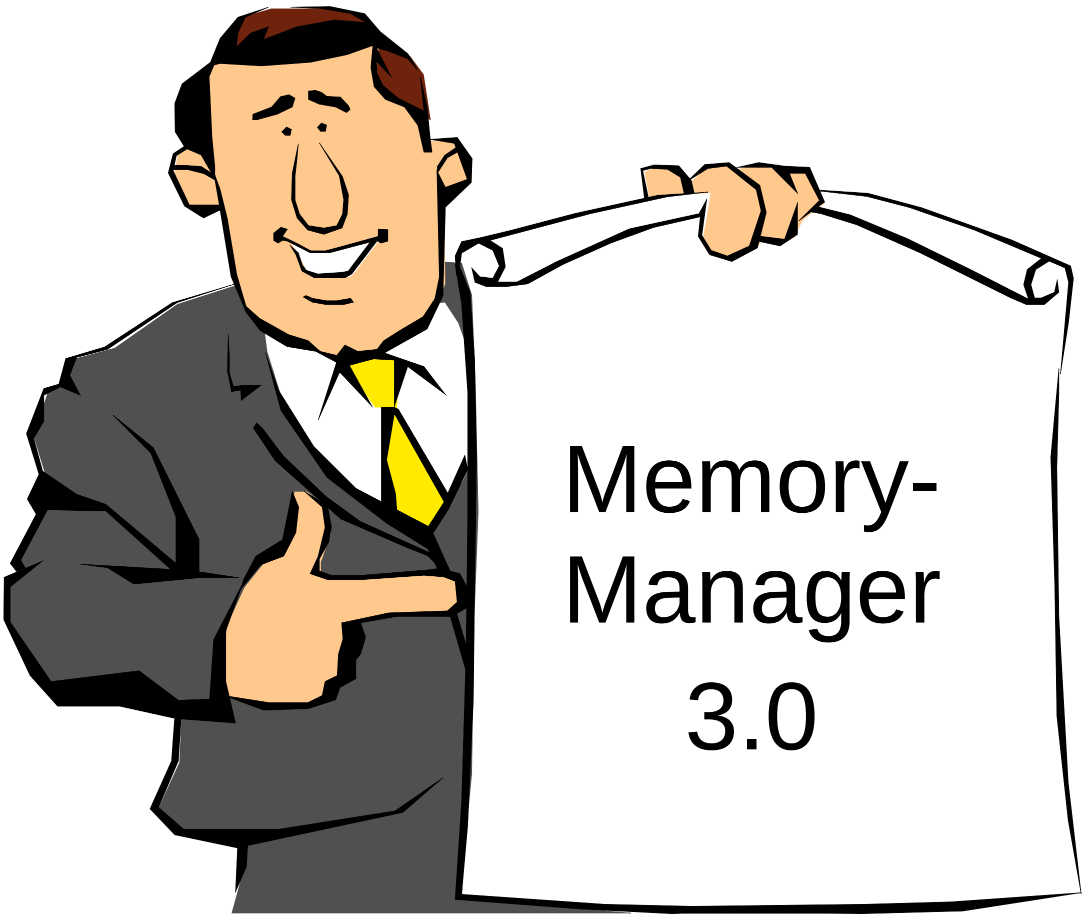

A Program

from the

system-99 user-group

Last Manual Edit: 2017-10-17
Translation by Bob Carmany

Actual versions at system-99 user-group

Note: This manual describes MM in a form that may not be your particular program version. Please read the file MM-README on the program diskette where the actual features of your program are listed!

Forward of the Program MEMORY – MANAGER

Memory-Manager is a new, innovative program for the TI-99/4A. It was developed between the years 1985 to 1989 and remains (even to this day!) the only program that gave/gives a comfortable means to allow working on individual files. In particular, program files.

The basic idea is no longer new, and the authors are astonished, that there have not been other programs to accomplish the same “software-smithing”

MM is present at a point in time when the TI market has become an absolutely “freaky”. At a time where each introduction into the market of a so-called compatible questions the existence of its ancestors; like the fragmentation of the 9640- and TI- software rather than increasing the utility and scope or improving the software for the TI, which should be necessary and straightforward for a TI-Program.

MM is not tailored to all of the possible stages of the development of the TI-99/4A.This is the result of the unfortunate developments in the last months and years, with the shrinking market, that software authors have seen that causes some to think that there is no longer a broad market to be spoken to.

The results are programs like TELCO, with an overlay concept that does not fully function or PRESS that does not exist as a Super-Duper text system.

Memory-Manager was developed in such a way that that only a minimum configuration is necessary and does not recognize any other extensions except for an 80-column card and Speech Synthesizer and nothing else. It uses no GRAM memory for administrative functions, does not pre-suppose a RAMdisk for overlays, or use any other unknown tricks. It is able to work with any memory storage of the TI (Module, RAM-Disks, GRAM-Cards, DSRs etc.). It is so compactly programmmed that it resides entirely within the 32K-memory with the program, help text and files to to be worked on. That being done, the entire low memory has remained free which benefited the auxiliary software concept.

Program Features

Memory-Manager (or abbreviated as 'MM') is a users program. That means - MM is a program for the advanced user who wants something to use as a beginning. It can do much but is not always simple or straightforward. That should not lead to the conclusion that it is complicated to understand!

Due to its universitality, is is sometimes not easy to see what MM does. However, through the use of prompts, fatal consequences are minimized.

MM uses a small character set (with short descenders) and German umlauts in the German messages. It uses a window technique (with rigid function borders) that not only looks nice but also improves clarity of function.

MM is freeware, and is not copy-protected and thus, is fully copiable. It is meant for your own backup copy and naturally for copies to be distributed to your friends. Please read section 2 where the rights and responsibilities of the users are explained!

Workings of MM

MM is constructed so that one must read the COMPLETE instructions at least once. During its use, it is not necessary to have the instructions on hand since there are help windows available. Due to storage restrictions, it is not possible to have extensive help windows available and it was through the concept of a manual that suggested this structure. In the long run, it was decided to have as many functions as possible and not carry any other baggage forward.

Program Structure

MM is structured in blocks with each individual function having its own function block that is associated with a “Submenu” or 'Module' display. These modules have names that agree, in the most part, with the functions that they employ. For example, the Transfer-Module contains the various storage types of the system and can be moved in and out of the VDP buffer.

The whole process can be compared to a wheel with the hub being the Main Menu. From there, the Submenus can be selected with the press of a key that corresponds to the first letter of one’s choice. The Submenus call themselves with the press of the designated key together with the CTRL key. The Main menu is the only point where one can leave the structure of MM!

Programming

MM is so programmed that the memory space of the TI-99/4A is optimally used. Functionality was chosen at the expense of elegance as is often done in small systems which is the only way to overcome the inadequacies of the known virtual memory limitations of the TI and lengthen the program. We ask the indulgence of whoever wishes to use disassemblers to look back on this point – when you come across Spaghetti-Code that causes one to swear. The length of the instructions in your version is sometimes longer than necessary and it may be that a revision for MM is in order. Otherwise: improvements of a sincere nature are always welcome!

The program is perfectly environmentally independent e.g., it does not pre-suppose any module routines or the like. It is such that it has its own (each ROM tested!). The program was developed and works on a TI-99/4A with a 16-bit 32-K memory expansion, and a CPU with a clock frequency of 3.5 MHz (14 MHz Quartz). Therefore, the program is 16-bit compliance (with no mischief with the VDP!).

Functions-Shorthand (Hot-Keys)

The individual submenus can be reached through a single keypress. Also, if one wishes to leave a submenu, most of them can be similarly exited. Usually very simple parameters will work.

Function Parameters

Nearly every function in MM is driven by a word (parameter) that forms the basis of the function for execution. One must, for example, before loading or saving files, know the name and length (and certain other information). The function will be executed after the parameters are set!

Prompts

When you exit a function that has permanent consequences (such as storing the buffer contents or altering the filename, etc.), and the program recognizes such action, a followup question will be asked. In this case, a warning window will ask if you wish to continue. One can continue (always with a capital “J”) or through any other keypress to cancel the action.

This is known in the new German computer slang the WYSIWYG Principle.- What You See Is What You Get. Freely translated it means: what you see happens directly! MM expects that BEFORE you leave a function, you give it all of the required data. With MM, an ALREADY PREPARATORY function is thus executed (always with the “A” key) but not until the start of the operation or new data has been requested (What would you like to save? How would you like to have it called?). The only exception is in the Buffer Editor and here there was not enough space on the screen.

The opposite effect of this parameter control is that a program of finite length has only a limited understanding of what a thing is. Error-trapping must be used economically and used only when something really catastrophic can occur. You should only execute a function when you are sure of what you are doing. A highly flexible program such as MM should not interfere everywhere so that the user has as much latitude as possible.

It is said: Crap in, crap out!

Naturally, there is also a HELP function in MM. One may call it by pressing the key combination of FCTN-7 (sometimes will not fit on the screen) and so is not always fully displayed. It is meant only as a small memory aid (and should be used as such) when the function list is actually needed.

MM is a dynamic program and is constantly being extended. Commands with known keypresses have been implemented, however, may not be implemented in your particular version and are indicated by inverse display in the auxiliary help window. Pressing these keys will have no effect! The other functions will bring up the appropriate warning window.

Word applications

MM knows three different kinds, arithmetic or parameters. They can be given as strings, hexidecimal characters (words) or through a Screen-Editor. There are also some less important details (such as disk drive number and time).

String-Applications

These are used by file and device names. Input is at the blinking cursor with automatic repeating keys and blank characters are not permitted by TI-I/O conventions. Therefore, the end of the string is fixed by the first blank character.

Hexadecimal Applications

Hex inputs are always 16 bits or one word wide. The input place is indicated by an inverse video display. The input procedure follows the '1 Nibble per keypress' with a running character displacement. The displayed word is always sent with ENTER. FCTN-9 (BACK) ALWAYS restores the old word!

Screen-Editing

MM has a number of so-called full-screen editors. The cursor has full freedom of movement throughout the zone and input can be entered anywhere within the area and sent with ENTER. FCTN-9 aborts the action.

Auxiliary Software

A special characteristic of MM is that it allows the free loading of AUXILIARY SOFTWARE as long as it follows certain boot conventions and it can even make full use of the MM system routines.

For such software the entire area of low memory as well as a loader that will directly load the file into this storage area and object code must be created with the E/A module assembler (no Mini-Memory is necessary!) This auxiliary software is used at present as a control program for a variety of EPROM programming devices but is by no means restricted to it. You will find the implementation in another section.

List of the abbreviations used in this manual:

♦	CTRL	Control-Key of the TI; always pressed with another key.
♦	FCTN	Function-Key of the TI; always pressed with another key.
♦	I/O	Abbreviation of Input/Output, (German: Ein-/Ausgabe).
♦	MM	Abbreviation for Memory-Manager.
♦	VDP	Stoage area of the TI; workspace of the Editors.
♦	ZP	Temporary Buffers. Transport- and Buffer options for each of the Editors.

Used terms/characteristics:

Header Bytes

'Beginning' of program files (see there) that contain auxiliary information about the contents of the file. It is usually 6 bytes in length (0-9 possible). Also see I/O-DISK.

Configuration File

This is a special file in which the default parameters of MM are stored. It loads immediately after the main program is loaded (Name: MM-PAR).

Main Keys

These are special keys that allow for the direct switching between modules within MM.

Program File

This is a special file, of non-relocatable code, in blocks of a maximum of an 8K less 6 bytes of auxiliary information. The auxiliary information consists of a count of the data bytes and the load address in CPU-RAM for the start of the file. Two bytes are used as concatenation information.

Relocatable

A characteristic of programs that function independently of the occupied memory. In the TI, there is no provision for this code and it must be accessed by the loading program.

Tagged-Object-Code

Aids relocatable code. It is a special file type in which the relative storage addresses are linked to the actual storage locations.

Capabilities of the Memory-Manager

Memory-Manager is a comprehensive program that contains all of the necessary functions by which the manipulation of files and memory in the TI-99/4A can be done.

