TWIN OPERATING SYSTEM

1. Table of contents

2. Introduction

This document contains a brief description of the Operating System that is currently available for the TWIN. This operating system is intended to execute, upon demand, a number of functions. These functions concern mainly system control and file management. The execution of functions can be requested by means of commands and supervisor calls. Commands are entered via the keyboard or they are read from a command file (a disk file containing commands). Supervisor calls are issued by programs.

The memory of the TWIN is partitioned in two main parts:

1. Master memory
   The master memory contains the locations with addresses H'0000' - H'3FFF'. Only the master CPU has access to this part of the memory.

2. Common memory
   The common memory can be accessed by both the master CPU and the slave CPU. It can contain up to 64 k (k = 1024) bytes. The slave CPU uses the addresses H'0' - H'FFFF' to address this memory.

   For the master CPU this memory consists up to 4 memory banks of 16 k locations each. The master CPU can select one bank at a time. It uses the addresses H'4000' - H'7FFF' for the selected bank.

The memory is also partitioned in another way, viz. in a number of overlay areas. To each overlay area a job is assigned. The overlay areas are:

1. Overlay area 1
   This area occupies the addresses H'1500' - H'187F' of the master memory. The corresponding job is job 1.

2. Overlay area 2
   This area contains the addresses H'1880' - H'1BFF' in the master memory. Job 2 is associated with this area.

3. Overlay area 3
   This area is located in the Common Memory. The corresponding job is known as job 3 or "slave job".

The remainder of the address space H'0' - H'3FFF' is the resident OS area. Job 0 operates in this area. Any of the jobs 1, 2 and 3 can also execute routines which are located in the resident OS area.

Each job executes functions which have certain specific characteristics. These characteristics are:

1. Job 0
   Command line processing, loading of overlays and signalling of errors.

2. Job 1
   System programs which may require an considerable amount of time to be executed. A special program to be executed in this area is DEBUG.

3. Job 2
   System programs which may also be executed when the system is in DEBUG-mode. Overlay area 2 can also be used by job 1.

4. Job 3
   Job 3 is associated to all programs that are executed by the slave CPU.

The progress of a job is indicated by the job state. The following states exist:

- 0: idle
  There is no program to be executed and the job will not be selected for performance.

- 1: loaded
  A program has been loaded, but the execution of this program has not been requested.

- 2: ready to start
  The execution of the program is to be started on first occassion. The address of the first instruction is given by the initial start address.

- 3: executing
  A program is being executed.

- 4: I/O wait
  The job is waiting for the completion of some input/output operation. Jobs in "I/O wait" will not be selected for performance.

- 5: I/O complete
  The execution of the job is to be resumed on first occassion. The address of the next instruction is given by the I/O complete return address.

- 6: suspended
  The job shall not be selected for performance. The job returns to its preceding state when the system command CONT is executed for the job.

- 7: aborted
  The job shall be terminated on first occasion. When the job has assigned channels, these channels will be closed first.

- 8: paused
  The job shall not be selected for performance. Certain specific operations will set the job state to its preceding value. This state is used with DEBUG, EXAM and DUMP.

The OS-programs can be partitioned as follows:

1. Command line processing
   The command line processing programs deal with the system commands which are entered via the console input or which are read from a procedure file.

   The command line processing comprises the determination of the specified system commands, loading of the associated programs and initialization of the related jobs.

2. Supervisor calls (SVC)
   Many standard operations (such as input/output) are controlled via SVC's. A SVC is identified by a SRB (service request block). A SRB describes the requested operation and it will contain the termination status. Any job can issue SVC's.

3. File management
   All input/output operations are related to files and abstract devices. Upon the initiation of input/output a file is mapped into a particular peripheral device. This apporach frees the user from most peripheral characteristics.

   The file management programs are a part of the SVC-programs as well as a part of the device handling programs.

4. Device handlers
   A device handler corresponds to a particular peripheral device. The system includes device handlers for:

   - Line printer;
   - Floppy disks (sector disks);
   - Teletype and console;
   - Papertape equipment.

5. Dispatcher
   The dispatcher is responsible for the selection of the next function to be executed. The dispatcher distinguishes the following classes of functions:

   - Device handlers;
   - Job 0;
   - Job 1, job 2 and job 3;
   - Procedure.

It is assumed that the reader is familiar with the TWIN Operator's guide.

3. File management

The TWIN Operating system is a file-oriented system. With the file management the concepts of "file", "device" and "channel" are used.

A file is a set of data. A file can be processed sequentially only; processing always starts with the first data item and it is terminated with the last data item. A file is identified by its name, the file name. In the TWIN system the files are stored on floppy disks (diskettes).

Devices are physical peripherals which provide for input/output services. The TWIN system can contain up to 16 devices. The standard devices have reserved names through which the user can access them. These names cannot be employed for file names. The reserved names with the associated peripheral devices are:

device name	device
CONI	console input
CONO	console output
TTYR	teletype reader
LPT1	line printer
LPT2	
HSPT	
PTPU	papertape punch
PTRD	papertape reader

Files may also be viewed as devices. A file can be specified as input or output device. To refer to a file as a device, the file name of that file serves as the reference. The file name can be appended by the number of the involved disk drive.

Channels are used by the software to refer to a device. Each job can use 8 channels. These channels are numbered 0 through 7.

The following operations can be executed for the devices:

1. Assign
   To assign a given channel to a particular device or disk file.

2. Read ASCII
   To retrieve a string of bytes from a device. The string is terminated when the associated input area is full or when the ASCII control character CR (H'0D') is encountered. In the first case a CR is stored in the location following the input area. With input from CONI, TTYR and PTRD the following ASCII control characters are recognized and processed:

   - NUL (H'00')
     The byte is ignored.

   - LF (H'0A')
     The byte is ignored.

   - DEL (H'7F')
     The preceding byte (if any) is deleted from the input area.

- CR (H'0D')
  This byte is the last byte of the current string.

- ESC (H'1B')
  All data in the input area are deleted and this byte is treated as CR.

- CTRL-Z (H'1A')
  This byte is replaced by CR and the end-of-file condition is set.

3. Read binary
   To retrieve a string of data from a device. The string is terminated when the associated input area is filled. No CR is placed at the end of the string and control characters are not processed.

4. Write ASCII
   To send a string of bytes to a device. The string is terminated when the associated output area is exhausted or when the ASCII control character CR has been transmitted. With output to PTPU the most significant bit is used as parity bit. This bit is set such that parity is even.

5. Write binary
   To send a string of bytes to a device. The string is terminated only when the associated output area is exhausted.

