CUBIX

SYSTEM

PROGRAMMING

and

REFERENCE

Revised: 06-Jul-91

Dunfield Development Systems

----------------------------

High quality tools for

Embedded Development

at low prices.

http://www.dunfield.com

Copyright 1983-2005 Dave Dunfield

All rights Reserved

CUBIX System Programing Guide

TABLE OF CONTENTS

Page

1. INTRODUCTION 1

2. DISK AND FILE SYSTEM DESCRIPTION 1

2.1 Drive Control Blocks 1

2.2 Disk Layout 2

2.3 Directory Structure 2

2.4 Sector Link Map Structure 3

3. PROGRAM LINKAGE 4

3.1 Program Initiation 4

3.2 Program Termination 4

4. SYSTEM SERVICE REQUESTS 5

4.1 SSR Macros 5

4.2 Description of system requests 5

4.3 Dos File Specification Pattern 15

5. DOS RETURN CODES 15

6. FILE CONTROL BLOCKS 16

CUBIX System Programing Guide Page: 1

1. INTRODUCTION

This document describes the internal operation of the CUBIX disk

operating system, and the facilities available to application

programs and utilities.

Detailed discriptions are provided for the disk, directory, and

sector link map formats, as well as descriptions of the appliction

program interface, and the functions which are available through it.

2. DISK AND FILE SYSTEM DESCRIPTION

The floppy disk used by CUBIX is formatted to contain a number of

512 byte sectors. The number of sectors on a given disk is equal to

the number of cylinders on the disk, multiplied by the number of

sides (heads) used, multiplied by the number of sectors on each

track.

2.1 Drive Control Blocks

Each disk drive on the system has its own drive control block,

which indicates the status of that particular drive. The format of

the drive control block is as follows:

ORG 0 Offset ZERO within DCB

DCB EQU \* Start of Drive Control Block

DRIVE RMB 1 Physical device ID of drive.

NCYL RMB 1 Number of CYLINDERS on disk

NHEAD RMB 1 Number of HEADS used on disk

NSEC RMB 1 Number of SECTORS/TRACK on disk

CYL RMB 1 CYLINDER to READ/WRITE

HEAD RMB 1 HEAD to READ/WRITE

SEC RMB 1 SECTOR to READ/WRITE

The DRIVE location indicates to the system, the physical drive

ID which is associated with this logical drive. It is used by the

lowest level disk drivers to determine which physical drive to

access.

The NCYL, NHEAD, and NSEC locations describe to the system the

exact format of the disk in the drive. These locations are used by

the system to determine the size (capacity) of the drive, and the

logical sector number to CYLINDER, HEAD, and SECTOR mapping.

The CYL, HEAD, and SEC locations are used by the lowest level

disk drivers in the system to determine where on the disk an

access is to take place. These locations must be set up by any

routine calling the low level drivers.

CUBIX System Programing Guide Page: 2

2.2 Disk Layout

The disks are logically divided into the following areas:

Sector(s) Contents

--------------------------------------

0 Root directory sector

1-<x> Sector link map

<x+1>-<n> File & directory data space

The highest sector number <n> is equal to the number of sectors

on the disk minus one. The highest link sector <x> is calculated

by the dividing <n> by 256, and adding 1.

The size of a disk (SECTORS, HEADS, and CYLINDERS) is

determined by the type of drive used, and the formatting of the

disk. This information is stored in the drive control block for

each particular drive.

2.3 Directory Structure

Each sector of the directory contains 16 file directory

entries, each of which is 32 bytes in size. The format of the

directory entry is defined in the following structure:

ORG 0 Offset ZERO within entry

DIRENT EQU \* Beginning of directory entry

DPREFIX RMB 8 File's DIRECTORY PREFIX

FNAME RMB 8 File's FILENAME

FTYPE RMB 3 File's FILETYPE

DSKADR RMB 2 Address of first sector in file

LODADR RMB 2 File's LOAD ADDRESS

FPROT RMB 1 File's PROTECTION BITS

SPARE RMB 8 Available to user applications

When a DIRECTORY PREFIX, FILENAME, or FILETYPE is shorter than

it's maximum length, it is padded with zero's on the right.

When a directory entry is not used (contains no file entry),

then the first byte of the directory prefix is set to zero. Thus

is is not possible to have a file with a zero-length directory

prefix.

The LOAD ADDRESS field contains a RAM address which is

associated with the file. For '.EXE' files, this is the address at

which the object code is loaded and executed. For files which are

not executable, this is simply the address at which the file will

be loaded in response to a 'LOAD' command if the address is not

explicitly given.

