0-253
  byte     byte        bytes              bytes
```

Relative Address - The term relative address refers only to non-zero page and zero-page relocatable texts and not absolute address. The 2 byte address is specified as an offset from the beginning of the assembled base address which is assumed to be 0.

Absolute Address - This term is only for absolute text. The following object text is to be loaded into specific portion of memory beginning at the absolute address.

Length - Counting from the byte after Length till the end of the record, not including TYPE ID and Length. Correct values shall range from 2 to 255.

Text Record Format (Cont')

Type ID - Each text type has a different type identifier:

Type	Type ID
Non-zero Page Relocatable Text	00
Zero-Page Relocatable Text	01
Absolute Text	0A

4.2 Information Records

Each information byte is an offset that points to the byte of the text record whose final contents needs modification.

4.2.1 Information Record Types

3. Non-zero page low byte references to non-zero page

   Points to the low byte data in the non-zero page Text record for relocation.

4. Non-zero page high byte references to non-zero page

   Points to the high byte data in non-zero page Text record for relocation.

5. Non-zero page word references to non-zero page

   Points to the first byte of a two byte (low, high) address in the non-zero page Text record for relocation.

6. Non-zero page one byte references to zero page

   Points to the zero page one byte value in the non-zero page Text record for relocation.

```
                .ORG $600      ;non-zero page area
                 LDA ZVAR
                .ORG $40       ;zero page area
ZVAR             LDA XXX
```

Records 7 through 10 are explained by means of examples as follows:

7. Zero page word references to non-zero page

```
                .ORG $40       ;zero page area
                 LDA VAR
                .ORG $600      ;non-zero page area
VAR              LDA XXX
```

Information Record Types (Cont')

8. Zero-Page low byte reference to Non-Zero Page

```
              .ORG      $40           ;ZERO PAGE AREA.
ZVAR          .BYTE     #.LOW.NZVAR   ;GET LOW BYTE
              .ORG      $600          ;NON-ZERO PAGE AREA.
NZVAR          LDA      $FF
```

9. Zero-Page high byte references to Non-Zero Page

```
              .ORG      $40            ;ZERO PAGE AREA.
ZVAR          .BYTE     #.HIGH.NZVAR   ;GET HIGH BYTE.
              .ORG      $600           ;NON-ZERO PAGE AREA.
NZVAR          LDA      $FF
```

10. Zero page one byte references to zero page

```
              .ORG      $40            ;ZERO PAGE AREA.
               LDA      ZVAR
NZAR           LDA ...
```

4.2.2 Information Record Format

```
    Text Record                         Word Information Record


    +-------+                           +--------------------+
    |type   |                           | type               |
    +-------+                           +--------------------+
    |leng.  |                           | length             |   offset
    +-------+                           +--------------------+   info.
    |rel-   |                 +--<------| 1                  |
    |ative  |                 |         +--------------------+
    |addr.  |                 |    +----| 4                  |
    +-------+                 |    |    +--------------------+
0   | LDA   |                 |    |
    +-------+                 |    |
1   |L. Var|-<----------------+    |
    |       |                      |
2   |H. Var|                       |
    +-------+                      |
3   | STA   |                      |
    +-------+                      |
4   |L. SYM|-<---------<-----------+
    +       +
5   |H. SYM|
    +-------+
```

4.2.3 Word, Low Byte and One Byte Record

Type ID	Length	Offset 1	Offset 2	Offset 3
1 byte	1 byte	1 byte	1 byte	1 byte

Type	Type ID
Non-zero page low byte references to non-zero page	02
Zero page low byte references to non-zero page	03
Non-zero page one byte references to zero page	04
Zero page one byte references to zero page	05
Non-zero page word references to non-zero page	06
Zero page word references to non-zero page	07

Length Byte

The length byte specifies the number of offset bytes following the length byte. The value ranges from 1 to 253.

Processing Algorithm

The word, low byte and one byte records shall be processed as follows:

1. (The proceding text record has been read and its object text has been placed into RAM at the appropriate displacement from the base address).

2. Each offset is used to obtain a data value (one or two bytes, depending upon the record type) from the proceding object text record.

3. The base address is added to the data value obtained.

4. The resulting value (one or two bytes, depending upon the record types) replaces the data value at the specified offset.

4.2.4 High Byte Record

The processing of a high byte data needs information about the low byte data, therefore the low byte of a full word data is recorded into the record. The format of this type is:

Type ID	Length	Offset 1	Low byte 1	Offset 2	Low byte 2
1 byte	1 byte	1 byte	1 byte	1 byte	1 byte

<u>Type</u>	<u>Type ID</u>
Non-zero page high bytes references to non-zero page	08
Zero page high bytes references to non-zero page	09

<u>Length</u>

The number of record bytes following the length byte. The value may range from 0 to 255.

<u>Processing Algorithm</u>

The processing of the High Byte record shall be as follows:

1. (The proceding text record has been read and its object text has placed into RAM at the appropriate displacement from the base address).

2. Each offset is used to obtain a single byte data value from the preceding object text.

3. The base address is added to the two byte value of which the most significant byte is the data byte obtained and the least significant byte is the low byte from the record.

4. The most significant byte of the resulting value replaces the data value at the specified offset. The least significant result value byte is discarded.

4.3 End record

This END record (type = 11) holds information about the self-start flag and the execution entry point. The self-start flag defines the nature of the execution entry point. The entry point value will be one of the three types: 1) An absolute address. 2) A relative start adress, that means a relocation bias has to be added before the address becomes an absolute address. 3) program execution is not desired after relocation, there fore the entry point value is 0. The Loader shall return to the user after this record is read and processed.

4.3.1 END Record Format