6. Close
   To terminate the operations on a device and to release the used channel.

The preceding read and write operations can be executed in two modes:

- Execute and wait
  The next instruction of the program that has requested the operation is executed after the completion of the operation.

- Execute and proceed
  The next instruction of the program that has requested the operation is executed after the operation has been started, so this program can proceed in overlap with the requested operation.

For disk files also the following operations are useful:

7. Rewind
   To position a disk file at its beginning. When a disk file has been "rewound", it is treated in the same way as it had just been "assigned".

8. Rename
   To change the file name of a disk file. Only disk files that had just been "assigned" or "rewound" can be renamed.

9. Delete
   To delete a disk file from a diskette and to release the used channel.

4. Device handlers

The TWIN system contains a unique device handler for each type of periperhal device. Device handlers for the following peripherals are included:

- floppy disk drives;
- line printer;
- papertape punch;
- papertape reader;
- console input (including teletype reader);
- console output.

These device handlers with their specific functions are described in the following subsections.

4.1. Floppy disk drive

A diskette contains 77 tracks on which data can be written and from which data can be read. Each track is partitioned into 4 blocks. A block comprises 8 sectors. 128 bytes of data can be stored in a sector.

To locate a file on a diskette, the directory of that diskette is used. The directory resides on track 0. The sectors of a directory are used as follows:

```
    sector  0:     bytes   0-1:    diskette identity;
                            2:      number of files on diskette;
                            3-51:   diskette identification;
                           52-89:   bad bit map;
                           90-127:  master bit map.

            1-5:   file names, each sector contains 16 slots for file names.

            6-31:  file bit maps; each sector contains 3 file bit maps.
                   the format of a file bit map is:
                   bytes   0-37:   bit map;
                          38-39:   number of sectors in the file;
                          40:      number of data bytes in last sector.
```

The diskette identity is a string of 16 bits that is used to identify the diskette. A bit map is a string of 304 bits. Each bit in a bit map corresponds to a block. When the bit is 1 the block is in use, if the bit is 0 the block is free.

From the directory format follows that a diskette contains 304 data blocks and that 78 files can be stored on a diskette.

The floppy disk drive handler can execute the following commands:

- ASSIGN
- READ
- WRITE
- CLOSE
- REWIND
- DELETE
- RENAME
- DIRECT I/O (privileged command)

With other commands the termination status H'05' (illegal function code) is returned.

4.1.1. ASSIGN

Upon the assignment of a channel to a disk file, the device handler starts to search the involved directories for the file name. If SEARCH (see 5.2) is "off", only the directory on the given drive (default system drive) is searched. Otherwise the directories on all drives in the system are involved.

If the file name has been found, the corresponding bit map is read; else a (empty) new file is created.

Termination status:

H'00' - FUNCTION COMPLETE
H'01' - CHANNEL ASSIGNED TO A NEW FILE
H'08' - ILLEGAL DRIVE NUMBER
H'09' - FILE IN USE
H'0A' - DEVICE NOT OPERATIONAL
H'0B' - DEVICE NOT AVAILABLE
H'0E' - DIRECTORY READ ERROR
H'0F' - DIRECTORY WRITE ERROR
H'10' - DIRECTORY FULL
H'16' - ILLEGAL FILE NAME

4.1.2. READ

The operation READ starts with reading of a sector into the buffer memory that is located in the floppy disk controller. Next these data are moved to the input area in store (either master memory or common memory).

Termination status:

H'00' - FUNCTION COMPLETE
H'05' - ILLEGAL FUNCTION CODE
H'09' - FILE IN USE
H'0A' - DEVICE NOT OPERATIONAL
H'11' - DEVICE READ ERROR
H'FF' - END OF FILE

4.1.3. WRITE

The operation WRITE starts with moving data from the output area in store (either master memory or common memory) to the buffer memory in the floppy disk controller. Next the contents of the buffer memory in the floppy disk controller is written on the diskette.

When data is to be written in an existing file, the original data in the file are deleted before the first write operation is executed.

Termination status:

H'00' - FUNCTION COMPLETE
H'05' - ILLEGAL FUNCTION CODE
H'09' - FILE IN USE
H'0A' - DEVICE NOT OPERATIONAL
H'12' - DEVICE WRITE ERROR
H'FF' - END OF DEVICE

4.1.4. CLOSE and REWIND

The function CLOSE differs for input files and output files. REWIND is identical to CLOSE for an input file. When an output file is closed, the last sector (if applicable) is written on the disk, the directory of the diskette is updated and the file administration is initialized. When an input file is closed, no data are written on the diskette but only the file administration is initialized.

Termination status:

H'00' - FUNCTION COMPLETE
H'0A' - DEVICE NOT OPERATIONAL
H'0E' - DIRECTORY READ ERROR
H'0F' - DIRECTORY WRITE ERROR
H'12' - DEVICE WRITE ERROR
H'FF' - END OF DEVICE

4.1.5. DELETE

With the command DELETE the file name is deleted from the directory and the blocks which were allocated to the file are released.

Termination status:

H'00' - FUNCTION COMPLETE
H'09' - FILE IN USE
H'0A' - DEVICE NOT OPERATIONAL
H'0E' - DIRECTORY READ ERROR
H'0F' - DIRECTORY WRITE ERROR

4.1.6. RENAME

The command RENAME can be executed successfully only for files which have just been assigned. With the execution of this command the file name of the file is changed. The new name is written in the directory of the diskette.

Termination status:

H'00' - FUNCTION COMPLETE
H'08' - ILLEGAL DRIVE NUMBER
H'0E' - DIRECTORY READ ERROR
H'0F' - DIRECTORY WRITE ERROR
H'15' - FILE NAME IN USE
H'16' - ILLEGAL FILE NAME

4.2. Line printer

The line printer handler controls the peripheral line printer. The program can execute the following commands:

- ASSIGN
- WRITE
- CLOSE

With other commands the terminations status H'05' (illegal function code) is returned.

The line printer can only be connected to one channel at a time, so it is not possible to merge a number of output streams on the printer.

4.2.1. ASSIGN

Upon the assignment of the line printer to a channel, the ASCII control characters FF and CR are sent to the peripheral printer in order to start the following output at the top of a new page. The line feed at the beginning of a line (see below) is supressed for the first line.

Termination status:

H'00' - FUNCTION COMPLETE
H'0A' - DEVICE NOT OPERATIONAL

4.2.2. WRITE

With the execution of the command WRITE, the contents of the output area is sent to the peripheral line printer. A distinction is made between "write ASCII" and "write binary".