The file protection bits used in the 'FPROT' field are defined

as follows:

RPERM EQU %10000000 READ permission

WPERM EQU %01000000 WRITE permission

EPERM EQU %00100000 EXECUTE permission

DPERM EQU %00010000 DELETE permission

CUBIX System Programing Guide Page: 3

The directory may be extended up to the available size of the

disk, and is linked together using the SECTOR LINK MAP, in the

same way as all files on the disk are linked.

2.4 Sector Link Map Structure

The sector link map has a 16 bit (two bytes) entry for every

sector on the disk.

Each entry of the sector link map contains the allocation

information about the sector it represents.

If the contents of a sector link map entry is $0000, then the

sector which corresponds to that entry is free, and may be

allocated to a file.

If the contents of a sector link map entry is $FFFF, the the

sector which corresponds to that entry is allocated to a file, and

is the last sector in that file.

If the contents of a sector link map entry is neither $0000 nor

$FFFF, then the sector which corresponds to that entry is

allocated to a file, and the value of the entry indicates the next

sector which is allocated to that file.

Since the directory entry contains the pointer to the first

sector in the file, and each sector link map entry indicates the

next sector, the position on the disk of the entire file can be

determined.

The disk directory is linked to its root sector zero (0), in

the same manner as a file is linked to its starting sector.

Using this approach, disk drives of up to 65535 sectors (32

Meg) may be accommodated by the file system.

CUBIX System Programing Guide Page: 4

3. PROGRAM LINKAGE

3.1 Program Initiation

When a program is executed under the CUBIX operating system, it

is loaded into ram at the address specified by the LOAD ADDRESS

field in it's directory entry.

The 6809 'B' register will contain the drive number from which

the program was loaded.

The 6809 'X' register will point to the first free memory

location following the loaded image.

The 6809 'Y' register is set to point to the first non-blank

character in the command line following the program name. If this

character is a carriage return or zero byte, indicating that no

parameters were supplied, then the 6809 'Z' flag will be set. The

actual character found will be passed in the 'A' register.

NOTE: The parameters to a program are contained in the DOS input

buffer, and will be destroyed if any System Service Requests

(SSR's) are executed which allow entry of data or commands

into the DOS input buffer.

Once the program is loaded, and the parameters are set up, the

DOS calls the load address of the file, transferring control to

offset zero within the loaded image. The stack pointer is

unchanged, and if called directly from the DOS, will point to the

DOS internal stack, which contains enough extra space to be used

by small utilitiy programs.

3.2 Program Termination

Under the CUBIX operating system, a program may terminate in

one of two possible ways.

If the stack has not been changed, the program may simply

execute an 'RTS' instruction, to return to the caller.

NOTE: This method of returning will return to ANY caller, this is

usually the DOS, but may be another program which has

executed this program via a DOS command. 'RTS' should only

be used by small utilities, that do not require or destroy

any memory other that it's own image. The top 3K (3072 byte)

block of memory (below the 1K used by CUBIX) is reserved for

such small utilities.

A program may use dos call number 0, to return to DOS command

mode. This method of termination will always return to the DOS,

and should be used by any program which is not a small utility, to

prevent returning to a calling program after it's image has been

destroyed by memory usage from this program.

In either case, the 6809 'Z' flag must be set if the program

completed successfully, and is to return a zero return code. An

easy way to do this is with the 6809 'CLRA' instruction.

CUBIX System Programing Guide Page: 5

In order for a program to return a specific value for it's

return code, it must clear the 'Z' flag, and return the value in

the A accumulator. In order to avoid ambiguity, the value returned

should never be 0 (zero). An easy way to do this is with the 6809

'LDA' instruction.

NOTE: In order that small utility can return values to their

caller in the 'A' register, The 'Z' flag indicates a zero

return code, reguardless of the actual contents of the

accumulator.

4. SYSTEM SERVICE REQUESTS

The DOS contains a built in interface, which allows requests to be

made by application programs. The request is handled by the DOS, and

a return code is returned to the the application, using the same

return code conventions described under the 'Program Termination'

section.

The system interface is handled by the 6809 'SWI' (Software

Interrupt) instruction, and is followed by a function code byte,

indicating the request.

The resident assembler 'ASM' supports a SSR directive, which can

be used to generate individual System Service Requests within an

assembly language program.

4.1 SSR Macros

The following is an example of a MACRO used with Dunfield

Development Systems XASM cross assembler package to simulate the

'SSR' directive. Similar macros can be constructed with any macro

assembler.

