CUBIX

System Porting

Guide

Revised: 30-Jul-90

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 Porting Guide

TABLE OF CONTENTS

Page

1. INTRODUCTION 1

2. HARDWARE INITIALIZATION 2

2.1 Initializing CUBIX 2

2.2 Initializing Hardware Devices 3

2.3 Initializing the Console 3

3. SERIAL DEVICE DRIVERS 3

3.1 Input Drivers 3

3.2 Output Drivers 3

3.3 NULL Device Driver 3

3.4 Similar Devices 3

4. DISK DEVICE DRIVERS 4

4.1 Home head 4

4.2 Read a sector 4

4.3 Write a sector 4

4.4 Format a disk 4

5. STTY UTILITY 5

6. INTIO UTILITY 5

7. TERMINAL HANDLING 5

8. MAJOR PORTS 6

8.1 Operating System 6

8.2 Applications and Utilities 7

9. PORTING FROM ROM (MINOR PORT) 8

9.1 Porting from an existing system ROM 9

9.2 Applications and Utilities 9

10. SAMPLE I/O DRIVERS 10

CUBIX System Porting Guide Page: 1

1. INTRODUCTION

The CUBIX operating system is designed to be ported at the object

(minor port), or source code (major port) level.

A minor port may be performed whenever the target system conforms

to the standard CUBIX memory map. The development system used must be

capable of linking a ROMable image. This memory map is as follows:

$0000-$1FFF - I/O devices (Incl. memory mapped video etc).

$2000-$DFFF - Random Access Memory.

$E000-$FFFF - CUBIX Operating System ROM

A minor port is accomplished by writing the I/O drivers, and

linking those drivers into the CUBIX image. The system application

and utility programs are provided on a CUBIX format 5.25" disk, and

are ready to use.

A minor port may also be performed from a CUBIX distribution ROM,

as described in the section on ROM porting.

A major port is required whenever the target system does not

conform to the above memory map, or the develoment system used does

not have the capability to link absolute location ROMable images.

A major port is accomplished by writing the I/O drivers, and

including them in the CUBIX operating system source code, which is

then assembled into a ROMable image. The memory addresses used by

CUBIX may be altered at this time.

The system application and utility programs must also be

assembled, to properly locate them in the target system memory map.

In all cases, CUBIX is placed in an 8K (2764 or similar) ROM and

placed at the top of the 6809 memory map (Locations $E000 to $FFFF).

The CHECKSUM of the rom must be stored as a 16 bit value in the first

two rom locations.

The ROM CHECKSUM is calculated as the 16 bit sum of all 8 bit

values from offset $0002 to the end of the rom (Offset $1FFF). This

16 bit checksum must be stored in the rom with the most signifigant

byte at offset $0000, and the least signifigant byte at offset $0001.

CUBIX System Porting Guide Page: 2

2. HARDWARE INITIALIZATION

The I/O drivers must include a routine called HWINIT which

establishes the CUBIX system default parameters, initializes hardware

devices, installs all device drivers, and prepares the console for

the CUBIX startup messages.

2.1 Initializing CUBIX

When HWINIT is called, the 6809 'Y' register is set to point to

an internal table which must be filled in by HWINIT. The format of

this table is as follows:

OSRAM EQU \* Address of OS initialized ram (passed in 'Y').

RMB 7 Default DCB for drive A

RMB 7 Default DCB for drive B

RMB 7 Default DCB for drive C

RMB 7 Default DCB for drive D

RMB 1 Default console input device (0-7)

RMB 1 Default console output device (0-7)

RMB 16 Serial device input driver vectors (8 words)

RMB 16 Serial device output driver vectors (8 words)

RMB 2 Disk HOME HEAD routine.

RMB 2 Disk READ SECTOR routine.

RMB 2 Disk WRITE SECTOR routine.

RMB 2 Disk FORMAT routine.

RMB 2 SWI handler vector address

RMB 2 SWI2 handler vector address

RMB 2 SWI3 handler vector address

RMB 2 IRQ handler vector address

RMB 2 FIRQ handler vector address

RMB 2 NMI handler vector address

RMB 1 Error messages enabled flag

RMB 1 SSR Debug enabled flag

RMB 1 Command file trace enabled flag

RMB 1 Default drive (0-3)

RMB 8 Default directory prefix

RMB 1 System drive (0-3)

RMB 8 System directory prefix

Notes 1) CYL, HEAD, and SEC values of DCB's should be set to zero.