With "write ASCII" each line will be preceded by the control character LF (except for the first line, see above) and the line is terminated when either the output area is exhausted or the control character CR is encountered.

With "write binary" a line is not preceded by the control character LF and the line is terminated only when the output area is exhausted.

Each line is terminated by the control character CR (with "write ASCII" as well as with "write binary").

Note: Underlining is possible with "write binary".

Termination status:

H'00' - FUNCTION COMPLETE
H'07' - NO EOL ON ASCII WRITE
H'0A' - DEVICE NOT OPERATIONAL

4.2.3. CLOSE

With the execution of the command CLOSE the device is released, so it becomes available for other users.

Termination status:

H'00' - FUNCTION COMPLETE

4.3. Papertape punch

The papertape punch handler controls the peripheral papertape punch. The program can execute the following commands:

- ASSIGN
- WRITE
- CLOSE

With other commands the termination status H'05' (illegal function code) is returned.

The papertape punch can only be connected to one channel at a time, so it is not possible to merge a number of output streams on a tape.

Note: The software cannot check the presence of papertape. The user has to assure that papertape has been loaded before the papertape punch is used.

4.3.1. ASSIGN

Upon the assignment of the papertape punch to a channel, a string of 128 bytes H'00' (tape feed) is sent to the peripheral device.

Termination status:

H'00' - FUNCTION COMPLETE
H'0A' - DEVICE NOT OPERATIONAL

4.3.2. WRITE

With the execution of the command WRITE, the contents of the output area is sent to the papertape punch. The commands "write ASCII" and "write binary" are executed in a different way:

1. Write ASCII
   The most significant bit of each outgoing byte is used as parity bit. Its original value is ignored and the new value is such that parity is even.

   The output stream is terminated when either the control character CR has been transmitted or the output area is exhausted.

2. Write binary
   All the bits in each outgoing byte are transmitted to the peripheral device without any modification.

   The output stream is terminated only when all the data in the output area has been sent to the peripheral.

The device handler does not accept both types of write commands for the same file; all write commands are to be either "write ASCII" or "write binary".

Termination status:

H'00' - FUNCTION COMPLETE
H'05' - ILLEGAL FUNCTION CODE
H'07' - NO EOL ON ASCII WRITE
H'0A' - DEVICE NOT OPERATIONAL

4.3.3. CLOSE

With the execution of the command CLOSE a string of 64 bytes H'00' (tape feed) is sent to the peripheral. With an ASCII file (write commands "write ASCII"), the tape feed is preceded by the character CTRL-Z (H'1A' end-of-file). This character is not punched in binary files.

After the completion of this transfer the punch motor is stopped and the device is released.

Termination status:

H'00' - FUNCTION COMPLETE
H'0A' - DEVICE NOT OPERATIONAL

4.4. Papertape reader

The papertape reader handler controls the peripheral papertape reader. The program can execute the following commands:

- ASSIGN
- READ
- CLOSE

With any other command the termination status H'05' (illegal function code) is returned.

The papertape reader can only be connected to one channel at a time.

4.4.1. ASSIGN

Upon the execution of the command ASSIGN, no specific operations are executed.

Termination status:

H'00' - FUNCTION COMPLETE

4.4.2. READ

With the execution of the command READ, a data string is retrieved from the peripheral papertape reader. This string is stored in the input area. A distinction is made between "read ASCII" and "read binary":

1. Read ASCII
   The input bytes are treated as ASCII characters of which the most significant bit is the parity bit. The parity bit is cleared and the resulting characters are stored at consecutive locations in the input area.

   The string is terminated when either the input area is full or the control character CR is encountered. In both cases CR is stored as last character. When the input area is full, CR is stored in the location following the input area.

   The ASCII control characters are processed too (see 3).

2. Read binary
   The input bytes are stored in the input area without any modification.

   The string is terminated only when the input area is full and no CR is added. Also no control characters are recognized.

Termination status:

H'00' - FUNCTION COMPLETE
H'06' - NO EOL ON ASCII READ
H'0A' - DEVICE NOT OPERATIONAL
H'11' - DEVICE READ ERROR
H'FF' - END OF FILE

4.4.3. CLOSE

With the execution of the command CLOSE, the motor of the papertape reader is stopped and the device is released so it becomes available for other users.

Termination status:

H'00' - FUNCTION COMPLETE

4.5. Console

The console handler consists of two closely related device handlers: the console input handler and the console output handler. The console input handler controls the console keyboard (or teletype reader), while the console output handler is responsible for the console display.

4.5.1. Console input handler

The console input handler recognizes only the command READ. With any other command the termination status H'00' (function complete) is returned. The console input handler distinguishes three types of read commands:

- read ASCII;
- read binary;
- read ASCII without echo.

With both "read ASCII" commands the ASCII control characters are processed as is described in section 3, but the control character ESC is processed further in order to detect consecutive ESC's (see 5.1.1).

With the ordinary "read ASCII" command each entered character is passed to the console output handler to be echoed on the display. This does not occur with "read binary" and "read ASCII without echo".

With both "read ASCII" commands the end of the input stream is detected when the input area is full or when the control character CR is found. In both cases CR is stored as last character. If the input area is full, CR is stored in the location following the input area.

With "read binary" the end of the input stream is detected when the input area is full. Then no additional data are stored.

Termination status:

H'00' - FUNCTION COMPLETE
H'06' - NO EOL ON ASCII READ
H'FF' - END OF FILE

Because the console is the system control device, the console input handler deals also with entered characters when it is not involved in a read operation. Then only the control characters ESC and SP are accepted (see 5.1.1 and 5.1.2).

4.5.2. Console output handler

The console output handler recognizes the commands:

- write ASCII;
- write binary;
- reserve console (privileged);
- release console (privileged).

With any other command the termination status H'00' (function complete) is returned.

Again a distinction is made between "write ASCII" and "write binary".

- with "write ASCII" the byte H'0F' (CTRL-O) is replaced by A'.', the ASCII control character LF is substituted for the control character FF and, if the end of the string is found, the control characters CR and LF are sent to the peripheral display. With "write ASCII the end of the string is attained when either the output area is exhausted or the control character CR is encountered.

- with "write binary" the byte H'0F' and the control character FF are not translated and, when the end of the string is found, no additional data are sent to the peripheral display. With "write binary" the end of the string is attained only when the output area is exhausted.