\*

\* MACRO TO IMPLEMENT 'SSR' FUNCTION

\*

SSR MACRO

\0 SWI GENERATE INTERRUPT TO OPERATING SYSTEM

FCB \1 ACTUAL FUNCTION CODE REQUESTED

ENDMAC

4.2 Description of system requests

Following is a description of the currently implemented system

requests. Requests obtaining information for the command line

(pointed to by Y) will accept a space, a carraige return, or a

null character ($00) for a terminator.

Unless otherwise stated, each request follows the return code

convention of setting the Z flag if success, and clearing the Z

flag, and returning a return code value in A if failure.

If an unimplemented DOS request is issued, a message is printed

indicating the value of the request, and the address of the

offending SSR instruction. DOS command mode is then entered.

CUBIX System Programing Guide Page: 6

SSR|Registers| Description of function.

# |Modified.|

---+---------+-----------------------------------------------------------

0 | N/A | Terminates the program and returns directly to the DOS.

---+---------+-----------------------------------------------------------

1 | A,B,Y | Prompts with '\*' and waits for a line of input from the

| | Console device, which is buffered within the DOS. On exit,

| | 'Y' points to the start of the buffered line, and 'B' has

| | The length of the line (excluding the carriage return).

---+---------+-----------------------------------------------------------

2 | A,B,Y | As above except Line-feed, Carriage-return is output prior

| | To the prompt.

---+---------+-----------------------------------------------------------

3 | A,B,Y | As SSR 1 except no '\*' prompt is issued.

---+---------+-----------------------------------------------------------

4 | A,Y | Advances 'Y' to the first non-blank character, and sets

| | the 'Z' flag if that character is a carriage-return, or a

| | NULL ($00). The character found is returned in 'A'

---+---------+-----------------------------------------------------------

5 | A,Y | Gets the character pointed to by 'Y', sets the 'Z' flag

| | if it is one of a space, carriage-return, or NULL ($00),

| | and returns the character in 'A'. 'Y' is advanced by one

| | if the character was not a carriage-return or NULL.

---+---------+-----------------------------------------------------------

6 | A,B,X,Y | Gets a 16 bit decimal number from the line pointed to by

| | 'Y', and returns it in the 'X' register. The lower 8 bits

| | of the number are also returned in 'B'. 'Y' is advanced to

| | the next non-blank character following the number.

---+---------+-----------------------------------------------------------

7 | A,B,X,Y | Gets a 16 bit hexidecimal number from the line pointed to

| | by 'Y', and returns it in the 'X' register. The lower 8

| | bits of the number are also returned in the 'B'. 'Y' is

| | advanced to point to the next non-blank after the number.

---+---------+-----------------------------------------------------------

8 | A,B,X,Y | Gets a 16 bit number from the line pointed to by 'Y', in

| | the same way as SSR 6 or 7 above. If the number begins

| | with '$', a hexidecimal number is expected, otherwise a

| | decimal number is assumed.

---+---------+-----------------------------------------------------------

9 | A,B,X,Y | Gets and buffers within the DOS a file specification

| | pattern from the line pointed to by 'Y'. On exit, 'Y'

| | is advanced past the pattern, and 'X' points to the start

| | of the directory name portion of the saved pattern which

| | has been converted to directory entry format.

---+---------+-----------------------------------------------------------

10 | A,B,X,Y | As above except that a DOS error (1) is generated if the

| | specification pattern does not specify a single file.

| | (I.E: It contains wildcard characters)

---+---------+-----------------------------------------------------------

11 | A,B,X,Y | As SSR 9, except that no FILETYPE is expected, on exit 'X'

| | Points to the FILETYPE portion of the saved pattern, which

| | must be set to a three character type.

---+---------+-----------------------------------------------------------

12 | A,B,X,Y | As SSR 11, except that the directory prefix defaults to

| | the SYSTEM directory, not the DEFAULT directory.

---+---------+-----------------------------------------------------------

CUBIX System Programing Guide Page: 7

SSR|Registers| Description of function.

# |Modified.|

---+---------+-----------------------------------------------------------

13 | A,B,X,Y | As SSR 11, except that the directory prefix defaults to

| | the directory prefix pointed to by the 'X' register.

---+---------+-----------------------------------------------------------

14 | A,B,X,Y | Gets and buffers within the DOS a directory prefix name

| | only. On exit, 'X' points to FILENAME portion of DOS

| | buffer, allowing it to be updated.

---+---------+-----------------------------------------------------------

15 | A,B,X,Y | As SSR 14, except that the directory prefix defaults to

| | the directory prefix pointed to by the 'X' register.

---+---------+-----------------------------------------------------------

16 | A,B,Y | Gets a drive specifier from the line pointed to by 'Y',

| | and returns the drive id number (zero origin) in 'A'.