Besides, MM opens the revolutionary concept of possible auxiliary software and user created extensions that can be integrated without any problems.

The main program (or the auxiliary software as a shell) makes the following functions possible:

It is a screen-based editor with full cursor control. Temporary buffers that work with the editor memory. Extensive shifting and manipulation possibilities of the editor buffers are supported.

It is possible to to read almost all of the memory ares of the TI-99/4A and to describe the flexible transfer possibilities.

Access to the data on the disk is at sector level (block sector editor). An error-tolerant disk catalog finds defective or protected programs or data. A genuine file editor makes it possible to work on files up to 12 Kilobytes in length at ONE TIME and to accomplish this will shift past the sector border.

The data in the editor buffers, which is fed from various sources, can be printed in various ways. Manipulations of ASCII representations are also possible as conditional offsets or virtual along with the running storage addresses which are not necessarily correlated with the address of origin.

Full configuration and the automatic return of the program to its last configuration is allowed. All preset values, file and printer options, mode of operation of the editors, etc. will permit the return to the last action without a large effort or expenditure. Experimentation can, for example, cause a system crash and the previous point can be established from where the program was exited.

Freely loadable auxiliary software with access to the resources of the main program is enabled. See the details in the appropriate section.

The virtual Tagged-Object-Code-Loader, allows for the first time, any machine code (Assembler/GPL) to be directly loaded into any memory location (DSRs, ROM at >0000 or >2000 etc.). It is now possible to load or program DSR-EPROMs without RAM-DSR-Cart or GROMs without a GRAM-Cart!

Through the concept of auxiliary software the function range of the Memory Manager main program is not limited to the functions described here. Through a special program interface, an experienced programmer, with the necessary knowledge of TMS9900 Assembler, can merge applications from the main program without having to resort to writing these functions again.

The following auxiliary software is now available:

Control of various EPROM program devices.

The current is enough for simple devices like those built in TI-Revue (Quick-Pulse etc.) and the version of the free standing Eprommer for the P-Box with the control software like the EPROG 27011 from Auerswald.

Adaptive Sound list handling.

By these means one can search all of the programs and files for sounds and tones. The results can be heard and stored.

In preparation (among other things): Flexible (virtual) Disassembler, the virtual memory can be worked in the editor buffer in MM independent of the address range.

The details are in the appropriate sections of this manual and the manuals of the individual auxiliary software.

Hardware requirements of the Memory-Manager

Memory Manager is programmed exclusively in machine (assembly) language. It is not a hybrid of c99 or TP99 and TMS 9900 machine code. For this reason, it works at all times at the highest possible speed of which the TI is capable. Therefore, however, a 32k memory expansion is necessary!

Absolutely necessary:

A TI 99/4A console (The GROM-Revision is not significant).

A 32K Memory Expansion or functionally equivalent RAM-Disk (CorComp) with 32K emulation 8- and 16 Bit-types are required.

A Disk system with TI, Atronic, CorComp, or SNUG-Diskcontroller installed.

A 5.25” disk drive (The distribution diskette format is SS/SD).

Recommended:

Interface card (CorComp, TI or compatible) also a serial or parallel printer.

A Horizon RAM-Disk (ROS 7.3 or higher is recommended). The capacity is not significant. A Mechatronic 80-column card (with a DSR version 2.0 or higher) or a Dijit AVPC card. With such video cards a SNUG-Disk Controller is recommended.

Not Recommended:

Any of the Myarc products. Errors are generated in MM by these cards and cards from this company should be removed. This applies to everything from the 32K to the 9640!

A Horizon RAM-Disk with a V 4.0 operating system.

Foundation 128K Card.

WARNING:

Memory-Manager works with TI-99/4A or fully compatible systems (functionally identical reproductions based on the TMS 9900). Due to design incompatibilities with the TMS 9995 with that of the 9900 the Memory Manager is not compatible with systems like the Myarc 9640 System and is NOT functionally secure!

MM was not developed on these devices and that is why its usefulness on these machines is not guaranteed. That does not mean that MM is completely not functional on the 9640 (a few traps have been added to account for known design flaws in the 9640 . . .). However, there are undocumented function errors that are due to the slipshod Myarc architecture and not to MM. If the 9640 were compatible, then this paragraph would itself be obsolete!

Important Copyright Notes!

This program Memory Manager, this manual, and all of the auxiliary programs are protected under the context of copyright. Those ignoring the copyright notice and each offense against it can be criminally prosecuted (the damage in confidence to the TI software market is what has made it a waste land). Those persons know what is meant here.

This program is distributed under the freeware concept. Subsequently, the copyright remains with the author and not the buyer and, although the use of Memory Manager is free, all who use Memory Manager are encouraged to send what they can. Such a donation identifies the user as a “registered User”.

The user of this program has the right to use it on as many TI systems as they wish. Furthermore, it is permissible to make additional copies of this program and pass them on unaltered (!) to a third party.

Updating the freeware donation entitles the new user to be on the Memory Manager list and receive up to 5 version updates without cost (more than DM 20). The option also allows for the update of the manual as well as a new disk to replace the old original. The manual will contain the details of any upgrade of options.

The user of Memory Manager is responsible for any copyright violations resulting from the use of an unauthorized original or backup disk.

The passing on of altered program copies or the manual to a third party that results in the unauthorized reproduction or facility to reproduce the program or manual in an altered form or the reverse engineering (disassembling the program code) and other copyright infringements are also prohibited activities.

The exceptions are changes to the main program or auxiliary programs that are done with the permission of of the author. Such altered programs are NO LONGER freeware programs and are no longer to be referred to by the original name. The author does not assume any liability for damages caused to software or hardware by the use of Memory Manager deliberately or negligently (negligence by not following references in the manual). Further, there is no liability associated by the author with third party users of of Memory Manager.

Contact addresses:

Christopher Winter	Harald Glaab
Xxxxxxxxxx xxx xx	xxxxxxxxxxxxxxx xx
xxxx xxxxxxxxxxxx	xxxxx xxxxxxxxxxxxx

Starting MM

Disk Drive

MM can be started from diskette or RAMdisk. The condition (simply expressed!) is that the least significant byte at the VDP address >3EEB according to the VDP RAM conventions is for the disk system and is parsed. By this means the loader knows which drive assembly or which disk drive MM was last addressed and MM is loaded from there. Also, the configuration file is loaded from this drive. Naturally, MM does not see >3EEB again and orients itself later to the VDP Block Link Pointer at >8370 and the concatenated list to which this pointer subsequently refers.

If it was indicated in the configuration file, the auxiliary software is also loaded from this disk drive. If MM cannot identify the loading disk drive, it is impossible to reload the configuration file. Likewise, in the case where the file MM-PAR (do not rename!) is not found on the disk drive it will not load properly. This all generally applies to the auxiliary software files as well. The MM inquiries are described in the following section. If the parameter file is not found, MM goes back to default values and continues to work.

For a reasonable program start, it is necessary that you have MM-PAR on the same disk as the main program. The load error message with the auxiliary software has no further effect. If you have a Horizon RAMdisk with the original Bud Mills operating system (Version 4.0) and encounter a problem loading MM, you can receive a new ROS from the author, which, among other improvements, allows you to boot MM correctly. To be clear, this service is for registered owners of MM.

Loader Module

MM can be started from anywhere when one knows the program filenames. This is the case with the E/A module, Option 5, the XB module with the appropriate loader (tested with Funnelweb 4.0), the TI-Writer module, Option 3, and the elegant and tested MENU program frm John Johnson for the Horizon RAMdisk. At least there were no problems in Version 7.3 (as long as there was not a previous disk access in VDP for the V9938!). Also, the CPS-99 Utility Loader from Michael Moeller is another elegant program that allows (like MENU from J.J.) the loading of MM without an assembly-type module inserted and read from the module port.

Do not load MM with the loader from DM1000 from Bruce Caron (Version 3.5 and other “homemade” versions) since they do not set the capital letter character set correctly.

Possible error message at the start of MM

There are only two error messages that can occur. A description of each and how to proceed from each follows:

WO IST MM-PAR? (Where is MM-PAR?)

The loader of MM could not find your starting drive number! Obviously, this error occurs when you have a Horizon RAMdisk AND an 80-column card in your working system. In such a case, enter the number (number) of the drive from which you loaded MM. Even if you specified this at the start of your auxiliary software, your input is needed so that the appropriate disk drive can be addressed.

Checksum Error

At the start of MM and the loading of the configuration file, a self-test is performed. This became necessary since the emergence of the Horizon RAMdisk and its rudimentary backup circuits that often drop bits and bytes much to the annoyance and consternation of the perfectionist TI users. When such an error occurs (for example, MM was loaded from a RAMdisk) then MM will wait for a keypress. Since the main program was obviously destroyed, working further with MM is dangerous, MM branches to the title screen. If such an error occurs please restart with a new copy of your original diskette. If this error occurs with your original diskette, please return it to the address on the previous page.

I say here and now that those “restless characters” that think that this sort of program security constitutes an ill-will restriction, that this is only a safety precaution! It is requested that this user-friendly function not be “patched”. If this instruction should not be heeded due to poor understanding and ambition, then a psychological investigation is recommended.

Functions of the Main Menu (Hauptmenü)

The main menu is a central program distributor with only a few key functions. Only from here is it possible to call the auxiliary software from its defined identification letter. The identification letter is freely defined, as long as it does not duplicate an already assigned. In this case, it is reachable only but the means of "Hot-Keys" (see below) from the main menu!

Short Cut Keys in the Main Menu

- D Calls the Printer module (DRUCKEN)
- E Calls the Editors (EDITIEREN)
- I Calls the diskette operations (I/O-DISK)
- K Calls the Configuration menu (KONFIGURIEREN)
- Q Ends the program (QUIT)
- T Calls the Transfer module (TRANSFER)
- FCTN-7 Help. Overview of the key commands (HELP)

CHECKSUM ERROR

In order to spare one from detouring through the main menu (however, the status of the auxiliary software must be explicitly known!) one can jump directly from one module to another.

This function is possible only from properly programmed auxiliary software. It is necessary that the key paths within the auxiliary software be seperately planned. The proper pointers must also be handed over.

The only exception is the VDP temporary editor buffer. This is a subordinated module that always leads to the temporary buffer in VDP. However, you can call other modules from the temporary buffer.

The respective module is called by pressing the CTRL key and the letter listed above!

The jump to auxiliary software from the MM module generally takes place with CTRL-Z. A test follows to see whether valid auxiliary software is present. In the event of an error (auxiliary software not present or other error!) then nothing happens after CTRL-Z is pressed.

Auxiliary Software and the Main Menu

You can only jump from the main menu to a defined letter of possibly-loaded auxiliary software. You can return only to the main menu with FCTN-9.

Each time the main menu is built or the shorthand key CTRL-Z is pressed, a search is conducted in low memory for signs of existing software. An implementation range is checked, among other things, for adherence to the convention for the first 20 bytes of the auxiliary software. If the test concludes successfully, then another selection appears on the screen and a further key selection is available. The section that explicitly concerns auxiliary software contains further references.

This procedure was selected because it was very possible during the test phase to load newly developed auxiliary software in the Disk-I/O part (for example with the virtual Tagged-Object Code loader) and then move it to low memory with the TRANSFER Option and merge it into MM, if necessary, for testing. [A-Z] tests the "sense" of the low memory data and with it the error prone auxiliary software (that uses low memory as a buffer) so that no "senseless" data or a program with a direct jump will simply be destroyed. From this, the user runs the risk of over-writing low memory (and possibly the contained auxiliary software) simply in the context of its use as a temporary buffer or by pressing "L" in the configuration module or newly reloading auxiliary software.

