**Timex Command Cartridges**

# http://timex.comboios.info/Timex/ts2068cartridge.jpgC:\Users\JCaetano\Downloads\Angled view-1.jpgC:\Users\JCaetano\Downloads\IMG_4750.jpgC:\Users\JCaetano\Downloads\Cartridge.jpg

# Timex Memory Bank Switching System

The Timex's system of memory bank switching supports three internal 64 KB memory banks called HOME (bank 255/#FF), EXROM (bank 254/#FE) and DOCK (bank 0/#00). Other 253 banks of 64 KB are reserved (bank 1 to 253) for external expansion units.

The HOME bank contains 16 Kb ROM (called HOME ROM) at addresses 0-16383, and the rest of the 48 KB is filled with RAM, like the Sinclair ZX Spectrum 48K.

The EXROM bank initially contains only 8 Kb ROM (called EXROM ROM). This ROM is not fully decoded, so its image repeats itself 8 times (e.g. at addresses 0-8191, 8192-16383, 16384-24575 and so on). Although the EXROM bank is not designed for expansions, cartridges exist which are mapped into this bank.

The DOCK bank is initially empty and connected via pull-up resistor (so reading from it will give #FF). Its purpose is memory expansions (ROM or RAM) using cartridges which you can plug into the cartridge dock on Timex computer.

Banks from 1 to 253 hare planned to be used with the TS2060 Bus Expansion Unit (BEU), enabling the use up to 16MB of RAM/ROM, but it was never released, so these banks are not used. For more information, read this article:

[The Mystery of the Missing 253](http://8bit.yarek.pl/interface/ts.beu/wes-1.html)

Each bank as 64 KB of the address area of the Z80 and is divided into 8 chunks of 8 KB. In each chunk can be selected Home or Dock/EXROM bank - it is defined by #F4 (244) port. Bit 0 means #0000..#1FFF area, bit 1 - #2000..#3FFF etc. Zeroed bit selects Home bank, set - Dock or EXROM, what depends on bit 7 in #FF (255) port. Dock and EXROM banks cannot be paged in the same time.

**Memory Map:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Section Addresses** | **Bit (#F4)** | **= 0** | **= 1** | |
| **Chunk** |  | **Bit 7, (#FF)** | |
| **= 0** | **= 1** |
| **D** #C000..#FFFF | **7** #E000..#FFFF | HOME7 | DOCK7 | EXROM7 |
| **6** #C000..#DFFF | HOME6 | DOCK6 | EXROM6 |
| **C** #8000..#BFFF | **5** #A000..#BFFF | HOME5 | DOCK5 | EXROM5 |
| **4** #8000..#9FFF | HOME4 | DOCK4 | EXROM4 |
| **B** #4000..#7FFF | **3** #6000..#7FFF | HOME3 | DOCK3 | EXROM3 |
| **2** #4000..#5FFF | HOME2 | DOCK2 | EXROM2 |
| **A** #0000..#3FFF | **1** #2000..#3FFF | HOME1 | DOCK1 | EXROM1 |
| **0** #0000..#1FFF | HOME0 | DOCK0 | EXROM0 |

HOME0 and HOME1 are the Home ROM. Other HOMEx is Home RAM (HOME2 contains the basic screen).

TS1000/TS1500 Cartridges:  
Timex made the TS1510 cartridge player to add the command cartridges to the TS1000 (needs a TS1016 - 16K RAM Module) and TS1500 computer. These cartridges are incompatible with the TS2068/TC2068/UK2086 cartridges.  
  
The TS1000/1500/ZX81 use the signal */ROMCS* to disable the internal ROM. The cartridge works by pulling this low and placing its own ROM in this range. The Spectrum 48K + Interface 2 (IF2) also works in this way.