---+---------+-----------------------------------------------------------

17 | A,Y | Gets a file attribute bit mask from the character string

| | ('RWED' etc.) pointed to by 'Y', and returns it in 'A'.

---+---------+-----------------------------------------------------------

18 | A,B,X,Y | Looks up word(Y) in table(X), returning the index of the

| | word (zero origin) in 'B'. If word is not found, the value

| | returned is equal to the number of words in the table.

| | A word is defined as any string of characters ending with

| | one of space, carraige-return, '/', '=', or NULL ($00).

| | Each entry in the table must begin with a flag byte, which

| | has the high bit set, and the number contained in bits 0-7

| | indicates the minimum number of characters which must be

| | supplied for a match. Following this flag byte is the

| | actual characters for the word, which ends with the next

| | flag byte. The table ends with a flag byte of ($80).

---+---------+-----------------------------------------------------------

19 | NONE | Compares the file specification pattern previously saved

| | in the DOS to the directory format filename pointed to by

| | the 'X' register. Sets the 'Z' flag if a match occurs.

---+---------+-----------------------------------------------------------

20 | NONE | Sets the 'Z' flag if the saved file specification pattern

| | is valid as a single file name, and does not contain any

| | wildcard characters.

---+---------+-----------------------------------------------------------

21 | NONE | Outputs a space character to the console device.

---+---------+-----------------------------------------------------------

22 | NONE | Outputs a line-feed and carriage-return to the console.

---+---------+-----------------------------------------------------------

23 | X | Outputs ZERO terminated string(X) to the console device.

| | 'X' is left pointing to character after terminating ZERO.

---+---------+-----------------------------------------------------------

24 | NONE | Outputs ZERO terminated string(PC) to the console device.

| | String is inline with code, and execution continues with

| | the instruction following the ZERO terminating the string.

---+---------+-----------------------------------------------------------

25 | NONE | As SSR 24 above, except that a line-feed, carriage-return

| | is output following the string.

---+---------+-----------------------------------------------------------

CUBIX System Programing Guide Page: 8

SSR|Registers| Description of function.

# |Modified.|

---+---------+-----------------------------------------------------------

26 | NONE | Outputs the 16 bit number(D) to the console in decimal.

---+---------+-----------------------------------------------------------

27 | NONE | Outputs the 16 bit number(D) to the console in hex.

---+---------+-----------------------------------------------------------

28 | NONE | Outputs the 8 bit number(A) to the console in hex.

---+---------+-----------------------------------------------------------

29 | A | Outputs the lower nibble(A) to the console in hex.

---+---------+-----------------------------------------------------------

30 | NONE | Display the file specification pattern previously saved in

| | the DOS, on the console device, as a full file name.

---+---------+-----------------------------------------------------------

31 | NONE | Displays the directory format file name pointed to by 'X'

| | on the console device as a full file name.

---+---------+-----------------------------------------------------------

32 | A | Displays the table entry indicated by the index in 'A', of

| | the table(X), on the console device. Table format is the

| | same as for SSR 18.

---+---------+-----------------------------------------------------------

33 | NONE | Outputs the character in 'A' to the console device.

---+---------+-----------------------------------------------------------

34 | A | Reads a character from the console device, returned in 'A'

---+---------+-----------------------------------------------------------

35 | A | Test for a character from the console device. If a char is

| | ready, then the 'Z' flag is set, and the character is read

| | and returned in 'A'. If no character is found, 'Z' will be

| | clear, and 'A' will contain $FF.

---+---------+-----------------------------------------------------------

36 | NONE | Writes the character in 'A' to device(B).

---+---------+-----------------------------------------------------------

37 | A | Reads a character from device(B), returned in 'A'.

---+---------+-----------------------------------------------------------

38 | A | Test for a character from device(B). If a char is ready,

| | the 'Z' flag will be set, and 'A' will contain the char.

| | If no character is found, 'Z' will be clear, and 'A' will

| | contain $FF.

---+---------+-----------------------------------------------------------

39 | A | Returns in 'A' the device number of the currently active

| | console input device.

---+---------+-----------------------------------------------------------

40 | A | Returns in 'A' the device number of the currently active

| | console output device.

---+---------+-----------------------------------------------------------

41 | A | Sets the active console input device to the device number

| | passed in 'A'. On exit, 'A' contains the previous console

| | input device number.

---+---------+-----------------------------------------------------------

42 | A | Sets the active console output device to the device number

| | passed in 'A'. On exit, 'A' contains the previous console

| | output device number.