Overview of the command keys and VDP Editor commands

- All alphanumeric keys. Depending on the mode, are replaced under the cursor
- FCTN-= Cursor set to top left
- (HOME) CTRL-0 Page-Undo – Restores the old page.
- CTRL-D Calls the printer modules, Buffer updated
- CTRL-I Calls the Disk-I/O, Buffer updated.
- CTRL-K Calls the configuration, Buffer updated.
- CTRL-T Calls the Transfer module, Buffer updated.
- CTRL-Z Jumps to auxiliary software, Buffer updated.
- FCTN-ESDX Used to move the cursor, page overflow Buffer updated.
- ENTER Switches between ASCII- and HEX-mode. Buffers updated!
- FCTN-0 Leaves the Editor - the last change is not maintained in the Buffer (UNDO also see CTRL-0).
- FCTN-1 Character under the cursor (1 Byte) deleted (DELETE). Buffer updated!
- FCTN-2 Character under the cursor (1 Byte) inserted (INSERT). Buffer updated!
- FCTN-3 Buffer emptied (security question). Buffer updated also with the answer of No! (NEIN!)
- FCTN-4 Page down to the next page. Buffer updated!
- FCTN-5 Activate the manipulators. Buffer updated!
- FCTN-6 Page to the previous page. Buffer updated!
- FCTN-7 Activate the shift mode (MOVER). Buffer updated!
- FCTN-8 Jump to the temporary buffer. Buffer updated!
- FCTN-9 Leave the Editor. Buffer updated.

Commands of the Temporary Buffers

- A Cut the specification range in the buffer and the assumed name in ZP, fill with >00 or >20.
- E Insert the ZP from the cursor position.
- F Enter the length of fill or search (Hex.)
- K Copy: like A. source area not changed.
- S Search for byte sequence or string.
- T Switch byte A and byte B.
- W Repeat the example under the cursor in the Buffer
- X Cursor position as Start Mark designation
- Y Cursor position as End Mark designation.
- CTRL-A Activate ASCII-Editor.
- FCTN-ESDX are active.
- CTRL-C Directly select cursor position.
- CTRL-H Activate HEX-Editor.
- FCTN-ESDX are active.
- CTRL-S Manually enter the Start Mark.
- CTRL-E Directly enter the End Mark.
- CTRL-D I, K, T, Z as above
- FCTN-7 Help. List of the substantial commands.
- FCTN-9 Return to the Editor
- FCTN-E Line by line upward in the buffer
- FCTN-X Line by line downward in the buffer

The VDP Buffer Editor

This editor is at the heart of MM. It allows the manipulation of the contents of the VDP-Buffer, whereby many of the useful help functions work. One can reach the Buffer Editor with “E” from the main menu and with CTRL-E from the other MM modules, for example.

Overview

The Buffer Editor works on a separate area of the VDP RAM in the console. All of the operations are contained in the VDP RAM and do not alter the CPU addressable memory with the exception of the small buffer zones behind the main program. The Buffer Editor is a so-called “Full Screen Editor” which means that you can move the cursor to any location and change what is there. These changes are nearly immediately valid without having to press another key. For timing reasons, the buffer is not actualized when a valid key is pressed (practical, you are always working on a cut copy of the buffer) and the screen buffer is used as a temporary buffer.

Data Transfer, Actualization, and UNDO Function

Whenever the screen contents have been changed (page forward and page back), help functions or temporary buffer has been called) your input accepted through the press of FCTN-9 (BACK). Two functions have been implemented which will nullify the last entered changes. FCTN-0 breaks off the work in the editor, returns to the main menu and ignores all previous changes. CTRL-0 restores the screen contents that were in place on the original page and the same as FCTN-0 leaves the editor. You can see which keys are associated with which function in the Editor by looking at the list of command keys at the beginning of this section.

Other issues

The Buffer Editor is page oriented. That means that at any time you see a cut of the buffer on the screen that covers >180 (decimal 384) bytes. You can switch with the FCTN-4 (CLEAR) or FCTN-6 (PROCEED) to the following or previous page, for example, if you are not at the end or beginning of the buffer.

In this case, the keypress will be ignored and the cursor wil remain at the top left or ‘stuck’ to the lower right. Normally, the cursor stays at the position on the page!! A page (the actual work area of the Buffer Editor) consists of 24 lines of 16 bytes of pure data information. On the left hand side you will see the assigned address of these bytes in the VDP buffer. This address will be known from here as the “buffer address”. The display of the buffer address on the line in which the cursor is located is the actual (effective) address in the buffer. More information will follow.

Modes of operation of the Buffer Editors

The Buffer Editor has two different modes of operation: The ASCII mode and the HEX mode. You can switch between both of these modes with the ENTER key. The display of the buffer address and effectively the cursor address is the same in both modes and only the data is represented differently. The ASCII mode is particularly suited to entering text whereas the HEX mode permits the entry of hexidecimal data. In the ASCII mode you can only enter 7-bit characters and the eighth bit is set to zero.

Therefore in the ASCII mode, key combinations using the CTRL (the key with the red dot) can work like control codes!

In the HEX mode only the input '0123456789ABCDEF' can be used – all other input is ignored.

When storing the configuration files, the actual current mode of operation is also stored. So you begin (after a restart) in the your chosen mode of operation when you first began your work!

Cursor Movement

You can move the cursor (recognized by the blinking inverse video) with the well-known function keys of E, S, D, and X to up, left, right, and down as displayed on the buffer page. The press of FCTN-6 takes you to the upper left corner of the buffer page.

Cursor movement beyond the Page borders

If you bring the cursor, for example, beyond the page borders then the following and/or preceding page will be automatically displayed in the buffer. If, as an example, you leave a page for one above with FCTN-S Left arrow) you will be switched to that page and the cursor will be placed at the lower right corner of the page. You can step through the buffer byte for byte backwards by repeatedly pressing FCTN-S.

The same is true for FCTN-D in the opposite direction. The cursor keys FCTN-E and FCTN-X will let you page forward or back (to higher and/or lower addresses in the buffer) whereby the cursor is brought to the HOME position in the upper left corner,

Basic Functions

Besides the commands described for page choice and cursor movement, there are some other basic commands that are implemented immediately without further action. These are delineated in the Help Functions.

BACK – Exit Editor

As soon as you press FCTN-9 (BACK) you will leave the editor and find yourself at the main menu of MM. If you have inadvertently pressed BACK, you can return to the editor with “E”. You will be at the same buffer page and in the same mode as you were before you pressed BACK. Also the changes that you made to the page will not be lost!

INSERT – Create space under the cursor

With the press of FCTN-2, all of the buffer contents from the cursor position are shifted down and the byte under the cursor becomes free. Practically, the byte is copied and not filled with >00. In both modes exactly one byte is shifted and not, as one would surmise, one byte in ASCII mode and one nibble in HEX mode. Since it works with the entire buffer in VDP RAM the cusor will disappear for a while with this function which will be about 1.5 seconds on a normal TI and 1 second on a 16-bit 32K and a 3.5 Mhz processor clock. The closer to the end of the buffer the cursor is, the shorter the time will be. The last character, placed in the buffer, appears at the old location and FCTN-1 (DELETE) removes the old character. That means that you can undo FCTN-2 by coincidentally pressing FCTN-1.

DELETE – Remove the character under the cursor

As suggested under INSERT pressing FCTN-1 deletes the character under the cursor and replaces it with the following one. Since the entire buffer is shifted, you have the same delays as described under INSERT. With DELETE you must consider. On the one hand, that the byte in irretrievably lost and, on the other hand, it is pushed to the end of the buffer where it was pushed with INSERT. If you want this pushed out byte, write it to the lower left of the last buffer page and press FCTN-2 (INSERT). Thus this byte can be found in the overflow buffer where it has been placed by DELETE.

Help Functions

The Help Functions differ from the other functions in that a dialog window appears asking if you really want to proceed (Pressing “J” will continue and any other key aborts). That is to guarantee that you will not inadvertently destroy your buffer contents. In any case, you can escape by pressing FCTN-9 (BACK). You will be returned from where you started.

Empty Buffer

After pressing FCTN-3 a dialog window is display and a security question is asked. If you press “J” (any other key aborts) the buffer is filled with >00 in HEX mode. Likewise, IN ASCII mode it is filled with >20 (space character). The bytes in the overflow buffer are set to >20 as well.

Shift Buffer Contents

With FCTN-7 you purge the dialog window. Here you can enter the start address of a block in the buffer area to be shifted. Then you determine where to push this block in the buffer. Subsequently, you specify the length of the block to be shifted.

All of the entries are in hexidecimal notation. The addresses are related to the buffer addresses and correspond to the values on the left hand side. A new call of MOVER uses the last displayed values as default. These parameter values are stored and loaded along with other configuration values at every new start of MM. The input can be aborted at any time by pressing FCTN-9 – after the entry of the length and pressing ENTER the shift starts. With shifting only that data is changed starting at the destination address. If the addresses do not overlap (that is also permissible!) the data at the start address is NOT altered!

Manipulate Buffer Contents

BY pressing FCTN-5 you activate the so-called MANIPULATOR. With this function a specified range in the buffer can be worked on with various computer procedures. After the dialog window is opened, you can successively enter the following parameters: the starting address of the range to be worked on and the ending address. The input is in hexidecimal and corresponds to the related buffer addresses on the left. You can specify, for example, if something is to be added to the contents and/or something subtracted from them. Recently, a Bit-Reversal Routine was added.

Through word operations it is possible to change only the high or low bytes or both of them differently. This has been made possible because the high byte might be at an odd address. The high byte is always the first specified in the ranges followed by a second. If you input >0000, the option is not executed.

If you have specified a range that exceeds the buffer borders, a warning window (Alert-Box) will appear. By pressing any key you can re-enter the parameters. If the range limits were correct, and you specified an addition or subtraction, then you can indicate whether the word that has been read is to be rotated. Valid inputs are from >0000 to >000F and all others are standardized on these values. An input of >0008 is the same as a SWPB. The description here is valid for both high and low bytes. The last parameter specifies whether the action is exclusive OR linked. It is here that the byte determined to remain untouched with the corresponding 8 bit >00 or the XOR mask. The manipulator is an efficient and variable instrument in function. If some of the points are unclear to you simply test some examples in the buffer!

As you will observe, the arithmetic operations are processed in order as they follow one another in the dialog window. If you wish to work on another section or in a different order, and wish to deactivate the function, do so by inputting >0000.

You can abort the input at any time with FCTN-9. Upon re-entering the manipulator, the parameters that you used last will be displayed as default. These defaults are stored in the configuration file for your next use as in the MOVER. This is the natural condition when you leave the program before starting anew!

The Temporary Buffer

The VDP buffer has a smaller brother: the TEMPORARY BUFFER. It contains 256 bytes (>100) and can be used for many things. By pressing FCTN-8 you bring in the contents and the usable options. After pressing FCTN-8 the following happens: a hexidecimal display of the temporary buffer contents will appear in the upper area of the screen. Under that, separated by a horizontal line will be the ASCII interpretation of the temporary buffer. The last two lines comprise a dialog window with the following functions (in the well-known single key short choices):

A – Cut the marked range

Like “K” (see there) only the range of the cut out portion of the buffer is filled with >00 or >20 (space character) depending on the mode of the buffer editor.

E – Insert into temporary buffer starting at cursor position

Starting at the current cursor position in the temporary buffer the length is determined by the calculation of the end address minus start plus 1 and inserted there but only up the the buffer maximum. The contents of the temporary buffer remain unchanged and can be inserted in another location if necessary. The contents of the buffer are NOT shifted backwards!

F – Search the current ring for the full specified string

This value fulfills two functions. On the one hand it determines how often the character under the cursor (in the buffer editor!) repeats from the cursor address – the maximum input is the buffer size (>2B80) occurrences outside that are ignored.

On the other hand, you can specify the length of the search before beginning from the start of the temporary buffer. You can find more under “S” and “W”.

K – Copy the marked range into the temporary buffer

Pressing this key will copy the contents in a given range into the temprorary buffer. One can see these contents in the upper screen area.

