Programmer's Guide
(Rev. 1.1)

Contents

Introduction
This manual is intended for coders to
'd like to write applications and system for SpartaDOS X.
The authors assume that the reader has the idea of writing programs
Atari, and is an advanced user SpartaDOS X, to
What message contained in the "Disk Operating System
SpartaDOS X User's Guide "and supplement" SpartaDOS
V. X 4.40 "omit here as self-evident. In addition, it is assumed that
concepts such as "PORTB" does not require an explanation.
Listingi are written in Assembler Pseudocode the syntax is consistent with
MAE assembler. It is most similar to the syntax of assembler
MAC/65, only the details differ from the syntax used by
crossassembler MADS. A programmer wants to adapt his own hand
them to your favorite assembler.
We wish you a pleasant reading
DLT Ltd.

Chapter 1: Symbols
What is the symbol
Manual programming SpartaDOS X is a good start with
explanation of the concept of *a symbol* with which the reader will meet
while reading.
The symbol in SpartaDOS X is a kind of rich-pointer.
It contains information on where the computer's memory is the
object: variable, table or procedure. One symbol represents
one such object. All symbols are about hundreds,
are linked in the list. It also guarantees access to the most important - from the point of
programmer's view - the procedures and internal structures SpartaDOS X
and the drivers and overlays, since the resident may,
naturally, to add to the list of their own symbols.
The main advantage of such a solution is that the procedures
system, so that they are indicated by symbols, are not
permanently assigned to specific addresses. Addresses can be
vary from version to version of DOS, and yet the programs will work.
What's more, thanks to define a new symbol of the same name,
as one of the existing overlay can easily take over mediation

between the program and procedure system, which he produces.

The structure symbol (Symbol)

The exact description of the symbol is a not-for-nothing
needed, because the programs do not benefit from them as to require a
Programmer its knowledge. However, this information can at least
help satisfy curiosity readers.
A single symbol occupies 13 bytes, consisting of him as follows:

+ $ 00 - $ 01 rate for the next symbol (2 bytes)
+ $ 02 - $ 09: the name of the symbol (8 ASCII characters)
+ $ 0A: byte control
+ $ 0B-$ 0C: address indicated by the symbol (2 bytes)

Symbol name meet the same conditions as the file name: is it to
eight characters ASCII, but they are generally large and sign letters of the alphabet
underscores. When a name is no less than eight characters, is completed
spaces.
Byte contains **control** bits in the two oldest index memory
indicated by the symbol: this is 0 for main memory and 2
notes. Six bits are known. PID *(Program Identifier):*
the number of the program, which consists of a symbol. Zero represents the library
system, which will still question.
The functions of libraries allow access to the list of symbols are described
in a subsequent chapters.
NOTE: at the time of the launching of the X command list of symbols
becomes unavailable.

Chapter 2: The format of binary blocks SpartaDOS

The types of blocks of binary

Atari DOS knows only one type of binary block, it is
sześciobajtowym segment with a header starting (optional
indeed), from FFFF $ signature. This format is known, therefore
his description is omitted.
SpartaDOS distinguishes a total of seven types of blocks of binary.
One of them is, naturally, above-mentioned segment Atari DOS.
Nierelokowalny SpartaDOS segment is the same structure, only
header signature is different: instead of $ FFFA $ FFFF. Changing signatures
is aimed at preventing such binary read by other DOS-y,
because this format is intended for system files
(drivers, etc.) SpartaDOS.
The other five types are:
1) memory block booking
2) relokowalny block binary
3) block update addresses inside the program (the so-called *fixupy)*
4) lists the symbols required by the program
5) lists the symbols defined by the program

Block booking memory

Block booking memory causes, as taught by the same name,

set aside by the loader indicated quantity indicated memory. Block
This is due to a header from a population of only eight bytes:
+ $ 00 - $ 01: $ signature FFFE
+ $ 02: sequence block

+ $ 03: byte control with a value of $ 80 index added memory
+ $ 04 - $ 05: the beginning of the transfer terms
+ $ 06 - $ 07: the number of bytes to book
The memory is reserved on the index of free memory
(ie over MEMLO in the main memory). Type of memory, the main or
Secondary is indicated by an index that is in bajcie
control. The indexes more memory is written in chapter
"The economy of memory."

Relokowalny block binary

Relokowalny block binary is heading in principle the same as
memory block booking.
+ $ 00 - $ 01: $ signature FFFE
+ $ 02: sequence block
+ $ 03: byte control with a value of $ 00 index added memory
+ $ 04 - $ 05: the beginning of the transfer terms
+ $ 06 - $ 07: the number of bytes to load
Both types of units differ only two things: first, byte
control in the header block relokowalnego is zgaszony bit 7; is
signal to the loader, that are headed for the data to load.
The second thing is precisely this fact. The binary data are loaded into the address
indicated by the rate of free memory. Type of memory, the main or
Secondary is indicated by the index recorded in bajcie control.

Block fixupów

Block update addresses internal, so called. *Fixupów,* block
relokowalnego is pięciobajtowy headline:
+ $ 00 - $ 01: $ signature FFFD
+ $ 02: the number of the next block, which fixupowanie
+ $ 03 - $ 04: two bytes reserved
The next bytes to transfer within the block, first in relation
to address the loading of the block, each of the next in relation to
the previous one. In other words, such a byte means "increase the address and so on
bytes and so do the fixup. " Fixupowanie lies in the fact that the address
charging unit is added to each word dwubajtowego
below the address indicated by the byte transfer.
Byte transfer shows always double-byte word - and so
Program relokowalny MAY NOT include the value of any of its
internal addresses divided into junior and senior byte located
separately. Structures such as:
lda # <address
ldx #> address
whether the plates together separately younger and older bytes indicators

are excluded.
Such is the importance of each value of less than $ FC. The values of
$ FC to $ FF is an additional control codes:
$ FF: increase by 250 bytes of address (anything beyond this is not done)
$ FE: change the number of block fixupowanego the value seq. byte
$ FD: change the beginning of the block to the address contained in the following. word, and fixup
$ FC: mark the end of the block

The lists of required symbols

This is a list of symbols, which must be defined in the system, to
loaded the program can work. The absence of any cause interruption
No charge and error 154 (Loader: The symbol not defined).
+ $ 00 - $ 01: $ signature FFFB
+ $ 02 - $ 09: the name or symbol (8 characters)
+ $ 0A-$ 0B: two bytes reserved
Next followed bytes transfers and control in the same way as in the block
fixupów. Address indicated by the symbol is added to
dwubajtowego words below the address indicated by the
byte transfer.

The definition of a symbol

Block definition symbol to add the new loader
symbol of the global list of symbols. Structure:
$ 00 - $ 01: $ signature FFFC
$ 02: the number of the block, where the symbol is defined
$ 03 - $ 04: Transfer
$ 05 - $ 0C: the name of the symbol
The address pointed to by a new symbol is the address block loading
number listed in bajcie $ 02, plus transfer of
bytes $ 03 - $ 04

Chapter 3: memory management

The use of memory by SpartaDOS

SpartaDOS X distinguishes the following types of memory:
1) Main memory: basic 48k of RAM to MEMLO
MEMTOP-u
2) the extension of memory: 14k additional RAM ("under the ROM-height",
Areas $ C000-$ CFFF and $ D800-$ FFFF) in computers 800XL and 65XE,
or additional RAM (Area $ 4000 - $ 7FFF) for up to 1 MB in
130XE computers, or up to 2 MB in the Atari 800 computers In
the absence of this memory instead of the allocated memory is
Home.
3) own module ROM: 128k ROM in the area of $ A000-$ BFFF.
System for their own needs does the memory as follows:
1) kernel main memory, from $ 0,700 to MEMLO.
2) drivers additional system memory
3) The library system in the module ROM

4) The device CAR: ibid
Already mentioned earlier *system library* contains a collection of
the intermediary between the user and
load DOS, and one of the software application - namely
SpartaDOS Formatter. The library occupies two banks (after 8k) module
ROM SpartaDOS X 4.20, and 3 banks in SpartaDOS X 4.40. Action
and use of the library is devoted to the main part of this
manual.

Leading module library (the bank No. 1) is normally present in
Under $ A000-$ BFFF, but it can be unplugged and replaced
RAM, if the program is run a command X.
Usually this happens in the case of ordinary binaries intended
Atari DOS-in (those with a header $ FFFF).
The allocation of RAM for the various elements of the system
The following depending on whether it is available, and from what he chose
you in your CONFIG.SYS. Here are four possible combinations:
1) the use of main memory: use none in the CONFIG.SYS.
All components of DOS are loaded in the main memory
ranging from $ 0700. address This configuration is chosen
automatically on computers Atari 800 with no
more than 48k of RAM.
2) the use of memory "under the ROM-em": USE OSRAM in
CONFIG.SYS. Kernel memory addresses from $ 0,700 to MEMLO, other
DOS-in components are loaded into memory in the area of $ E400-$ FFFF,
area $ D800-$ DFFF occupied on data (from version 4.40 --
was previously vacant), the area of $ C000-$ CFFF remains slow for the rest of
the system is assigned the main memory. This configuration is chosen
automatically on computers Atari 800XL and Atari 65XE
with no more than 64k of RAM.
3) the use of memory "under the ROM-em": USE OSRAM / DEVICE
SPARTA OSRAM in CONFIG.SYS. As above, except that the memory of $ C000 -
$ CFFF utilization is at the DOS buffers respectively in reducing
occupancy main memory.
4) the use of memory bankowanej: kernel is loaded in the area
from $ 0,700 to address MEMLO, additional components of DOS, data and
buffers occupy an additional memory bank located in
Under $ 4000 - $ 7FFF. This configuration is automatically selected,
Atari 800 when the computer is at all equipped with such an extension

---

(Axlon type, up to 2 MB of RAM banks), or when the computer XL / XE is
More than 128k of RAM.
Additional memory is distinguished by DOS on the bank system
where the drivers reside, above all, the procedure
SPARTA.SYS, and all the rest. The procedures in DOS ensure easy
access only to the bank system. This is justified by the fact that memory
allotted for the system may be in the main area

RAM, the ROM-in memory or in bankowanej, connecting
adequate memory system takes so himself. In contrast, "the whole
the rest "to extend the type 130XE (or Axlon). SpartaDOS offers here
some support - as below - but the switch banks
must do it on their own by interfering in the relevant records
I / O.