Termination status:

H'00' - FUNCTION COMPLETE
H'07' - NO EOL ON ASCII WRITE

The console output handler deals also with the console input data that are to be echoed. All characters are without any change translmitted to the peripheral display, except DEL, CR, FF and CTRL-O. These characters are replaced by the following data:

DEL: ESC - BS - SP - ESC - BS (erases character on screen)
CR: CR - LF (positions cursor at beginning of next line)
FF: LF (proceeds at next line)
CTRL-O: A'.' (provides for a "printable" character)

5. System commands

A system command consists of a command code followed by a number of operands, separated by blanks and/or comma's:

```
<cmdcode> [ <sep> <operand> ]
```

The command code may be preceded by a number of blanks. Blanks following a separator (this is the comma or blank next to the command code or an operand) have no meaning, the subsequent operand starts with the next non-blank character.

If the command line is read from a command file, the strings of 2 characters beginning with '$' are processed as follows:

1. $D
   The string '$D' is replaced by the current date;

2. $T
   The string '$T' is replaced by the current time;

3. $<n>
   The string '$<n>' is replaced by the procedure parameter designated by <n>. The serial number of the procedure parameter is given by the 4 least significant bits of the ASCII representation of <n>. <n> does not need to be a decimal number.

For most system commands two formats exist, the full format and the abbreviated format. The abbrivated format comprises only the first characters of the full format. In the following sections the abbreviated form is marked by a blank in the full form, for example:

```
A BORT
```

Which indicates the abbreviated form 'A' and the full format 'ABORT'.

With several system commands addresses shall be specified. Such an address may be given in the form of an expression. The syntax of an address expression is:

```
<addrexpr>     ::= <term> + <addrexpr>
                 ! <term>

<term>         ::= string of hexadecimal characters
```

The value of any term and the expression shall be in the range 0 - H'FFFF'. Leading zeros are allowed.

The system commands are partitioned into the following groups:

System control commands;
System options;
System utilities;
Object code utilities;
Debug commands;
PROM programming utilities;
Miscellaneous commands.

5.1. System control commands

The user can control program execution by means of the keys "ESC" and "SPACE BAR" and the control commands:

S USP
C ONT
A BORT
AS SIGN
CL OSE

5.1.1. ESC-key

The ESC-key can be used to request attention of the operating system. The system reactions depend on the use of the console:

1. Console in use for BINARY input:
   No system reaction.

2. Console in use for ASCII input:
   EOL is returned as the only entered character, see further below.

3. Console not in use:

   1. First ESC:
      A "prompt" will be issued on first occasion.

   2. Second ESC:
      An active job 1, an active job 3 and an active procedure are suspended; while an active job 2 is aborted. Next a "prompt" is issued.

      Note: The suspended jobs can be resumed by means of the system command CONT.

5.1.2. SPACE BAR

The space bar can be used to control the console output when the console input is not in use. Depressing the space bar once will stop output to the console; a second depression of this bar resumes the console output.

5.1.3. SUSP, CONT and ABORT

These commands can be used to control the state of job 1, job 3 and the procedure. The operands are optional. When the operands are omitted, the command concerns job 1, job 3 and the procedure. The possible operands are:

1: job 1
3: job 3
P: procedure

The operands may be supplied in any sequence and more than one operand is allowed.

SUSP suspends the execution of the designated job, CONT restores the suspended jobs to their preceding states and ABORT terminates jobs.

5.1.4. ASSIGN and CLOSE

See TWIN Operator's guide.

5.2. System options

The user may set the value of various system options with these commands:

```
SEA RCH
SY ST
DEV ICE
SL AVE
TMR
TIME
ST AT
TY PE
K ILL
*
```

5.2.1. SEARCH, SYST and DEVICE

See TWIN Operator's guide.

5.2.2. SLAVE

This command designates the active slave CPU and sets its mode of operation. The format is:

```
SLAVE <chip> [ <mode> ] [ <mem> ] [ <iolim1> <iolim2> ]
```

Only the first operand, <chip>, is obliged. This operand gives the name of the slave CPU. The allowed values are '2650' and 'Z80'.

The parameter <mode> designates the mode in which the slave CPU will operate. The allowed values for this parameter are:

0: TWIN mode
   Uses TWIN common memory and TWIN I/O.

1: Partial TWICE mode
   Uses TWIN common memory, user I/O and user clock.

2: Full TWICE mode
   Uses user memory, user I/O and user clock.

The default value of the operand <mode> is 0.

The parameter <mem> specifies the common memory block size and the common memory base. The possible values are:

```
0: no protection, base 0;
1: block size 16 k, base 0;
2: block size 16 k, base 16 k;
3: block size 16 k, base 32 k;
4: block size 16 k, base 48 k;
5: block size 32 k, base 0;
6: block size 32 k, base 32 k.
```

The default value of this operand is 1.

The operands <iolim1> and <iolim2> apply only to a Z80 slave CPU. These operands give the limits for the "memory mapped I/O".

When the Z80 slave operates in partial TWICE mode, references to addresses in the range <iolim1> - <iolim2> are considered as references to peripheral input/output devices and not as references to the common memory.

<iolim2> shall be higher than <iolim1>. The default value for these operands is 0. The default values suppress the "memory mapped I/O".

Note: The address of the slave CPU board cannot be specified. These addresses are related to the chip name, as follows:

- 2650: 'E0'
- Z80: 'E1'

5.2.3. TMR

The command TMR is introduced to control timer interruptions of the master CPU. The command requires a single operand. The possible values of the operand are 'OF', 'OFF' and 'ON'.

If the operand is OFF or OF, timer interruptions are inhibited and the time of the day is not displayed with EOJ-messages.

If the operand is ON, the time of the day is set to '00.00.00' and timer interruptions are enabled.

5.2.4. TIME

The command TIME can be used either to display the current time of the day or to allow timer interruptions of the master CPU and to set the time of the day. In the first case no operand is given. In the second case a single operand is required. The format of this operand is:

<digit> <digit> . <digit> <digit> . <digit> <digit>

In which <digit> is a decimal digit. The operand gives the current time of the day in hours, minutes and seconds.

5.2.5. STAT, TYPE, KILL and *

See TWIN Operator's guide.

5.3. System utilities

The user can perform disk and file maintenance and move data around the TWIN system with these commands:

```
FORMAT
V ER
DUP
LD IR
REN AME
DEL ETE
COP Y
PR INT
PRINTL
PRH
PU NCH
```

5.3.1. FORMAT, VER, DUP, LDIR, RENAME, DELETE and PRINTL

See TWIN Operator's guide.

5.3.2. COPY and PRINT

The commands COPY and PRINT can be used to copy files or data from devices to other files or devices, as is described in the TWIN Operator's guide.