S – Search for byte sample

Permits a search for a given string or byte sample. The search begins at the start of the temporary buffer. Before starting one must use “F” to give the length, for example, the byte count (string length) of the string to compare. You can give the sample in either ASCII or HEX since it plays no part in the search. If you use >0000 or any value over >0100 for the string length then the search function is not initiated. Otherwise, the search begins at the current cursor position in the buffer. If the string sequence is found in the VDP buffer, the cursor position will be set behind the string or byte sequence. Thus, you can search again if necessary. Generally speaking, if a match is not found in memory, an appropriate reference box is displayed that you can exit by pressing any key. In this case, you will remain in the temporary buffer menu.

T - Exchange

In order to initiate this function, you must use a two-part input in either ASCII or HEX window. At the beginning of the appropriate buffer the searched string (search byte sequence) and the length of the fill or searched string must be specified. The sequence is replaced by one of the same length as soon as the primary sample is found. The fill/search length in this case can be a maximum of 128 bytes. Starting from the current cursor position each of the discovered samples is exchanged. The conditional search will be realized in an update.

W – Repeat the byte under the cursor

This function has nothing to do in principle with the temporary buffer. However, thematically it fits. Pressing "W" the character on which the cursor stands is repeated the number of times as previously given in 'fill/search" The maximum is to the end of the buffer. Subsequently, return is to the Buffer Editor.

X – Set the beginning mark

Here you take the actual cursor position as the start address in the range, which is what you have to do somehow in the temporary buffer. The actual address is displayed in the VDP buffer window and, after pressing "S" is displayed and copied into "Start Mark (Startmarke):" The value of end mark is set at >FF or decimal 255 higher than Start Mark with a maximum at the end of the buffer!

Y – Set the end mark

Like 'X', only the end of the block can be set. You must have previously set a "Start Mark" that is not more than 256 bytes lower than the end address. Otherwise, an error tone is generated and your input is ignored. With a correct choice, the actual cursor position is copies and the End mark (Endmarke) is displayed.

CTRL-A – Edit the temporary buffer ASCII window

This is a small "goodie" with many possibilities. After pressing CTRL-A the cursor appears in the ASCII field. You can now freely work on this field. You can move the cursor anywhere and change any of the 256 bytes. The FCTN-E, S, D, X, ENTER, and FCTN-9 keys are supported. ENTER ends input and makes the changes. FCTN-9 aborts input and reinstates the old temporary buffer. After pressing ENTER the ASCII block within the HEX range is converted and represented.

CTRL-H – Edit the temporary buffer HEX window

An analog to the ASCII Editor except that you can work in half bytes. The same key functions are supported here as in the ASCII mode.

Error messages of the VDP Editor

Naturally, the error messages in these editors are quite sparse. Either you have left the buffer area addresses or you looked for something but did not find it.

Keep Buffer Borders (Puffergrenzen einhalten)

This applies to MOVER or MANIPULATOR. It indicates that you have chosen start or end addresses that would cause you to leave the boundries of the editor buffer range. In this instance, the previous function is not implemented and you must re-enter the parameters.

Not Found (Nicht gefunden)

This is only in the temporary buffer when a string or byte sample is searched or exchanged from the cursor position and it is not found.

The obligatory question is asked before losing the buffer contents, Really delete? (Wirklich löschen?) To which you can answer No (Nein).

Some functions in MM are just plainly and simply ignored without an error message. If you are at the beginning or end of the buffer and want to go up or down to the next page, for example. It also appears if you want to insert something at the end of the buffer.

Also, in the case of an amount of filling that would spill over the end of the buffer, the value is reduced to one that would fit in the maximum buffer size.

An error message in the editor can be seen to a certain extent backwards through the program modules that bring data into the VDP buffer.

Remove changes?

This message appears if at any time changes have been made to the VDP buffer and have not been saved in the meantime. Since the editing range is only available once, this message is triggered and also when you wish to remove changes that you have made.

Overview of the command keys in the I/O-Disk Module:

Standard Functions

- A Execute desired function
- D Diskette catalog
- E End offset for Editor buffer
- F Input Filename
- H Input Header bytes
- I I/O-Specify action (Load/Save)
- L Determine logical end of buffer
- M Mode of the I/O choice (P-File, Datei, File etc.)
- R Skip RAM-Disk at >1000 (J/N)
- S Start offset in Editor buffer
- ^ Set Offset of the upper end of the buffer
- + Last character in the filename plus 1
- FCTN-A Automatic buffer delete on/off (an/aus)
- FCTN-E End offset with File- Sektor-I/O
- FCTN-L Length byte option with file input
- FCTN-R Relocation base of the Tagged Object Code Loaders
- FCTN-S Start offset with File- Sektor-I/O
- FCTN-7 Help function Key overview
- FCTN-9 Back to the Main Menu
- CTRL-D Jump to Printer module
- CTRL-E Jump to Editor Module
- CTRL-K Jump to Configuration Module
- CTRL-T Jump to Transfer Module
- CTRL-Z Jump to auxiliary software (if available!)

Functions in the Disk Catalog

- ENTER Run selected filename
- FCTN-E,S,D,X Cursor movement keys
- FCTN-6 Next catalog page
- FCTN-8 Catalog again on a different drive
- FCTN-9 Abort catalog

Functions in the Status Generator

- A Execute – produce file
- H Hexadezimal parameter input
- P Filetype - Program
- I Filetype - INT/FIX
- i Filetype - INT/VAR
- D Filetype - DIS/FIX
- d Filetype - DIS/VAR S Alter protection status (input)
- FCTN-7 Help fuction key overview
- FCTN-9 Abort return to Disk-I/O-Menu

Disk-I/O

That is one of the terms that has been kept from its English contraction 'I/O'. That is because nearly everyone knows it means 'Eingabe/Ausgabe'. At times, honestly – 'Disk-E/A' looks painfully comical, does it not?

Let's get to it. This submenu is the one that causes the most confusion. It is not intended to do so but it is probably unavoidable. Straight disk access from programs of the MM type usually comes with carelessness that can be incomprehensible. So either no file can be loaded (debugger solutions) or it can be loaded only as a D/F 80 (among other things debugger solutions) or it can be saved not at all or insufficiently. And then, the files are usually always in a format that the resultant software cannot read straight away.

It is not like that with the Disk I/O in MM!

You can load each file type, in principle, in the buffer and save it as a P-file. You can work on large files (it is simply more logical to, for example, to work on the contents of 27256 EPROMs as a single 129-sector file) and smaller files can be inserted in the buffer at will and only parts of the buffer saved without the files becoming jumbled.

The I/O Parameters

Below the window identification you will find a window, it which all of the parameters can be found that, during reading and writing of a diskette, become important. For an understanding of each one, they are described below.

Start-Offset

With this you determine which address in VDP buffer the file read from disk is to be placed and from which the data from the specified file is to be written. During reading the data in front of it in buffer is not affected (during writing, not so).

End-Offset

Specifies up to which byte (inclusive) the buffer contents are to be saved and where the loaded file must stop in order not to endanger data lying behind it. In saving files, you can specify this input of file length as the End-Offset minus the Start-Offset plus the header bytes plus 1. Please note that with certain RAMdisks the end monitoring does not function correctly!

Header Bytes

All program files (or P-files for short) consist of pure data of program area with usually a 6-byte header of additional information placed at the front. Exceptions are files that are loaded with a special loader like GRAM modules and the like. Because of this exception to the rule, we have provided exactly this option. Thus you specify a boundary in front of which this additional information is found and after which is the pure data. The boundary is at the beginning of the VDP buffer. Before the boundary you can have a maximum of 9 bytes of header information that cannot be changed in the VDP buffer. When reading it is loaded a little before and when saving also a little before (the beginning of the buffer).

What does this mean, you ask. Naturally, if you load with Headerbytes = 0, then you will see these bytes in the buffer. But if you know what the headerbyes are and want them to remain unchanged, you can hold them out of the buffer and protect them from inadvertent change. You can then work correctly with the logical addresses in the buffer without having to constantly refer to the number of header bytes and where you really are in the buffer (in relation to the beginning).

Start sector

This parameter is only of interest with the modes of "Sector" and "File". It specifies which logical sector (I/O mode SEKTOR) in the specified file is to be read first. This sector is loaded, beginning at the Start-Offset, in the VDP buffer.

End sector

This specifies up to which logical sector of the diskette (I/O-Mode SEKTOR) in the specified file will be read. This sector is inclusive and always completely (256 bytes) read. More under "Operating Modes".

Load- or Save Mode

Specifies whether LOAD or SAVE is to be performed.

A- Is for better recognition of the word “Save” which is displayed in inverse video. By pressing “A” the function is executed and the SAVE mode is activated with a safety question appears and one can proceed by pressing “J” or abort by pressing any other key.

Operating Modes

Which type of data that you wish to load or save is displayed next to the word “Modus”.

P-File Mode

'P-File' is is simplest and most space-saving filetype as long as the file length remains under 44 sectors. Note that loading P-files from a RAMdisk often works incorrectly in that the specified maximum length (from Start-Offset to End-Offset) that would indicate that the protection information behind the data is not taken into account and the entire file is always loaded. If the file on the disk is longer than the buffer area, it is not loaded.

File-Mode

P-Files that are longer than 44 sectors must be loaded in the less specialized 'File-Modus' and cannot be loaded all at once (but in pieces) in the buffer. You must specify a 'Start-' and 'End sector' and which section of the file you wish to read/write back.

As soon as the loading overlaps the end sector and the End-Offset in such a manner that the buffer range is exceeded, then the loading process is interrupted. With file I/O the filetype does not play a role but it can concern overly long program and data files. The file is read and no consideration is taken of its internal structure! A sector editor is available that, contrary to other programs, that cannot exceed the individual sector borders, allows for the logical editing (and non-overlapping) pieces without borders that permits constructive work without the 256-byte restrictions that hinder such work.

Data Mode, virtual Tagged-Object-Code-Loader

Deficiencies in file structure are eliminated in the Datei-Modus” with input when data-oriented files are read. In order to be able to write such files after reading them into buffer further boundary conditions must be kept (byte length option) which are described further down. If you specified a file input in the “LOAD-Modus” it is tested to see if it is in DIS/FIX 80 format. In this case, you will be asked if it is so-called Tagged-Object Code. Files of this kind were produced by the Assembler in the E/A module or other compatible programs (like the Macroassembler or GPL Assembler) and contain machine or GPL code. Answer this question with “J” and the file will be loaded with the virtual Tagged-Object-Code Loaders that evaluate each data record in detail. Virtually; because the machine code (or GPL) is not directly loaded into the computed place in CPU/GRAM but correctly relocated into VRAM.

Otherwise, the data records are loaded one after another in the buffer where the running string length and buffer pointers indicate how long each data record is and if the buffer is filled. A Start-Offset different than zero is considered and loading is aborted when the End-Offset is reached. After the file is loaded, a message appears that everything is correct or that the buffer borders forced an abort. The load data is displayed (last string length and buffer pointer) and disappears with the press of any key.

The Tagged-Object-Code-Loader shows the entire work window that truly expresses the nature of the load. The loader works on both compressed and uncompressed code, both TMS9900 machine code and GPL code. The type of code (compressed or uncompressed) but is really without meaning. Likewise, it indicates the code length. Depending on the assembly, this is immediately displayed at the start of the load and counted up (AORG or not). The running buffer pointer is also added.

At the end of the loading procedure (end of file or buffer full) any code references are indicated (the MM loader is not a linker – such a file is not so easily usable) which was the last definition and if any checksum error arose. The TOC loader knows all of the tags of the E/A loaders and does not ignore the unassigned tags without an error message like a checksum error. Tags over “I” break off the loading procedure with an error message since no normal TO code is present. The loader loads the code directly into VDP RAM (in the VDP buffer) where the AORGs are resolved both forward and backwards if they remain within the indicated buffer borders. Relocatable code causes a few annoyances.

In connection with the file loader the terms of importance are the relocation basis and the file length options.

The Relocation Basis (FCTN-R) indicates which address is to be used with relocatable machine code. If absolute code (AORG) is loaded then this input has no effect and the address references are resolved as if they at the buffer start or correspond to the CPU storage address.