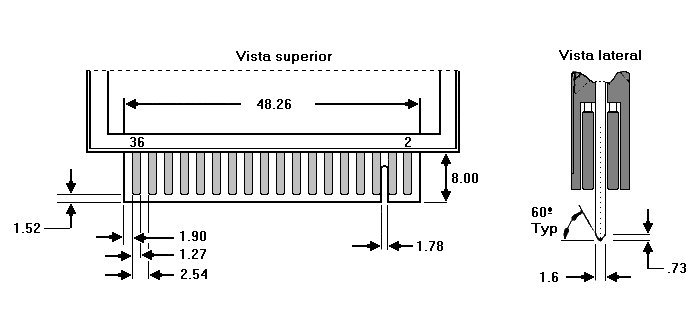
TS2068/TC2068/UK2086 Cartridges:  
  
Timex Command Cartridge (TCC) are a solid state technology that allows to load instantly any kind of program (games, business programs, word processors, etc.). They plug into the dock port of the Timex computer (**Never remove or insert a cartridge with the computer on!**) and the software in it will start to work. These cartridges actually add up to 64K of additional ROM and/or RAM memory to the computer.

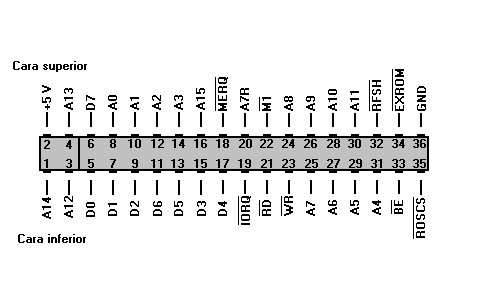
Timex Computer 2068 (TC2068) and Unipolbrit Komputer 2086 (UK2086) cartridges, made by Timex Portugal, are slightly higher than the ones made by Timex USA for the Timex Sinclair 2068 (TS2068). Otherwise the TS2068 cartridge slot top casing having to be enlarged to fit TC2068 cartridges (“Spectrum Emulator” and “Time Word”), they are 100% compatible and interchangeable.

Due to the incompatibility and useless signals placed in the TCC slot of TS2068 and TC2068/UK2086, the safest way is to build DOCK cartridges (using /ROSCS) and I/O cartridges only.

Design of safe HOME or EXROM cartridge would require a duplication of #F4 and #FF ports. Anyway, in each 2068 model the memory is disabled in other way (low or high state).

**Cartridge details as in the manual (sizes in millimetres):**





**The cartridge slot has following 36 lines:**

|  |  |  |
| --- | --- | --- |
| **Line** | **Signal** | **Description** |
| 1 | A14 | Address Lines: A14 |
| 2 | +5V | Power: +5V |
| 3 | A12 | Address Lines: A12 |
| 4 | A13 | Address Lines: A13 |
| 5 | D0 | Data Lines: D0 |
| 6 | D7 | Data Lines: D7 |
| 7 | D1 | Data Lines: D1 |
| 8 | A8 | Address Lines: A8 |
| 9 | D2 | Data Lines: D2 |
| 10 | A1 | Address Lines: A1 |
| 11 | D6 | Data Lines: D6 |
| 12 | A2 | Address Lines: A2 |
| 13 | D5 | Data Lines: D5 |
| 14 | A3 | Address Lines: A3 |
| 15 | D3 | Data Lines: D3 |
| 16 | A15 | Address Lines: A15 |
| 17 | D4 | Data Lines: D4 |
| 18 | /MREQ | Z80 CPU control signals |
| 19 | /IORQ | Z80 CPU control signals |
| 20 | A7R | Address Lines: A7 with 7th refresh bit |
| 21 | /RD | Z80 CPU control signals |
| 22 | /M1 | Z80 CPU control signals |
| 23 | /WR | Z80 CPU control signals |
| 24 | A8 | Address Lines: A8 |
| 25 | A7 | Address Lines: A7 |
| 26 | A9 | Address Lines: A9 |
| 27 | A6 | Address Lines: A6 |
| 28 | A10 | Address Lines: A10 |
| 29 | A5 | Address Lines: A5 |
| 30 | A11 | Address Lines: A11 |
| 31 | A4 | Address Lines: A4 |
| 32 | /RFSH | TS2068: Memory refresh signal: /RFSH of Z80 |
| /ROMCS | TC2068: Active low chip select signal for HOME ROM bank |
| 33 | /BE | TS2068: Active low memory access disable: /BE of ULA |
| /???? | TC2068: Active high input that disables both internal ROM and also forces high state on both /ROMCS and /EXROM in the slot |
| 34 | /EXROM | Active low chip select signal for EXROM bank |
| 35 | /ROSCS | Active low chip select signal for DOCK bank |
| 36 | GND | Power: Ground |

Types of TS2068/TC2068/UK2086 Cartridges:  
 **Actually exists 3 types of cartridges: LROS, AROS and ROMs.  
  