2) Default console input device is usually 1

3) Unused serial device drivers should be set to zero.

4) SWI vector should be set to execute the CUBIX SSR handler,

which is located at address $E00B.

5) Unused SWI2, SWI3, IRQ, FIRQ, NMI vectors should point

to an 'RTI' instruction.

6) Usual settings of ERROR, TRACE and DEBUG are $FF, 0, 0.

($FF = ENABLED, 0 = DISABLED).

7) Drives are offset zero ('A'=0, 'B'=1, 'C'=2, 'D'=3).

8) Unused portions of directory prefix's should be padded with

zeros to the right.

CUBIX System Porting Guide Page: 3

2.2 Initializing Hardware Devices

HWINIT must initialize the hardware in the system, this

includes such things as the disk drive controller, serial devices,

video displays, and any other devices which must be available for

CUBIX to operate.

2.3 Initializing the Console

For systems using a serial terminal as the console, HWINIT

should output a line feed, and carraige return to the console

device to establish the cursor at the left hand margin.

For systems using an integrated memory mapped video display,

HWINIT should perform a clear screen operation, and home the

cursor.

3. SERIAL DEVICE DRIVERS

The serial device drivers installed by HWINIT are free to use the

6809 'B' and 'X' registers without saving them. When the driver is

called, the 'B' register will contain the CUBIX device number

multiplied by two.

3.1 Input Drivers

Input device drivers should test for a character from the

device, and return with the character in the 'A' register, and the

'Z' condition flag set if a character was found. If no character

is found, the 'A' register should be loaded with $FF, and the 'Z'

flag cleared.

3.2 Output Drivers

Output drivers should write the character passed in the 6809

'A' register to the device. It should not return until this has

been done.

3.3 NULL Device Driver

The NULL DEVICE (device 0) drivers are easily implemented as

follows:

RDNULL LDA #$FF INDICATE NO CHARACTER READY

WRNULL RTS RETURN TO CALLER

3.4 Similar Devices

When several similar devices are available, the device drivers

usually load the 6809 'X' register with the address of the device,

and then jump to a common handler.

The standard 'STTY' utility supplied with CUBIX, which

functions with 6551 uarts, examines the first instruction of the

input driver, looking for an 'LDX' instruction, from which it can

determine the address of the device.

CUBIX System Porting Guide Page: 4

If no 'LDX' is found, STTY aborts with the error message:

'Device has fixed comunication format.'

4. DISK DEVICE DRIVERS

The disk drivers installed by HWINIT are free to use the 'A', 'B',

'X' and 'Y' registers without saving them. When a routine is invoked,

the 6809 'U' register will be set to point to the drive control block

(DCB). See the CUBIX system programming guide for a description of

the DCB.

If an operation is unsuccessful, the disk driver should return

with the 'Z' flag cleared, and a value in the 'A' accumulator

indicating the proper error message to display as follows:

0 - 'Disk system error'

1 - 'Disk format error'

2 - 'Bad sector'

3 - 'Sector not found'

4 - 'Disk write protected'

If 'A' is set to any other value, it is treated as zero (0).

The following disk routines are required:

4.1 Home head

The home routine resets the head of the physical drive

indicated in the DCB to track zero.

4.2 Read a sector

The read routine reads sector (512 bytes) indicated by the CYL,

HEAD and SEC entries in the DCB from the indicated physical drive,

and places it in memory at the address passed in the 'X' register.

The routine must perform a seek operation to position the head to

the correct cylinder. If a read error occurs, the routine should

simply return with an error code. The operating system will call

this routine up to five times to retry in case of errors. After

the third retry, the home routine will be called to recalibrate

the head.

4.3 Write a sector

The write routine writes a sector (512 bytes) indicated by the

CYL, HEAD and SEC entries in the DCB to the indicated physical

drive, from memory at the address passed in the 'X' register.

Seeking and retry operation is the same as for the read sector

routine described above.

4.4 Format a disk

The format routine formats the disk in the indicated physical

drive with the number of cylinders, heads, and sectors/track

indicated by the NCYL, NHEAD, and NSEC values in the DCB. The

sector Interleave factor is passed in the 6809 'A' register.