The length byte (FCTN-L) is important in the case of a standard file, not only when you read a file but if you want to save it later. Since each data record can, in principle, have a different length MM must store this byte with it when loading. These bytes are stored in the buffer. The file can be manipulated purposefully by inserting or deleting or altering the length byte. For the sake of simplicity, file with a fixed length can also be edited in this manner.

Sector-Mode

A further function is the mode of operation – “Sektor”. Here not the relative sectors of a file but the physical sectors of a diskette are read and written. To do this, it is necesary that the filename also includes a valid disk drive number. The starting and final sector indicates from which sector and to which sector reading and writing will be done. During writing the entire 256-byte block is always read into the VDP buffer and during reading the bytes between the Start- and End-Offset are read. If the sector count indicates that (End minus Start plus 1) would bring more data into the VDP buffer than the End-Offset permits from the temporary VDP sector buffer, then the bytes are brought into the VDP buffer until the End-Offset is reached. A warning then follows that the read is incomplete and you can quit by pressing any key.

That is the principle of the sector editor but not all of it. On the one hand you can move the sector contents into the buffer built in a quasi measured dissection and, on the other, you can get an equal lot of sectors from the diskette to the VDP buffer comfortably and simultaneously work on them. The sector editor reads the sectors in a numerically ascending order – it is not suitable as a file editor!

Status-Mode, Load

The last mode of operation is the ”Status-Modus”. In the load mode, the status of the file is displayed with the name, disk drive number. The filetype, protection code, length of the data without the FDS, record length, record count per sector, and the EOF of the last data sector and, this is new, the date of the last update or the creation of the file.

The bytes in the File-Descriptor-Sector (FDS) were reserved at the outset by TI and contain date and time code. So far, these bytes have not been used. But since the Geneve that was the first system to use a hardware clock, these bytes have now found a use. After an analysis of the format used by the Geneve, we have decided to take over and support this standard. This also applies to the TI, which in principle applies to it as well, with only some boundary restrictions.

In order for the pertinent operating systems not to allow some of the data to spread beyond the VDP RAM and not to allow the program to exceed the allowed memory borders, the search for of the FDS of the files and memory for the date codes (more in the configuration section) the built-in disk controller rouitines have been taken over. They are also in RAMdisks but are only known by the developers.

If the disk I/O part activates the date code storage option in the configuration section, then your RAMdisk operating system will work with errors. If you have a Horizon RAMdisk you can be helped with a more conforming ROS. Since the operating system is variable, there seemed to be little meaning to adapt MM to the actual version 7.3. In this case, please deactivate this option and please use your disk drives. This is also the case with the Myarc Disk Controller since this company does not administer the VDP RAM correctly.

Status-Mode, Save

If you call the Status Mode in the Save-Mode, you can create a new file of arbitrary type. First, on the basis of the indicated filename a check is made to see if this file already exists. If the meaningful data appears as default, all of the file parameters appear as zero. They are now in the input mode of the Status Generator – abort as usual with FCTN-9.

File parameters are adapted. The following are the possible key codes (also displayed with FCTN-7). The premissable key functions are indicated and displayed in the Disk Catalog:

A - Execute

The file is produced on the basis of the adjusted parameters. There is NO safety question!

D,d – Display File

Sets the type of file as DIS/FIX (D) or DIS/VAR (d).

I,i – Internal File

Sets the type of file as INT/FIX (I) or INT/VAR (i).

P - Program-File

Produces a Program file.

S – Protection status

Permits the input of 'J' or 'N', if the produced file is to be protected or unprotected.

H - Insert Hex-Parameters

By pressing H you can in sequence specify all numeric file parameters.

U – Set Time and Date

Similar to the function in the configuration module.

More Direct Functions in the I/O-Module

Insert Filename

This input serves two purposes. On the one hand, the first character after “DSK” must be a number from 1 to 9 that represents the disk drive on which the modes “File”, “Sector” and “Status work. If none of these options are to be used then DSKR and DSKG, for example, for the CorComp Memory-Plus or GRAM-Disk will (Level-2 Access).

The period that comes after the disk drive number is compellingly described. When editing the name INSERT, DELETE, ERASE, LEFT- and RIGHT ARROW as well as BACK and ENTER are available. The first blank character marks the end of the string. You can also enter the filename directly with “F” or take it from the disk catalog. How that is done is shown in the following clarification.

Disk Catalog

After selecting this function with 'D' a small dialog window appears askling for a disk drive number. The allowed values are 1 through 9. The default disk drive number is the one that was shown on the last catalog or where the configuration file resides. ENTER accepts this value.

When the disk drive is found the catalog appears. The basis of the catalog admittedly (which we will gladly use) is the ‘SD’ catalog from Clint Pulley. You will quickly notice that it has been improved in the meantime.

The catalog is error tolerant. That means that if sector 0 and sector 1 are found or no >0000 is found in sector 1 then the display of the filenames will not be disturbed. The catalog is terminated when the physical end of sector 1 is reached or you press (and hold) the space bar until it is aborted.

Of course, the identification of each file that appears in this “table of contents” is found can be read. A change after writing should not be a problem. Files that MM cannot read are otherwise unreadable and as a predominant file are useless. With large requirements an appropriate disk editor becomes necessary – and MM is not a suitable program for this!

A short overview of the valid functions is at the beginning of this section.

Automatic Buffer Clear

It is often an advantage if a file can be loaded again (if it is the same type) into a perfectly empty buffer. To make it easier for you, instead of pressing FCTN-3 and confirming each time, by pressing FCTN-A this option can be switched off an on. Before the load when the option is active the entire buffer is filled with >00 whether the ASCII- or HEX- mode editor is active.

Building an Error List

This error message list consists of two equally developed blocks: First the error message in plain language (MM does not know code number messages) then (in bold print) the I/O mode in which this message may occur, and last a description of the cause of the error.

List of the Error Messages (in alphabetical order)

Changes that were removed since you last edited the buffer contents that were not overwritten and not stored. The buffer limits are not tested and there is not a global statement that the buffer contents were altered.

Diskzugriff gefährdet P-File Siehe 'Puffer reinitialisiert' (Disk access endangered P-File see Buffer reinitialized)

The file exists but Status has produced another file. Due to the TI- File System, the file cannot be edited by Status. Because a file of the same name has been produced that no longer contains the old data!

Puffergrenzen einhalten (Keep Buffer Borders)

Everything in the execution of this function is an invalid configuration from the Start- Endoffset or Start-, Endsector that were determined as outside of the buffer area. After this message MM jumps to an input cell where the author of the “supposed” error is allowed to make the correction.

Puffer reinitialisiert P-File (Buffer reinitialized P-File)

After loading P-Files from a RAMdisk an inadmissible destruction of the VDP block of the disk controller occurred. In order to avoid this, at the next disk access the system “hangs” and the buffer area is restored. The message 'Diskzugriff gefährdet', is always displayed to underline the meaning of this message. It is best to restart MM at the next opportunity.

Schreibgeschützt (Write Protected)

Always occurs during the writing of a file or, in general, writing to a file/ disk that is write-protected. Remove the mechanical write protection or likewise remove the file write-protection with a suitable program. Later versions of MM allow you to make this modification immediately.

Teilweiser Lesevorgang Sektor(Partial Sector Read)

The buffer end was at an address that was not with the integral multiple of 256 or was beyond the buffer start address (Offset). The last sector could not be completely shifted.

Unverträglichkeiten des Disk-I/O von MM (Incompatibility of Disk I/O with MM

In general, MM uses the Disk-DSR made available, excluding the routines. Since Texas Instruments never explicitly listed the memory area assignment changes in the VDP RAMs, administrative support remains for the original Disk Controller in the P-Box. This means that the MM I/O works with TI Controllers and or software compatible devices without any problems. Additions are the BwG-Disk Controller (SNUG), the CorComp Disk Controller, the CPS-99 and the Atronic PEB Disk Controller. These systems have been previously tested with MM.

The VDP RAM is NOT correctly administered by either the Myarc Disk Controller (DDCC-1 old and new) or the Horizon RAMdisk (all past ROS Versions!). The Myarc Controller defies discussion and there are similar problems with Horizon systems.

The Subroutine >14 and the DSR function LOAD of the Horizon ROS 7.3 are as much incorrect as the SBR >14 extended status in the VDP area prepared for the TI Controller and its obedient brothers in that LOAD Program File is placed in PAB up to the End Offset and all else is ignored. Therefore, the MM Status provides NO date codes on RAMdisk and they can also not be stored!

Also the destruction of the disk control blocks in VDP RAM (with a following crash) can occur when loading P-Files from a RAMdisk if the file end is overwritten despite the setting of the maximum buffer offset.

These effects are attributed simply to incompatibile DSRs. It cannot be the task of a user program to take on the lack or awareness or ignorance of “moonlight programmers” in retrospect!

In order not to drive this “bird bunch politics” to the extreme, a monitoring function has been integrated in MM. After each P-File read access (before securing), the VDP Init Block of the last device that used VRAM is checked. In the event of an error, MM implements a reduced Power Up Link (a full link is not possible for various other reasons!) and avoids a system crash with the next disk access. Such 'Holzhammer' procedures are not with out error and that is why the two “endangerment” messages continue working.

Overview of the Command Keys in the Printer Module

Actions

- ♦ A Execute printing
- ♦ C Input Control codes
- ♦ D Input Printer name
- ♦ E End page of the printer output
- ♦ F Copy File namen in Head line
- ♦ K Edit Head line
- ♦ L AscertainLogical Buffer End
- ♦ M Switch mode of operation
- ♦ P Set ouput Parameters
- ♦ S Start page of output

Function Keys

- ♦ FCTN-9 Jump to the Main menu
- ♦ FCTN-7 Display necessary key codes

Shortcut Keys

- ♦ CTRL-E Jump to the Editor Module
- ♦ CTRL-I Jump to the I/O-Disk Module
- ♦ CTRL-K Jump to the Configuration Module
- ♦ CTRL-T Jump to the Transfer Module
- ♦ CTRL-Z Jump to Auxiliary Software (if available!)

The Printer Controls

This program module can be called with 'D' from the Main Menu or CTRL-D from another module and allows one to print the contents of the VDP buffer (or a part thereof) in a 'Display-Variable-80' format or save it to disk. Thus you have a simple procedure on hand to put the buffer contents in a form that can be worked on by one of the many text-processing systems.

The Screen Layout

The uppermost window contains the well-known identification of the module concerned. Under it you will find two windows. The one on the left gives options to assist in adapting the buffer contents to your requirements.

Printer Mode

You have the choice here between ASCII- and Hex mode – switchable with the 'M' key.

In the HEX mode, which you will probably use more frequently, is expressed in three columns. The left column shows the pseudo (symbolic) address that you can specify (see below). The middle column (better the ‘Middle Block’) represents the data in its hexidecimal form (1 character per nibble) and the right column shows the data in its ASCII form. Sixty-four (64) lines are printed per page with 16 bytes per line or 1 Kilobyte of data per page. In the ASCII mode, likewise the address column appears and right beside it is a whole block of ASCII symbols. Here, effectively, 64 characters per line and 4 Kilobytes of date per page (64 lines) are printed.

In both modes after the completion of a page a Form Feed Code is sent so that you can begin a new print page on a new sheet. One wonders if there are printers that do not understand this. It is hardly conceivable that there are such dumb devices in free trade and there is no option to help these existing printers!

Pseudo-Address

Normally the expression of the addresses takes place in the left column that corresponds with the placement of the data in the VDP buffer (as you see in the VDP buffer on the left). If you know the original address of where your data was located, you can let these expressed addresses run along with it. By pressing “P” you can see the virtual Start Address at the time of printing and stop at any value. Consider that this address concerns the beginning address of the buffer when printing other than the first page (print page) and the addresses are accordingly advanced (with Hex 1K and with ASCII 4K per page more). It is best if you test several times!

Standardization Mask

Immediately after specifying the Pseudo address, you can input the so-called Standardization Mask. This mask becomes necessary when you express ASCII characters that your printer does not interpret. Thus codes below the space character and above >7E are not interpreted.