LROS takes control of the computer, the internal ROMs are not used;  
AROS uses BASIC interpreter;  
ROM is simply a ROM chip. In TS2068 to a ROM work, you must type OUT 244,3 to "page" the DOCK port.  
  
ROM extensions mapped into the DOCK bank are mostly implemented as LROS (Language ROM Oriented Software) and AROS (Application ROM Oriented Software). LROS programs are mapped at address 0 in the DOCK bank and they must be written in machine code. AROS programs are mapped at address 32768 in the DOCK bank, and may be either in machine code or in BASIC (The Timex BASIC interpreter allows the running of BASIC programs from the DOCK bank). LROS programs always have autorun and they will be started after initialization of the Timex computer is finished (an example of such software is Zebra OS\_64). AROS programs may be or may not to be autorun programs. Both type of programs have a short header at the beginning which contains the necessary informations (start address etc.) for their execution.**

**DCK FILES - Used by Warajevo emulator  
  
DCK files keeps information about memory content of various Timex memory expansions, and information which chunks of extra memory are RAM chunks and which chunks are ROM chunks. Such files have relatively simple format. At the beginning of a DCK file, a nine-byte header is located. First byte is bank ID with following meaning:  
  
0: DOCK bank (the most frequent variant)  
  
1-253: Reserved for expansions which allow more than three 64 Kb banks  
  
254: EXROM bank (using this ID you may insert RAM or ROM chunks into EXROM bank, such hardware units exist on real Timex Sinclair)  
  
255: HOME bank (mainly useless, HOME content is typically stored in a Z80 file); however, using this bank ID you may replace content of Timex HOME ROM, or turn Timex HOME ROM into RAM   
  
This numbering of banks is in according to convention used in various routines from Timex ROM.  
After the first byte, following eight bytes corresponds to eight 8K chunks in the bank. Organization of each byte is as follows:  
  
bit D0: 0 = read-only chunk, 1 = read/write chunk  
  
bit D1: 0 = memory image for corresponding chunk is not present in DCK file, 1 = memory image is present in DCK file  
  
bits D2-D7: reserved (all zeros)  
  
To be more clear, these bytes will have following values:  
- 0, for non-existent chunks (reading from such chunks must return default values for given bank; for example, #FF in DOCK bank, and ghost images of 8K Timex EXROM in EXROM bank)**

**- 1, for RAM chunks, where initial RAM content is not given (in the emulator such chunks will be initially filled with zeros)**

**- 2, for ROM chunks**

**- 3, for RAM chunks where initial RAM content is given (this is need to allow saving content of expanded RAM; also this is useful for emulating non-volatile battery-protected RAM expansions) After the header, a pure image of each presented chunk is stored in DCK file. Some examples will help understanding of such organization. 16 Kb long LROS program needs header 0,2,2,0,0,0,0,0,0 in front of pure binary image of this program. 24 Kb long AROS program needs header 255,0,0,0,0,2,2,2,0 in front of binary image of it to become a valid DCK file. 64 Kb DOCK RAM disc cartridge (64K of empty RAM) may be described as only 9-byte long DCK file with content 0,1,1,1,1,1,1,1,1. 32 Kb EXROM RAM disc cartridge mapped at address 32768 may be described also using 9-byte long DCK file with content 254,0,0,0,0,1,1,1,1. If you put a 9-byte header 255,2,2,0,0,0,0,0,0 in front of binary image of standard ZX Spectrum ROM, you will get DCK file which will replace Timex HOME ROM with ordinary Spectrum ROM (e.g. you will achieve Timex Computer 2048). At the last, if you put a header 255,3,3,0,0,0,0,0,0 in front of binary image of Timex HOME ROM, you will allow writing in the HOME ROM!  
  
