



# S3C6400/6410 HW Multimedia Codec (MFC) User's Guide

**S3C6400/6410**  
August 29, 2008  
(Preliminary) REV 3.20

# Important Notice

The information in this publication has been carefully checked and is believed to be entirely accurate at the time of publication. Samsung assumes no responsibility, however, for possible errors or omissions, or for any consequences resulting from the use of the information contained herein.

Samsung reserves the right to make changes in its products or product specifications with the intent to improve function or design at any time and without notice and is not required to update this documentation to reflect such changes.

This publication does not convey to a purchaser of semiconductor devices described herein any license under the patent rights of Samsung or others.

Samsung makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Samsung assume any liability arising out of the application or use of any product or circuit and specifically disclaims any and all liability, including without limitation any consequential or incidental damages.

"Typical" parameters can and do vary in different applications. All operating parameters, including "Typicals" must be validated for each customer application by the customer's technical experts.

Samsung products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, for other applications intended to support or sustain life, or for any other application in which the failure of the Samsung product could create a situation where personal injury or death may occur.

Should the Buyer purchase or use a Samsung product for any such unintended or unauthorized application, the Buyer shall indemnify and hold Samsung and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, expenses, and reasonable attorney fees arising out of, either directly or indirectly, any claim of personal injury or death that may be associated with such unintended or unauthorized use, even if such claim alleges that Samsung was negligent regarding the design or manufacture of said product

## S3C6400/6410 RISC Microprocessor User's Guide

Copyright © 2007-2007 Samsung Electronics Co., Ltd.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electric or mechanical, by photocopying, recording, or otherwise, without the prior written consent of Samsung Electronics Co., Ltd.

Samsung Electronics Co., Ltd.  
San #24 Nongseo-Dong, Giheung-Gu  
Yongin-City Gyeonggi-Do, Korea  
446-711

Home Page: <http://www.samsungsemi.com/>

E-Mail: [mobilesol.cs@samsung.com](mailto:mobilesol.cs@samsung.com)

Printed in the Republic of Korea



Preliminary product information describe products that are in development, for which full characterization data and associated errata are not yet available. Specifications and information herein are subject to change without notice.

## Revision History

Revision No	Description of Change	Refer to	Author(s)	Date
1.00	Initial draft(refer to WinCE v5.4)	-	Simon Chun	Aug. 13, 2007
2.10	Linux version document		Jiun Yu	Aug. 31, 2007
2.11	Fig 2-3 is added		Jiun Yu	Sep. 1, 2007
2.20	loctl command is added		Jiun Yu	Oct. 15, 2007
2.21			Jiun Yu	Dec. 21, 2007
2.30	Rotation function is added		Jiun Yu	Jan. 24, 2008
2.40	Encoding options are added		Jiun Yu	Mar. 31, 2008
3.00	Supporting S3C6400 and 6410		Jiun Yu	April 14, 2008
3.10	Supporting Hybrid divx decoder		Jiun Yu	July 5, 2008
3.20	Tested on s3c-linux-2.6.24 kernel		Jiun Yu	August 29, 2008

## Contents

<b>1</b>	<b>INTRODUCTION .....</b>	<b>1</b>
1.1	PURPOSE.....	1
1.2	SCOPE.....	1
1.3	INTENDED AUDIENCE .....	1
1.4	SUPPORTED HW & SW .....	1
1.5	DEFINITIONS, ACRONYMS, AND ABBREVIATIONS .....	1
1.6	REFERENCES.....	1
<b>2</b>	<b>INSTALLATION GUIDE.....</b>	<b>2</b>
2.1	DIRECTORY STRUCTURE.....	2
2.1.1	<i>Driver Source Code Structure .....</i>	<i>2</i>
2.1.2	<i>API Test application Source Code Structure .....</i>	<i>2</i>
2.2	INSTALLATION STEP (LINUX BSP) .....	3
2.2.1	<i>Kernel configuration and building.....</i>	<i>3</i>
2.2.2	<i>Module compilation .....</i>	<i>6</i>
2.2.3	<i>Test application compilation.....</i>	<i>7</i>
2.2.4	<i>Download module and test application(At target side) .....</i>	<i>9</i>
2.2.5	<i>Loading module and executing test application .....</i>	<i>9</i>
2.3	MEMORY CONFIGURATION.....	9
2.3.1	<i>Configuring the MFC Buffer Address.....</i>	<i>10</i>
2.3.2	<i>Required Memory Sized Calculation .....</i>	<i>12</i>
<b>3</b>	<b>FILE I/O OPERATION .....</b>	<b>15</b>
3.1	LINUX DEVICE DRIVER'S FILE I/O OPERATION .....	15
3.1.1	<i>open .....</i>	<i>15</i>
3.1.2	<i>ioctl .....</i>	<i>16</i>
3.1.3	<i>mmap .....</i>	<i>16</i>
3.1.4	<i>close.....</i>	<i>16</i>
3.2	CONTROL CODES FOR IOCTL() .....	17
3.3	DATA STRUCTURE FOR PASSING THE IOCTL ARGUMENTS .....	19
3.3.1	<i>MFC_ENC_INIT_ARG.....</i>	<i>19</i>
3.3.2	<i>MFC_ENC_EXE_ARG.....</i>	<i>20</i>
3.3.3	<i>MFC_DEC_INIT_ARG.....</i>	<i>20</i>
3.3.4	<i>MFC_DEC_EXE_ARG.....</i>	<i>20</i>
3.3.5	<i>MFC_GET_BUF_ADDR_ARG .....</i>	<i>20</i>
3.3.6	<i>MFC_GET_CONFIG_ARG.....</i>	<i>21</i>
3.3.7	<i>MFC_SET_CONFIG_ARG.....</i>	<i>21</i>
3.4	OBTAINABLE PARAMETERS FOR MFC.....	21
3.5	CONFIGURABLE PARAMETERS FOR MFC.....	21
3.5.1	<i>MFC_SET_CONFIG_DEC_ROTATE.....</i>	<i>22</i>
3.5.2	<i>MFC_SET_CONFIG_ENC_H263_PARAM.....</i>	<i>23</i>
3.5.3	<i>MFC_SET_CONFIG_ENC_SLICE_MODE.....</i>	<i>23</i>
3.5.4	<i>MFC_SET_CONFIG_ENC_PARAM_CHANGE.....</i>	<i>24</i>
3.5.5	<i>MFC_SET_CONFIG_ENC_CUR_PIC_OPT.....</i>	<i>24</i>
<b>4</b>	<b>ANNEX A (H.264 DECODER USAGE).....</b>	<b>26</b>