What you specify in the mask is the argument for the command 'SZC' to be used, whereby only the High-Byte is of interest or effective. More simply stated that the bits that are normally set in the Standardization Mask are not set or will not be lost in ASCII ouput.

ASCII-Codes under the space character (>20) will generally sent as a space so the printer will not get entangled in control codes.

8-Bit-Correction

After in input of the Standardization Mask you will be asked about the possible subtraction of 8-bit codes. Only the high bit here is significant. If you have already planned a standardization, which already produced 7-bit codes, and an 8-bit ASCII code appears, then this number is subtracted from the ASCII code. So you know that the patterns that you have seen in the 'Screen-Offset' are again converted into the correct symbols (input >6000). Values under >7F are not changed. You can abort all three of these inputs with BACK (FCTN-9). Make sure that your 8-Bit Correction does not produce new 8-bit numbers or control codes (words like >8000). Since the standardization has taken place previously further 8-bit codes are not prevented. They will be treated as control codes and replaced with a space character.

Page Data

In the window on the upper right you will find the amount of paper that can be printed. You will know what page (inclusively!) from where to where the printing will be done. The input can take place arbitrarily (not that one has to stand for it or like it) but must lie between 1 and 11! Unreasonable values are standardized so that you can see after pressing ENTER what the program understood them to be. Your further input is not tested (Start page larger than End page), but the printer does not begin until the correct page input has been made. In such a case an error message is given.

Next to the Start- and the End page is 'Laufd.'. that indicates during printing which page is being worked on. With slow printers or large page counts it can be otherwise useful when one must take a possible coffee break.... When one does not print there is a “0” – logically for both the Start page (Startseite) and end page (Endseite).

Logical Buffer End

With the press of “L” a search begins in the VDP buffer for the first null byte different than zero – it it is from the back! The address of the first null byte and what follows in the buffer is displayed. Besides this page, you can still print the entire buffer contents. Since this page depends on the printing mode, it is deleted when you switch modes. The search (the VDPand backwards) can last for a while. Meanwhile, the word “AKTIV” appears in the upper window on the right in inverse video. That means: ‘Please Wait! ‘

Text Options

In the lower window you will see these options: the filename display of which file was last loaded and/or saved. This explanation gives you the high probability of what data is in the buffer. This name can be used in the Head Line of each printed page.

Filename

The expression is sometimes quite informative, if one sees the origin of the data. You have a headline of 25 characters to use. If you do not want to enter them directly, by pressing “F” then the actual filename with the prefix ‘Filename :’ appears.

Control Codes

Before the first page is printed, 11 bytes of control codes are sent to the printer. Which ones they are is specified after “C” is pressed. The input, for the sake of simplicity, takes the form of an unstructured hexidecimal number field. The advantage is that you can send any code from >00 to >FF but those that you do not wish to use please set to >00! If your printer later prints strange things here, check here first to see if everything is correct. The codes are sent only once at the beginning of printing without a line feed!

Header Line

The header line of each page is, to a large degree, fixed. On the left you will find (constant) the Copyright Information, then your headline follows (of various ASCII codes) and, on the right is page ('Seite') and the page number. By pressing “K” you can jump into a small editor where you can develop the headline. Except for the page number all of the headlines are identical.

Printer Name

What is at the bottom in the corner is the printer name. It can be a maximum of 37 characters long and will, along with the control codes, be saved in the parameter file by pressing “D”. It must be at least 3 characters long! If the name should not match a printer in the system, then an error message at the start of the output (Eröffnungsfehler) will be issued. Naturally the printer name can also be a diskette. More information is in the following section.

Output to Printer

Pressing “A” irrevocably starts the output. The file to be printed by the device will be opened as a Dis/Var 80 file in the APPEND Mode. Nothing will be output if you designate the name of an existing DV80 file. It will always create a hangup. You must see that it does not become so long that the Text Editor cannot clearly use it! While printing the 'AKTIV' message will appear. You do know that at any time (if your interface allows it!) that you can abort with FCTN-4.

The error message will disappear again. Avoid using the .NT specification with the Atronic I/O card (or CPS99)! If an error should arise during printing an error message (Warning Box) will appear. After acknowledgement, the file is closed and the message disappears.

Error Messages during Printer Output

The printer control recognizes 3 visible and 2 invisible error messages. The visible have already been described several times and appear in the warning boxes and read 'Eröffnungs-Fehler!', 'Ausgabefehler!' and 'Falsche Seitenangabe!'. The invisible are present if the cursor does not disappear after pressing ENTER.

''Eröffnungs-Fehler!' (Open Error!)

Occurs when the printer name is wrong. When you try to write to a diskette and the file or the whole disk is write-protected or there is no more space on the disk. Also, when a file of the same name already exists or it is not a DV80 type file. You can change the name and try again.

'Ausgabefehler!'(Output Error)

A cause can be a time out of the printer or a defect on the disk. It will happen also if the disk becomes full. The file is closed and the data is secured.

'Falsche Seitenangabe!' (Wrong Page Indication)

In the Hex mode all 11 pages are necessary and the entire buffer is “printed empty”. In the ASCII mode only 3 pages are needed. The ASCII mode output will only start if no page number is over 3. Otherwise, an error is displayed when the starting page number is higher that the ending page number. In this case input the correct page numbers!

Errors and their Causes/Recovery

The principle is that you must be certain first that your system is functioning correctly on all points. Then we can work in a well-founded manner.

When you see differently mutilated data on different lines, you have made the wrong choices for Standardization and 8-bit correction.

Give it the Standardization mask of >8000 and 8-Bit-Korrection of >0000 and try again.

Another cause can be the control codes. Test those or give the entire row '00's. If you cannot repair the error, then please see the detailed instructions that I addressed in section 2.

Should the headline “wander” i.e., the printer starts a new page either up or down then the printer does not know how long your sheets of paper are. This can be adjusted by DIP-switches or the control sequence (lines per page/ sheet length in inches). See your printer manual for installation.

Overview of the Command Keys in the Transfer Module

Standard Functions

- A Execute Transfers
- B Input Banking Address
- C Specify CRU Base for Bit pattern
- E External Bank of the V9938 (Target)
- G Install GROM Parameters
- M Mask (Bit pattern) for CRU control
- O Offset in the Editor buffer
- P Buffer bank V9938 (Source, Editor buffer)
- R Direction of the Transfer (in, out, compare, buffer)
- S Start Address of external Memory (with End Address)
- T Type of Memory (Addressing mode)
- V VDP Parameter
- +,- Switch extended Speech Address

Comparison Functions

- A Ignore all unequal but summerize
- E End of comparison
- W Go further

Function Keys

- FCTN-7 Help, Display the appropriate key codes
- FCTN-9 BACK, return to the Main Menu

Shortcut Keys

- CTRL-D Jump to the Printer Module
- CTRL-E Jump to the Editor Module
- CTRL-I Jump to the I/O-Disk Module
- CTRL-K Jump to the Configuration Module
- CTRL-Z Jump to Auxiliary Software (if available!)

The Transfer Menu

Here you can play in all of the memory areas in your system to your heart’s desire. One can create and shift different storage ranges from here into the VDP Buffer. You can even get a program in sections, change it and then write it back, although for obvious reasons it comes with a warning. Not that it would be a concern with copyrighted text - they are, in principle, safe. No, to the contrary, you could shoot your program beyond recovery – and that is not in your best interest.

Transfer Direction

With 'R' you switch the direction in which the data is to be shifted. As previously stated it either goes into or out of the VDP buffer or the memory area is compared with the contents of the buffer. And so, you only have 'in'(in), 'aus'(out) or 'vgl' (compare)' buffer.

Offset

That is the relative buffer address, starting from which the data in the VDP buffer is stored and/or from which it is read into the buffer. It is completely independent from the address in memory outside of the buffers!

Memory Type

With 'T' you switch between the diverse memory types that the TI (at present) has. It is possible to do transfers in and out of GROM/GRAM (Type: GRM), out of CPU-RAM/ROM (Type: CPU), out or in to the Speech

Synthesizer (Type: SPEECH) and over an external VDP (Type: extVDP). The instructions for CPU and SPEECH are de facto system-inherent ranges that have firm address assignments. There are no options here, otherwise. GRAM/GROM and VDP are looked at a bit differently. Here the addresses can be freely set. More information follows:

Transfer-Options

Certain storage areas of the TI are opened only if certain system parameters are active. For example, the device control programs (DSR) that are accessed only when the CPU and its own CRU, segments of them only by simultaneously switching various 'Hardware-Options'.

The transfer part, before the actual beginning of the shift, the bringing in of a Bit Mask (16 Bit) to a fixed CRU address and the eventual activation ROM banking by writing to a certain address. It proceeds so carefully that a RAM Bank can be switched and the contents are not altered.

As soon as one of the three options indicates >0000, it is not worked on!

'C' adjusts the **CRU-Address, 'M'** specifies the **Mask** and **'B'** specifies the **Banking-Address**, which must be written in order to activate the desired bank.

Start- and End Address

Enter with 'S' and specify which other memory range (except VDP) is involved in the trnsfer.

System Addresses

GROM/GRAM and external VDP have no fixed address that one can always ‘capture’ with certainty. One can, after pressing 'G', all of the relevant GROM/GRAM parameters and through the press of 'V', adjust the external VDP as well. The GROM parameters are always phased in.

Speech

The Speech Synthesizer is always good for a surprise. Here it is the 20-bit wide address that causes problems. Since, however, not more than 256 Kbytes per speech ROM are not to be expected, one can preset this extra address (MSB) with the '+' and '-' keys. The existence of the Speech Synthesizer is NOT tested!

With the transfer of the Speech Synthesizer into the editor buffer the speech ROMs are copied. (Attention: the data bits are set in reverse order!).

With the transfer of the Speech Synthesizer the external command of the TMS5200 SPEAK is used.

In the case where the data transfer does not take place in the proper manner, after a time a BONK tone is issued and the communication is aborted, MM30 is fully functional afterwards. With comparison the buffer end is automatically adapted. Unfortunately, the newer TMS5220C behaves a bit differently than its predecessor so that the time until abort is a few seconds more.

External VDP

This point is meant for self-built and systems with the MSX2-VDP. These parameters are themselves covered in the original VDP but this VDP has the characteristic of opening in one of the new modes. For these the option WA, Start and End is given. This means that the address of VDPWA before the transfer of the option the value will be written in this manner (first the pattern of the Low-Byte, then the High-Byte) and after the transfer the end pattern will be written. The joke is that this option changes the screen colors during transfers (VDPint afterwards VDPext remain!) or when switching the VDP-RAM banks of the MSX2-Chips. Since this point is only for professionals, this is enough. To check for the existence of MSX-2 chips, there is an auxiliary insertion for Start and Target in the lower window.

These two banks do not have a function that can be checked completely or simply. At the time of the transfer, which bank will be the buffer will be interpreted and which is the target of the shift operation. The designation does not reverse with the direction of the transfer! The buffer bank is always in the VDP buffer (only when a genuine Bank 0 is there – otherwise the change is not recognizable). The external bank is another thing. With the VDPWA option, for example, if Register 14 is switched, then the bank logic of the transfer part detects this. VDPWA can, however, be used as an XRAM index in the present MM version shifts can only be done within the XRAMs.

Transfer Execution

Pressing 'A' begins the transfer in the desired direction. Actually, there is only one thing that can happen with which further inquiry would be necessary – the transfer into the CPU-RAM, where MM resides. In this case, a message comes up asking the seriousness of your intentions. You have the chance to avoid possible mischief. Otherwise – you were warned!

Error Messages in the Transfer Module

Error message in a Single Transfer

Änderungen aufgeben? (Remove Changes)

You tried to over-write the contents of the editor buffer that have not yet been saved. This warning can be over-ridden by pressing 'J' and the transfer to the VDP buffer will take place. Please consider the global character of this message since all of the contents of the buffers are classified as endangered and your transfer would not lead into the range in the buffers that were not changed, then you can ignore this message with ‘J’ as mentioned above.