---+---------+-----------------------------------------------------------

43 | A | This SSR outputs the canned DOS error message:

| | 'Operand missing or invalid'

| | And returns with a return code of (1).

---+---------+-----------------------------------------------------------

CUBIX System Programing Guide Page: 9

SSR|Registers| Description of function.

# |Modified.|

---+---------+-----------------------------------------------------------

44 | A | This SSR outputs the canned DOS error message:

| | 'Error processing file: <file specification>'

| | 'File not found'

| | Where <file specification> is the previously saved (in DOS)

| | file specification displayed as a full file name.

| | And returns with a return code of (2).

---+---------+-----------------------------------------------------------

45 | A | This SSR outputs the canned DOS error message:

| | 'Error processing file: <file specification>'

| | 'File protection violation'

| | Where <file specification> is the previously saved (in DOS)

| | file specification displayed as a full file name.

| | And returns with a return code of (3).

---+---------+-----------------------------------------------------------

46 | A | This SSR outputs the canned DOS error message:

| | 'File not open for read'

| | And returns with a return code of (4).

---+---------+-----------------------------------------------------------

47 | A | This SSR outputs the canned DOS error message:

| | 'File not open for write'

| | And returns with a return code of (5).

---+---------+-----------------------------------------------------------

48 | A | This SSR outputs the canned DOS error message:

| | 'File already exists'

| | And returns with a return code of (8).

---+---------+-----------------------------------------------------------

49 | A | This SSR outputs the canned DOS error message:

| | 'Insufficent disk space'

| | And returns with a return code of (9).

---+---------+-----------------------------------------------------------

50 | A | This SSR outputs the canned DOS error message:

| | 'Invalid device''

| | And returns with a return code of (11).

---+---------+-----------------------------------------------------------

51 | A | This SSR outputs the canned DOS error message:

| | 'Download format error'

| | And returns with a return code of (12).

---+---------+-----------------------------------------------------------

52 | NONE | This SSR outputs the canned DOS error message:

| | 'Error processing file: <file specification>'

| | Where <file specification> is the previously saved (in DOS)

| | file specification displayed as a full file name.

| | The ZERO terminated error message(Y) is then output.

---+---------+-----------------------------------------------------------

53 | A,B,X | Loads the entire file named by the previously saved DOS

| | file specification into memory at the address in the 'X'

| | register. On exit, 'X' points to the first free memory

| | location following the loaded file. (This location will be

| | a multiple of 512 bytes beyond the first address).

---+---------+-----------------------------------------------------------

CUBIX System Programing Guide Page: 10

SSR|Registers| Description of function.

# |Modified.|

---+---------+-----------------------------------------------------------

54 | A,B,X | Saves the contents of memory pointed to by 'X' for the

| | number of 512 byte blocks in 'D' to the file named by the

| | previously saved DOS file specification pattern.

---+---------+-----------------------------------------------------------

55 | A | Opens the file named by the previously saved DOS file

| | specification for read. The 'U' register must point to an

| | empty file control block which will be filled in.

---+---------+-----------------------------------------------------------

56 | A | Opens the file named by the previously saved DOS file

| | specification for write. The 'U' register must point to an

| | empty file control block which will be filled in.

---+---------+-----------------------------------------------------------

57 | A | Closes an open file(U). If the file was being written, the

| | last block in the file is padded with $FF's and written to

| | disk, then the file linkage chain is closed off.

---+---------+-----------------------------------------------------------

58 | A | Reads a 512 byte block from the open file indicated by the

| | file control block(U), and places that block in memory at

| | the address in the 'X' register.

---+---------+-----------------------------------------------------------

59 | A | Reads a single character from the open file indicated by

| | the file control block(U), and returns it in 'A'.

---+---------+-----------------------------------------------------------

60 | A | Writes a 512 byte block from memory(X) to the file

| | indicated by the file control block(U).

---+---------+-----------------------------------------------------------

61 | A | Writes a single character passed in 'A' to the file

| | indicated by the file control block(U).

---+---------+-----------------------------------------------------------

62 | NONE | Rewinds the file indicated by the file control block(U),

| | so that subsequent reads or writes will occur at the

| | beginning of the file.

---+---------+-----------------------------------------------------------

63 | NONE | Seeks forward relative within the file indicated by the

| | file control block(U), by the number of bytes in 'D'.

---+---------+-----------------------------------------------------------

64 | NONE | Seeks to the absolute byte location in 'D' within the file

| | indicated by the file control block(U).

---+---------+-----------------------------------------------------------

65 | A,B | Returns in 'D' the current byte location within the file

| | indicated by the file control block(U).