## Figures

Fig 2-1 SW Layer of MFC Driver Source Code .....	2
Fig 2-2 MFC Test applications using File I/O .....	
Fig 2-3 MFC Test application using API .....	3
Fig 2-4 Setup reserved memory layout .....	3
Fig 2-5 Select "Boot option" .....	4
Fig 2-6 Select 3 <sup>rd</sup> line .....	5
Fig 2-7 Confirm "Default kernel command string" .....	5
Fig 2-8 Memory Layout in RAM for MFC .....	10
Fig 2-9 Size of BITPROC_BUF .....	12
Fig 2-10 Size of STRM_BUF with LINE_BUF and RING_BUF not-shared (Example) .....	13
Fig 2-11 Size of STRM_BUF with LINE_BUF and RING_BUF shared (Example) .....	13
Fig 2-12 Size of FRAM_BUF (Example) .....	14

Tables

Table 2-1 Directory Structure of MFC Driver and Test application Source Code .....	2
Table 2-2 MFC's Buffer Description .....	10
Table 2-3 Required Memory Size (1 or 2 SD images) .....	14
Table 2-4 Required Memory Size (1 or 2 CIF images).....	14
Table 3-1 Configurable parameter types for IOCTL_MFC_SET_CONFIG.....	22

# 1 Introduction

## 1.1 Purpose

This document is prepared for the purpose of describing the S3C6400/6410 HW codec (MFCv1.0) device driver's API so that users can implement their multimedia application easily.

## 1.2 Scope

The scope of this document is to describe

- How to call the device driver's API to decode/encode.
- Usage example of Decoder

## 1.3 Intended Audience

Intended Audience	Tick whenever Applicable
Project Manager	Yes
Project Leader	Yes
Project Team Member	Yes
Test Engineer	Yes

## 1.4 Supported HW & SW

Intended Audience	Tick whenever Applicable
HW	Samsung S3C6400/6410 HW Multimedia Codec
OS	Linux 2.6.16 and 2.6.24

## 1.5 Definitions, Acronyms, and Abbreviations

Abbreviations	Description
MFC	Multi-Format Codec (HW codec in S3C6400/6410 Samsung AP)

## 1.6 References

Number	Reference	Description
1	S3C6400 Datasheet	S3C6400 Datasheet
2	S3C6400WM60MfcDriver_UserManual_REV5.60_20071228	UserManual of the MFC Windows Mobile device driver

## 2 Installation Guide

### 2.1 Directory Structure

FIMV_MFC_V1.0	
Doc	Documents(API specification, User's guide, release note)
Mfc_app/File_Operation	MFC test application using File I/O
Mfc_app/API	MFC test application using API made by Samsung
Mfc_drv	MFC Linux device driver source code

Table - Directory Structure of MFC Driver and Test application Source Code

#### 2.1.1 Driver Source Code Structure

Mfc driver source code was separated into three layers which is shown in .