Im Systembereich! (Within System range)

This message appears as soon as you transfer into the buffer in CPU-RAM from >A000 upwards where part of the main program of MM will be encountered. You can confirm with ‘J’ if you purposefully want to change MM or the function is otherwise aborted. This option should only be chosen with absolute knowledge of the main program since this function endangers MM and the auxiliary software to the extreme. It is not to be excluded that this function might be blocked in future versions without reference.

Puffergrenzen einhalten (Keep Buffer Borders)

You have tried to shift a data block whose size is larger than the capacity of the VDP buffer, either into or from that buffer. This message has an abort of the selected function as a consequence. In such a case, enter the Start- and End address of the transfer target and, likewise, the buffer offset so that a valid amount of data is moved.

Error Messages When Comparing

The comparison buffer and the external memory do not go well or evenly (no deviations). With a successful comparison a keypress is awaited so that the protocol of the comparison will not simply immediately disappear. If conflcting bytes are found, the value and the address of the bytes will be stored in buffer and in external memory. The buffer address is always be displayed from the logical buffer start from >0000 from which the buffer contents are interpreted. The external address is the actual address in the 64-K range of tested memory.

During an error condition the comparison can be ended with ‘E’, ‘W’ continues to the next byte, and ‘A’ continues all further byte comparisons with a renewed error message – at the end of the count the non-identical storage locations appear.

Overview of the Command Keys in the Configuration Module

Standard Functions

- A Store the date codes on/off (an/aus)
- B Ready tone on/off (an/aus)
- C Adjust cursor parameters
- D Enter date (only without a real-time clock!)
- F Input color code
- I Buffer-Initialization on/off (an/aus)
- S Save the configuration
- U Input time (only without a real-time clock!)
- Z Input auxiliary software name

Function Keys

- FCTN-7 Help, Dsiplays the necessary key codes
- FCTN-9 BACK, return to Main Menu

Shortcut Keys

- CTRL-D Jump to Print module
- CTRL-E Jump to Editor module
- CTRL-I Jump to the I/O-Disk module
- CTRL-K Jump to the configuration module
- CTRL-Z Jump to the auxiliary software (if available!)

The Configuration Menu

You can get to know MM and 'find your way' to personalize your settings. For example, if your cursor blinks too quickly or too slowly, you can change it or you can change the repeat rate or wait time for the automatic key repeat.

All of the parameters will be kept in the file 'MM-PAR' until the next start of MM. When this file is saved, the exact conditions present at that moment will be restored for the new start.

Practically, this means that all of the Hex parameters of the submenus Transfer, Disk-I/O, Editor, Printer etc. as well as the certain parameters of the auxiliary software are saved. Also, I/O directions, transfer type and direction and much more are saved.

All of this happens naturally if you have saved the configuration file before you left MM – self-explanatory!

And why is all of this? With this characteristic you can stop working with MM, whenever you wish, and one will find the program exactly as it was when you last left. For which parameters are stored in detail you can reread the sections for further guidance. Temporary values e.g. like the buffer contents of the temporary buffers are naturally not saved. Also, the buffer contents are not automatically saved. There must be an end to it because it would have been possible to save the buffer contents and the configuration file before leaving the program. On the one hand, you can do that yourself and on the other the program does not have to take any arbitrary actions that are not absolutely necessary.

By the way: The terms 'Configuration file' and 'Parameter file' are equivalent in this regard. What are put into this file are parameters that configure the program.

Program Start and Configuration Parameter File

When starting MM the loader looks for this file on the diskette from which the main program was loaded. Therefore, the name 'MM-PAR' must in no case be changed due to internal references in the program itself!! After the program start, the parameters that were found there are assembled as the program defaults. That means that with a load error of this configuration file these defaults are not in effect.

If loaded correctly, the MM loader in the configuration file looks for the name of the auxiliary software. This is done in such a way that it is the last file saved. You must also provide for the possibility of recognizing an empty string, for example, in the auxiliary file ring. This generates an immediate error while loading so that you do not lose anything if you do not wish for auxiliary software except the length of time it takes to do a self-search on the diskette. Since the main program works adaptively, this load error has no further effect.

The Options in the Configuration Menu

Also, the functions can be called here with a single key command. The following instructions are available:

A – Store the Date Codes

This gives you the possibility to use all of the hidden possibilities of the TI disk memory system and all of the TI programs (and that the Geneve-MDOS provides as well). If you do not have a real-time clock, you can input the time and date when you save a file from the Disk-I/O menu.

If you have a hardware clock in your system, the actual time is installed. You can later read this information directly from the file (with Status) if you want to get a picture of when the file was created or when it was last updated.

This option pre-supposes the correct administration of the VDP storage by the addressed storage medium. For inexplicable reasons, a RAMdisk operating system hardly adheres to these conventions so there are problems when the date codes are brought in from the RAMdisk. A RAMdisk operating system (ROS) that contains these conventions is currently in the test phase and will be offered as Freeware (only for Horizon Cards and has full backwards compatibility for registered MM owners!).

B – Readiness Tone On/Off

Normally the program gives diverse acoustic signals to you when, as an example, the input readiness signal (Beep) or an error notification (Bonk). The ‘bonks’ cannot be switched off (except at the most by the volume control) but the ‘beeps’ can. Press the ‘B’ to activate or, likewise, deactivate what is known as the 'Accept Töne'.

C – Set Cursor parameters

Here you can successively enter the flash rate, repetition delay, and repetition rate/period. All of the values are 16-bit hexidecimal data that have no current correlation to seconds or Hertz rate. These entries are a matter of taste and it is reasonable that you experiment with a several abstract numbers.

The value ‘Blink frequency’ specifies how long a blink phase lasts. If the cursor makes you nervous because it blinks too fast, you can increase this value. Doubling the value makes the cursor about half as fast, half the value (if watching it makes you fall asleep ,,,) causes a doubly nervous cursor. All intermediate values are permitted (except those that are unreasonable!) and the terms ‘double’ and ‘half’ refer to even hexidecimal numbers.

The repetition delay refers to how long you must hold down a key until its automatic repetition begins. Every-thing is the same is with blink frequency. A higher value lengthens the wait time and a smaller value shortens it. The repetiton delay, finally, determines how long the wait is before the automatic repetition begins.

All of these parameters are used when you have a visible cursor if the input permits a repeating cursor at all.

The defaults were developed after many attempts and represent, however, only ONE opinion of how the cursor should behave, This should not keep you from experimenting (>0001 for the repetition period makes for some interesting insight into the programming of the Buffer Editor).

D – Enter Date

For this there is not much to say (see ‘A’ and ‘U’). You can move the cursor freely and enter the date (in European notation). The data delimiters are jumped over. Please do not enter any nonsense here since it will bring the mechanism into question! If you have a hardware clock, and the input was correctly recognized, then you cannot use this input. With re-entry into this submenu, the current value was updated then a CorComp-compatible clock was found.

F – Color Code

With this, you can specify the color codes for the screen display. Input is by the cursor that can be moved left or right. The left code stands for the foreground color, for example, the character color, the right for the background. Only after ENTER is pressed are the codes transmitted and the screen display appears in the colors that you desire. The numerical values are the CALL COLOR values in TI BASIC that have been decreased by 1!

I – Initialize Buffer

If ‘J’ is displayed here, then the VDP buffer at the start of MM is completely emptied – switchable by pressing ‘I’. If you do not want to empty the buffer, then you will find there the remainder of the auxiliary software, or if this was not loaded, the remainder of the configuration file.

S – Saving the Configuration

In this window is a display of the disk drive on which you want to save the configuration file. As is the case for all disk drive numbers, values between 1 and 9 are permitted. FCTN-9 aborts the input and ENTER stores it on the specified drive/disk. The MM-PAR file should not be protected for this reason and, in the case it is, a warning tone sounds indicating that the file could not be saved.

U – Set Time

This is only for people who do not have a clock and want to adjust this value for the time. As with ‘D’, ‘Enter Date’, if you have a clock, stop.

Z - Filename of the Auxiliary Software

Pressing ‘Z’ takes you to the bottom line on the screen. Here you can input the name of the file that that is loaded into Low Memory when MM boots. The format is ‘DSK’, ‘Drive number’, ‘Filename’. In principle, the specified- filename should be correct or completely wrong (see above).

If the file name is completely wrong (for example, 'WGRKSFT' or completely empty) then nothing is loaded. The ‘Obelisk” quotation causes an error tone whereas an empty string is handled and no tone is produced.

Giving, for example, ‘DSK4’ plus the filename causes disk drive number 4 to be searched (DSKR for the CorComp RAMdisk). An asterisk ‘*’ as the disk drive number means that the loader will search the disk drive number from which MM and MM-PAR were loaded.

Error Messages in the Configuration Module

The Configuration Module only has a few situations where an error can be issued. Some errors produce the so-called BONK tone and others a warning box that must be confirmed.

Error Messages through the BONK Tone

Either MM-PAR cannot be loaded or the auxiliary software cannot be reloaded. Possible causes are: write protected disk, write protected file, or the wrong name for the auxiliary software.

Error Messages in the Warning Box

Änderungen aufgeben? (Remove Changes)

There is data in the Editor Buffer that has not been saved. Reloading the auxiliary software changes the first 8 Kbytes of the buffer so it might be that the buffer contents are destroyed.

Reload nur ohne '*' (Reload only without ‘*’)

The asterisk as the identification of the boot drive number is only available during the initial start of MM. An access of the boot drive is usually neither possible after one disk access nor is it meaningful. For reloading (e.g., in the test phase) the disk drive number must be given. If you want to save MM-PAR you can, naturally, use the asterisk to do so.

Concept of the Auxiliary Software

The principle of the creation of the auxiliary software was a logical cosequence of the needing ever more capacity by the main program Memory Manager over time.

The desire arose to use the I/O-, Transfer-, Editor- and Print options with other programs. Therefore MM was reprogrammed so it gets along with small buffer zones that are in the PAD-RAM or behind SLAST that left the entire Low-Memory in the extent of 8Kbyes free. Since MM is in program format, no module routines are need and/or used.

As the size of the main program grew larger, it seemed unreasonable to reassemble the main program and the auxiliary software again and again for different tasks and to produce the different versions of the same program bodies with slight differences.

The auxiliary software created a memory pool (Low–Mem) in the development phase and was a top priority in the main program and was granted a place in the configuration that became more extensive during the course of development. In addition to that there was a problem with the lack of an adequate loader for Tagged Object Code in the range of >2000 - >3FFF. The only way, at that time, was through the Mini-Memory that still needed a P-File saver.

With in inclusion of the TOC loaders in the Disk I/O part of MM this loader became available. Thus you need only an editor and assembler to produce auxiliary software; combining and testing can be done with MM and small changes to the object code be done with the MM editor!

Together with the auxiliary software the problem of the compatibility of all of the parameter files developed. The parameter file is neutral at least as far as the auxiliary software is concerned. Recognition and length identification now guarantee that old AND new program versions of MM and the auxiliary software and EVE-RY parameter file will be ‘transparent’! Since this causes certain restrictions, the first edition of MM could only take place following the standalone definitions of the parameter core!

All of this, among other things, is why the development of MM was extended again and again and the re-lease date was extended again and again – with the inevitable effect that gave bad-willing contemporaries material for ill-will speculation!

Short References to the Implementation of the Auxiliary Software

To all of the experienced machine language programmers please accept this explanation for the limited style of this section. This manual was written as a guide to the use of MM and to give the inexperienced user the means to use the program successfully. That brings us to the problem that the different auxiliary software must be compatible in any form!

Finally, it must be possible for the respective auxiliary software to recognize whether a parameter file fits it or not. Therefore, the authors of MM and the existing auxiliary software ask, that those who would program their own auxiliary software refer to the address given in section 2.

Implementation - or – What one must do in order to arrive at the necessary references and source code!

For connoisseurs of the Hitchhiker it is said: They do not need to forget it!