---+---------+-----------------------------------------------------------

66 | A | Suspends reading or writing the file indicated by the file

| | control block(U). Following this SSR, eight (8) bytes will

| | have been placed on the stack, for later use by SSR 67.

---+---------+-----------------------------------------------------------

67 | A | Resumes a previously suspended file, by reading the eight

| | (8) bytes which were stacked, and restoring the file

| | control block which must be pointed to by 'U'.

---+---------+-----------------------------------------------------------

CUBIX System Programing Guide Page: 11

SSR|Registers| Description of function.

# |Modified.|

---+---------+-----------------------------------------------------------

68 | A,B,X | Locates the file indicated by the previously saved DOS

| | file specification in the directory on the currently

| | selected disk drive. On exit, 'D' contains the sector

| | number of the directory sector containing the entry, and

| | 'X' points to a copy of the entry in the DOS work buffer.

---+---------+-----------------------------------------------------------

69 | A,B,X | As above, except that an error message is issued if the

| | file is not found.

---+---------+-----------------------------------------------------------

70 | A,B,X | As SSR 69 except that file must have READ permission, or

| | an error occurs.

---+---------+-----------------------------------------------------------

71 | A,B,X | As SSR 69 except that file must have WRITE permission or

| | an error occurs. If the file does not exist, it will be

| | created, and given a single free disk sector.

---+---------+-----------------------------------------------------------

72 | A,B,X | Creates a file named with the previously saved DOS file

| | specification, and gives it a single free disk sector. On

| | exit, 'D' contains the number of the sector given to the

| | file, and 'X' points to a copy of the newly created

| | directory entry in the DOS work buffer.

---+---------+-----------------------------------------------------------

73 | A,B,X | Deletes all files on the currently selected disk which

| | have names which match the previously saved DOS file

| | specification pattern. If the pattern contains wildcards,

| | and hence could indicate more that one file, each matching

| | filename is prompted for, as with the DELETE command.

---+---------+-----------------------------------------------------------

74 | NONE | Sets the current DEFAULT directory prefix to the directory

| | prefix and drive contained within the previously saved DOS

| | file specification pattern.

---+---------+-----------------------------------------------------------

75 | NONE | Sets the current SYSTEM directory prefix to the directory

| | prefix and drive contained within the previously saved DOS

| | file specification pattern.

---+---------+-----------------------------------------------------------

76 | NONE | Sets the disk drive which will be used for all subsequent

| | file system operations to the drive index passed in 'A'.

---+---------+-----------------------------------------------------------

77 | A,B,X | Locates the disk sector link for the sector number passed

| | in 'D', and returns it in 'D'. The 'Z' flag will be set

| | if the link if $FFFF, indicating end of file. The 'X'

| | register will be left pointing to the sector link within

| | a ram copy of the disk link table in the DOS work buffer.

---+---------+-----------------------------------------------------------

78 | A,B,X | Loads the chain of disk sectors, starting with the sector

| | number passed in 'D', into memory at the address in 'X'.

| | On exit, 'X' points to the first free memory location

| | following the loaded sectors.

---+---------+-----------------------------------------------------------

79 | A,B | Allocates a free disk sector on the currently selected

| | drive, and returns it's number in 'D'.

---+---------+-----------------------------------------------------------

CUBIX System Programing Guide Page: 12

SSR|Registers| Description of function.

# |Modified.|

---+---------+-----------------------------------------------------------

80 | A,B,X | Releases (deallocates) a sector chain from the currently

| | selected drive, starting with the sector passed in 'D'.

| | On exit, 'X' points to the sector link entry for the last

| | sector in the chain. The 'D' register will contain $FFFF

| | if the chain ended normally, or $0000 if an un-allocated

| | sector was found in the chain.

---+---------+-----------------------------------------------------------

81 | A,B | Returns in 'D' the size in sectors of the disk drive

| | corresponding to the index passed in the 'A' register.

---+---------+-----------------------------------------------------------

82 | A,B | Returns in 'D' the size in sectors of the currently

| | selected disk drive.

---+---------+-----------------------------------------------------------

83 | A,B | Returns in 'D' the number of free sectors remaining on the

| | currently selected disk drive.

---+---------+-----------------------------------------------------------

84 | X | Reads the disk sector passed in 'D' into the DOS internal

| | work buffer. The old buffer will be re-written to disk if

| | it had been marked as modified. On exit, the 'X' register

| | points to the beginning of the buffer.

---+---------+-----------------------------------------------------------

85 | NONE | Marks the DOS internal work buffer as having been

| | modified, insuring that it will be written to disk the

| | next time a new buffer is read, or the disk is updated.