Recognizing the configuration memory

Even programs not in principle for SpartaDOS X may
be interested in taking the current configuration of the memory
DOS, and especially that expanded memory banks are occupied by it.
Part of this information is contained in an array COMTAB
DOSVEC indicated by the vector ($ 0A-$ 0B), and by the symbol
COMTAB.

*Caution should refrain from treatment value DOSVEC
as a constant. The address pointed to by this vector is different in different
SpartaDOS versions (and even different versions SpartaDOS X) and not
is no guarantee that it will not change in the future. Solids are only shifts
(offsets) for the specified address, as shown below.*

COMTAB + $ 1D (NBANKS) contains a number of free banks
additional memory type or 130XE (for the Atari 800) Axlon. Where is
here is zero, this means that additional memory or does not exist, or is

fully occupied by components DOS, ie, drivers, etc. ramdyski

COMTAB + $ 1E (BANKFLG): If there is $ FF, the system loaded
is the memory bankowanej (USE BANKED).

COMTAB + $ 1F (OSRMFLG): If there is $ FF, the system loaded
to the memory "under the ROM-height" (USE OSRAM). Memory bank, the
if any, may be used as ramdysk and allocated
drivers such as CON64.SYS and CON80.SYS.

In addition, the developer may want to check out the type of enlargement
Memory: When the **COMTAB + $ 1B (_800FLG)** is $ FF, we have to
dealing with a computer Atari 800 and a Axlon. In
Otherwise (if the _800FLG is zero) is a computer
XL / XE with a type of 130XE.

The calculation of how much the system in general is an additional memory (used,
or not) is possible on the basis of so-called masks PORTB
(PBMASK). It can be found at COMTAB + $ 1C, and includes
one in all the bits, which makes set-up in the PORTB
Switching memory banks in the area of $ 4000 - $ 7FFF. What counts is also
bit 4 of the port, so for example, simply 130XE mask is set to $ 1C,
namely% 00011100. Except bit 4 are set here two bits, which means
four additional banks of memory, or 64k.

By analogy with the extension to the 320k type of RAM Compy Shop
mask has a value of $ DC, which is 11011100.% Except bit 4 are here
4 bits set. This means 16 additional memory banks.

Information, which specifically memory bank notes is
"The system" is contained at T_ + $ 06 or $ COMTAB-0150
1)

1 The only appropriate method of access to the array reference T_ is for

T_ symbol means. However, due to programs
Not for SpartaDOS X, and wishing to take advantage of these instructions
data, we - as an exception - an alternative method of calculating its
address, namely on the basis of a shift in relation to COMTAB
(ie, in relation to the value of the vector DOSVEC). Should not be there with
conclude that each of the symbols is achievable this way, rather the contrary,
the location of all other defined objects symbolically

For use none is $ FF, and for the use and BANKED USE
OSRAM there is value to be put into the registry
PORTB, to connect the bank, in which resides the main driver
SpartaDOS (ie SPARTA.SYS).

The list of free memory banks (T_)

Obtaining information reverse, ie not that banks are busy, but
rather, which are free, is slightly more complicated. Nevertheless,
it's a lot more useful, because they are occupied by the bank
SPARTA.SYS need not be the only one in which they reside drivers DOS
u. By now skip the question ramdysków, SpartaDOS X from version 4.41 is
Two Drivers, CON64.SYS and CON80.SYS, which allocates each
additional memory banks. Overwrite them anything at the moment
when they are activated, of course, result in suspension of computer
sooner or later.
The main entrance to calculate the list of free memory banks
their number is shown by NBANKS (COMTAB + $ 1D). For the calculation
necessary data are also included in the array T_ and the register PBMASK
(COMTAB + $ 1C). The procedure produces a list of banks is free
as follows:

```
; FREELIST
sav
= $ 80
savy
= $ 81
temp
= $ 82
index = $ 83
lda COMTAB + $ 1d
Solid sav
LDY # $ 00
Jan savy
```

(variables, arrays, procedures) is nohow to address and may COMTAB
be any change in future versions of SpartaDOS development.


```
January index
loop
LDY sav
dey
Jan sav
```

```
bmi exit
yeah
and # $ 03
asl
asl
Solid temp
yeah
LSR
LSR
tax
lda T_ + $ 08, x
now temp
eor PORTB
and COMTAB + $ 1C
eor PORTB
PHA
iny
yeah
LDY index
Solid axlon, and
pla
Solid list, and
inc index
bne loop
exit
rts
axlon. SB 128
letter
. SB 128
```

In the place labeled "list" procedure will list
PORTB registry values corresponding to free memory banks,
while under the label "axlon" is corresponding with her list
Axlon banks enlargement. Number of entries will be equal to the number of
free banks shown by NBANKS (COMTAB + $ 1D). Banks
in accordance with the 130XE (or first 64k extension) will be on this list
featured as the last one. This is in line with the overall logic of this allocation
memory, according to the first-come, banks far, so that

for example on a computer equipped with 192k of RAM is SpartaDOS ulokował
in the area over the basic 128k, and banks remained free 130XE
programs for wanting to possibly benefit from it.

The allocation of main memory and an additional (malloc)

There are few roads, which installs the program may be rezydentnie
SpartaDOS tell, that took for himself a piece of RAM.
Regular programs in binary format Atari DOS-u (with a header
$ FFFF) may, in order to raise the rate MEMLO. After paying
controls for DOS in memory of information (contained in the table
H_FENCE) will be created on this basis.
The second way is available for relocatable binaries SpartaDOS,

which are always loaded in the place indicated vector MEMLO --
or, more precisely, in the place indicated by the first word plate
H_FENCE. Automatic allocation of the area occupied by the program
is obtained by inserting a $ FF to the indicated variable symbol
INSTALL and return control to the DOS-u. Before you start
scheme, resetting the variable INSTALL, so the normal way
"Insert $ FF" is its reduction by 1
Additional areas of memory may be allocated by the
malloc indicated symbol. "Saw" only two types of
Memory: Memory and the central bank of the extension, which resides in
SPARTA.SYS procedure (bank system). The number of bytes that are
be reserved more than an indicator of free memory, we in
FAUX4 / 5 ($ 0,785 / 6). To register X code representing the load
type of memory (the so-called index memory: 0 Main, 2 bank system) and Y
reset. With the additional allocation of memory, when a bank
system lacks this place, the system tries to allocate
main memory. When correctly performed the procedure (ie completed

positive), a pointer to the assigned area is in
FAUX1 / 2 ($ 0782 / 3), and index of memory - in the register X.
Memory used by malloc then it can not be exempt, but this feature
is intended for programs to reside permanently in RAM.

The allocation of memory banks

In addition to main memory and the bank system, an area which may
TSR want to take, are free additional memory banks.
Allocation procedure is very similar to the above zademonstrowanego
generates a list of sub-free banks.

```
; BANKALLOC
temp
= $ 80
LDY COMTAB + $ 1d
beq error
dey
Jan COMTAB + $ 1d
yeah
and # $ 03
asl
asl
Solid temp
yeah
LSR
LSR
tax
lda T_ + $ 08, x
now temp
eor PORTB
and COMTAB + $ 1C
eor PORTB
iny
```

```
clc
rts
error sec
rts
```


Subprogramme reserves (on a permanent basis, until the next cold start-up)
a memory bank notes. When executed correctly (with C = 0)
the battery pack will contain a code PORTB podłączający reserved
bank in the computer XL / XE, while the register Y - appropriate value
Axlon to register in the computer Atari 800

Access to the bank system

Access to additional memory, as it is written above, you will
differently depending on its type. First, we'll simplify the issue
access to the bank system.

*NOTE: the area of RAM is conventionally called the bank
the system can be located at different addresses, depending on the
memory configuration selected by the user. In particular, need
believe that the code Switching was outside the $ 4000 --
$ 7FFF!*

Connecting the bank system is realized by calling
EXT_ON the system ($ 07F1) in argument
transmitted in the accumulator. Disconnection together with the reintroduction
the previous system of banks - not necessarily because originally it must be
connected to main memory bank - provides a procedure EXT_OFF
($ 07F4). For the latter not to pass on any arguments.

