```
                ;MOVER MOVES B BYTES FROM THE ADDRESS IN HL TO
                ;THE ADDRESS IN DE
FC30 78         MOVER:  MOV     A,B
FC31 B7                 ORA     A       ;IS LENGTH ZERO?
FC32 C8                 RZ              ;YES - RETURN TO XYBASIC
FC33 7E         MOVES:  MOV     A,M     ;FETCH SOURCE BYTE
FC34 12                 STAX    D       ;STORE IT
FC35 23                 INX     H       ;INCREMENT SOURCE POINTER
FC36 13                 INX     D       ;INCREMENT DEST POINTER
FC37 05                 DCR     B       ;DECREMENT BYTE COUNTER
FC38 C233FC             JNZ     MOVES   ;KEEP MOVING
FC3B C9                 RET             ;RETURN TO XYBASIC
                ;GETIP GETS AN INTEGER PARAMETER TO DE.
                ;JUMPS TO ERROR IF PARAMETER IS NOT
                ;A SIMPLE INTEGER VARIABLE.
FC3C CD0301     GETIP:  CALL    GTPAR   ;GET A PARAMETER
FC3F FE01               CPI     1       ;INTEGER?
FC41 C26801             JNZ     ERROR   ;NO - TAKE ERROR EXIT
FC44 79                 MOV     A,C     ;EXPECTING SIMPLE VARIABLE
FC45 B7                 ORA     A       ;ARRAY PASSED?
FC46 C26801             JNZ     ERROR   ;YES - TAKE ERROR EXIT
FC49 5E                 MOV     E,M     ;LOW BYTE OF VALUE
FC4A 23                 INX     H       ;POINT TO HIGH BYTE
FC4B 56                 MOV     D,M     ;HIGH BYTE
FC4C C9                 RET             ;RETURN TO CALLER
```

With these routines resident in memory, it is a simple matter to save the contents of a string variable in nonvolatile memory (say, at address 0C000H). The string variable DATE$ could be saved as follows:

```
NVM% = #C000: CALL #FC00, NVM%, DATE$
```

It could be recalled at power-up time by:

```
NVM% = #C000: DATE$ = "        ": CALL #FC1A, NVM%, DATE$
```

Note that a string of blanks is assigned to DATE$ before the call to RCLVAR. This is an absolutely necessary formality, because storage space for string variables is allocated dynamically by XYBASIC. This allows the length of a string variable's value to vary dynamically without wasting memory space. If a string variable contains the null string (as it does if it has not been previously defined), no storage space has been allocated for it. Thus a string variable must be assigned a string of length at least as great as the length of the string to be recalled before the CALL to RCLVAR. This extra step is necessary only for string variables.

Another example using GTPAR is included in Section 3 of Chapter II, in the paragraph Saving and Loading Under Operating Systems.

SCALL

The SCALL (Short CALL) command is similar to CALL, except that parameter values are passed to and from your assembly language routine directly (through registers) rather than with GTPAR. This makes linking to an assembly language routine faster and easier. However, the parameters which are passed must be integers; you must use CALL rather than SCALL to pass