The command PRINT always employs the input/output commands "read ASCII" and "write ASCII". The input/output commands which are used by COPY depend on the involved devices. Then both "ASCII"-commands and "binary"-commands are possible. The commands to use are determined by "ANDING" the bits 0 - 3 of the types of the devices (see 6.3.4). If "binary"-commands are allowed (bit 1 is a 1), then the commands "read binary" and "write binary" are used with a block length of 128. In all other cases the commands "read ASCII" and "write ASCII" are employed.

5.3.3. PRH

The command PRH can be used to print the contents of a file in hexadecimal characters. The format of the command is:

```
PRH <file> <device>
```

The first operand, <file>, is obliged. It specifies the file from which the data are to be taken. The second operand designates the output device. Its default value is 'LPT1'.

The command produces 2 output lines for every 32 bytes of data (16 bytes if <device> is 'CONO'). The first line contains the ASCII characters associated to the data bytes and the second line gives the values in hexadecimal notation.

5.3.4. PUNCH

The command PUNCH can be used to punch a readable text in papertape. The format of the command is:

```
PUNCH <data>
```

In which <data> represents the character string to punch. <data> may contain any ASCII character that is represented by a code in the range H'20' - H'5F'.

Each character is replaced by a string of 6 bytes (giving the original character in a readable form) and these strings are punched in papertape.

Note: The system command line processor considers <data> as a string of parameters. This results in the replacement of a comma by a blank and a number of consecutive blanks are replaced by a single blank.

5.4. Object code utilities

The user may manipulate files containing load modules and the data in memory with these commands:

```
G O
LO AD
X EQ
M OD
D UMP
E XAM
R HEX
W HEX
```

5.4.1. GO, RHEX and WHEX

See TWIN Operator's guide.

5.4.2. LOAD and XEQ

The commands LOAD and XEQ can be used to load, respectively load and execute, programs for the slave CPU. With these commands the system selects automatically the corresponding slave CPU (either 2650 or Z80).

The format and meaning of the operands of these commands are conform to the TWIN Operator's guide.

5.4.3. MOD

The command MODULE is to be employed when data in the common memory are to be stored in the form of a "load module". The format and the meaning of the operands of this command are described in the TWIN Operator's guide.

The type of the generated module depends on the current slave CPU. If the 2650 is the slave CPU, a 2650 load module is stored; if the Z80 is the slave CPU, a Z80 load module is formed.

5.4.4. EXAM and DUMP

The command EXAM is to be used to examine and to alter the contents of the memory. The command DUMP may be used to display the contents of the memory. Both commands can operate on the following memories:

- master memory;
- common memory;
- external user memory.

The master memory is selected if the address in the first operand is preceded (without any separation) by the character '*'. In other cases, either the common memory or the external user memory is involved.

The external user memory is accessed if the system is in full TWICE mode. In this case a little routine to access the external memory is placed in the common memory. The user shall reserve 135 bytes at the end of the first bank of the common memory for these routines. These routines are stored at the end of the first common memory bank, preceding the area that is reserved for the TRACE package (see DEBUG). The user cannot specify another load-address for these routines.

The format and meaning of the operands of the commands EXAM and DUMP are conform to the TWIN Operator's guide.

1. Note: The programs for the command EXAM and DUMP differ for a 2650 slave and a Z80 slave. This difference is significant only in the full TWICE mode.

5.5. Debug commands

The user can use the following commands with the debugging of programs:

```
DEB UG
TR ACE
RES ET
SE T
BK PT
CLB P
AC CNT
DS TAT
```

The implementation of the debug commands depends on the used slave CPU. The type of the slave CPU has also some consequences for the allowed operands of the debug commands.

5.5.1. DEBUG

See TWIN Operator's guide.

5.5.2. BKPT, CLBK and RESET

See TWIN Operator's guide.

5.5.3. DSTAT

The command DSTAT can be executed only when the corresponding slave CPU is active. With the execution of this command the following data are displayed:

- address of current slave instruction;
- active break points;
- contents of slave CPU registers (see also TRACE).

5.5.4. SET

This command can be executed only when the corresponding slave CPU is active. Also, the format of the command depends on the used slave CPU. The format for a 2650 slave is given in the TWIN Operator's guide. The format of the command SET for the Z80 slave is as follows:

```
SET <reg> <value1> [ <value2> [ <valuei> ] ]
```

In which <reg> designates the first register to set and <value> gives the corresponding values. Only the registers for which a value is given are changed. <value1> is assigned to the specified register, <value2> to the next register and so on. The register sequence is:

```
B C D E H L A F IX IY SP I R
```

The limits for the registers B, C, D, E, H, L, A, I and R are 0 and FF. The limits for the registers IX, IY and SP are 0 and FFFF.

5.5.5. TRACE

The command TRACE will be executed only when the corresponding slave CPU is active.

The meanings of the TRACE command are given in the TWIN Operator's guide, but the format of the operands has been simplified and certain limitiations do not exist.

The format of the TRACE command is:

```
TRACE [ <op1> ] [ <op2> ] [ <limit1> <limit2> ]
```

All operands are optional. The possible values for the first operand are:

```
A LL
O FF
J MP (not allowed with Z80 slave)
```

The default value being ALL. TRACE JMP is not allowed in full TWICE mode.

The only accepted value for the second operand is:

```
S TEP
```

Only the first character of these operands is significant, the remainder is ignored.

In all slave modes (including full TWICE) a complete trace line is generated. The format of a trace line for the 2650 slave is described in the TWIN Operator's guide, except that no special trace menmonics are used but the standard instruction mnemonics.

The format of a trace line for the Z80 slave is as follows:

```
LOC  INSTR    MNEMONIC        B  C  D  E  H  L  A  F   IX   IY   SP  I  R
```

Where:

LOC: the address of the last executed instruction
INSTR: the value of the instruction
MENMONIC: the instruction menmonic and operands
B .... R: the contents of the registers with the same name after the execution of the instruction

5.5.6. ACCNT

The command ACCNT can be used to retrieve the accumulated execution time of traced instructions. The execution times are determined only with the 2650 slave.

With each instruction trace the execution time of the traced instruction (expressed in machine cycles) is computed. These execution times are accumulated. The accumulated execution time is displayed by the command ACCNT. After the display the accumulator is cleared.

With the Z80 slave always the value 0 is returned.

5.6. PROM programming utilities

The following commands are available to program PROM's:

```
RP ROM
WP ROM
CP ROM
WB RN
CB RN
```

5.6.1. RPROM, WPROM and CPROM

See TWIN Operator's guide.