That's all if only one bank is stored in DCK file. Else, after the memory image, a new 9-byte header for next bank follows, and so on.  
  
The Spectrum Emulator Cartridge:  
  
This cartridge made by Timex of Portugal, contains a ROM image of TC2048. I think that this isn't a ROM cartridge, but a LROS cartridge, because it is simply plugged to TC2068 and it starts automaticaly. Many American Timex Sinclair 2068 users made a cartridge with a ROM chip of a ZX Spectrum and because of this, they have to type the OUT 244,3 command.  
  
Cartridge Software Overview**  
  
**The TS2068 supports two basic types of Cartridge or ROM-Oriented Software designated as LROS (Language ROM-Oriented Software) and AROS (Application ROM-Oriented Software) which pluq into the cartridge connector. They are identified via overhead bytes at Location 0 for an LROS or 32768 (80#00) for an AROS. The fundamental difference is that an LROS contains Z80 machine code in memory chunk 0 and is in total control of the TS2068 hardware including the RESTART implementation and Interruption Mode setting and handling, while an AROS is dependent on the System ROM or an LROS for these functions if needed. An AROS written in BASIC, which may also include machine code accessed via the USR function, is supported from the System ROM BASIC Interpreter and is mapped beginning in memory chunk 4. An AROS may also be written entirely in Z80 machine code. An AROS written in any other high-level language would require an LROS supporting that language and would have to be integrated with the LROS in a single cartridge.**  
 **BASIC AROS Support**  
  
**BASIC Application Cartridges are supported by special code in the Home ROM. A program line is copied from the cartridge to a buffer in the Home RAM (ARSBUF) and is then executed from there by the BASIC Interpreter. When a READ command is executed, the line containing the appropriate DATA statement is also copied from the cartridge to the RAM. The cartridge memory is enabled only fur search and copy operations for both program lines and DATA statements, and when executing a USR function, otherwise the entire Home Bank is enabled while executing in the BASIC Interpreter. There is no support for User-Defined Functions which insert the expanded definition parameters directly into the program and then require search of the program area to find these parameters whenever a function is invoked.**  
 **Cartridge Software/Hardware  
  
LROS**  
  
**An LROS is identified by the following overhead bytes:**

|  |  |
| --- | --- |
| **Location** | **Description** |
| **0000** | **Not Used** |
| **0001** | **Cartridge Type -> O1=LROS** |
| **0002/0003** | **Starting Address (LSB/MSB) Address to be jumped to after Operating System initialization is complete. Order of bytes is as for a JP instruction.** |
| **0004** | **Memory Chunk Specification. Bits 0-7 represent Chunks 0-7 respectively in the Dock Bank in low active format:  0 if in use 1 if not in use  NOTE: When writing to the Horizontal Select Register (Port F4H), the Chunk Specification is High Active** |

**The Memory Chunk Specification is used to enable the specified chunks in the Dock Bank prior to jumping to the address specified in Location 2 and 3. Control is transferred from the Initialization code in the Extension ROM via the GOT0 BANK routine in Home Bank RAM Chunk 3, therefore Bit 3 of the Memory Chunk Specification must be set to 1 in order for the transfer to be accomplished as designed (Chunk 3 also contains the Machine Stack).  
  
CAUTION: If Chunk 3 is marked for' use in the Dock Bank, then when the Memory Chunk Spec. is written to Port F4H by the Sank Enable code, execution will continue from that point in Chunk 3 in the Dock Bank with the Stack Pointer addressing ROM.  
  