The argument for EXT_ON code is the additional memory, we wish
connect. It can not be permanent, since after the first call at the time
it is nowhere said that the additional memory at all, and after
Second, even if, whether any portion of it was us
assigned by DOS. *Therefore, the value is passed on to
EXT_ON always have, in one way or another, to obtain the first from
operating system.*

The number one method applies only to programs stored in

relokowalnym SpartaDOS format, in which at least one block
Binary is to be automatically loaded into the extra memory. This
memory is always a bank system, unless there is no place in it --
then block the program is loaded into main memory.

Code memory, which loaded the binary blocks intended to
additional memory can be found at the time of its launch in
variable indicated symbol EXTENDED. The program installs the
rezydentnie in the system should remember it was then the value (after
return to DOS in it is set at zero), then, during its
action, pass it as an argument for the EXT_ON when
you need to refer to the part residing in the bank system.

Method number two concerns TSRs, which
need access to memory is not by himself but by other

System drivers. A classic example of such a procedure is Ramdisk.sys. The data may be requested from the program
user, you should generally be saved for home
memory. However, as well the program may be causing
SPARTA.SYS procedure for requesting to read buffers DOS, and these
may be in memory of the notes. In any case,
Ramdisk.sys must therefore decide to which memory is
perform record, or he read, either sector, or block status, or
PERCOM block.
Such a decision is taken on the basis of the indicated variable symbol
SYSCALL. Each contains a memory index, to which the desired
is reading, or with which the desired word, and then, if necessary,
need, it is the value SYSCALL then be transmitted as
EXT_ON argument to the call to connect the memory
audience.
Connection EXT_ON memory must have its counterpart in a later call EXT_OFF,
where access to the bank

---

the system will cease to be needed.
Access to other banks enlargement
As mentioned above, access to other memory banks
realized by saving directly to the respective values
registers controls memory, ie PORTB ($ D301) in computers
XL / XE and Axlon ($ CFFF) in the 400/800 computers. As the value of the
the record should take the results of procedures, or BANKALLOC
FREELIST printed on the previous pages. Due to the
Separate addressing ANTICA memory and CPU, when expressing
PORTB good practice is to change only

the current value of those bits,
which are responsible for switching the
bank, and leave the rest without
changes. This is done in the following
manner:
```
; BANKSWITCH
lda PORTB
Solid old_pb
lda new_pb
eor PORTB
and COMTAB + $ 1c
; PBMASK
eor PORTB
Solid PORTB
```
To date, the value of PORTB is stored in
bajcie
labeled "old_pb", "new_pb" is taken from
FREELIST code
bank, which want to connect.
Switching program memory on one of the
banks of enlargement, when
will end the use of additional memory,
it should restore its
initial configuration. In the case of XL
/ XE is quite simple,
simply, as shown above, before remember
somewhere
PORTB value, and then change it back.

With the extension Axlon matter is
complicated because the control register
banks only to write - so you can easily
switch banks in
one-way, but to restore the system
before the switch already not so easy
You can, because reading from $ CFFF not
pay the state registry, but some
random garbage. Do not be so remember
your current configuration
memory before.
This difficulty can be resolved in two
ways. The first - and
easier - is blocking a program runs on
computers
other than XL / XE. This is achieved by
checking _800FLG
(COMTAB + $ 1B). When switching memory

is critical for
the speed of the action - as it is, for
example, in the case of drivers
CON64.SYS and CON80.SYS where to send
the screen of each
single character entails the memory
banks - then after
Just no other way out, how to reconcile
with the fact that our code to
400/800 with a Axlon will not work.
The second method involves the use of a
trick. But the current
banking system extension Axlon know DOS,
it is (in principle only) on
it switches. Remember we achieve this
state by calling
EXT_ON to the value of ensuring the
provision of bank system
(ie, EXTENDED or code returned by malloc
in the registry
X, see above). DOS switches the memory
banks, but it is not us
interested. When EXT_ON give control,
you can enter your desired
value to the registry Axlon control,
which the bank will make available a
program for
in question. In contrast, the previous
state is obtained
simply by calling EXT_OFF.

Chapter 4: Handling errors

The standard procedure for handling
error (U_FAIL)

SpartaDOS handling error in the creation
procedure
U_FAIL library with an error code in the
accumulator. U_FAIL may
be called explicitly to the user, but
generally there
This is the secret, if this procedure,
he finds some
irregularities, it automatically
produces the library system.
U_FAIL has the unpleasant feature that
will never come back. In other
words, calling the program is

interrupted and removed from memory,
open files are closed, and the system
shown on the appropriate console
the error message gives control to the
command interpreter. This is
Comfortable in the case of critical
errors, which occurred after
program, and so can not carry on,
however, relatively
often there is also a need to handle the
error within the program.
The creation of traps (U_SFAIL)
Program and want to take over the
handling of a particular error may
zastawić
trap. Use the library U_SFAIL procedure.
Before her
calling for the AX must registers load
properly and younger
Senior byte address space in the
program, which will be transferred to
control at the time the error occurred.
You can sign up for more than one trap -
reacts always the one that
was established recently. However, while
the number of traps set up
should not be prejudged, the program can
not have more than 10 at a time.
When an error occurs in the trap set up,
it will be above
all removed (this is a one-off). Then
loads the library
stack pointer value memorized by
U_SFAIL, switches banks

---

memory stored in a state there and
executes the stroke at the address JMP
indicated by the user in setting up the
last trap. Code
an error that occurred, is loaded into
the battery, a marker
N register markup processor - set to
one. The rest remains
unchanged, ie in particular do not close
the files that the program previously
opened.
Removing the trap (U_XFAIL)

As a rzekło, the trap is removed
automatically when there's
mistake. However, if the error does not
occur, the trap usually becomes
unnecessary
and should be removed "by hand". Use the
procedure U_XFAIL. No
pass it any parameters, it simply
removes the trap
set up recently. U_XFAIL does not alter
the content of CPU registers.
All the trap set up by the program are
removed at the time
its completion and commissioning of
control in DOS.
Error message (U_ERROR)
U_ERROR displays on the screen system
error message,
whose code was passed in the
accumulator. Before that is sent
editor, character EOL ($ 9B). This is
part of standard maintenance procedures
Boot error by U_FAIL.
An example of the use of traps
; setting pulapki to the address
over the vector trapptr, it contains the
address
; marked a label trap
lda trapptr
ldx trapptr 1

JSR U_SFAIL
; a procedure call system
, which is expected in the speeches of
error
...
; when the error was not, we remove
pulapke
JSR U_XFAIL
; when the error occurs, the system will
automatically put off
; pulapke and give control to that place
; with N = 1 and the code of error in
the accumulator
TRAP
bmi error
; here, the proceedings in the case

```
; the successful conduct of the
procedure
...
rts
; a service error
JMP error U_ERROR
```

LBUF and buffer index BUFOFF

Interpreterowi command issued commands at the DOS, either directly by a user with the keyboard, or read from a batch file, is saved in the buffer LBUF (line buffer). It has 64 bytes and is at COMTAB + $ 3F. Current position in the buffer, ie the Generally the next parameter indicates the index BUFOFF (buffer offset, COMTAB + $ 0A).

LBUF buffer BUFOFF index and provide input for library procedures dealing with command-line treatment, and programs normally do not have the necessary attention to their content. May however, sometimes have to read more than once a element of the command line. You'll need to remember the state and BUFOFF restore it before you call the library (with All of which is more BUFOFF setting it on the next item for processing).

It must be noted that the first parameter is the LBUF Generally caused the program name. At the time of its launch BUFOFF points out, however, for the next parameter - as the name of the program has already been read by the interpreter in DOS commands. To download this name, you must reset the BUFOFF.

Buffer COMFNAM

Exit procedures for processing the command line, those at the very least,

which results in the form of text can be found at
COMTAB + $ 21 This is a buffer COMFNAM (complete file name) on
30 bytes in length. The associated variable is the TRAILS (trailing

---

space, COMTAB + $ 1A), it contains length in the
COMFNAM parameter.
While LBUF contains a command line, it is COMFNAM
intended for a single parameter. LBUF is just behind
COMFNAM, in principle, these two buffers connect with each other. It follows from this,
that a very long parameters (more than 30 characters) collected by the library from the command line and wkopiowane to COMFNAM can override
the initial part of LBUF.
Reading the command line
The command line consists of a group of elements, ie, separated from
each file name spaces, switches, options, etc. stored on
generally in the order, which is known in advance and defined as syntax
the command. It follows from this that the program to realize
reading the command line from beginning to end as part of element
in most cases can anticipate what kind of
parameter will be reached next. This program does not need to parse
the command line to walk - though, if the need arises,
of course, can - as the library offers a procedure for processing
the most common types of parameters. We will be discussing now so simple
cases where the parameter type is known in advance, the question of analysis more
complex command line leaving at the end.