5.6.2. WBRN and CBRN

The commands WBRN and CBRN deal with files (papertapes) for PROM-burners. These commands are useful with PROM's that cannot be programmed on the TWIN.

The first data item in a "BRN"-file is preceded by the ASCII control character "SOH" (H'01'). The contents of the file before this control character is not relevant. Each subsequent data item in a "BRN"-file is preceded by a blank. Two types of data items exist:

1. Data byte
   A data byte is represented by two consecutive hexadecimal digits.

2. EOT
   The ASCII control character "EOT" (H'04') indicates the end of a "BRN"-file.

Any ASCII character, except a blank, may be inserted between a pair of data items. A "BRN"-file contains 512 data bytes.

The commands associated with "BRN"-files are:

1. WBRN
   This command is to be used to create a "BRN"-file from the contents of the common memory.

   The first bytes in the produced file are "SP" and "SOH". Next follow 32 groups of 16 data bytes each. Every group is terminated with "CR" and "LF". The last group is followed by "SP", "EOT" and "CR".

2. CBRN
   This command can be used to compare a "BRN"-file with the contents of the common memory. Detected differences are displayed on CONO.

Both commands have 2 optional operands, <address> and <device>.

<address> is to be the first operand; it specifies the first location in common memory to use. The default value is 0.

<device> is the second operand it and gives the involved device or disk file. The default value of this operand depends on the command: it is 'PTPU' with the command "WBRN" and it is 'PTRD' with the command "CBRN".

5.7. Miscellaneous commands

The miscellaneous commands are related to programs which are to be executed by the slave CPU:

EDIT
MACRO
ASM
LNK
TXT
PLT
PASCAL

These commands are described in the corresponding documents.

6. Supervisor calls

A program which wants to perform a system function (such as an input/output operation) can request the execution of such a function by means of a "supervisor call" (SVC). All the data required to perform the requested function are contained in a Service Request Block (SRB). The address of the SRB is is passed to the SVC processor with the supervisor call.

A program running in the master CPU passes this address in the register pair R2-R3 of the current register bank. Programs running in the slave CPU have to use another mechanism. Therefore the SVC interruption is introduced. The slave CPU generates a SVC interruption when it executes an I/O instruction (either "write" or "read") to on of the I/O-ports H'F7' - H'F2'. The addressed I/O-port designates the locations in the slave memory at which a pointer to the SRB is stored:

I/O-port	SRB pointer location
H'F7'	H'40'
H'F6'	H'42'
H'F5'	H'44'
H'F4'	H'46'
H'F3'	H'48'
H'F2'	H'4A'

Note: The I/O-ports with addresses H'F1' and H'F0' are also used to interrupt the master CPU, but these interrupt conditions apply to DEBUG, EXAM and DUMP.

A SRB is a data structure with the following format:

```
bytes  0:    SFC  - SVC function code
       1:    SCH  - channel number
       2:    STAT - status
       3:    SDAT - single byte data
       4:    BCNT - buffer byte count
       5:    BMAX - buffer length
       6-7:  BPTR - buffer pointer
       8-9:  URET - master SVC complete return address
      10-11: CSTR - pointer to DIRECT I-O command string
```

A SRB shall not cross a 8 k boundary. The meaning of the components is:

1. SFC - SVC function code
   The SVC function code specifies the requested function. Certain bits of the function code have a general meaning:

```
Bits 0:   1 - execute and proceed
          0 - execute and wait
     1:   with I/O-functions 1 - "binary"
                             0 - "ASCII"
     2:   special function
     3-4: 00 - a channel shall have been assigned
     6-7: with read/write functions 10 - write
                                    01 - read
```

   The SVC function code is to be set by the program which issues the SVC. The allowed function codes are given in the following sections.

2. SCH - channel number
   A channel number, in the range H'00' - H'07', is required for all I/O functions. It is not significant with other functions. The user shall supply the channel number.

3. STAT - status
   The status reflects the state of the requested function. It is set by the SVC processor. The possible status conditions depend on the requested function and they are listed with the description of the functions.

4. SDAT - single byte data
   This byte can be used to pass a single byte data item between the user and the SVC processor. Its usage depends on the requested function. After the termination of I/O operations it contains the status of the peripheral device.

5. BCNT - buffer byte count
   With READ and WRITE functions this count denotes the number of data bytes which have been moved between the input/output area and the device. It is also used in conjunction with SDAT to return 2 byte data requested by non-I/O functions. The information in this byte is always supplied by the SVC processor.

6. BMAX - buffer length
   A binary number denoting the length of the buffer, expressed in the number of occupied bytes. The buffer length is supplied by the user.

7. BPTR - buffer pointer
   The address of the buffer. The buffer is the input/output area with READ and WRITE functions. If applicable, it is the address of the involved data string with non-I/O functions. The buffer pointer is to be set by the requestor.

   The buffer shall not cross a 8 k boundary.

8. URET - master SVC complete return address
   The address of the instruction that is to be executed next when the program, which has issued the SVC, is contiued.

   This component is relevant only for SVC's that are issued by programs being executed by the master CPU. The return address is supplied by the program that issues the SVC.

9. CSTR - pointer to DIRECT I/O command string
   The address of the command string that is to be used with the SVC "DIRECT I/O". This is a privileged function. The address is to be set by the user.

The SVC functions are partitioned into two categories: privileged functions and non-privileged functions. The privileged functions may only be requested by system programs; non-privileged functions may be requested by any program. A system program is program that is executed in the master CPU or a program which name starts with CTRL-O (H'0F').

The SVC functions are partitioned into the following groups:

- input/output functions;
- control functions;
- miscellaneous functions.

With all SVC's the SVC function code is checked and with I/O requests (except ASSIGN) the given channel number is considered. Further execution is function dependent and is

described in the following sections.

General termination status:

H'02' - ILLEGAL CHANNEL NUMBER
H'03' - CHANNEL NOT ASSIGNED
H'04' - CHANNEL BUSY
H'05' - ILLEGAL FUNCTION CODE

Note: The format of the pointers (both SRB pointer and buffer pointer) related to SVC's which are issued by the slave CPU depends on the current slave CPU. With a 2650 slave, the first byte contains the bits 0 - 7 of the address and the second byte the bits 8 - 15. With a Z80 CPU, the contents of these bytes are exchanged (first byte bits 8 - 15 and second byte bits 0 - 7).