CUBIX System Porting Guide Page: 5

5. STTY UTILITY

The complete source for the standard CUBIX 'STTY' utility is

provided in the EXAMPLES directory. This routine is designed to

function with 6551 type uart devices, and may be modified to handle

the serial devices in the target system.

6. INTIO UTILITY

The complete source for the standard CUBIX 'INTIO' utility is

provided in the EXAMPLES directory. This routine is designed to

function with 6551 type uarts, and assumes a common interrupt line

from all uart IRQ outputs to the 6809 IRQ input. If the standard

serial device drivers do not support interrupt I/O, this utility

should be modified to operate in the hardware environment of the

target system.

7. TERMINAL HANDLING

The TTYPATCH utility is provided to modify the terminal

charactistics of screen oriented programs such as the 'EDT' screen

editor. See the 'TTY' manual, and the 'TTYPATCH' documentation in the

CUBIX users guide for details.

CUBIX System Porting Guide Page: 6

8. MAJOR PORTS

When performing a MAJOR port of the CUBIX system, you will be

provided with "porting" source files for the operating system and its

utilities. The "porting" sources have all comments stripped out, and

all symbols which do not need to be modified as part of the port are

changed to meanless alphabetic sequences.

The "porting" sources have been made purposly difficult to modify,

in order to minimize the alterations to CUBIX, and subsequent non

standard releases. Please resist the temptation, as compatability

must be maintained between CUBIX releases in order for it to be

successful.

8.1 Operating System

The following symbols must be set in the CUBIX "MAIN" source,

to adapt it to the system memory map.

The symbol 'ROM' must be set to the starting address of the

system ROM (8K rom is required). The ROM must control the 6809

vectors which are located at the top of the memory map at address

$FFF2. The usual address of 'ROM' is $E000.

The symbol 'RAM' points to a 1K block of memory which is used

by CUBIX for internal variables, buffers and stack. It is usually

set to the top 1K of Random Access Memory, which is $DC00 in the

standard system.

The symbol 'USRRAM' points to the lowest memory address which

is available for general use. It set to $2000 in the standard

system.

The symbol 'USREND' indicates the highest memory address which

is to be tested by the startup memory test. It is usually set to

RAM+1023, but is available separately in case OS memory is not

contiguous with application memory.

The driver routines should be included in the source file

immediately preceding the 6809 hardware vectors (ORG $FFF2) near

the end of the file.

CUBIX System Porting Guide Page: 7

8.2 Applications and Utilities

All application and utility programs must be assembled to fit

the memory map of the system. A 'macros' file is provided which

must be included at the beginning of each source file. This

provides the 'SSR' macro which implements the CUBIX System Service

Request instruction. This macro must be modified to suit your

particular macro assembler. Also provided in the 'macros' file are

the following symbols used by the application and utility sources.

The symbol 'OSRAM' must be set to the lowest memory address

which is available for general use. Is it set to $2000 in the

standard system

The symbol 'OSEND' must be set to the highest memory address

which is available for general use. It is set to $DBFF in the

standard system.

The symbol 'OSUTIL' indicates the top three Kbytes of general

use memory. This is the location reserved for CUBIX utility

programs which may be executed from within an application. It is

set to $D000 in the standard system.

The following the a sample 'macros' file which is compatable

with the my XASM Development Tools macro processor and assembler

(ASM09).

OSRAM EQU $2000 APPLICATION RAM AREA

OSEND EQU $DBFF END OF GENERAL RAM

OSUTIL EQU $D000 UTILITY ADDRESS SPACE

\*

\* MACRO TO IMPLEMENT THE SSR FUNCTION

\*

\* This is not needed if using CUBIX's own assembler,

\* since it has an internal 'SSR' directive.

\*

SSR MACRO

\0 SWI INTERRUPT TO OS

FCB \1 FUNCTION CODE NUMBER

ENDMAC

CUBIX System Porting Guide Page: 8

9. PORTING FROM ROM (MINOR PORT)

A minor port may be performed directly from a CUBIX distribution

ROM. This ROM contains only the operating system code, without the

I/O drivers.

When performing a system port from a ROM, the distribution rom

should first be copied, and all changes made on the copy. This leaves

the master rom unchanged and safe from accidental damage.

