x**KIM**

**Extended KIM-1 Monitor**

Last update: 10/10/2021

Bob Applegate [bob@corshamtech.com](mailto:bob@corshamtech.com)

Corsham Technologies, LLC

[www.corshamtech.com](http://www.corshamtech.com/)

# Introduction

XKIM is the extended monitor included on some Corsham Technologies, LLC products such as the KIM-1 RAM/EEPROM board and the KIM Clone system. It is written in 6502 code without use of any extended 65C02 instructions. It provides a number of useful commands as well as vectors to numerous subroutines.

It is meant to provide an easy interface to the Corsham Tech SD Card system to save and load hex files, as well as access the real-time clock.

Full source code is provided so anyone can make changes, take useful code for other projects, etc.

## Key features

* Load Intel hex files from either the console or an SD Card.
* Directory of SD Card.
* Display time from SD Card RTS.
* Examine/modify memory.
* Memory test.
* Extendable.
* Provides subroutines for accessing the SD Card.

The code is written in plain 6502 instructions so it can work on any 6502-based processor.

## Requirements

* KIM compatible monitor, although similar functions are available in other monitors.
* For SD related commands, a Corsham Technologies SD Card System.
* 128 bytes of RAM from DF80 to DFFF.

# Commands

All commands start with a unique first character. If additional arguments are required, such as addresses, the command handler will wait. All addresses must be entered as four hex digits. For commands that require more input, they will prompt the user.

## ? - Help

Show a list of all commands.

## C – Current Time and Date

Gets the current time and date from the SD Card System and displays it.

## D – Directory of SD Card

Does a directory of the SD card. Only files at the top level are displayed, as there is no option to move into any subdirectories.

## E - Edit Memory

The user will have to enter a four digit hex address and then the editor will start. Entering a two digit hex value will put the value at the current memory location and then move to the next address. Pressing RETURN will advance without changing the current value. Backspace will move to the previous address. Any other non-hex value exits the editor.

There is another fun feature to help with code entry. If a branch instruction is put into an address, when the editor is at the next address (where the offset goes) press the R key. It will prompt for the target address. Enter the four byte address of where the branch should go and xKIM will put in the proper offset at this memory location, then move to the next location. If the branch is out of range then an error will be displayed.

## H - Hex Dump

The user enters a starting and ending address. This will round down the starting address to a 16 byte boundary.

## J - Jump to Address

The user enters a four digit hex address and xKIM will JMP to that location with undefined register values.

## K – Return to KIM-1 Monitor

Exits xKIM and returns to KIM. The K command will be deprecated.

## L - Load HEX File

The user will be prompted for either a filename or whether to load from the console (upload from terminal emulator software). If the file sets the AutoRun address then xKIM will jump to that address after the load is completed.

## M - Memory Test

This waits for a starting and ending address, then does a rolling nine bit pattern indefinitely. At the end of each successful pass, a dot is printed. An message is displayed if an error is found.

This instruction runs forever, so press the ST key on the keypad to stop it.

## O - Calculate Branch Offset

To help those writing code, this will take the address of the branch instruction (first argument) and compute the offset to the target address (second argument). If the branch is out of range then an error will be displayed.

## P – Ping (“Ah, I see you have the machine that goes PING!”)

Pings the SD Card System to verify it is on-line. This was an earlier debugging tool and might be removed in future versions. The quote is from Monty Python about a machine that everyone is quite impressed with because it makes a pinging noise but does not seem to have any other purpose.

## S - Save Memory to an SD File

On the command line, enter the starting and ending address. It will then ask for a filename on the SD drive, or allow the dump onto the console. The range is inclusive, ie, it saves from the starting address up to and including the end address.

## T - Type a File on SD Card

Prompts for a filename, then dumps the contents of the file to the console.

## X – Return to KIM-1 Monitor

Exits xKIM and returns to KIM. The K command will be deprecated.

## ! - Cold Start

This forces a cold start of xKIM. Any additional handlers added by the user are removed.

# xKIM Subroutines

All addresses assume xKIM starts at E000. If relocated then these addresses will differ. The subroutine names match those in xkim.inc.

## E000 – extKIM

Enters the xKIM monitor. Can be used for either cold or warm start.