6.1. Input/output functions

The following input/output functions exist:

	function	SVC function code
Non-privileged:	assign	H'10'
	read ASCII and wait	H'01'
	read ASCII and proceed	H'81'
	read binary and wait	H'41'
	read binary and proceed	H'C1'
	read block and wait	H'62'
	read block and proceed	H'E1'
	read CI with no echo	H'21'
	write ASCII and wait	H'02'
	write ASCII and proceed	H'82'
	write binary and wait	H'42'
	write binary and proceed	H'C2'
	write block and wait	H'62'
	write block and proceed	H'E2'
	close	H'03'
	rewind	H'04'
	delete	H'05'
	rename	H'06'
Privileged:	special close	H'23'
	special assign	H'30'

During the execution of an input/output operation the SRB status is set to H'7F', indicating "I/O-operation in progress".

6.1.1. Assign

The SVC "assign" can be used to assign a channel to a device or a disk file. The requestor has to set the following components in the SRB:

- SFC - H'10'
- SCH - the number of the concerned channel
- BPTR - the address of the first byte of the device name

The device name is to be an ASCII character string terminated by CR.

The SVC processor checks the channel number and the device name. When all checks are passed correctly, the channel is occupied. Next the corresponding device handler is invoked in order to execute the device dependent operations. Finally the SVC processor sets the termination status and the device status in the SRB and, in case the assignment is not successful, the channel is released.

The following data are returned:

- STAT - termination status
- SDAT - device status

Termination status:

H'00' - FUNCTION COMPLETE
H'01' - CHANNEL ASSIGNED TO A NEW FILE
H'0B' - DEVICE NOT AVAILABLE
H'0D' - DEVICE IN USE
H'16' - ILLEGAL FILE NAME
H'18' - CHANNEL ALREADY ASSIGNED

Other status conditions can be returned by the device handler.

6.1.2. Read

With the SVC's "read ASCII/binary" a distinction is made between devices and disk files. The SVC's "read block" are treated as "read binary" from a device (also in the case of disk files). The requestor has to supply the following data in the SRB:

SFC - the appropriate SVC function code
SCH - the number of the concerned channel
BMAX - the maximum number of data bytes to read
BPTR - the address of the input area

Note: In case of "read ASCII", the input area must be able to contain 1 byte more than is indicated by BMAX; with the other read functions at most the indicated number of data bytes will be returned.

When data are to be read from a device or with "read block", the SVC processor invokes the device handler to perform the requested read operation. With "read ASCII/binary" from disk files, the SVC processor reads first a sector from the concerned disk file into a buffer area that is located in the master memory. Next the data are deblocked into the user's input area. The SVC processor also distinguishes "read ASCII" and "read binary":

1. Read ASCII
   A string of bytes is moved from the system buffer to the user's input area. The string is terminated when either the ASCII control character CR has been transferred or the input area is full. In the latter case CR is stored in the location next to the end of the input area. The ASCII control characters are not processed.

2. Read binary
   A string of bytes is moved from the system buffer to the user's input area. The string is terminated only when the input area is full. No additional CR will be stored.

The following data are returned in the SRB:

STAT - termination status
SDAT - device status
BCNT - number of bytes moved to the input area

Termination status:

H'00' - FUNCTION COMPLETE
H'06' - NO EOL ON ASCII READ

Other status conditions may be returned by the device handler.

6.1.3. Write

With the SVC's "write ASCII/binary" also devices and disk files are distinguished. The SVC's "write block" are treated as "write binary" for a device. The user has to pass the following data in the SRB:

SFC - the appropriate SVC function code
SCH - the number of the concerned channel
BMAX - the maximum number of data bytes to write
BPTR - the address of the output area

When data are to be written on a device or with "write block", the SVC processor invokes the device handler to perform the requested write operation. With "write ASCII/binary" into disk files, the SVC processor first blocks the supplied output data into a buffer area which is located in the master memory. When this buffer area is filled, a complete sector is written on the disk.

The following data are returned in the SRB:

STAT - termination status
SDAT - device status
BCNT - number of bytes taken from the output area

Termination status:

H'00' - FUNCTION COMPLETE
H'07' - NO EOL ON ASCII WRITE

Other status conditions may be returned by the device handler.

6.1.4. Close and delete

To execute the SVC's "close" and "delete", the user has to supply the following data in the SRB:

SFC - the right SVC function code
SCH - the number of the associated channel

Upon execution of the SVC's "close" and "delete", the SVC processor invokes the corresponding device handler to execute these functions. When the device handler has completed the function, the SVC processor releases the channel.

The following data are returned via the SRB:

STAT - termination status
SDAT - device status

The device handler generates the termination status.

6.1.5. Rewind

With the SVC "rewind", the SRB has to contain the following data:

SFC - H'04'
SCH - the number of the corresponding channel

To execute the SVC "rewind", the SVC processor transfers control to the associated device handler.

The next data are returned in the SRB:

STAT - termination status
SDAT - device status

The termination status originates from this device handler.

6.1.6. Rename

With the SVC "rename", the SRB has to contain the following data:

SFC - H'06'
SCH - the number of the involved channel
BPTR - the address of the new file name

The SVC processor starts to check the device to which the channel is assigned. If this is not a disk file, the function is terminated. Next the new file name and the state of the file are checked. If these checks are passed correctly the device handler is invoked to rename the file.

The following data are returned in the SRB:

STAT - termination status
SDAT - device status

Termination status:

H'00' - FUNCTION COMPLETE
H'16' - ILLEGAL FILE NAME
H'17' - FILE IN READ/WRITE PROGRESS

Other status conditions can be supplied by the device handler.

6.2. Control functions

The following control functions are defined:

	function	SVC function code
Non-privileged:	load overlay	H'17'
	execute overlay	H'18'
	suspend	H'19'
	exit	H'1A'
	abort	H'1F'
Privileged:	execute overlay and wait	H'98'
	pause	H'1B'
	reserve console	H'24'
	release console	H'25'

6.2.1. Load overlay and execute overlay

The SVC functions "load overlay" and "execute overlay" are similar to the system commands LOAD and XEQ (see 5.4.1). With the execution of these functions a load module is loaded into the corresponding overlay area (see 2). The requestor has to supply the following data in the SRB:

SFC - the appropriate SVC function code
BPTR - the address of the module name

The module name is an ASCII character string terminated by CR.

Upon completion of the function, the termination status is stored in the component STAT of the SRB.

Note: With "execute overlay" the requestor's job remains in the state "I/O wait" (see 2), except if the load module is loaded into the requestor's overlay area.