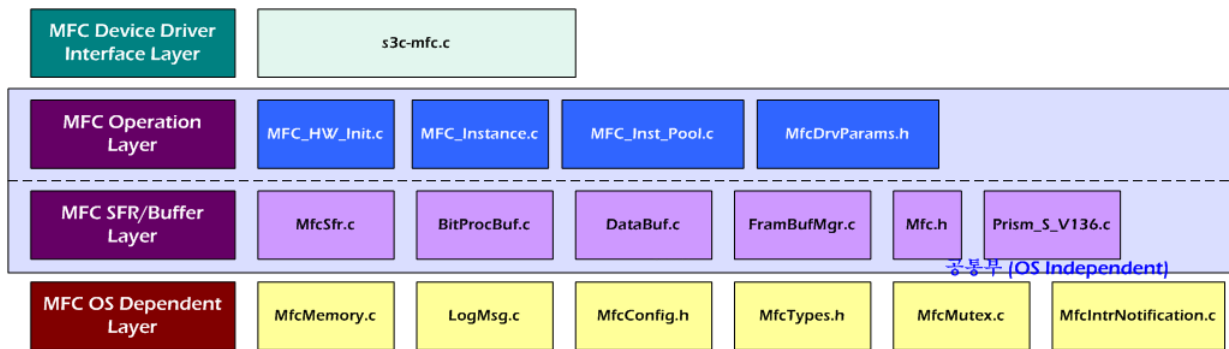


Fig - SW Layer of MFC Driver Source Code

- [MFC Device Driver Interface Layer]** It is the layer for the Linux OS device driver interface to be communicated with VFS. This layer is implementing the Linux character driver's API specification.
- [MFC Operation Layer]** It is implementing the operation logics for letting the MFC HW work.
- [MFC SFR/Buffer Layer]** This layer is for handling components such as MFC SFR, Bit Processor's buffer, etc.
- [MFC OS System Layer]** It is an abstraction layer for the OS system call.

#### 2.1.2 API Test application Source Code Structure

Fig 2-3 shows source code structure of the MFC test application using API. Blue boxes are API Layer.



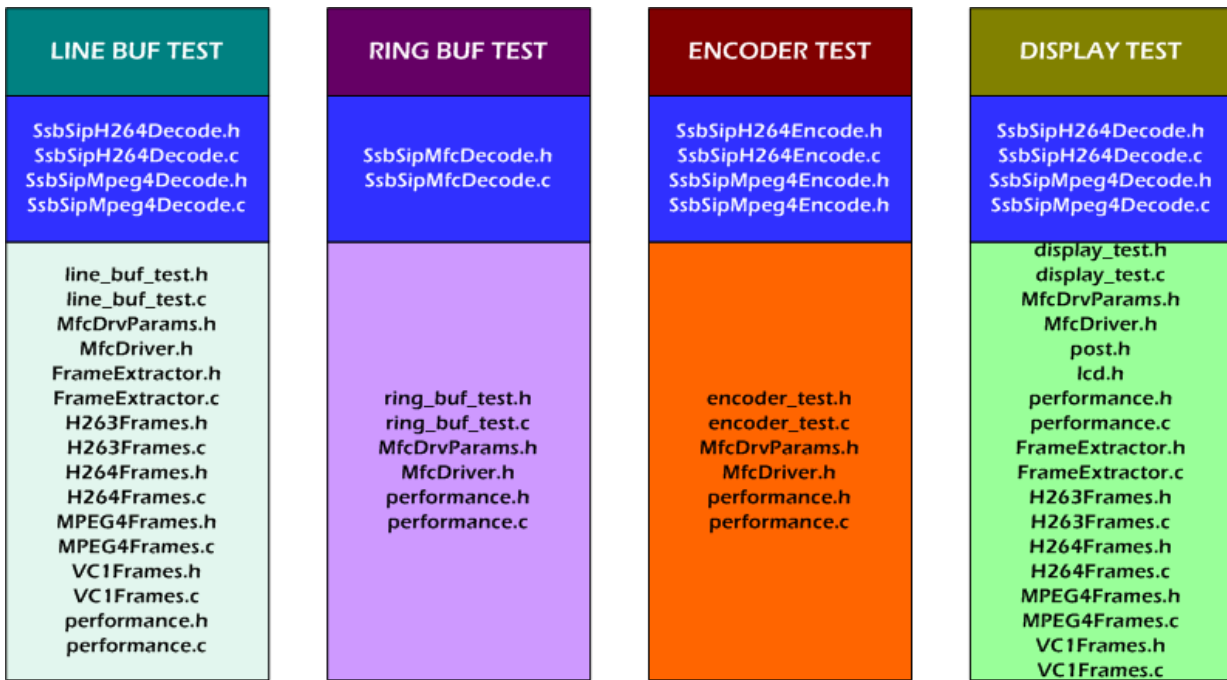


Fig - MFC Test application using API

## 2.2 Installation Step (Linux BSP)

### 2.2.1 Kernel configuration and building

Some devices need reserved memory because they have to allocate physically continuous memory. So You must setup reserved memory layout using below file named "reserved\_mem.h" before kernel compilation.

```
#ifndef _ASM_ARM_ARCH_RESERVED_MEM_H
#define _ASM_ARM_ARCH_RESERVED_MEM_H

/*
 * Default reserved memory size
 * MFC      : 6 MB
 * Post     : 8 MB
 * JPEG     : 8 MB
 * Camera   : 15 MB
 * These sizes can be modified
 */

// #define CONFIG_RESERVED_MEM_JPEG
// #define CONFIG_RESERVED_MEM_JPEG_POST
#define CONFIG_RESERVED_MEM_MFC
// #define CONFIG_RESERVED_MEM_MFC_POST
// #define CONFIG_RESERVED_MEM_JPEG_MFC_POST
// #define CONFIG_RESERVED_MEM_CAMERA
// #define CONFIG_RESERVED_MEM_JPEG_CAMERA
// #define CONFIG_RESERVED_MEM_JPEG_POST_CAMERA
// #define CONFIG_RESERVED_MEM_MFC_CAMERA
// #define CONFIG_RESERVED_MEM_MFC_POST_CAMERA
// #define CONFIG_RESERVED_MEM_JPEG_MFC_POST_CAMERA
```