The number (U_GETNUM)
Relatively simple to download and
interpretation of parameter
number. This may be a number between 0
and 65,535 given in
decimal or hexadecimal notation. The
call returns U_GETNUM
zero (Z = 1), where the parameter is not
a number. In wpadku (Z = 0)

AX registers is appropriately junior and
senior byte values
binary equivalent to the number.
When the parameters are many more may be
separated by commas or
spaces. With U_GETNUM use a command such
as PEEK and Poke
SpartaDOS command interpreter.
The dwustanowy: ON and OFF (U_GONOFF)
Some commands you just something turns
on or off
giving respectively ON or OFF as a
parameter. For the treatment of this
type
parameter used procedure U_GONOFF
library. When the parameter
is ON, marker C register markup
processor is set to 1, and
If OFF - is at zero. When the parameter
is not specified neither he nor
OFF, the procedure reported error No 156
(Bad parameter) giving control
U_FAIL to the procedure and thus
disrupting the program. How to deal with
this problem, described in the chapter
titled "Using the errors".
In this way the command line overlay
KEY.COM proceed.
Options (U_SLASH)
The parameters for some of the programs
is conveniently provided in
a single "options" preceded by the
character slash "/".
To read such from the command line
procedure is used U_SLASH.
All options must be recognized place in
the table. One entry in this

array of options for one, two bytes, in
turn marker
of options and the corresponding letter.
For example, if
program wants to respond to the options
/ A / X, table should look like
as follows:
switch

```
? a
. byte 0
. byte 'A'
? x
. byte 0
. byte 'X'
```

Address the array must pass the U_SLASH
in the records
AX, and the total size in bytes - in the
register Y. When the
analyzed the command line item is given
one of the expected
options, then the flag, in the above
example, they are
marked as "switch? a" and "switch? x ',
will take the value of $ FF.
When given the option lacks the plate,
U_SLASH forward
U_FAIL control to the program by cutting
the error No 156 (Bad
parameter). In contrast, kept machined
when the parameter is not at all
option (ie does not start with a "/"),
U_SLASH returns to the place
Call not doing anything.
In this way the command line interprets
the MEM command.
Keyword (U_TOKEN)
Analysis of the command line for the
occurrence of certain words
provide key two procedures: U_GETPAR and
U_TOKEN.
U_GETPAR copies the next (ie those
indicated by BUFOFF)
parameter of LBUF to COMFNAM, BUFOFF
update, add length
parameter to the trails, and the vector
FILE_P - COMFNAM address.

U_TOKEN gets parameter text that is in
COMFNAM
and tries to find it in the array of
keywords, which address the
transmitted in the registers AX.
Keywords must be placed in
plate for a second, with the last
character of each must be
negatywie (ie setting bit 7). The end
plate determinded by
zero.
When U_TOKEN not match a keyword in the
table, returns from

---

skasowanym tag C. Otherwise, the finds
marker C is set, a battery pack contains
a sequence (ie
token) of the keyword in the table,
counting from zero.
In this way instruction is progressing,
and a batch file IF
SpartaDOS command interpreter.
Device Name (U_GETPAR / U_GEFINA)
When the parameter is the name of the
device itself, for example,
ID disk, you must download it to use
steam procedures
U_GETPAR and U_GEFINA. U_GETPAR, as has
been written above,
gets the parameter of LBUF and insert it
into COMFNAM, however, address
COMFNAM writes to the vector FILE_P.
U_GEFINA but gets parameter from the
place indicated by
FILE_P, interprets it as the name of the
device and, according to the
accordingly sets a record DEVICE ($
0761). His younger four bits
means the number of devices (such as the
number of disk drives), while older
encode the type of equipment as follows:
$ 0x - drive
$ 1x - a clock
$ 2x - a device CAR:
$ 3x - a device CON:
$ 4x - a device PRN:
$ 5x - a device COM:

$ 6x - a device NUL:
$ 7x - reserved
When no ID is not specified, the code is adopted
keep the set device (generally, the current drive) downloaded from

---

variable symbol CURDEV indicated. On the contrary, when given
ID is invalid and can not be decode, control
U_FAIL is transmitted to the error code number 130 (nonexistent
Device).
In the given way of U_GETPAR and U_GEFINA uses such as command
CHKDSK.
The specification of the file (U_GETPAR / U_GEFINA)
Described above sequence U_GETPAR / U_GEFINA may also be
used to get the file name. ID devices, as above,
is translated, then the value of DEVICE ($ 0,761), while the rest of the specifications
is allocated to the path copied to the PATH ($ 07A0, 64
bytes), and the name of the file-filled NAME ($ 0,762, 11 bytes).
The name is translated into the format of an internal DOS,
ie to form NNNNNNNNXXX. When one part of the name, ie main or
extension, are less characters than the 8 or 3, is
supplemented by spaces, if any jokery (wildcards) are developed
in strings of question marks, etc.
This file is a specification for the kernel SpartaDOS Strawn,
primarily for the driver DSK: in SPARTA.SYS.
Calling it features a library system in general, the same exercise
U_GEFINA to jump in, so the user needs only to use

U_GETPAR before.
Specification directory (U_GETPAR / U_GEPATH)
When the expected parameter is the specification of the directory, moving in the same way as above, with the only change that, as with the other procedures, instead of U_GEFINA, you must call U_GEPATH. Review

---

It translated the name of the device on a code device ($ 0,761) in the same way U_GEFINA as a complete path is copied to the PATH
($ 07A0, 64 bytes). NAME to nothing is inserted.
This form of the specification file is useful for calling
the kernel number 16 (kchdir) and 17 (kgetcwd). The relevant functions libraries (CHDIR and getcwd) U_GEPATH same cause, the program
user needs only so U_GETPAR call before.
The specification and file attributes (U_GETATR)
Some commands, such as COPY or DUMP, accepts as a parameter
specification file detailing the optional attributes. For example, DUMP + H FOO.BAR displays the contents of a hidden file (with the attribute + H) FOO.BAR. Normally, however, a hidden file will be ignored.
The task of analysis of such a parameter U_GETATR performs the procedure.
He does it alone. ie there is no need to call first
U_GETPAR. The battery must be transmitted before the default attributes for the file, if the user has not submitted any. Mask
bit attributes compiled by default following
Scheme:
+ $ 01: protected (+ P)
+ $ 02: hidden (+ H)
+ $ 04: archive (A)

+ $ 08: catalog (+ S)
+ $ 10: not protected (P)
+ $ 20: not hidden (H)
+ $ 40: Do not archive (A)
+ $ 80: no directory (S)

Default masks are used most often: $ 20 (not hidden) and
$ A0 (not hidden and not directory).
U_GETATR returns to zero (Z = 1), whereas in LBUF is not enough parameters to be taken - such as given attribute, but not given names file. Mask attributes is inserted into FATR1 ($ 0779), this is one of the criteria to seek the files in the office Library FFIRST and FNEXT (and thus also by all Other features of them benefiting from it, above all, fopen). Besides U_GETATR detail is the same as U_GETPAR, that is, as has been written above, it retrieves the parameter of LBUF and insert it into the COMFNAM, saves in parameter length trails, while the address COMFNAM entered in the vector FILE_P.

Specification file with the default Mask (U_FSPEC)

Some commands, such as COPY, accepts as a parameter path access with the specification of a file or a mask of selecting a group of files. This mask can be omitted, if it is to be'*.*' - COPY FOO> is to copy all of the directory (*.*) files FOO. Program executing the command must, of course, in the treatment posed parameters to identify whether a file name or mask was given, and when finds that it does not, add it.

It is indeed simple, but onerous task - the city check several conditions - performs a function library U_FSPEC.

Called immediately after U_GETPAR or
U_GETATR check whether
located in COMFNAM parameter satisfies
the conditions necessary for
that add at the end of the mask *.* and,
if so, add it
whilst improving the value of trails.
Combinations of different types of
parameters

The most common situation, when the
program gets the first
the name of the file, and then the
options that begin with slash. Analysis
such a command that is the difficulty,
the program first
should proceed as downloading a file
specification
(specific cases that are described
above), and then call
U_SLASH to read the options.
Difficult case to the command of a kind
in Pakistan, which has two
form: ECHO ON / OFF or ECHO TEXT. In the
first form
enabled or disabled is performed by the
echo command
command interpreter. The second form
simply displays the text on a given
screen. The difficulty here lies in the
fact that to identify the
ON / OFF U_GONOFF need to call the
procedure, and that if the parameter
is neither an ON or OFF (but TEXT),
permanently interrupt the program
jumping to U_FAIL.
The only solution is mortgaging the trap
on the error, which may
U_GONOFF generate. This has been
described in the chapter titled
"Handling errors".

Chapter 6: the working name of the file
Converting from 8 +3 format for internal
(PRO_NAME)
PRO_NAME library function converts the

file name
bufadr indicated by the vector ($ 15)
with a Y-1 format
Internal and stores it in the buffer
NAME ($ 0,762 - $ 076C). In
Where the name is the name of the file,
the function returns the result of
non-zero (Z = 0) and a sign of EOL ($
9B) in the accumulator. When it is
directory name, the result is zero (Z =
1), and the accumulator is
separator '>' or '<'.
Transforming the internal format
consists of splitting
File names for parts of central and
extension possible addition
both parts of spaces as possible and
develop Jokers
'*' In strings of question marks.
PRO_NAME is a service from which the
benefit described in
the previous chapter features U_GEFINA
and U_GEPATH. As a result,
including the user never normally have
no need to
directly cause.
Converting from an internal format for 8
+3 (U_EXPAND)
Reverse transformation, ie the name
stored in a format
internal and appearing in NAME ($ 0,762
- $ 076C) to a 8 +3
U_EXPAND function of the library. The
result will be entered into
buffer address AX, from the position Y.
The resulting string of characters ended
is a sign of EOL ($ 9B).