An LROS is Z80 machine code and is in complete control of the TS 2068 hardware after transfer to the starting address has been made. It can directly implement an application, or it can support multiple applications by implementing a language other than basic. An AROS dependent on such an LROS would have to be part of the same cartridge since there is only one cartridge connector.**

**Interruption Mode 1 has been set by the TS 2068 and interruptions are enabled prior to passing control to the LROS starting address, therefore the LROS must contain appropriate code at location 56 (38H) to cover the case where the interruption occurs after Chunk 0 in the Dock Bank has been enabled, but before any action by the software cartridge to disable the interruption has been taken. Once control is transferred, the LROS may then disable the standard TS 2068 interruption by setting bit 6 of Port #FF, mask the interruption by executing a DI instruction, or set a different Interruption Mode. It may change the location of the Machine Stack. It may also change the memory selection by writing to Port 0F4H with each bit set to 1 for the corresponding chunk to be enabled in the Dock Bank (high active format) or 0 to be enabled in the Home Bank. Thus, an LROS may contain code in Chunk 3, but it should be enabled after the OS RAM code has finished execution. Now that your LROS is in the driver's seat, you are on your own! Some important points to remember when,, mapping your Dock Bank memory and doing bank switching are:  
  
1. The Display RAM is in Home Bank Chunk 2 for the primary display file and Chunk 3 for the second display file. This memory is accessed independently by the video hardware. The software only needs to enable it when actually reading or writing it.  
  
2. The Dock Bank and Extension ROM Bank are mutually exclusive since they share the Horizontal Select Register in Port F4H. You will need a routine in the Home Bank RAM to do any switching between the two. You must also be careful to have the appropriate Home Bank Chunks enabled which are referenced by the Extension ROM code, e.g. the System Variables in Chunk 2 or possibly the bank switching code in Chunk 3.  
  
3. Some interesting switching routines can be constructed by having parallel code in shadowing chunks of memory to take advantage of the"instant" switch in execution from one hank to another when the memory selection is made. E.g. a routine in the Dock Bank ROM in Chunk 6 could push a Home Bank address on the stack, write to Port F4H enabling Chunk 6 and any other desired chunks in the Home Bank (by deselecting them in the Dock), and have code at the next sequential instruction address in Home Bank RAM Chunk 6 to continue the path. A Return instruction, for example, would pass control to the address on the stack. Code to switch memory back to the Dock Bank could be mapped in a similar way.  
  
4. If you plan to use any of the System software routines, unless you know otherwise it is probably necessary to maintain the contents of Home Bank Chunks 2 and 3 intact (and Chunk 7 if the OS RAM routines have been relocated). The system routines rely heavily on the System Variables and assume that any pointers in them are pointing to the Home Bank. See Section 3.3.4.1 for details on using the RAM Interruption Handler and Section 6.0 for known corrections when using System S/W.  
  
5. If you design an LROS implementing a higher-level language and want to support an AROS application, you must design your own initialization code to detect the presence of such an AROS. The TS 2068 will not look for the presence of an AROS if an LROS is present, therefore there will be no entry for the AROS in the System Configuration Table. Note that since there is only one cartridge connector, such an AROS would also have to be integrated with the supporting LROS in a single cartridge or cartridge board.**

**AROS**  
**An AROS is identified by the following overhead bytes**

|  |  |
| --- | --- |
| **Location** | **Description** |
| **32768 (80#00)** | **Language Type 1 = BASIC [and machine code] 2 = Machine code only (Any other value will result in Error S, Missing LROS)** |
| **32769 (8001H)** | **Cartridge Type 2= AROS** |
| **32770/32771 (8002/8003H)** | **Starting Address(LSB/MSB) BASIC AROS= Addrs. of First Program Line Machine Code AROS = Addrs. of First Z80 Instruction** |
| **32772 (8004H)** | **Memory Chunk Specification Bits 0-7 represent Chunks 0-7 respectively in the Dock Bank in low active format as follows: 0 if in use 1 if not in use  NOTE: Bits 0-3 must he set to 1 for proper execution.** |
| **32773 (8005H)** | **Autostart Specification: 0= No Autostart                                          1 = Autostart** |
| **32774/32775 (8006/8007H)** | **Number of bytes of RAM to be Reserved for Machine Code Variables (LSB/MSB - 01#00=1 byte Reserved; 0002H=512 bytes Reserved.** |