---+---------+-----------------------------------------------------------

86 | NONE | Writes the DOS internal work buffer back to disk if it has

| | been marked as modified.

---+---------+-----------------------------------------------------------

87 | NONE | Writes the DOS internal work buffer to disk regardless of

| | whether or not it has been marked as modified.

---+---------+-----------------------------------------------------------

88 | NONE | Writes the DOS internal work buffer to disk if it has been

| | marked as modified, and clears the buffer, causing the DOS

| | to update it from disk the next time it is used.

---+---------+-----------------------------------------------------------

89 | A,B,U | Returns in both 'D' and 'U' the address of the disk drive

| | control block for the drive index passed in 'A'.

---+---------+-----------------------------------------------------------

90 | A,B,U | Returns in both 'D' and 'U' the address of the disk drive

| | control block for the currently selected drive.

---+---------+-----------------------------------------------------------

91 | NONE | Sets up the disk control block(U) with the CYL, HEAD, and

| | and SECTOR values required to access the sector number

| | passed in the 'D' register.

---+---------+-----------------------------------------------------------

92 | NONE | Reads the sector(D) from the currently selected disk drive,

| | and places it in memory at the address in 'X'.

---+---------+-----------------------------------------------------------

93 | NONE | Writes the sector(D) on the currently selected disk drive,

| | from memory at the address in 'X'.

---+---------+-----------------------------------------------------------

CUBIX System Programing Guide Page: 13

SSR|Registers| Description of function.

# |Modified.|

---+---------+-----------------------------------------------------------

94 | A,B,X | Displays the directory using the previously saved DOS file

| | specification pattern as a mask.

---+---------+-----------------------------------------------------------

95 | \*ALL\* | Re-Initializes all hardware devices installed in the DOS,

| | restores the DOS tables and vectors to the default values.

---+---------+-----------------------------------------------------------

96 | A,B,X,Y | Restores (homes) the head on the currently selected drive.

---+---------+-----------------------------------------------------------

97 | A,B,X,Y | Reads the sector indicated by the current CYL, HEAD, and

| | SECTOR setting in the drive control block(U), from the

| | currently selected drive, and places it in memory at the

| | address passed in the 'X' register.

---+---------+-----------------------------------------------------------

98 | A,B,X,Y | Writes the sector indicated by the current CYL, HEAD, and

| | SECTOR settings in the control block(U), to the currently

| | selected disk drive, from memory at the address passed in

| | the 'X' register.

---+---------+-----------------------------------------------------------

99 | A,B,X,Y | Formats the currently selected disk drive using the NCYL,

| | NHEAD, and NSECTOR value from the disk control block(U)

| | The interleave factor is passed in 'A'. This is a physical

| | format only, the file system is not initialized.

---+---------+-----------------------------------------------------------

100| \*ALL\* | Executes the DOS command pointed to by the 'Y' register.

---+---------+-----------------------------------------------------------

101| \*ALL\* | Enters dos command mode, allowing dos commands to be input

| | and executed. Returns to the instruction following SSR 101

| | when the 'RETURN' command is entered. Any program which is

| | run during this time and returns to dos using SSR 0. Will

| | terminate the calling program, making DOS entry permanent.

| | NOTE: The DOS uses its own stack, which it resets when

| | command mode is entered. Any program issuing this

| | SSR must have its own stack, and not rely on the DOS

| | stack space available when it was invoked.

| | Use of this SSR will cancel any command file which is

| | executing.

---+---------+-----------------------------------------------------------

102| A,B | Returns in 'D' the dos device vector indicated by the

| | index passed in the 'A' register. Device vectors are:

| | 0-7 = Serial device 0-7 INPUT drivers.

| | 8-15 = Serial device 0-7 OUTPUT drivers.

| | 16 = Disk HOME head routine.

| | 17 = Disk READ SECTOR routine.

| | 18 = Disk WRITE SECTOR routine.

| | 19 = Disk FORMAT routine.

| | 20 = SWI vector. (Used by DOS for SSR interface).

| | 21 = SWI2 vector.

| | 22 = SWI3 vector.

| | 23 = IRQ vector.

| | 24 = FIRQ vector.

| | 25 = NMI vector.

---+---------+-----------------------------------------------------------

CUBIX System Programing Guide Page: 14

SSR|Registers| Description of function.

# |Modified.|

---+---------+-----------------------------------------------------------

103| A,B | Sets the dos device vector indicated by the index in 'A'

| | to the address passed in the 'X' register. The old vector

| | address is returned in 'D'. See SSR 102 for vectors.