## E003 – xkOUTCH

Sends the character in A to the console. All registers modified.

## E006 – xkGETCH

Waits for a character from the console and returns it in A. Modifies all registers. This will block until a character is received. On a KIM, the character is always echoed.

## E009 – xkGETCHne

This is a place holder for a subroutine to get a character from the console without echoing it. The KIM-1 hardware does the echo so this will get a character but always echoes it.

## E00C

Reserved for future subroutine to get console status. Not implemented. Always returns with carry clear.

## E00F - putsil

Putt string in-line. This prints the text immediately following the call to putsil, up until a zero byte. Allows printing messages directly where they are used. After printing, this returns to the instruction following the zero byte.

## E012 - getHex

This gets two hex characters and returns the 8 bit value in A with carry clear. If a non-hex digit is entered, then A contans the offending character and carry is set.

## E015 - xkPRTBYT

Prints the contents of A as two hex digits. All registers modified.

## E018 - getStartAddr

Gets a four digit hex address amd places it in SAL and SAH. Returns C clear if all is well, or C set on error and A contains the character.

## E01B - getEndAddr

Gets a four digit hex address amd places it in EAL and EAH. Returns C clear if all is well, or C set on error and A contains the character.

## E01E - getAddrRange

Get an address range and leave them in SAL/SAH and EAL/EAH. Calls getStartAddr to get the SAL/SAH value, prints a “-” and then calls getEndAddr.

## E021 – xkHexDump

This subroutine does a hex dump from the address in SAL/H to EAL/H. Rounds down the start address to a 16 byte boundary.

## E024 – xkMemEdit

This subroutine edits memory. On entry, POINT has the first address to edit. Upon exit, POINT will have been updated to next address to edit.

This is the same code used for the E command, so all editing options explained there will apply when this function is called.

## E027 – loadHexConsole

This subroutine is called to load a hex file from the console. A success or error message will be displayed but there is no status returned.

## E02A – loadHexFile

This subroutine loads a hex file from the SD. On entry the pointer to the filename is in X (MSB) and Y (LSB). This will print dots as the file is loading and will display either a success or error message. Provides no status indication on return.

## E02D – doDiskDir

Does a directory of the SD card and displays the file names to the console. Note that if no SD card is attached, this will hang.

## E030 – calcOffset

This computes the relative offset between the address in SAL/SAH (address of branch instruction) and EAL/EAH (address to jump to). If a valid range, returns C clear and the offset in A. If the branch is out of range, C is set and A undefined. Modifies A, SAL and SAH.

# SD Card System Subroutines

These are helper subroutines to work with the Corsham Technologies SD Card System. All of the xPar functions are low-level calls, while the Disk calls are higher level. There is support for only one open file at a time, so when working with files, applications must close one file before opening another.

## E033 – xParInit

This is the initialization subroutine for talking to the SD Card System. This is called by xKIM but it does not hurt to call it again. Takes no input parameters and destroys A on return.

## E036 – xParSetWrite

This is called to prepare to start a write sequence to the SD Card System. This needs to be called before the first call to xParWriteByte, but not again for subsequent writes. The interface remains in write mode until xParSetRead is called. Returns immediately.

After completing a transaction it is recommended to call this subroutine again so that the interface is normally in the write state.

## E039 – xParSetRead

This is called to prepare to start a read sequence to the SD Card System. This needs to be called before the first call to xParReadByte, but not again for subsequent reads. The interface remains in read mode until xParSetWrite is called. Returns immediately.

## E03C – xParWriteByte

Writes the byte in A to the SD Card System. This will block until the byte is transferred. It will block indefinitely if there is no SD system attached. I.e., there are no timeouts. The interface must be in the write mode before calling this.

## E03F – xParReadByte

Reads a single byte from the SD system, blocking until a byte is received, then returning it in A. This will block indefinitely if no SD card system is attached. The interface must be in the read mode already.

## E042 – DiskPing

Sends a ping request and waits for the pong response from the SD card. Returns with C clear if a good response received, C set on a bad response or never returns if no SD card is attached.

## E045 – DiskDir

Starts a directory read of the SD card. Always returns C clear.

## E048 – DiskDirNext

Read the next directory entry. On input, X (MSB) and Y (LSB) point to a 13 byte area to receive the drive data.