Fig - Setup reserved memory layout

**[NOTE]****This file is in**

include/asm/arch-s3c64xx/reserved\_mem.h (Linux2.6.16)

include/asm/arch-s3c2410/reserved mem.h (Linux2.6.21 and 2.6.24)

And then you must confirm "Boot option" in menuconfig using below procedure.

```
#cd [installed directory]/s3c-linux-2.6.24
```

```
[root@localhost s3c-linux-2.6.24]# make smdk6410mtd_defconfig
```

```
[root@localhost s3c-linux-2.6.24]# make menuconfig
```

```
.config - Linux Kernel v2.6.24.2 Configuration
Linux Kernel Configuration
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing
<Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for
Search. Legend: [*] built-in [ ] excluded <M> module < > module capable

  General setup --->
[*] Enable loadable module support --->
-*- Enable the block layer --->
  System Type --->
  Bus support --->
  Kernel Features --->
  Boot options --->
  Floating point emulation --->
  Userspace binary formats --->
  Power management options --->
  Networking --->
  Device Drivers --->
  File systems --->
[*] Instrumentation Support --->
  Kernel hacking --->
  Security options --->
  [ ] Cryptographic API --->
  Library routines --->
---
  Load an Alternate Configuration File
  Save an Alternate Configuration File

<Select>  < Exit >  < Help >
```

Fig - Select "Boot option"

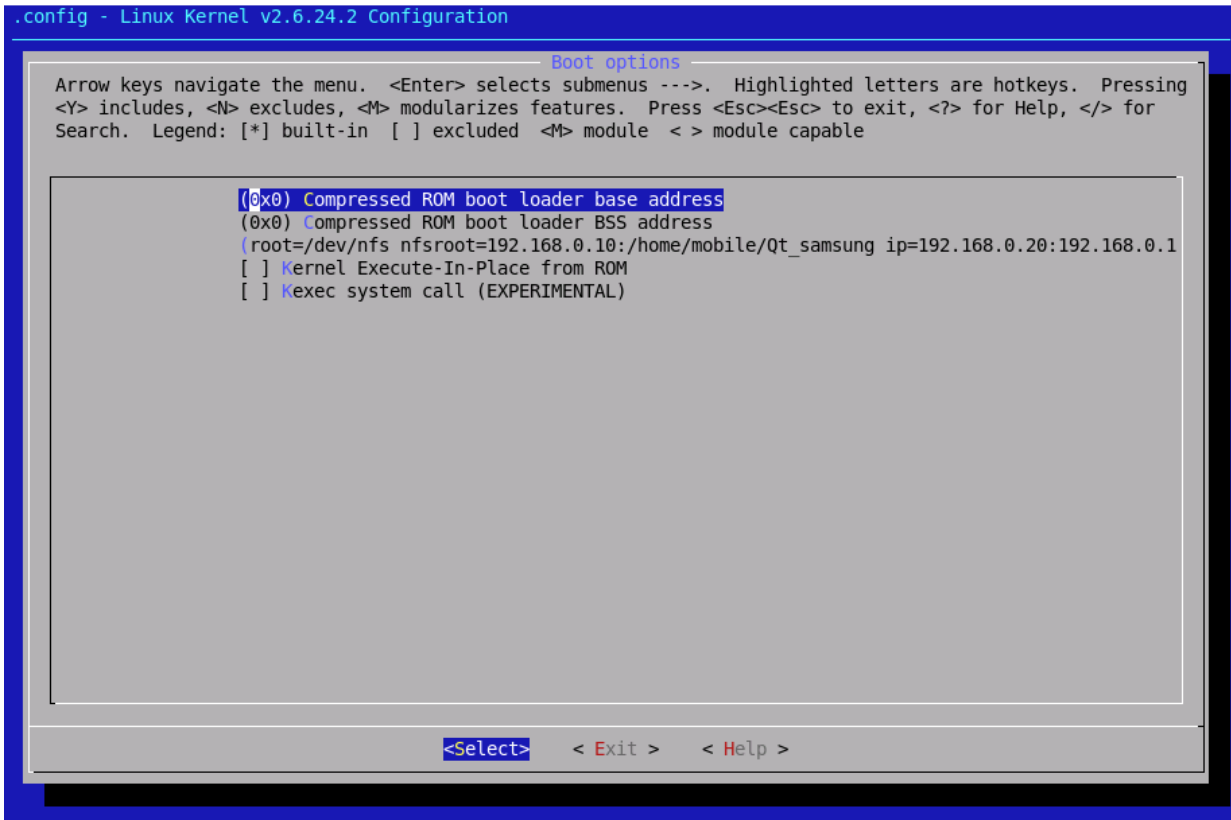


Fig - Select 3<sup>rd</sup> line

If there is "mem=128M" in "Default kernel command string", please erase because you already modify reserved memory layout previous step.