---+---------+-----------------------------------------------------------

104| A | Sets (enables) the dos function indicated by the index in

| | the 'A' register. The functions are:

| | 0 = Error message output

| | 1 = Debug output of SSR's executed.

| | 2 = Command file tracing.

| | On exit, 'A' contains the old flag setting (0=disabled).

---+---------+-----------------------------------------------------------

105| A | Clears (disables) the dos function indicated by the index

| | in the 'A' register. See SSR 105 for function numbers.

| | On exit, 'A' contains the old flag setting (0=disabled).

---+---------+-----------------------------------------------------------

106| A,Y | Sets the command file parameters for a command file which

| | is currently executing to the string pointed to by the 'Y'

| | register. The string may be terminated by either a ZERO

| | byte or a carriage return character.

---+---------+-----------------------------------------------------------

107| A,B | Performs unsigned 16 bit multiply of 'D' by 'X'. The

| | result (truncated to 16 bits) is returned in 'D'.

---+---------+-----------------------------------------------------------

108| A,B,X | Performs unsigned 16 bit division of 'X' by 'D'.The result

| | is placed in 'X', and the remainder is placed in 'D'.

---+---------+-----------------------------------------------------------

109| NONE | Displays the contents of all the processor registers on

| | the console device.

---+---------+-----------------------------------------------------------

110| A,B,X | Downloads a file to memory in motorola hexidecimal format

| | from the device indicated by the index passed in 'A'.

---+---------+-----------------------------------------------------------

'\*ALL\*' indicates SSR's which cannot guarantee the integrity of

any processor registers when they return.

CUBIX System Programing Guide Page: 15

4.3 Dos File Specification Pattern

Some the System Service Requests make use of the DOS Saved File

Specification Pattern. This file specification is contained within

DOS, and once set up by the apprioprate SSR's, informs DOS of the

file specification pattern to be used for subsequent operations

requiring filenames.

The Saved File Specificaton Pattern is the following format:

ORG 0 Offset ZERO within FSP

FSP EQU \* File Spec. Pattern Starts Here

FSDRIVE RMB 1 Drive Index (0-3)

FSDIR RMB 8 DIRECTORY PREFIX

FSNAME RMB 8 FILENAME

FSTYPE RMB 3 FILETYPE

With the exception of the FSDRIVE field, the DOS saved file

specification pattern is in exactly the same format as the file

specification portion of an entry in the disk directory.

For consistency with directory entries, the SSR's which return

pointers to the saved file specification pattern return a pointer

to the FSDIR entry, with the FSDRIVE entry being referenced at an

offset of -1 from that pointer. This allows the DIRCTORY PREFIX,

FILENAME, and FILETYPE to be referenced at the samed offsets into

either a directory entry of a file specification pattern.

5. DOS RETURN CODES

The following return codes are generated by DOS commands, Utility

programs, and System Service Requests.

0 - Success, command, program, or system request worked.

1 - Operand to command or SSR was invalid or not supplied.

2 - Requested file does not exist on the disk.

3 - Operation failed because file is protected.

4 - Attempt to read file not open for read.

5 - Attempt to write file not open for write.

6 - End of file encountered while reading.

7 - Seek beyond end of file.

8 - Attempt to create a file failed because the file

already exists on the disk.

9 - Attempt to allocate a disk sector failed because there

is no available free space on the disk.

10 - Physical disk error occured while reading or writing

to the disk.

11 - Attempt to access a serial device for which no device

driver is installed.

12 - Attempt to download a file from a device failed because

of a format error in the file.

255 - Command was not recognized as a valid DOS command.

CUBIX System Programing Guide Page: 16

6. FILE CONTROL BLOCKS

When a file is opened via a system service request, the DOS uses

its FILE CONTROL block to maintain its state until it is closed. This

file control block is an area of memory supplied by the user issuing

the OPEN request, and is pointed to in all cases by the 6809 'U'

register.

The format of the file control block is as follows:

ORG 0 Offset ZERO within FCB

FCB EQU \* Begining of control block

OPEN RMB 1 Type of OPEN (1=READ, 2=WRITE)

FDRIVE RMB 1 Drive index where file is located

FIRST RMB 2 First sector in file

SECTOR RMB 2 Sector being accessed

LASTSEC RMB 2 Last sector accessed

OFFSET RMB 2 Offset into character buffer

CHRBUF RMB 512 Buffer for character access

CHRBUF is used for CHARACTER accesses only, this space is not

required if the file is only being BLOCK accessed. OFFSET will always

be zero for BLOCK accessed files.

Once a file has been accessed using the CHARACTER system calls, it

should not be BLOCK accessed until enough characters have been read

or written to set OFFSET back to zero. This will be a multiple of 512

characters.