```
+-----------------------------------+---------+
| Type ID |  Self-start  |  Execution entry   |
|         |   flag       |       point        |
+-----------------------------------+---------+
                            low   &  high
 1 byte      1 byte         2 bytes
```

The value in the self-start flag is as follows:

Value	Interpretation
00	Program execution after relocation is not needed. The 2 bytes value of execution entry point is ignored but must exist.
01	The start execution address is an absolute address.
02	The start execution address is a relative address. The Loader adds a relocation base to make the start execution address an absolute address.

Execution Entry Point - The content may vary depending the self-start flag described above. The record must contain these two bytes even if the self-start flag is zero. The value may range from 0 to $FFFF.

Type	Type ID
End record	OB

The data indicated in the END record shall be placed in the Loader-user Parameter Block as defined in 2.3. The loader's caller shall then be returned to via the RTS instructions.

5.0 LOADER ENTRY POINT ADDRESS

The OS ROM data base shall contain a jump-to-loader instruction at location $E486.

6.0 BLOCK DIAGRAM FOR THE RELOCATING LOADER

The Loader shall process each type of record according to the following block diagram:

```
                  +---------------+         +---------------------------+
                  | Non-zero page |         | load text into specified  |
               +-->| relocatable   |----->----| address + base address    |----->+
               |  | text          |         |                           |      |
               |  +---------------+         +---------------------------+      v
               |                                                               |
               |  +---------------+         +---------------------------+      |
               |-->| Absolute      |----->----| load text into the        |---->-|
               |  |   text        |         | absolute address          |      |
               |  +---------------+         +---------------------------+      |
               |                                                               |
               |  +-----+   +----------------------------------------+         |
               |  | Low |   |       fixup instructions/data referring to |     |
               |-->| byte|--> |       low byte address.                |---->|
               |  |     |   |                                        |         |
               |  +-----+   +-------------------+--------------------+         |
               |                                                               v
+----+   +-+   |  +-----+   +-------------------------------------------+      |
|CALL|   |T|   |  |High |   |       fixup instructions/data referring to |     |
|GET |-->|Y|-|-->|byte |--> |       high byte address.                  |->|
|BYTE|   |P|   |  |     |   |                                           |      |
|SUB.|   |E|   |  +-----+   +-------------------------------------------+      |
+----+   |?|   |                                                               |
         +-+   |  +-----+   +------------------------------------------+       |
          |    |-->|one  |--->|       fixup instructions/data referring to |--->|
          |    |  |byte |   |       one byte variable.                  |       |
          |    |  |     |   +------------------------------------------+       |
          |    |  +-----+                                                      |
          |    |  +-----+   +-------------------------------------------+      |
          |    |-->|Word |--->|    fixup instructions/data referring to a full|->|
          |    |  |     |   |    variable.                              |      |
          |    |  +-----+   +-------------------------------------------+      |
          |    |                                                               v
          |    |  +---------------+         +---------------------------+      |
          |    |  | Zero-Page     |         |  execution concept is     |      |
          |    |-->| related record |--->-----|  the same as none-zero page |---->-|
          |    |  | types.        |         |  record types             |      |
          |    |  +---------------+         +---------------------------+      |
          |    |                                                               |
          |    |  +-------+   +-----------------------------+                  |
          |    |-->| End   |---->| fixup executive entry point |--------->----|
          |       +-------+   | and sets up parameter block |                  |
          |                   +-----------------------------+                  |
          |                                                                    |
          |                                                                    v
          <-------------------------------<-------------------------------------+
```