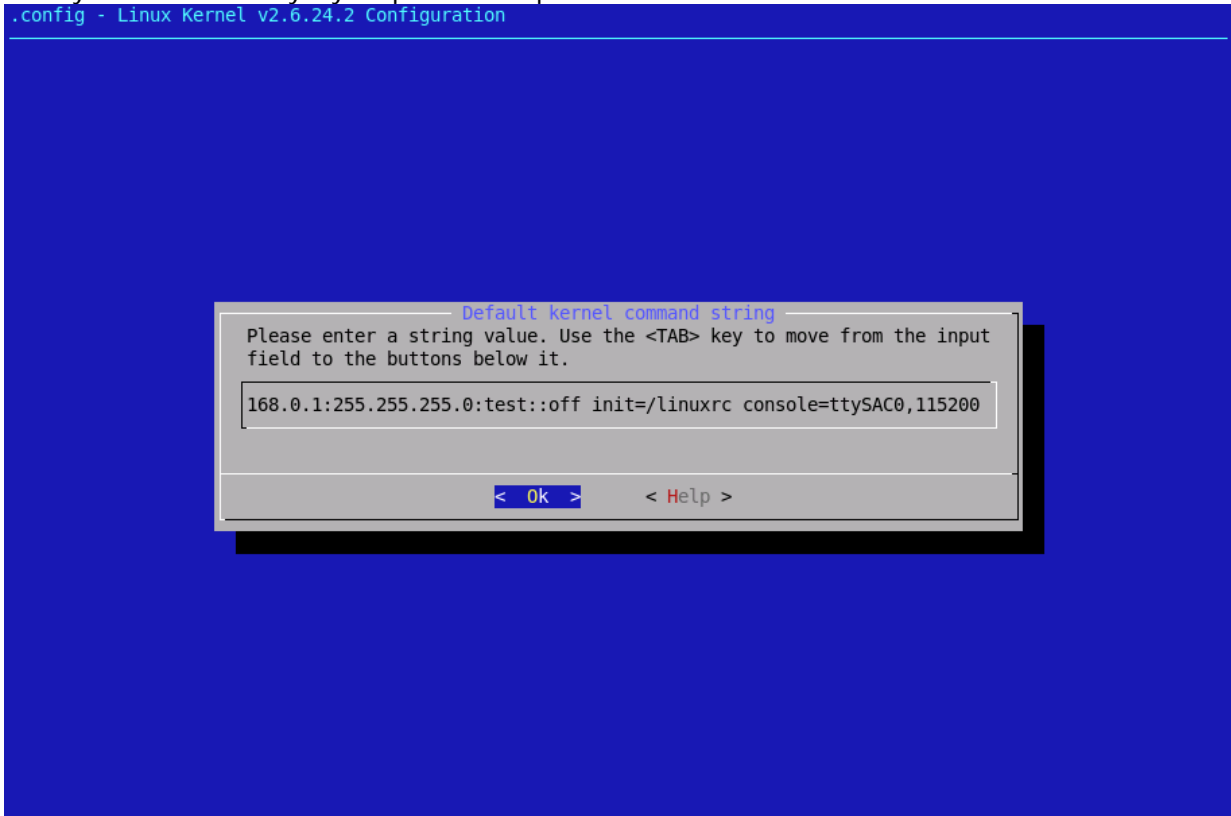


Fig - Confirm "Default kernel command string"

Save and exit in "menuconfig"

```
[root@localhost s3c-linux-2.6.24]# make clean

[root@localhost s3c-linux-2.6.24]# make

[root@localhost s3c-linux-2.6.24]# cp arch/arm/boot/zImage /tftpboot
```

When "zImage" is ready in the directory "/tftpboot", then you can run "minicom" to download it to the target board.

#### [NOTE]

For detailed information about how to build Linux kernel and how to download kernel image and cramfs, please refer to related porting guide documents.

## 2.2.2 Module compilation

MFC device driver should be compiled as a kernel module.

### 2.2.2.1 Makefile

You should modify "Makefile" to set configuration according to your environment.

```
[root@localhost s3c-linux-2.6.24]# cd [module directory]/Multimedia_DD/FIMV_MFC_V1.0/mfc_drv

[root@localhost mfc_drv]# vi Makefile
```

```
#####
# Makefile for MFC Driver
# 2007 (C) Samsung Electronics
# Author : Jiun. Yu <jiun.yu@samsung.com>
#####

#where the kernel sources are located
KERNEL_DIR := ../../../../ s3c-linux-2.6.24

CFLAGS_MODULE += -DLINUX
CFLAGS_MODULE += -DDIVX_ENABLE

obj-m          := s3c_mfc.o

s3c_mfc-y := Prism_0503.o BitProcBuf.o DataBuf.o FramBufMgr.o \
          LogMsg.o MFC_HW_Init.o MFC_Inst_Pool.o MFC_Instance.o MfcMemory.o
MfcMutex.o MfcSfr.o \
          s3c-mfc.o MfcIntrNotification.o MfcSetConfig.o

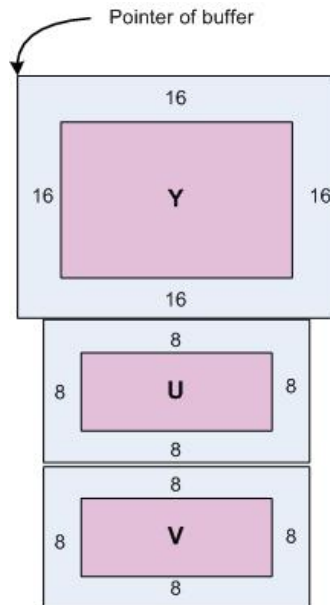
PWD            := $(shell pwd)

here:
    (cd $(KERNEL_DIR); make SUBDIRS=$(PWD) modules)

clean:
    rm -rf *.ko
    rm -rf *.mod.*
```

```
rm -rf *.cmd
rm -rf *.o
rm -rf Module.*
```

In case of enabling DIVX\_TEST macro, MFC's output buffer's scheme is changed like below. Below buffer scheme is used for Hybrid divx decoder.