Chapter 7: environmental variables
What is the environment variable
SpartaDOS X is the only DOS on
ośmiobitowe Atari, which
implements of the larger computers known
environmental variables.
The environment variable is a string of
ASCII characters assigned
a unique name. These variables hold some

global settings
the system interpreter commands or
application programs.
An example of the first kind are the $
PATH and $ daytime,
$ COPY second, third $ MANPATH.
Environment variables are stored in the
additional memory
cache size of 256 bytes. The library
offers three functions
allowing for easy access to the data
contained therein.
Read variable by its name (getenv)
To read the variable name with a known
feature
Getenv. Name address the variable
function is found, must give
AX in (name should be completed sign EOL
mark
equality "="). Completion of a negative
result (N = 1) indicates that
buffer, this is not a variable. In the
opposite case, the result is
positive (N = 0), and the value of the
variable is enshrined in the form of a
string
ASCII character EOL culminating in the
buffer output package
mathematical LBUFF ($ 0580).
Read by a variable number (NUMENV)
Alternatively, you can search for
variables not by name, but
according to their numbers further in
the buffer. We will do this procedure

NUMENV. The number of variable need
provide in its battery pack when the
result
is positive (N = 0), the value of a
variable is recorded in the same
format and the same place as in the case
of getenv (see above).
This feature is generally used to read
all
environmental variables at a time, for
example, view them on
screen (as it does to the SET command

interpreter), search
whole, etc.
Record and erase the variables (putenv)
For the record variable in the buffer is
the putenv. New address
variable content to be given in AX. In
this address should be
find a string of ASCII characters end
with a EOL ($ 9B) in the
the form: NAME = TEXT
This creates a variable NAME and assign
the text
"TEXT" as a value. When the buffer
environmental variable that
name already exists, is erasable before.
Comment on putenv same name variable is
a string
Ascii completed by EOL and free trade
equality
will remove the buffer variable with
that name.
NOTE: when combined with NUMENV and in
one putenv
loop in return for specific variables
and delete them, should be
remember that clears the variable
results in a reduction in numbers by 1
all the variables that are followed in
the buffer. Therefore, in
such a loop after calling putenv
kasującym variable DO NOT
increase the counter for NUMENV.

Chapter 8: read and write files
Opening the files (fopen)
The device performs the function fopen.
Required parameters
Opening transmitted are as follows:
1) specification file should indicate
the vector FILE_P
2) the opening of the type opmode ($
0,778)
3) mask attributes wanted to FATR1 ($
0779)
The specification of the file indicated
by FILE_P should take the form of
text (ASCII string end with a EOL).

Specification
equipment must enter the convention
SpartaDOS X, and not the Atari OS - ie
eg A:> Catalog> textfile instead of D1:>
Catalog> textfile.
Opmode has the same function as a byte
ICAX1 Channel I / O XL OS
when you open the file (when the device
functions OPEN D: for
opmode value through the CIO is charged
with the ICAX1).
The value of this variable consists of
two półbajtów. Younger signals
mode of access to the data:
$ x4 - reading
$ x8 - record
$ x9 - annotations
$ xC - data exchange (read / write)
Senior to mask, which has set subsequent
bits
the following meanings:
7 - long format directory

---

6 - attributes tracking mode
5 - opening up the route from the search ($ path)
4 - direct access to the directory
Mask attributes wanted FATR1 ($ 0779) is used at
opening the file to read. The file name will be retrieved
in the directory and open only when they meet the condition
encoded
FATR1 bits:
+ $ 01: protected (+ P)
+ $ 02: hidden (+ H)
+ $ 04: archive (A)
+ $ 08: catalog (+ S)
+ $ 10: not protected (P)
+ $ 20: not hidden (H)
+ $ 40: Do not archive (A)
+ $ 80: no directory (S)
Normally when you open a regular file, the mask $ A0
(Not hidden and not directory).
In the event of error control is transferred to U_FAIL (see
Chapter on "Handling errors"). When the opening was
successful, the file handle
(Eng handle) is entered into FHANDLE ($ 0760).
Closing files (fclose / FCLOSEAL)
The library offers two functions here: fclose closes a
specific file,

namely that the handle at the time of the call is in FHANDLE ($ 0760). FCLOSEAL while closing all the files, which are open at the time.

Page 41

There is a possibility of security before closing the file FCLOSEAL. It is for this purpose to enter its grip FHANDLE ($ 0760) and call the FCLEVEL with a value of $ FF in the accumulator.

The same procedures for the value set on the battery the current value of SYSLEVEL abolishes protection.

Read and write individual bytes (FGETC / FPUTC)

Read and write a single byte file, whose handle is in FHANDLE ($ 0760), exercise the functions FGETC and FPUTC. FGETC calls byte read in the accumulator. When the end there file, in the accumulator will mark EOL ($ 9B), X value in the registry $ FF, a register of tags indicates the result will be negative (N = 1).

Any other error causes the transmission to control U_FAIL.

FGETC does not change the contents of the register Y.

FPUTC sends a byte to a file over in the battery. Call does not change the contents of the registers A, X and Y. Speeches error causes the transfer of control to U_FAIL.

I / O performed by FGETC and FPUTC for mobile devices DSK: and CAR: mikrobuforowane by the library, so that proceed much faster than the readings and records of individual bytes fread functions and fwrite.

Read and write records (fgets / FPUTS)

Read and write records file, which is in the grip FHANDLE ($ 0760) and carry out the functions fgets FPUTS. The parameters for both to be in the registers in the AX address a buffer overflow in Y its length.

The record is a string of ASCII characters end with a EOL and not longer than 255 bytes.

Page 42

When read by fgets will correctly, the function returns from positive (N = 0), zero in the accumulator and the number of read bytes in Y. When there is excess data, the result is negative (N = 1), and battery pack is $ FF. This situation represents a crash 137 (Truncated record) in the XL OS. Any other error results in automatic the transfer of control to U_FAIL.

When the entry by FPUTS, if the string is shorter than the value of Y
Calling the time, must be completed sign EOL ($ 9B).
An error causes an automatic call U_FAIL.
Fgets and FPUTS use of the functions of the library and FGETC FPUTC, so for the device CAR: readings, and for DSK: records and
readings results are mikrobuforowane.
Record formatted text file (FPRINTF)
Record formatted text file to implement the library
FPRINTF. It works identically to the one described in the next chapter
function printf (these are two different entry to the same functions), but
only that it is performed to save the file, which is in the grip
FHANDLE ($ 0760), and not on the console.
As in the case of record results, FPRINTF sends data
FPUTC through, so write to the disk is
mikrobuforowany.
Read and write blocks of binary (fread / fwrite)
Read and write binary blocks and carry out the functions fread Fwrite. Parameters, in addition to the file handle FHANDLE ($ 0760)
pass with:
- FAUX1 / 2 ($ 0782 - $ 0783): address buffer

- FAUX4 / 5 ($ 0785 - $ 0786): the amount of buffer
The size of the buffer must be greater than zero. In the case of
the success of the functions of the result comes back positive (N = 0). When the
Fread end of the file, the result is negative (N = 1). Any other error
button to U_FAIL.
Fread and fwrite are not mikrobuforowane, because the use of these
functions to read or write very small portion of the data (after a few or
a dozen bytes) does not pay off. It is better for this purpose and use FGETC
FPUTC.
Read the length of the file (FILELENG)
FILELENG read length feature an open file, which
handle is in FHANDLE ($ 0760), and puts the result in FAUX1-3 ($ 0,782 - $ 0,784).
Changing positions in the file (FTELL / FSEEK)
To read the current position of reading or writing to the file serves
FTELL function. It saves the records FAUX1-3 ($ 0,782 - $

0,784);
from the beginning of the file, the number of byte to be read or
saved in the next I / O.
The operation is the opposite, namely a change in this position carries out the function
FSEEK. New position have to give in FAUX1-3 ($ 0,782 - $ 0,784).
In both cases, handle the file must be saved in FHANDLE ($ 0760).
When you try to set the position beyond the end of the file to open
Read control will be transferred to U_FAIL error No 166 (Range error). In contrast, a similar operation on a file open for writing
does not produce error, following that record data and closure

_____

will generally rise to the discontinuous (with the "hole", which are not
assigned to any sectors of data).

_____

Chapter 9: console, entrance and exit
Record of individual characters on the screen (PUTC)
Record of individual characters on the screen (ie to the CON: --
Here again, that the output CON: it can be user -
routed to any other file) performs a function library
PUTC. Mark it on to record a pass in the accumulator.
The call does not change the contents of the registers A, X, Y or variables
FHANDLE ($ 0760) and DEVICE ($ 0761).
Record record on the screen (PUTS)
Record record text to the CON: realizes function
Library PUTS. Address string pass in the library
AX registers, and Y serves its length. If the string is shorter than
Y value at the time of the call must be completed sign EOL ($ 9B).
PUTS does not change the contents of the registers A, X, Y or variables
FHANDLE ($ 0760) and DEVICE ($ 0761).
Record formatted text on the screen (printf)
Printf function sends it to the console and the text strings of data
processed in the form of text and formatted according to the pattern. One of the features is that all data is transmitted by placing them directly for the upsurge in JSR printf.
Schematic call is as follows:
JSR printf

```
. byte pattern formatujący, 0
any data
...
```
"The pattern formatujący" is a string of a length to 253 characters ending zero. In the simplest case, the plain text to view, for example:
```
JSR printf
. byte "Sugar krzepi", 0
```
A window is simply a given text on the screen, stopping the cursor
at the end. In order to connect with the transition to the new line, you need to
the end of the string (before zero) add character EOL ($ 9B). But the whole strength of the printf function lies in the fact that given the
characters, which is, again, only the model format text output, you can put formatting commands. Here is a list of them:
%% - Write a single character%
% c - write a single character that is located at the following address
% s - write a string of address
% p - the same, only instead of the address is the address of its index
% x - value from the specified address, write as many 16-bit hex
% b - value from the specified address, write to be 8-bit number dec
% d - the same as the only 16-bit number dec.
% e - the same value of 24-bit
% l - the same, the 32-bit (from SpartaDOS X 4.40)
One command formatującemu must correspond to separate placed as a benchmark indicator format (in the above the pattern of these indicators have been named any data).