**BASIC AROS**

**A BASIC AROS is supported by special code in the System ROM (Section 3.2.1.2). The portion of the cartridge containing BASIC program lines is restricted to the upper half of the memory space beginning at location 32776 (8008H) in the Dock Bank. Support for User-Defined Functions, which requires searching for the definition parameters within the program, is not implemented. Also, because the support code interfaces directly to the bank switching code in.Home RAM Chunk 3 (does not allow for it to be relocated to Chunk 7), a BASIC AROS cannot utilize the advanced video modes and also execute BASIC program statements. If the cartridge contained machine code supporting advanced video modes, the TS 2068 would have to be returned to "Normal " video mode with the RAM mapped accordingly (see Figure 1.1-3) if control were to be returned to the BASIC Interpreter USR code.   
Since execution of the cartridge BASIC program is done by copying program lines to a buffer in the Home Bank RAM (ARSBUF), the most efficient cartridge execution is obtained by making program lines as large as possible, making use of the multi-statement feature of the TS 2068. The reverse is true concerning execution of READ commands. An entire DATA statement is copied to the Home Rank RAM, but only the current item is accessed. It therefore will be more efficient to not make DATA statements excessively long. The BASIC program lines appear in the cartridge in exactly the same format used in the RAM, i.e. Line Number (2 bytes), Length (2 bytes), Command Token, etc. terminated by an Enter (ODH). Numerical constants appearing in a proqram line are followed by the CHR$ (OEH) byte and 5-byte floating point format described in the User Manual (see Appendix C of the TS 2068 User Manual). The Variables area is built in the RAM (address in VARS) exactly as though the program were in the RAM. All variables, including arrays, are built at the time of program execution - there is no provision for copying or accessing ore-defined: variables from the cartridge, however, see Section 5.3.2. The last program line must be followed by a terminator byte having the Most Significant Bit set (e.g. 8OH), otherwise the Interpreter cannot detect the end of the program.  
A BASIC AROS may contain machine code accessed via the USR function. If the machine code address is within the memory designated by the AROS Memory Select Specification as 'in use', the Dock Bank will be enabled, otherwise the machine code address is assumed to be in the Home Bank. (See Section 6.0 for details on known problems i n this area of the code.) Obviously, once control is transferred to the machine code in the AROS, the ball is now in your court. You could have additional machine code residing in the lower half of the Dock Bank memory space which you can now switch in. You only have to know what you're about. If and when you are ready to go back to executing your BASIC program, you must enable Chunks O-3 in the Home Bank and have the stack and other Home Bank RAM in the proper state for return to the USR function code in the BASIC Interpreter, i.e. what it was when the USR function passed control to you. The Autostart feature begins execution out of the BASIC AROS immediately after system initialization. If the Autostart parameter is zero, control will go to the BASIC Interpreter as if there were no cartridge installed, althouqh internal flags have been set noting that a BASIC AROS is present. The cartridge will be started when you execute a RUN or GOT0 Line Number command.  
The final parameter in the overhead bytes allows you to reserve RAM beginning in Chunk 3 at Location 26688 (6840H) for machine code and/or machine code variables. The designated number of bytes are reserved by the AROS support code prior to beginning program execution. The AROS buffer (ARSBUF) begins immediately following this reserved area. Note that this area is part of the RAM that gets relocated if the second display file is opened. Therefore access to your machine code and/or variables should he conditional on the video mode rather than direct if you are going to be using the advanced video modes,. This reserved area begins at 31488 (7B#00) when the second display file is open. Remember-- use of the second display file and execution of BASIC program from the cartridge are mutually exclusive.  
The standard technique of reserving space for machine code by modifying RAMTOP could also be used to place machine code/variables at the top of the Home Bank RAM. If you place code above (RAMTOP) which is to be accessed via the BASIC USR function, the affected memory chunk(s) cannot be marked as "in use" in the cartridge in the AROS Memory Selection Specification.**  
  