Termination status:

H'00' - FUNCTION COMPLETE
H'06' - DEVICE READ ERROR
H'0E' - INVALID INPUT DEVICE
H'30' - LOAD FILE NOT FOUND
H'31' - LOAD FILE ASSIGN FAILURE
H'32' - FILE NOT A LOAD MODULE
H'33' - INVALID LOAD REQUEST

6.2.2. Suspend

With the execution of the SVC function "suspend" the requestor's job enters the state "suspended" (see 2). The requestor has only to set the right SVC function code in the SRB and no status or other data are returned.

Note: The suspended job can be resumed only by means of the system command CONT (see 5.1.3).

6.2.3. Exit and abort

Upon the execution of the SVC functions "exit" and "abort" the state of the requestor's job is set to respectively "idle" and "abort" (see 2). When the execution of the functions is requested by the slave job and the slave job is under DEBUG control, DEBUG is aborted too.

The requestor has only to set the right SVC function code in the SRB and no data are returned to the requestor.

With "abort", the assigned channels are closed too; the channels remain assigned with "exit".

6.3. Miscellaneous functions

The group of the miscellaneous SVC functions comprises the remaining functions. These functions are:

	function	SVC function code
Non-privileged:	get time	H'11'
	get overlay addresses	H'12'
	get parameter	H'13'
	get slave job parameter	H'1C'
	get device type	H'14'
	get device status	H'15'
	get last console input	H'16'
	get date	H'1D'
	get time of the day	H'1E'

6.3.1. Get time

With the execution of the SVC function "get time" the accumulated execution time of traced instructions (see 5.5.3) is returned to the components SDAT (high-order byte) and BCNT (low-order byte) of the SRB. The requestor has only to supply the appropriate SVC function code.

Termination status:

H'00' - FUNCTION COMPLETE

6.3.2. Get overlay addresses

With the execution of this functions the load addresses of the last loaded slave module are returned. The requestor has to supply the following data in the SRB:

SFC - H'12'
BPTR - the address of a 6 byte area into which the load addresses are stored

The returned data have the format:

bytes 0-1: address of first byte
2-3: address of last byte
4-5: entry point

Termination status:

H'00' - FUNCTION COMPLETE

6.3.3. Get parameter and get slave job parameter

The SVC function "get parameter" is useful only for job 1 and job 2, while the function "get slave job parameter" can be used by all jobs. The requestor has to supply the following data in the SRB:

SFC - the right SVC function code
SDAT - parameter number
BMAX - maximum number of bytes to be returned
BPTR - address of area into which the parameter shall be stored

Note: If the most significant bit of SDAT is a 1, the requested parameter and all subsequent parameters are returned. The parameters are separated by a single blank.

The returned data string is always terminated by CR. If the parameter area is exhausted before the end of the parameter is attained, CR is stored in the location next to this area.

When the requested parameter has not been specified, only the terminating CR is returned. If the given parameter number exceeds the number of the last parameter, the datum H'FF' is returned.

Termination stauts:

H'00' - FUNCTION COMPLETE
H'06' - PARAMETER AREA FULL or END OF PARAMETER LIST

6.3.4. Get device type

Upon the execution of the SVC function "get device type" the device identification number and the device type will be returned. This function can be executed successfully only if indicated channel has been assigned to a device. The requestor has to supply the following data in the SRB:

SFC - H'14'
SCH - the number of the concerned channel

The following data are returned in the SRB:

STAT - termination status
SDAT - device identification number
BCNT - device type

The possible device identification numbers and device types are:

device	device id	device type
CONI	1	H'01'
CONO	2	H'02'
LPT1	3	H'02'
TTYR	6	H'21'
PTPU	8	H'42'
PTRD	9	H'41'
disk file	-1	H'43'

Termination status:

H'00' - FUNCTION COMPLETE
H'02' - ILLEGAL CHANNEL NUMBER
H'03' - CHANNEL NOT ASSIGNED

6.3.5. Get device status

With the execution of the SVC function "get device status" the status of the device that is connected to the involved channel will be returned in SDAT. The requestor has to set the following components of the SRB:

SFC - H'15'
SCH - the number of the concerned channel

The following data are returned to the SRB:

STAT - termination status
SDAT - device status

Termination status:

H'00' - FUNCTION COMPLETE
H'02' - ILLEGAL CHANNEL NUMBER
H'03' - CHANNEL NOT ASSIGNED

6.3.6. Get last console input

Upon the execution of the SVC function "get last console input", the last entered console input character is returned in SDAT. The data entered during a "read binary" for the console are not considered as characters within this context.

The requestor has only to set the component SFC of the SRB and the following data are returned in the SRB:

STAT - termination status
SDAT - last console input character

Termination status:

H'00' - FUNCTION COMPLETE

6.3.7. Get date and get time of the day

With the execution of these functions respectively the current date and the current time of the day are returned. The requestor has to supply the following data in the SRB:

SFC - the appropriate SVC function code
BPTR - address of 8 byte area into which the data are stored

The date can only be set at IPL-time (see 7). The time of the day can be initialized at IPL-time too, but this datum is also invloved with the system commands TMR and TIME (see 5.2.3 and 5.2.4). The time of the day is updated continueously, except when timer interruptions are inhibited.

Only the termination status is returned in the SRB.

Termination status:

H'00' - FUNCTION COMPLETE

7. Initial program loading

The Operating System is loaded into the master memory after "power on" and upon a "system reset". When the Operating System has been loaded successfully, it displays the message:

```
** TWIN OS **
```

Next the current date and the time of the day are requested. The date is asked by means of the message:

```
DATE:
```

Then the operator may enter the current date in the form:

```
<y> <y> / <m> <m> / <d> <d>
```

In which <y>, <m> and <d> represent respectively "year", "month" and "day". The entered date is checked and the accepted date is recorded. When an error is found, the date is asked again.

Then the current time of the day is requested by means of the message:

```
TIME:
```

The operator has to enter the time in the form:

```
<h> <h> . <m> <m> . <s> <s>
```

In which <h>, <m> and <s> represent respectively "hours", "minutes" and "seconds". These data are checked for decimal digits in the allowed ranges. When an error is detected, the time is asked again. After the time has been accepted the timer interruptions are disabled, so the time of the day can be updated each second.

Finally the message:

```
IPL COMPLETE
```

Is displayed, which terminates the initial program loading.

When the operator enters the ASCII control character ESC, the preceding data in that line are ignored and the current request is repeated.

If the operator enters an empty string (by pressing only the "return"-key), no date and time, respectively no time, will be stored. The default date consists of 8 consecutive blanks, while the default time is 00.00.00. When no time is entered, the timer interruptions remain inhibited.