Using this is very simple. Suppose that the label symbolizes value
address 16-bit words stored in the usual convention junior / senior. We want to transform this value to strings representing the number in hexadecimal and decimal:
```
JSR printf
. byte "VALUE = $% x =% d", $ 9B, 0
. word value, value
```
When we value at the value of $ 98AB, appears on the screen We think:
```
VALUE = $ 39,083 = 98AB
```
Wrote above that% x is to format the "16 --
bit "hexadecimal numbers. That is indeed, but only if

zażyczy programmer does not itself differently (ie,% x defaultowo treats
values as 16-bit). In fact, the library reads
All numeric values as 32-bit (or, in versions SpartaDOS older than 4.40 - 24-bit). When the value is less than or greater
defaultowe than 16 bits, we need to provide, for example,
JSR printf
. byte "VALUE = $% 2x", $ 9B, 0
. word value
take into account only the youngest byte value at the address value and bring it as a hexadecimal number dwuznakową.
By contrast:
JSR printf
. byte "VALUE = $% 8x", $ 9B, 0
. word value

Doing so that the number printed on the screen will have all eight digits. The number specified here as a sign of% must be a number of decimal
range from 1 to 255 (in fact, it may be up to 65535,
but the older byte will be ignored).
In all the examples given, when a string of digits to print starts with zeroes, they will be truncated. However, you can enforce their
removal preceded by zero value, meaning the
the expected length of the string, for example,
JSR printf
. byte "VALUE = $% 04x", $ 9B, 0
. word value
In the end, it (ie, the length of the string, in the most recent example, "4") does not
must be constant, can induce a library to retrieve it with a variable in
this way:
JSR printf
. byte "VALUE = $% 0 * x", $ 9B, 0
. numlen word, value
The label indicates numlen place in memory where they should printf
get the desired length of the digits.
Similarly, formatting strings decimal digits: giving the target number of digits for wildcard% b% d%% l, we can not truncate
derived string of characters to their desired number.
In the case% c derived is always one character, all
Additional attributes formatting commands are ignored.
% s removal causes the console ASCII string
located at the following address. This string should be ended zero. At the% may show the number of characters. Then,

when

---

string is longer, it will be truncated, and when it is shorter, it will be dopełniony
spaces. % p works identically to% s the only place that address
the address given dwubajtowego index containing the address. Printf does not change the contents of the registers A, X, Y or variables
FHANDLE ($ 0760) and DEVICE ($ 0761).

Read a single character to the console (GETC)

The task of reading a single character to the console meets function
GETC. Read byte is returned in the accumulator. When reading to be successful, the result is positive (N = 0), and X is loaded
$ 00 When there end, it comes back negative
(N = 1) and the value of $ FF in the register X. Any other error causes
automatic transmission control to U_FAIL (see Chapter "Handling errors").

When you need a byte is not reading from the console, but the keyboard
should be used U_GETKEY function. It is waiting for the press key, then returns the ASCII code in the accumulator. Remember However, the latter must, that it exists only in versions of the X SpartaDOS
4.40 and above.

Read the record from the console (GETS)

Read the record from the console carry out the function gets. The record is
a string of ASCII characters end with a EOL ($ 9B). As for PUTS parameters to be in the registers in the AX address a buffer overflow in Y
length. Records can not be longer than 255 bytes.

When the transition is read correctly, the function returns the result of
positive (N = 0), zero in the accumulator and the number of bytes read in
Y. When there is excess data, the result is negative (N = 1), and

---

battery pack is $ FF. This situation represents a crash 137 (Truncated record) in the XL OS. Any other error results in automatic
the transfer of control to U_FAIL (see the "Handling errors").

Features PUTC, PUTS, printf, and GETC GETS performing reading and
record data for the console are just a separate entry (the so-

called wrappers) to
described in the previous chapter, general features and write
read files FPUTC, FPUTS, FPRINTF, FGETC and fgets. Using
the former rather than the latter allows painlessly passed on
to
library system to handle redirects I / O.
Redirection of I / O (DIVIO / XDIVIO)
Entry and exit from the console can be redirected to
any other file by using the DIVIO. She is the
SpartaDOS command interpreter in a situation where the user
for example, command
DIR>> file
.
DIVIO requires a vector FILE_P specification for the file to
which (or whose) is to be established with the console
redirection
(or her). The register Y included $ 00, where is redirected
output (ie PUTC features, etc.), a $ 01, where the entrance is
to be rerouted
(GETC). Possible errors in the opening are reported to
redirect
U_FAIL. In the case of the success of your grip, where it is
redirected data will be in FHANDLE ($ 0760).
The task of the opposite, this is the end of the redirection,
pursues
XDIVIO function. Parameter serves as $ 00 or $ 01 in the
register Y, so
Same as above. Handle the file should be redirected
be FHANDLE ($ 0760). The call will close XDIVIO
this file.
When entering the console has been redirected and end of the
file,
XDIVIO library calls automatically. The same thing with

closing all files and after the onset FCLOSEAL
error.
Vector output (PUT_V / VPRINTF)
Another method of transfer, this time the only exit for the
console, is
PUT_V defense of the vector. Normally he is not any meaningful
value, the program must set the address of the servicer
record. It may be, for example, the procedure for making
direct entries to
memory screen, with the exception of equipment CON: or E:. It
should be
by the end RTS, it is desirable to store the value of records.
When the file handle in FHANDLE ($ 0760) is set to 100 ($ 64),
FPUTC call button indirect (JMP) by PUT_V with the
to the view set out in the accumulator.
Since FPUTC is implicitly called by FPUTS and

FPRINTF, so the output of these functions is thereby also relayed.
FPRINTF Instead, it is better to do so by using VPRINTF - is Another (third now) entrance to the same procedure, and its action is
exactly the same as printf, with the only difference being that the active channel I / O is
automatically switched to handle 100 before
text, and also automatically switched back after him.
A programmer does not have to be so concerned about the content of FHANDLE ($ 0760)
in this case.

Search files (FFIRST / FNEXT)
To search for files in directories indicated designed two Features: FFIRST and FNEXT. Parameter input is the first complete catalog with specifications given at the end of a mask files
FILE_P indicated by the vector, and attributes sought in FATR1 ($ 0779). Found the entry is placed in DIRBUF ($ 0,789 - $ 79F) in the
the form of "raw" entry in the directory format SpartaDOS (ie such as is saved to disk, see the "X Handbook SpartaDOS user ", Chapter 7).
FNEXT not require any additional parameters, searches simply following the entry criteria set by the previous function
FFIRST.
Signaled the end of the catalog is in both function by the return of
a negative result (N = 1). Any other error button to U_FAIL.
NOTE: FFIRST in fact open to specify a directory
read a file handle is placed in FHANDLE ($ 0760). Therefore, after
finished searching, it is necessary to call for fclose
the handle in order to close the file directory.
Read the catalog (FDOPEN / FDGETC / FDCLOSE)
The library includes three features which provide a catalog of formatted, which is legible for humans (such as such, what appears
on the screen after the interpreterowi commands command DIR). They are
turn FDOPEN, FDGETC and FDCLOSE. FDOPEN open stream data directory, FDGETC collected from him asking for it in a byte
accumulator (and showing a positive result in the registry Tag

---

CPU), and FDCLOSE directory open previously closed by

FDOPEN. FDGETC errors reported to U_FAIL, with the exception of status
the end of the file - when takowy occurs, the function returns to the program
calling from a negative result (N = 1).
Each of them does, in principle, only one: call the appropriate
misc_ entry feature, namely the order No. 6 (misc_fdopen), 7 (misc_fdgetc) and 8
(misc_fdclose). Read formatted catalog so you can pursue
by the way (ie by the vector misc_), where the library system is
unavailable due to the launching of the command X. You just need to
remember that misc_ never gives control to U_FAIL, paying
Just after the error code in the battery pack instead of a further byte
read by misc_fdgetc not be provided in the registers, but on
Page zero ICAX6Z (address $ 2F).
The opening is a specification for FDOPEN directory
FILE_P determined by the desired manner and format in
Opmode ($ 0,778). SpartaDOS X versions older than 4.41 are familiar with only two
how to format: The format of "long" SpartaDOS (opmode = $ 80) and
format "short" AtariDOS (opmode = $ 00).
In SpartaDOS from version 4.41 and above is considered by the opmode
FDOPEN as a mask selecting bits (separately) the means
formatting the different elements of the directory. The
importance of bits:
+ $ 80 - long format directory
+ $ 40 - display the attributes
+ $ 20 - insert a space between the name and extension
+ $ 10 - full stop after the name (if bit 5 = 1)
+ $ 08 - two spaces before the name, '*' for a secure file
+ $ 04 - the time in the 24-hour format
+ $ 02 - when the long format directory (Bit 7 = 1), without a second time in
short format will be displayed name extensions directory

(instead of the standard
ƒ ... "
), The directory will be denoted by a colon
before the name.
+ $ 01 - display the size of the sectors in (the bit 7 = 0).
To maintain compatibility with older versions SpartaDOS X
given by the user is transmitted to $ 00 to $ 0B, and $ 80 to $ A2.
NOTE: misc_fdopen, and thus also function libraries

FDOPEN, the discreet opening the file directory to read. In
Therefore, (and) should not be neglected FDCLOSE call (or
misc_fdclose), when the reading is completed, and (b) may be
open
only one file at a time, because misc_ is unable to remember
more handles.