**Machine Code AROS**  
  
**A machine code AROS is similar to an LROS with the exception that it is dependent on the System ROM for interruption handling if the interruption is enabled. This implies that Chunks O-3 are enabled in the Home Bank.  
The Autostart parameter should be set to 1 since if it is zero, control will be passed to the BASIC Interpreter as if the cartridge were not present. There is no BASIC command to directly start execution of a Machine Code AROS.  
Because of a "bug"in the Initialization code handling a Machine Code AROS, the parameter specifying the number of bytes to be reserved for machine code variables must be adjusted by adding 21 (15H) to the actual number of bytes needed. This preserves the 21 byte CHANS area starting at 26688 (6840H). The reserved area then starts at 26709 (6855H) (or 31488 (7B15H) when the second display file is open). Access to the variables should be conditional based on the video mode rather than direct if you plan to use the advanced video modes. If you do not plan to utilize any of the system software, you can disregard the above and "do your own thing" with the RAM.**  
  
**BASIC AROS Variables**  
  
**In order to use pre-defined arrays and/or other BASIC variables, store them in the cartridge (possibly in the lower half of the addressable space which is not usable for BASIC program) and branch to a machine code routine via the USR function at the beginning of your BASIC AROS program. U se this routine to do the necessary memory selection and copy your data from the cartridge to the RAM (address in VARS). Adjust the System Variables E LINE, WORKSP, STKBOT and STKEND to all point to the first free memory following your BASIC variables. Of course, all BASIC variables must conform to the format expected by the BASIC Interpreter. In addition to BASIC structures, you can also store screen images and machine code/variables in the cartridge for transfer to the RAM under your control. Consider using the XFER BYTES service in the OS RAM.**  
**Known "BUGS" and Corrections**   
  
**This section describes the known problems in the TS 2068 System Software and gives corrections or work-arounds where these have been defined.  
  
1 LROS and Autostart Machine Code AROS  
  
1.1 If you will be using the System ROM Keyboard routines and accessing the input character code from system variable LAST K (5C08H), you must initialize the TS 2068 to "L" mode by setting the system variable MODE at 23617(5C41H) to zero and setting Bit 3 of FLAGS (23611 -5C3BH) to 1. (The TS 2068 is in "K" mode when control is passed from System Initialization to the Cartridge; Keyword Token codes will be placed in LAST K instead of character codes.**

**1.2 If you will be using the System ROM Calculator routines (RESTART 40 (28H) ) or any ROM routines that invoke them, you must initialize the System Variable YEM by doing the following:  
LD HL,5C92H Set HL=MEMBOT  
LD (5C68H),HL Initialize MEM  
  
1.3 Chunk 3 must not be designated as "in use" by the Cartridge Memory Selection Specification byte. This will cause deselection of the bank switching code prior to completion of the transfer of control to the cartridge starting address. Once control has been transferred, the cartridge code may then enable Chunk 3 in the Dock Bank if desired. (See Section 5.1.)  
  
1.4 No entry is made in the System Configuration Table for an RROS if an LROS is present. This means that an LROS designed to support either RAM based or cartridge based applications must include code for detection of an AROS.  
  
2 Machine Code AROS  
  
When setting the AROS Overhead parameter requesting RAM space for machine code variables, 21 + n bytes (15H + n) must be requested where n is the number of bytes needed. The machine language variables area then starts at 6 85 5H immediately following the 21-byte CHANS area.**

**NOTE: This does not apply to an AROS that contains both BASIC and machine code.  
  
3 BASIC AROS  
  
3.1 USR Function - When testing the USR address against the Cartridge Memory Selection byte to determine if the address is in the Home Bank or the Dock Bank, the wrong nibble is tested in the register thus a valid cartridge address could be erroneously processed as a Home Bank address. Since the ROM code cannot be corrected, the machine code in the cartridge would have to be moved to an address that does not cause a problem.  
  
3.2 FOR/NEXT - If the limit of the FOR statement has already been passed on its initial execution, (e.g. FOR A=1 TO 10 and A has been set to 12), control is passed to the statement following the corresponding NEXT. In the AROS support code, the address of this statement is lost giving unpredictable results. Since the ROM code cannot be corrected, care must be taken not to use this technique in an AROS Cartridge. Normal usage of FOR/NEXT loops is not affected.  
  
3.3 Advanced Video Modes- Because the BASIC AROS support code interfaces directly to the Bank Switching code in Chunk 3 (does not access based on its relocatability), the second display file cannot be open when executing BASIC program from an AROS.**