### 2.2.2.2 Compile

After compilation, you can find newly created files. "s3c\_mfc.ko" file is module of MFC device driver.

```
[root@localhost mfc_drv]# make
```

### 2.2.3 Test application compilation

In this sub chapter, you should know how to compile and how to use test application. It is included encoder and decoder test

First, you should modify the "Makefile". There are several macro. you should enable macro according to your development environment.

```
[root@localhost mfc_drv]# cd ../mfc_app/API
```

```
[root@localhost API]# vi Makefile
```

```
CC = /usr/local/arm/4.2.2-eabi/usr/bin/arm-linux-gcc
```

```
#[Definitions]
```

```
#FPS : performance measurement. It doesn't make the output file
```

```
#ROTATE_ENABLE : Rotation mode enable when testing line buffer and ring buffer.
```

```
# and if you want to test rotation, you must modify definition as "#define
```

```
# MFC_ROTATE_ENABLE 1" in MfcConfig.h file
```

```
#RGB24BPP : display as 24bpp. default is 16bpp
```

```
#DIVX_ENABLE : test for hybrid divx decoder
```

```
CFLAGS = -Wall -Os
```

```
CSRCS = ./MPEG4Frames.c
```

```
./H264Frames.c
```

```
./VC1Frames.c
```

```
./H263Frames.c
```

```

./FrameExtractor.c      \
./line_buf_test.c      \
./ring_buf_test.c      \
./display_test.c       \
./display_optimization1.c \
./display_optimization2.c \
./encoder_test.c       \
./SsbSipH264Decode.c   \
./SsbSipMpeg4Decode.c   \
./SsbSipVC1Decode.c    \
./SsbSipMfcDecode.c    \
./SsbSipH264Encode.c   \
./SsbSipMpeg4Encode.c  \
./SsbSipLogMsg.c       \
./performance.c        \
./FileRead.c           \
./demo.c               \
./test.c

```

```
OBJS = $(CSRCS:.c=.o)
```

```
.SUFFIXES: .c.o
```

```
all: mfc
```

```
mfc: $(OBJS)
      $(CC) $(CFLAGS) -g -o $@ $(OBJS) -lpthread
```

```
.c.o:
      $(CC) $(CFLAGS) -g -c -o $@ $<
```

```
clean:
      rm -f mfc $(OBJS)
```

Second, you should decide to test encoder or decoder.

```
[root@localhost mfc_drv]# cd ../mfc_app/API
```

```
[root@localhost API]# vi test.c
```

Modify "test.c" file. You can select encoder or decoder test

```

#include "line_buf_test.h"
#include "ring_buf_test.h"
#include "encoder_test.h"
#include "display_test.h"
#include "demo.h"
#include "display_optimization1.h"
#include "display_optimization2.h"

int main(int argc, char **argv)
{
    Test_H263_Decoder_Line_Buffer(argc, argv);
    //Test_H264_Decoder_Line_Buffer(argc, argv);
    //Test_MPEG4_Decoder_Line_Buffer(argc, argv);
    //Test_VC1_Decoder_Line_Buffer(argc, argv);
    //Test_Decoder_Ring_Buffer(argc, argv);
    //Test_Display_H264(argc, argv);
    //Test_Display_MPEG4(argc, argv);
    //Test_Display_H263(argc, argv);

```

```
//Test_Display_VC1(argc, argv);
//Test_H264_Encoder(argc, argv);
//Test_MPEG4_Encoder(argc, argv);
//Test_H263_Encoder(argc, argv);
//Demo(argc, argv);
//Test_Display_Optimization1(argc, argv);
//Test_Display_Optimization2(argc, argv);

return 0;
}
```

Third, you should compile the test application.

```
[root@localhost mfc_app]# make
```

After compilation, you can find newly created files. "mfc" is executable file.

### 2.2.4 Download module and test application(At target side)

Use "rz" command in root file system. This command enables you can transfer test application from host PC to target board through uart console by zmodem.

```
/tmp $ rz <ENTER>
```

After downloading completely, you need to change access right of this application. Use "chmod" command. If you are using NFS root file system, you just copy these created files to your NFS root directory.