Page 55
Chapter 11: loading of binary
Load the program into memory (U_LOAD)
U_LOAD symbol indicates SpartaDOS binary loader. It fills
three tasks: (a) the loading of binary programs into memory
along with
mobilization, (b) without loading run, and (c)
run them.
Inputs U_LOAD is a specification file indicated
FILE_P vector control and value in the registry FLAG.
Indicated
the file is opened for read-compatibility mode with $ PATH
(Opmode = $ 24) and, if this is correct binary is loaded into
memory. Any errors cause interruption of the operations and
jump to
U_FAIL.
FLAG register can decide what to do with the loaded program.
When
7 FLAG bit is zero, the program is started immediately. In
Otherwise, only the updates are free
memory (for programs in the format relokowalnym SpartaDOS),
and
control returns to the caller. Mobilization (already without
rechargeable) follows the re-call of U_LOAD
the same specification file and registry skasowanym bit FLAG
7.
To remove the program from memory (U_UNLOAD)
When the program was running normally, its removal from memory
The following soon as the interpreter to give control commands
SpartaDOS. In the opposite case (loaded with FLAG> $ 7F) in
the
to remove the code from memory must call the U_UNLOAD.
It removes everything that the user previously load
by U_LOAD.

Page 56
Chapter 12: File Management Functions
Change the file name (RENAME)
Change the name of that file carry out the function RENAME.
Parameter input specification is complete a file name which
is to be changed. These specifications should indicate the
vector
FILE_P. After the source specifications should be, separated
a space or a comma, name, which is to be attributed to the

file. Possible
Errors are reported to U_FAIL.
In earlier versions SpartaDOS X can be, using
RENAME, several files in the directory to give this one
same name. From version 4.40 is impossible.
Deleting a file (REMOVE)
Erases the file performs the function REMOVE. Parameter
input specification is to remove the file indicated by
FILE_P. An error button to U_FAIL.
Removing directory (RMDIR)
Erases the directory function RMDIR implement. Parameter
input is the specification of the directory (without final
separator)
indicated by FILE_P. An error button to
U_FAIL.
The catalog must be empty. Removing the directory together
with those in
the files and folders to other opportunities outside the
library. When
necessary, call contained in the module ROM program
DELTREE.COM with the name of the directory to delete it as a
parameter.

The creation of a directory (MKDIR)
Creates a new directory feature MKDIR. Parameter input is
specification of the directory (without final separator)
indicated by
FILE_P. An error button to U_FAIL.
Changing the attributes (chmod)
Change the attributes of the file performs the function of
chmod. Required
parameters are determined by specification file FILE_P, mask
wanted attributes in FATR1 ($ 0779) and the new mask
attributes in FATR2 ($ 077A).
Mask wanted attributes elect files, which attributes
to be amended. It works the same way as the other
taking into account the functions (such as fopen):
+ $ 01: protected (+ P)
+ $ 02: hidden (+ H)
+ $ 04: archive (A)
+ $ 08: catalog (+ S)
+ $ 10: not protected (P)
+ $ 20: not hidden (H)
+ $ 40: Do not archive (A)
+ $ 80: no directory (S)
In the mask of attributes defined individual bits have the
following
the importance (when set to 1):
+ $ 01: security (+ P)

+ $ 10: unprotection (P)
+ $ 02: hide (+ H)
+ $ 20: disclosure (H)
+ $ 04: archive (A)
+ $ 40: niearchiwalny (A)
Attribute S (directory), naturally, there is no way to change this.
Error gives control to U_FAIL.
Make sure you download the BOOT (SETBOOT)
SETBOOT sets indicated (by FILE_P) as the binary file, at startup, which is to be automatically loaded by loader located at the beginning of the drive. This function of course only on drives formatted SpartaDOS. The possible causes confusion the transfer of control to U_FAIL.

Chapter 13: Other features Disc
Formatting the drive (format)
FORMAT function does not do anything, other than to call on the screen SpartaDOS Formatter menu. The caller did not provide it no parameters.
In earlier versions of X SpartaDOS call by the formatter require prior "manual" (by interference in the records banks transfer module) cartridge'a switch to the bank 0 From DOS version 4.40 at this inconvenience has been removed, the system itself selects the appropriate bank, to the formatter no longer resides in a bank 0
However, if we want to build compatibility, it is better to use the XIO 254
Record fresh directory (builddir)
Builddir takes a "soft" format, it's just new blank saves from the root, bitmap disk and bootsektor SpartaDOS format. It is the only method of formatting ramdysków and partition the hard disk. Input parameters are:
1) code device in DDEVIC ($ 0300). $ 31 for the drive
2) the number of devices in DUNIT ($ 0301).
3) a pointer to the new label disc in AX.
The label is eight characters ASCII. If he were to be shorter, you must supplement it with spaces to that length. The label can not start with sign space.
The function automatically reads the configuration of the drive and on this determining the rest of parameters (number of sectors, etc.).

In SpartaDOS
X 4.40 is automatically turned on optimizing bitmap (function
Optimize formatter). The status of the operation is returned
to the registry Y.

Page 60
Can not read the disk parameters (GETDFREE)
GETDFREE read different information on the disk and puts them
in
PATH ($ 07A0-$ 07DF). Before you call have to give
specifications
equipment indicated it FILE_P vector. Any errors are
reported to U_FAIL.
Data (17 bytes) are placed in the PATH as follows:
+ $ 00: the identification code file
+ $ 01: the number of encoded bytes in the sector
+ $ 02: the total number of sectors (2 bytes)
+ $ 04: current number of free sectors (2 bytes)
+ $ 06: the name of the disk (8 bytes)
+ $ 0E: the number of sequential
+ $ 0F: the number of random
+ $ 10: byte irrelevant
Identification code file to see what system
Files can be found on the disk, as follows:
$ 0x - filesystem AtariDOS vx0 (eg $ 02 = DOS 2.0)
$ 11 - disk file SpartaDOS v. 1.1
$ 2x - disk file SpartaDOS v. 2.x
$ 3C - PC-MIRROR
$ 40 - filesystem module SpartaDOS (CAR:)
$ FF - disk file MyDOS
NOTE: GETDFREE will call the kernel No $ 13 From version
4.41 SpartaDOS format of the data returned by the kernel
call differs completely from what draws GETDFREE. In
SpartaDOS 4.2x it was not so - the kernel $ 13 in the 4.4x is
incompatible
backward-compatible with 4.2x. For the sake of compatibility
programs should be

Page 61
refer to it only by the library (ie by GETDFREE), and
in the absence of such a possibility, the system function XIO
47
operational. Both of them to convert data returned by the
kernel
to the "old" format, in accordance with 4.2x
Change the current directory (CHDIR)
CHDIR changes the current directory on the drive, which
specification
FILE_P vector points. Error causes U_FAIL to jump.
Read the current directory (getcwd)
Getcwd reads the path from the root of the drive to

current directory and puts it in your PATH ($ 07A0-$ 07DF).
Specification of the disk (as text), indicate the vector
FILE_P. Error results in calling U_FAIL.
In earlier versions of the X SpartaDOS getcwd not pay
name extensions directory. From version 4.40 has been
corrected.

Chapter 14: ancillary functions
32-bit multiplication and division (MUL_32/DIV_32)
The library contains two procedures for calculating the total
figures
Free trade: MUL_32 performs the multiplication of two numbers,
32-bit, and
DIV_32, respectively, sharing none.
Ingredients operations must be entered in the order of bytes
the reverse of the normal (oldest first byte!) as follows:
1) multiplicand (or dividend) under COMTAB + $ FF (4 bytes)
2) the multiplier (or divisor) under COMTAB + $ 0103 (4 bytes)
The result of the calculation, in the same way in reverse
order of bytes
We take from COMTAB + $ 0107. Both the signal
overflow by setting bit register markers C (C = 1).
Replacing the small letters on the large (ToUpper)
The call ToUpper code ASCII lower-case letter in the
accumulator
turns it into a large letter code. If the code does not mean
small
letters, it is not done nothing whatsoever.
Checking the directory separator (CKSPEC)
The procedure CKSPEC verify that the mark passed in the
accumulator
is one of the following: ':', '>', '\', '<' (separators are
likely to
occur in the specification file), and if so, returns with the
result
zero (Z = 1). In the opposite case, the result is non-(Z = 0).