The first few locations in the distribution rom have the following

contents:

E000 FF FF CHKSUM FDB $FFFF CHECKSUM GOES HERE

E002 10 00 MBASE FDB $2000 BASE RAM ADDRESS

E004 10 8E ?? ?? INIT LDY #IRAM POINT TO INITIALIZED RAM

E008 BD XX XX JMP HWINIT EXECUTE USER INIT. ROUTINE

E00B XX XX XX SSR EQU \* SSR HANDLER ROUTINE

The 16 bit value at offset 0 (Address $E000) must be set to the

checksum of the ROM, which is the sixteen bit sum of all ROM memory

locations from $E002 (Offset 2) to $FFFF (Offset 8191). This checksum

must be calculated AFTER the drivers are installed AND any other

modifications to the ROM contents have been are completed.

The 16 bit value at offset 2 (Address $E002) identifies to the

operating system the lowest available RAM location. This is used to

determinate where to begin the memory test, and for the default load

address of CREATE'd files.

The 'JMP' instruction at offset 8 (Address $E008) addresses the

user supplied 'HWINIT' routine. In the distribution ROM, this jumps

to the first free memory location in the ROM following the operating

system code. The address of this location can be determined by

reading the address from the 'JMP' instruction which is at offset 9

and 10 (Address $E009 and $E00A) in the ROM.

The I/O drivers should be assembled and placed into the rom at

this address, with the HWINIT routine being the first routine so that

it will be properly entered when the 'JMP' executes. All remaining

rom space up to the 6809 hardware vectors (Address $FFF2) is

available for use by the driver routines.

Since the operating system accesses all other driver routines only

via the table filled in by 'HWINIT', only 'HWINIT' must be placed at

a known location (As determined earlier).

The 'SWI' vector filled in the the HWINIT routine should be set to

point to the SSR handler which begins at address $E00B.

If the base (lowest) ram location of the target system is not the

CUBIX standard ($2000), the application and utility programs which

load at the bottom of ram will have to be re-generated to reflect the

new load address (Partial MAJOR port).

CUBIX System Porting Guide Page: 9

9.1 Porting from an existing system ROM

The CUBIX system may be ported from any existing CUBIX ROM,

provided that the location of the ROM, and the highest RAM address

are the same in the existing and target systems. This will be

$E000 and $DFFF in most systems.

Ports for systems in which the ROM location or the highest RAM

address differs from the above must be performed by re-generating

the CUBIX image with the new memory map (MAJOR port).

If the base (lowest) RAM location differs from the existing

system to the target system, the application and utility programs

which load at the bottom of ram will have to be re-generated to

reflect the new load address (Partial MAJOR port).

9.2 Applications and Utilities

If you have the CUBIX version of the minor port package, you

will have a CUBIX format disk containing the system applications

and utilities. In this case, the programs are ready to go, and you

may use them directly from the disk.

The HOST version of the CUBIX minor port package does not

include a CUBIX format disk, but instead has the ".HEX" download

files for each of the CUBIX system programs.

Once you have ported the CUBIX rom, you may use it to format a

disk, and download the system software. For information on

FORMATTING and DOWNLOADING, see the CUBIX user guide.

CUBIX System Porting Guide Page: 10

10. SAMPLE I/O DRIVERS

The EAMPLES directory contains a sample source file for a very

simple set of CUBIX device drivers. The drivers are fully functional,

and require minimum hardware support (ie: no interrupts are used

etc).

This system has two 6551 serial uart devices, and a NEC 765 type

floppy disk controller. The 6551 drivers are fully compatable with

the standard STTY and INTIO utilities (Also provided in EXAMPLES).

The two 6551 uarts occur in the 6809 memory map at addresses $0000

and $0100. For simplicity in this example, the serial I/O drivers

operate by software polling the uart. However, the IRQ outputs from

the 6551's would normally be connected in a wired OR configuration to

the 6809 IRQ input, allowing the INTIO utility to used. In a more I/O

intensive system, it may be desirable to directly implement interrupt

driven serial drivers.

The floppy disk controller is a NEC 765 or compatible, which

controls up to 4 standard 40 track single or double sided diskette

drives. Like the serial I/O drivers, all operations are performed by

software polling. The 765 is accessed at address $0200, and its

TERMINAL COUNT pin is pulsed by reading or writing at address $0300.