Returns C set if end of directory (ie, attempt to read and there are none left). Else, C is clear and X/Y point to the null at end of filename.

## E04B – DiskOpenRead

This opens a file on the SD for reading. On entry, X (MSB) and Y (LSB) point to a null-terminated filename to open. On return, C is clear if the file is open, or C set if an error, usually meaning the file does not exist.

## E04E – DiskRead

Reads bytes from an open file. On entry, A contains the number of bytes to read from the file, X (MSB) and Y (LSB) point to the buffer where to put the data. On return, C will be set if EOF was reached (and no data read), or C will be clear and A contains the number of bytes actually read into the buffer.

Modifies A, X and Y. Also modifies INL and INH (00F8 and 00F9).

## E051 – DiskClose

Call this to close any open file. No parameters and no return status.

## E054 – DiskOpenWrite

This opens a file on the SD for writing. On entry, X (MSB) and Y (LSB) point to a null-terminated filename to open. On return, C is clear if the file is open, or C set if an error.

## E057 - DiskWrite

Writes bytes to an open disk file. On entry, A contains the number of bytes to write to the file, X (MSB) and Y (LSB) point to the buffer where to get the data. On return, C will be set if an error was detected, or C will be clear if no error. Note that if A contains 0 on entry, no bytes are written.

Modifies A, X and Y. Also modifies INL and INH (00F8 and 00F9).

# Useful RAM Locations

xKIM uses RAM from DF80 to DFFF. The locations documented here are fixed and will not move, whereas other locations might change in future versions. Beware if you access those locations.

## DFF8 – AutoRun

Then a hex file is loaded from either the console or SD Card, if it places an address in the AutoRun vector then xKIM will jump to that location once the file is fully loaded. This provides a way to auto-run a hex file on loading. If this is not set then control is returned to xKIM when the load completes.

## DFFA – ColdFlag

This 16 bit value is used to indicate to xKIM if a call to extKIM is a warm start or a cold start. To force a cold start, modify either of these addresses before calling extKIM.

## DFFC – ExtensionAddr

A pointer to a user-written extension to the xKIM monitor. It is possible to add new commands to xKIM’s command handler and help system. Please refer to the samplext.asm file or the [Sample Extension](#Sample_Extension) section of this document.

## DFFE - HighestAddress

This is the higest location in RAM usable by user programs. Nobody should go past this address. If you are writing extentions to the monitor, it's okay to load before the address and then adjust this down to keep others from stomping on your extension.

If your program modifies this value, it needs to set it back before terminating.

# Sample Extension

;=====================================================

; A sample extension for the Extended KIM monitor.

; This is a very simple example of how to write an

; extension (adding a new command) for the

; Extended KIM monitor.

;

; How can you test this? Easy. First, use the "?"

; command in the extended monitor and verify the

; "Z" command is not listed, then load the binary

; version of this file. Do "?" again and you'll see

; the new command has been added and can be used.

;

; 12/26/2015 - Bob Applegate, bob@corshamtech.com

; 09/29/2021 - Bob Applegate

; Minor cleanup

;

; Consider buying a KIM-1 expansion board or a

; KIM Clone computer from us:

;

; www.corshamtech.com

;

;=====================================================

;

; First, define some common ASCII characters

;

LF equ $0a

CR equ $0d

;

include "xkim.inc"

;

; There are more vectors but I didn't need them

;

;=====================================================

; The actual sample

;

code

org ExtensionAddr

;

; Set up the pointer to our sample extension...

;

dw Extension

;

; This is the table of commands being added. Each

; entry has exactly five bytes:

;

; Single character command

; Address of code for this command

; Descriptive text for this command

;

; After the last entry, the next byte must be zero

; to indicate the end of the table.

;

org $0400

Extension db 'Z' ;adding the 'Z' command

dw zCode ;pointer to code

dw zHelp ;pointer to help

;

db 0 ;END OF EXTENSIONS

;

; The descriptive text...

;

zHelp db "Z ........... Describe a zoo",0

;

; And the actual code...

;

zCode jsr putsil ;call display function

db CR,LF

db "A Zoo is a place with "

db "lots of animals."

db CR,LF,0

jmp extKim ;return to Extended KIM