Chapter 15: procedures for initiating
Initiating overlays after RESET (S_ADDIZ)
Part overlays must be initiated after each
Pressing button. For example, it can be all kinds of drivers
Install the CIO, as contained in SpartaDOS X 4.41 CON64.SYS
and CON80.SYS. When you reset the system they need at least
re-install the OS array handlerów-in.
For the recording of such overlays is the S_ADDIZ library.
Please pass to it in the registers AX address procedures for
initiating
overlay. DOS entered this address to a special queue, and
after RESET

performs in turn sub-registered there.
A queue is only five seats, the lack of registration
is signaled by the return from S_ADDIZ with an tag
C Tag Registry (C = 1).
Exit to DOS-u (_DOS)
Calling JMP _DOS has the same effect as JMP (DOSVEC)
- The caller is finished and removed from memory, and
control returns to the command interpreter. A better way to
the end
program is to implement an order to RTS.
Warm restart SpartaDOS (_INITZ)
Calling _INITZ causes warm restart SpartaDOS. All
the files are closed, the settings reset, resident drivers
kernel initiated from the beginning.
_INITZ Is one of the procedures triggered after hitting

_____

Reset by the user. In earlier versions SpartaDOS X
execution resulted _INITZ the current paths of all
drives on the main directories. From version 4.40 driver
SPARTA.SYS
distinguish the call origination from the measure _INITZ from
each other and
paths are not reset.

_____

Chapter 16: the manipulation of symbols list
Searching the list of symbols (S_LOOKUP)
The system contains the procedure for searching the list of
symbols, and
add and remove symbols. Normally, however, these programs do
not
benefit from them directly, because all matters related to
deals with the symbols of binary loader SpartaDOS. List of
symbols
attached to blocks of binary program is, after loading the
code to
memory, it automatically translated into addresses, and these
are inserted in
adequate space program. Similarly the case definition
new symbol of the loaded program, this definition encoded
is the same binary structure, and creating new
symbol and supplying it to the list of global deals with the
loader.
Sometimes, however, there is a need to "manual" search of the
list
symbols by the program. For example, is located in the ROM
disk CAR:
Z. SYS driver needs access to a symbol I_FMTTD
defined by TD.COM, but can not be dependent on its
presence, because then it would not be possible without the

load Z. SYS
prior to load TD.COM (inability "zresolwowania" symbol
included on the list annexed to interrupt the
loading and "Loader: Symbol not defined").
In this context, you must use a procedure S_LOOKUP.
It requires registration of the name of the symbol, as
supplemented by up to eight characters
spaces, if necessary, under the symbol + $ 02 Calling JSR
S_LOOKUP
Calls to zero (Z = 1), where the wanted symbol does not exist.
Otherwise
Otherwise, the result is different from zero (Z = 0), and
under the symbol + $ 0B and
SYMBOL + $ 0C sequentially is a junior and senior byte address
reported by a symbol, but we have EXTENDED index
memory. EXTENDED byte is well before reset.

_____

Since the same procedure S_LOOKUP, structure and characters
EXTENDED variable symbols are indicated, it is clear that this
path
access to the list of symbols is open only for programs
written in
SpartaDOS binary format. Besides simple binary Atari DOS-in
usually do not need access to the list of symbols. However, if
there had been such a
need, there is a second entrance to S_LOOKUP, not quite, that
are
under constant address, it's a lot more simpler to use.
This procedure, called the FSYMBOL, introduced in
SpartaDOS v. 4.40. Entry to it is at $ 07EB.
An input address is supplemented by space-name symbol
over in the registers AX (mł. / st.). In the event of a lack
of symbol
procedure returns to zero (Z = 1), otherwise the records will
be AX
contain the address pointed to a register Y - an index of
memory.
NOTE: at the time of the launching of the X command list of
symbols
and the procedures described here will become available to
you.
The addition symbol (S_ADD)
After calling S_ADD symbol indicated parameters will be
included on the global list of symbols. Data entry pass in
SYMBOL structure, it seems, for the record, as follows:
+ $ 00 - $ 01 rate for the next symbol (2 bytes)
+ $ 02 - $ 09: the name of the symbol (8 ASCII characters)
+ $ 0A: byte control
+ $ 0B-$ 0C: address indicated by the symbol (2 bytes)
S_ADD requires the fulfillment of calling names (with

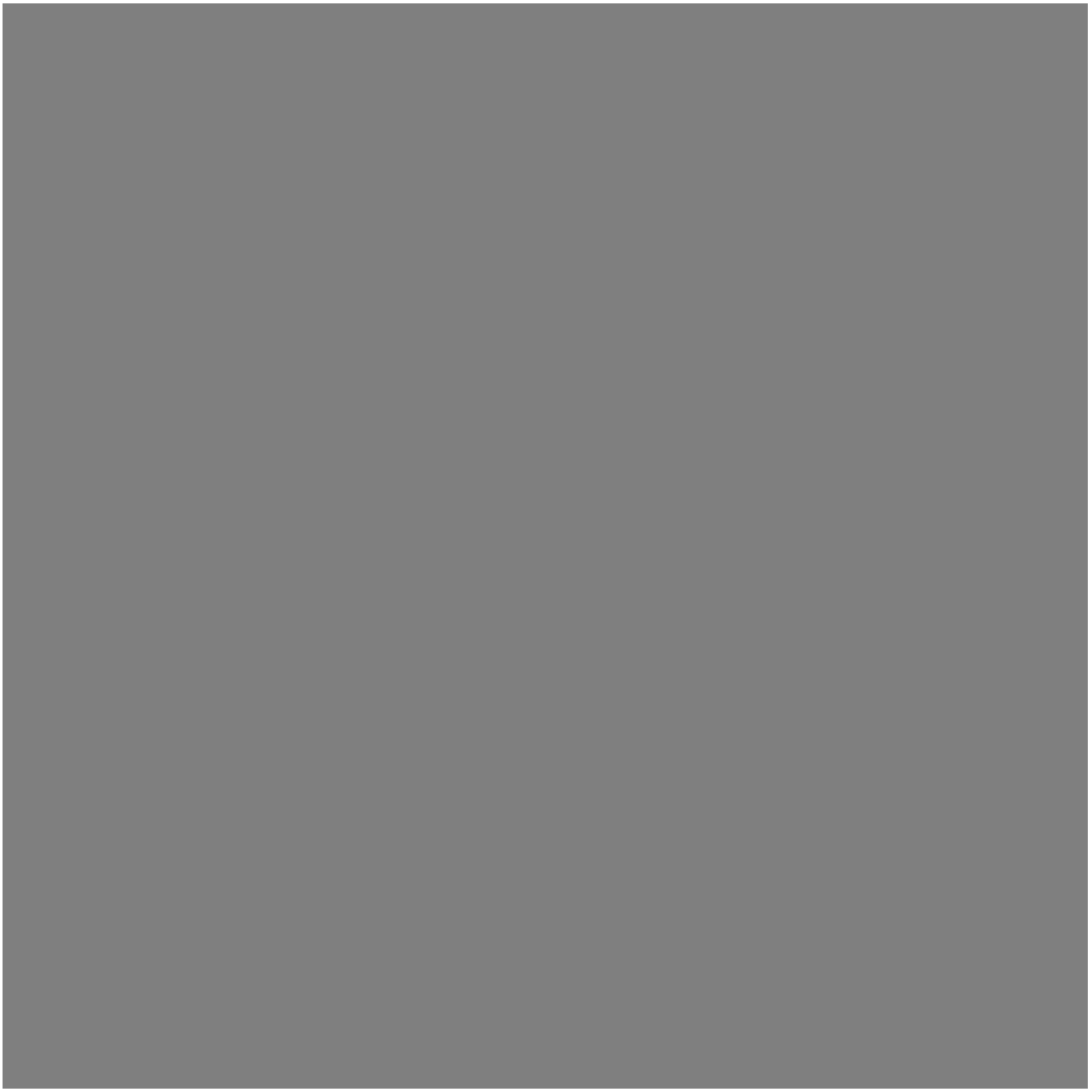
complementary spaces to eight characters, if shorter) and
the address in bytes SYMBOL + $ 0B and the + $ 0C. Index

---

memory should be recorded in the extended, as long as there is
no longer there.
Bayt control is completed automatically, just a pointer to
symbol next on the list.
The symbol is added in the place indicated by MEMLO,
This rate is then increased by 13 bytes.
Stripping (S_CLEAR)
S_CLEAR procedure removes the global list of the symbols of
symbols,
where PID is greater than keep the set. It is fundamentally a
part of
U_UNLOAD procedures, and "keep your 'number of applications
may not
be explicitly changed by the user. Therefore, the objective
S_CLEAR export to the global list of symbols is not quite
clear.

Chapter 17: other symbols library
In addition to the above in the list of symbols, there is
still a handful so far
not covered, and some were not even mentioned. These are the
variables:
STATUS, SYSLEVEL, H_FENCE, tables CARVARS, DEVSPEC,
Devname, COMTAB2, procedures _CIO, _edit, and _CRUNCH
XDFREE. Most of them applies only to special
system drivers, a description of their functions and their use
will be in
future in a supplement to this paper on
write drivers.
XDFREE symbol indicates an extensive procedure to replace
GETDFREE, but since it was introduced to SpartaDOS
X 4.41 so fresh that, in principle, it can be described as
experimental, its
description is omitted for the time being.

Index procedures and system variables



Original Polish text:
je sobie do ulubionego asemblera.
Suggest a better translation