### 2.2.5 Loading module and executing test application

```
/ $ insmod s3c_mfc.ko

S3C6400 MFC Driver, (c) 2007 Samsung Electronics

S3C6400 MFC driver module init OK.

./mfc [input filename] [output filename]
```

## 2.3 Memory Configuration

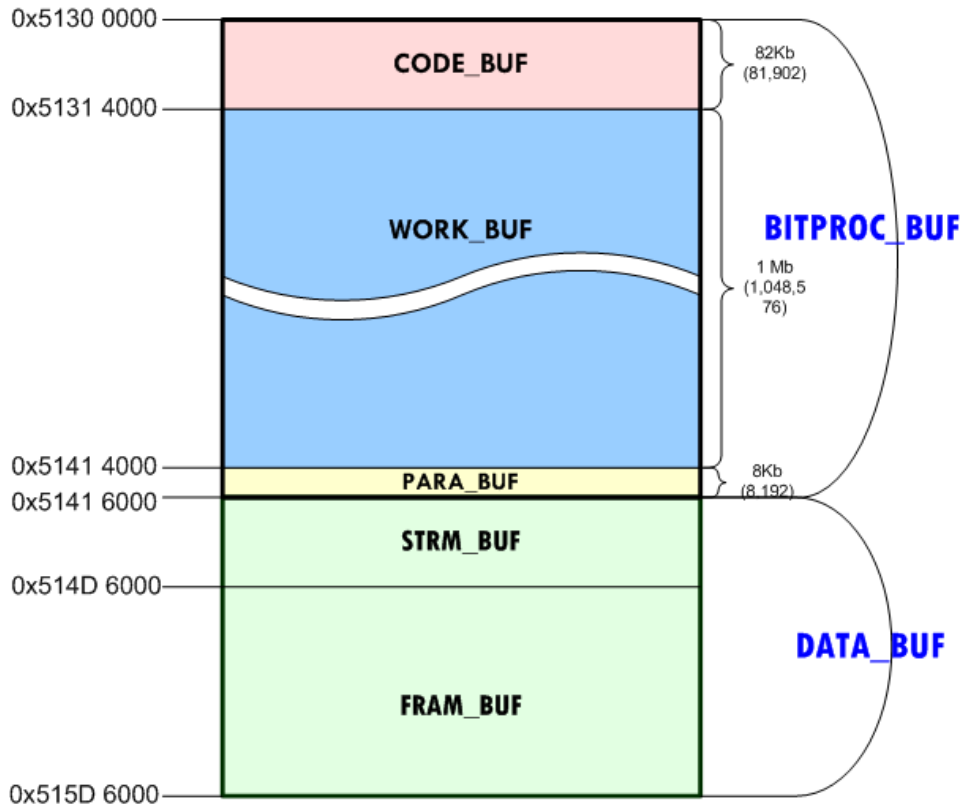


Fig - Memory Layout in RAM for MFC

There are two memory region to be reserved for the MFC to operate. One is BITPROC\_BUF and its size is fixed as 1,138,688 bytes. The other is DATA\_BUF, the size for this buffer is application-dependent.

BUF name	Description	BUF size (Bytes)	
<b>BITPROC_BUF</b>	Reserved area for the BITPROCESSOR(MFC's internal processor) Once the MFC is started, the address cannot be changed.	<b>1,138,688</b>	
	<b>CODE_BUF</b>	BITPROCESSOR's F/W code	81,902
	<b>WORK_BUF</b>	BITPROCESSOR's working buffer	1,048,576
	<b>PARA_BUF</b>	Parameters on issuing command to MFC	8,192
<b>DATA_BUF</b>	Input/Output buffer for encoding and decoding		
	<b>STRM_BUF</b>	Buffer for the compressed video stream.	<b>①</b>
	<b>FRAM_BUF</b>	Buffer for the YUV420 frame	<b>②</b>

Table - MFC's Buffer Description

BITPROC\_BUF is reserved for the BITPROCESSOR and the size is fixed as 1,138,688 bytes. DATA\_BUF consists of STRM\_BUF and FRAM\_BUF for input and output buffer. In the STRM\_BUF area, we allocates the LINE\_BUF

### 2.3.1 Configuring the MFC Buffer Address



FIMV\_MFC\_V1.0/mfc\_drv/MfcConfig.h

DEFINE	Description
S3C6400_BASEADDR_MFC_SFR	<ul style="list-style-type: none"> <li>Base address of MFC SFR</li> <li>[Value = 0x7e002000] is fixed. (Refer to S3C6400 Datasheet.)</li> </ul>
S3C6400_BASEADDR_MFC_BITPROC_BUF	<ul style="list-style-type: none"> <li>Base address of MFC BITPROC_BUF</li> <li>Value is in RAM region</li> </ul>
S3C6400_BASEADDR_MFC_DATA_BUF	<ul style="list-style-type: none"> <li>Base address of MFC DATA_BUF</li> <li>Value is in RAM region (Better if it is consecutive to BITPROC_BUF.)</li> </ul>

DEFINE	Description
MFC_CODE_BUF_SIZE	<ul style="list-style-type: none"> <li>Size of CODE_BUF</li> <li>[Value = 81920] is fixed.</li> </ul>
MFC_WORK_BUF_SIZE	<ul style="list-style-type: none"> <li>Size of WORK_BUF</li> <li>[Value = 1048576] is fixed.</li> </ul>
MFC_PARA_BUF_SIZE	<ul style="list-style-type: none"> <li>Size of PARA_BUF</li> <li>[Value = 8192] is fixed.</li> </ul>
MFC_BITPROC_BUF_SIZE	<ul style="list-style-type: none"> <li>Total size of BITPROC_BUF</li> <li>Value = MFC_CODE_BUF_SIZE + MFC_WORK_BUF_SIZE + MFC_PARA_BUF_SIZE</li> </ul>