For your own auxiliary software you will need the source code for the routines in MM as well as the descrip-tion of the source code and its implementation of it, and the memory use plan (Memory Map). Some of these routines are quite distorted and it might be necessary that an interested customer to prove that he possesses the knowledge of TMS9900 machine language for them to be of use.

If a person has not been proven to be appropriate through the submission of his own program, the submis-sion of his own commented source code is desired. It can be a segment of a larger program that is useless by itself. The main thing is that it makes a statement about programming knowledge. The confidential treat-ment of this material is a matter of honor. Everything afterwards will be personally regulated. Using this pro-cedure will guarantee that the concept of the form of auxiliary software will remain intact

The Form of the Auxiliary Software

Each of the auxiliary software must be a P-File with a maximum length of 8202 bytes. That corresponds to 8192 bytes of program plus 10 header bytes. This is the concept of ‘over-counted’ header bytes that auxiliary software uses is to allow a regular EA5 branch, for example, to a small program at the end and at the start like a normal P-File to a text message that this file is for use only with MM. All existing auxiliary software is already equipped with this sequence. In the context of MM use, this can be above SLAST (in the auxiliary software!) or in the buffer.

Access to the Auxiliary Software by Routines in the Main Program

The input and output routines and conversion routines for the auxiliary software are available on the screen. Both the storage area and the VDP buffer editor range are defined by pointers. Besides the cursor position, the revision number of the main program and some other characteristics of the hardware are conveyed to the auxiliary software if the main program initialized it (if a V9938 VDP or a TMS9995 CPU were recognized and indicated). Thus, the auxiliary software can position the cursor freely for its own use within the editing range in the 80-column mode, etc. A separate revision number is reserved for the 80-column version of MM!

Restrictions for the Auxiliary Software

The auxiliary software is not permitted to access a segment of the main program for which the pointer has not been defined! Since auxiliary software is indepently processed, it cannot be controlled, however. It is because from there the programmer of the auxiliary software cannot manipulate the main program.

Available Programs

At the present the following software is finished as auxiliary software:

MM-EPROG

Control software for EPROG 27011 Author: H.Glaab

JS-EPROG

Control software for the modified TI-Eprommer Authors: H.Glaab, J.Stelter, Ch.Winter

MB-PROG

Control software for P-Box Eprommer Authors: M.Becker, D.Lotze, H.Glaab

MM-SOUND

Adaptive treatment of Sound lists Author: H.Glaab

In Development

BM-EPROG

Control software for Böhm Prog Authors: Ch.Winter, H.Glaab

MM-DISASS

Virtual Disassembler as a counterpart to the TOC Loader Author: H.Glaab

TI-PROG

Control program for the original TI-Eprommer MM compliant Author: H.Glaab

MM-EPROG and MM-SOUND belong to the ‘standard issue’ of the freeware version of MM and are included on the distribution diskette.

These programs (like MM) are released under the freeware concept and are available without cost. In each case, there are instructions either written or on disk - depending on the author. There is also an update service available for those who have freely paid the freeware obligation (see the respective notice) for MM and the auxiliary software. These programs can be requested from the address in section 2 or directly from the authors. Conditions: a stamped, addressed return envelope with a formatted, single sided disk and a short written request. If one of the enumerated this is missing, then no action can be taken – also no return!

Error Messages in General

Memory-Manager knows a set of error messages. However, not every error inevitably triggers an error message. Those that do always do so in a warning window (also known as an Alert Box) that exhibits a representation of a stop sign in the upper left and the word ACHTUNG (Attention) and some text on the right. There are two lines under the error with the lowest tells you which key allows you to continue and not selecting the key allows you to abort.

Automatically Clearing Error Situations

Various error conditions that can be settled easily are, to a certain extent handled automatically and the function is simply not executed. So, for example, exceeding the buffer end in ZP is one of these and it avoids having to correct the counter.

Error Messages that only Warn

These error messages warn before any action and/or side effects. In order to continue, one must ALWAYS press ‘J’ (capital ‘J’) and any other key aborts the function. With repeating functions (comparison, etc.) the next execution is prevented. Here pressing FCTN-9 (BACK or ESCAPE) completely aborts the function.

The following messages fall into this category:

ÄNDERUNGEN AUFGEBEN? (Remove Changes)

This always appears when you have changed the Editor Buffer and want to load another file but have not saved the contents. This is to avoid ruining your work if you inadvertently press ‘A’ in another module.

IM SYSTEMBEREICH! (In System Range)

This message in the Transfer Mode is displayed when the target area for the transfer from the buffer would affect the main program. This option was maintained for experimental purposes in order to underline the open concept of the main program (open in the sense of application!). You can manipulate the main program but what follows is your problem!

Error Messages that Block Functions

This group states that an illegal or unreasonable function was executed. For example, you entered a program function that is not yet in the actual current version of MM or under the parameters is not executable. In such a case, pressing any key will cause the message to disappear and the function is cancelled.

The following messages are in this category:

PUFFERGRENZEN EINHALTEN (Keep Buffer Borders)

You have indicated an address range that ‘jumps’ the buffer borders. Examine your inputs and enter the correct values. Please consider this data in the context of the revision supplement of MM (in the context of the update service that is sent to registered owners)! Function not available indicates that the chosen function is not yet available in your actual MM version. To avoid malfunctions or those that are problematic or unfinished, they are blocked. These cause no essential restrictions.

Update Service, Hotline

MM will never have all of the functions that are conceivably accommodated or wished for in memory. Therefore, extensions are planned. For this reason we have the concept of ‘registered owner’. Whenever a new MM version with updates appears, all of these owners are contacted and advised of these changes.

In this letter, the details of the updates are described.

This is not a Hotline service. This was to be in the event MM was released as a commercial product that was not realized for various reasons (MM would have cost about DM 200,- which not cost-efficient). Nevertheless, questions from registered owners are answered if they provide a stamped, self-addressed envelope and the question is posed in writing. Freeware is not cost-efficient and things must be done in this way.

Tips and Tricks for the Use of MM

Tip 1: Adjustment of GRAM Headers

1. Method: Load the file with header bytes 0. Jump to the Editor (CTRL-E) and press DELETE 2 to 6 times (depending on the type) to remove the header bytes and INSERT the new bytes in order. Make certain that the new bytes are correct and leave the editor and go to I/O Disk (CTRL-I) and change the length of the file (L) to the new length and change Load-Modus to Save-Modus. Execute Save with ‘A’ (and confirm the save with ‘J’). Repeat with all appropriate files. To see which header bytes must be examined, go to the appropriate header overview.

2. Method: Some GRAM files contain redundant header bytes when compared to other formats. If the first two bytes are redundant, then you load with 6 header bytes but save later with only 4 header bytes without changing the Start- and Endoffset. Or, you load with header bytes 0 and set the Startoffset two bytes higher and save the file back.

Tip 2: Edit from Files (LBO active)

Load an existing file with the Length Bye Option activated and produce a record-oriented file with the status Generator. You can now enter the new record you can now put a length byte in front which determines the number of characters in the file (without the length byte). When you insert of delete characters from the existing string, you must increase or decrease the length byte. The following length bytes remain unchanged.

Make sure that the last data byte is >00 as the length byte states since this is saved as the abort identification.

When saving it must be determined that the appropriate file already exists. One must make sure that the resord length is kept.

Tip 3: Special Functions in the use of MM as a SAVE Utility

The virtual TOC loader from MM loads, in principle, up to the EOF tag of the designated file. The SLAST label does not play a role. Also, when saving always check your work (with L) to the buffer end. Thus, you can use the area behind SLAST for unique initializations or messages and still use, nevertheless, SLAST as a buffer label.

Tip 4: Producing Low Memory Program files, Auxiliary Software or DSR Eproms

Assemble your program with an AORG of >2000, or >4000 as you wish. If you are working with relocatable (RORG or nothing at all) make sure you are loading on a relocatable basis. The TOC loader produces absolute addresses and needs a computing basis! With >0000, >2000 or >A000 use the buffer offset of >0006 to leave space for the header bytes. After loading edit these bytes save the file or ‘burn’ the EPROM.

Tip 5: Transfer in 9938 Systems

In MM version 3.x the buffer always is found in Bank 0 of an existing 80-column card (more precisely in VRAM). If you went through a VDP-VDP transfer, the buffer contents are copied into VRAM Bank 1. The reverse direction (VDP-VDP from 0 to 1 in the buffer) copies the contents of the VDP buffer back to Bank 0. If you exchange internal and external Banks, in general, the reversal occurs.

Tip 6: MM as a GRAM or Module File

This is possible in principle in version 3.0. It is important that through the Transfer, the file is not brought from the Editor Buffer and then to diskette but rather that the files are brought directly from a disk in the GRAM format of your card/cartridge. Otherwise the check routine will issue an error message and the CMOS GRAM-Karte will be robbed of its functionality.

The parameter file would naturally not be found (a pathway patch of the asterisk in DSK*.MM-PAR and nothing else will get you through this fatal error!) but the auxiliary file could be designated as DSKG.<filename>.You would then only have to indicate the drive number of MM-PAR when MM starts and

this question is asked. Subsequent versions will be optimized on this point.

Tip 7: Filling the VDP Buffers with a Word or Words

The Temporary Buffer (Function W) can only be filled byte by byte – the filling with words can only be done with the Manipulator!

Empty the buffer and it fills with >00 (Hex Mode) if data is present (ZP has been used). Activate the Manipulator (FCTN-5) and set the range. Use the filler in the XOR operator and set all of the other operators to >0000. After ending the Manipulator the buffer contains the desired pattern. Instead of XOR you can naturally use ADD – the results are the same! The start address does not have to be given!

Tip 8: Splitting a File Longer than 8K

Some EPROMMERS (notably the Mechatronics) will not load files that are over 8 Kbyes in length. This presents a problem for files that are, say, 16 Kbytes in length. With MM, however, there is an easy way to split them into usable pieces. Load MM and go to I/O Disk and follow this procedure:

Enter the filename in the format 'DSKx.LARGEFILE', 'Load-Modus': File, 'Start rel:' >00,'Ende rel:' >1F and read the file.

For the file to be saved input 'DSKx.SAVEFILE1', Save-Modus:'Program, 'Start-Offs.:' >0000, 'Ende-Offs.:' >1FFF and save the file.

Then, simply repeat the process with the following changes: 'Start rel:' >20. 'Ende rel:' >3F and read the file. Use 'DSKx.SAVEFILE2' for the second segment and input the same offset values as for the first segment and save the file. You will now have a perfectly divided 16 Kbye file saved as SAVEFILE1 and SAVEFILE2 that are both exactly 8 Kbytes in length.

Thanks to those to whom MM Owes its Existence

MM is – as shown by its extent not the work of a single person. The program body was developed in 1985, modified in 1987, and put together piece-by-piece for use in the private sector (for members of the System 99 Users Group). From 1988 to 1989 we rested from the project as bit by bit the freeware version was built. .

Every assembler programmer knows that starting point is a well-informed co-worker who, starting in the development stage, leads one out of dangerous blindness. My thanks especially to Harald Glaab, without who the auxiliary software would have little meaning, and who as co-author pointed the way out of other dark dead ends.

Likewise to be thanked is Michael Becker from the program user side who ‘agonized’ so long that the program is relatively error free. Particularly in the auxiliary software was Franz Neudert, who secured the compatibility for the Auerswald Programmer (EPROG 27011) and MM-EPROG and, through observation and knowledge, made it possible for the 9640-System from Myarc as was necessary for MM. This manual is the product of Roland Meier whose the finely sharpened grammatical view ensured that no error escaped.

View of the Updates

The following extensions are planned (after all of the specified functions are realized!) and the updates made available if the user reaction to MM deems them to be meaningful:

- Separate the parameter file from the main programs. Special reloadable parameter files (with different names for different functions).
- An 80-column version with the following options: a buffer area of 14.5 Kbytes with real window technology and up to 7 callable auxiliary software programs.
- Mouse control (without changing the operating system!). And much more...
- All new developments will work with 80-column cards (separate versions!) but pre-suppose an 80-column card (DIJIT, Mechatronic or the like!).