DEFINE	Description
MFC_NUM_INSTANCES_MAX	<ul style="list-style-type: none"> <li>Maximum number of MFC instances</li> <li>Value = 1 ~ 8</li> </ul>
MFC_LINE_RING_SHARE	<ul style="list-style-type: none"> <li>LINE_BUF &amp; RING_BUF are shared?</li> <li>Value = 0 or 1</li> </ul>

DEFINE	Description
MFC_LINE_BUF_SIZE_PER_INSTANCE	<ul style="list-style-type: none"> <li>Size of LINE_BUF per instance</li> <li>[Recommended Value for VGA = 200 * 1024]</li> <li>[Recommended Value for QVGA = 100 * 1024]</li> </ul>
MFC_LINE_BUF_SIZE	<ul style="list-style-type: none"> <li>MFC_LINE_BUF_SIZE_PER_INSTANCE * MFC_NUM_INSTANCES_MAX</li> </ul>
MFC_RING_BUF_SIZE	<ul style="list-style-type: none"> <li>Size of RING_BUF</li> <li>[Recommended Value = 256000 * 3]</li> </ul>
MFC_FRAM_BUF_SIZE	<ul style="list-style-type: none"> <li>Size of FRAM_BUF</li> <li>[Recommended Value = 720*480*3*4]</li> </ul>

MFC_RING_BUF_PARTUNIT_SIZE	<ul style="list-style-type: none"> <li>• Size of one PART of RING_BUF</li> <li>• MFC_RING_BUF_SIZE / 3</li> </ul>
MFC_STRM_BUF_SIZE	<ul style="list-style-type: none"> <li>• MFC_LINE_BUF_SIZE + MFC_RING_BUF_SIZE</li> </ul>
MFC_DATA_BUF_SIZE	<ul style="list-style-type: none"> <li>• MFC_STRM_BUF_SIZE + MFC_FRAM_BUF_SIZE</li> </ul>

### 2.3.2 Required Memory Sized Calculation

In this section, how to calculate the memory will be explained step by step.

① BITPROC\_BUF size is fixed. (size = 1,138,688 bytes)

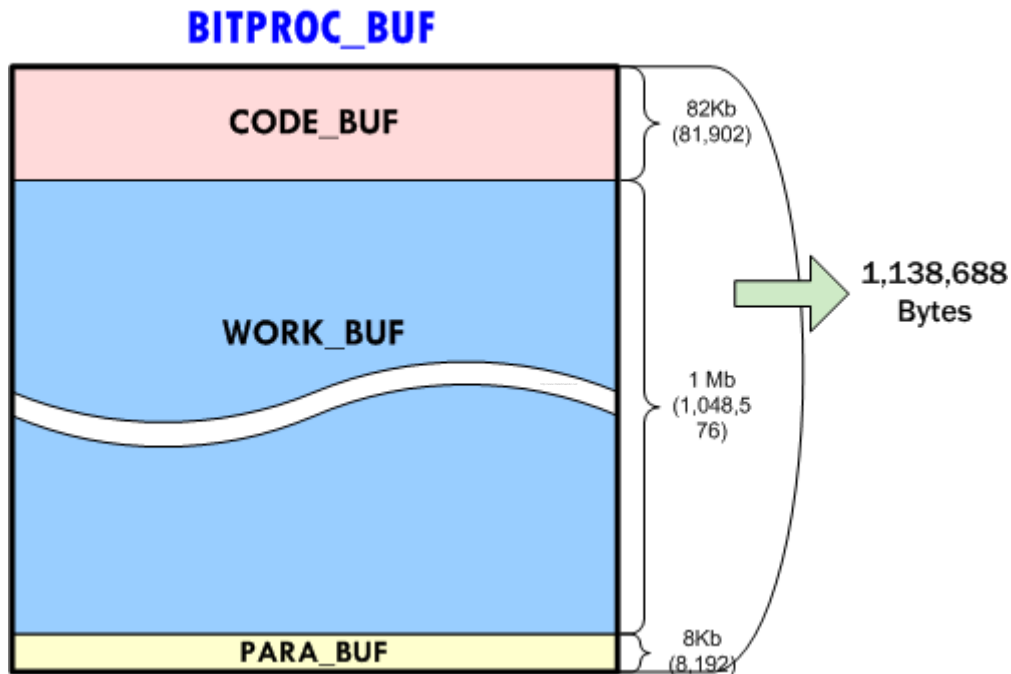


Fig - Size of BITPROC\_BUF

② STRM\_BUF size calculation.

STRM\_BUF has LINE\_BUF and RING\_BUF. They can be separated or shared.

LINE\_BUF & RING\_BUF are in separate mode.

As shown in , LINE\_BUF and RING\_BUF consume memory spaces respectively. For the 720x576-sized video, the possible maximum length of frame is 622,080 (720x576x1.5) bytes that is when it is not totally compressed. By assuming that it can be compressed to 30%, 204,800 bytes is the proper value for the maximum length of compressed frame.

Therefore, we can assume that the maximum length of compressed frame for different video size is

- SD image : 204,800 bytes or less
- CIF image : 102,400 bytes or less

LINE\_BUF & RING\_BUF are in shared mode.

As shown in , LINE\_BUF and RING\_BUF are shared so that the required size of STRM\_BUF is larger one. Typically, the RING\_BUF size is equal to three times of LINE\_BUF\_PER\_INSTANCE, the size for STRM\_BUF can be set 614,400 bytes for SD-sized image.

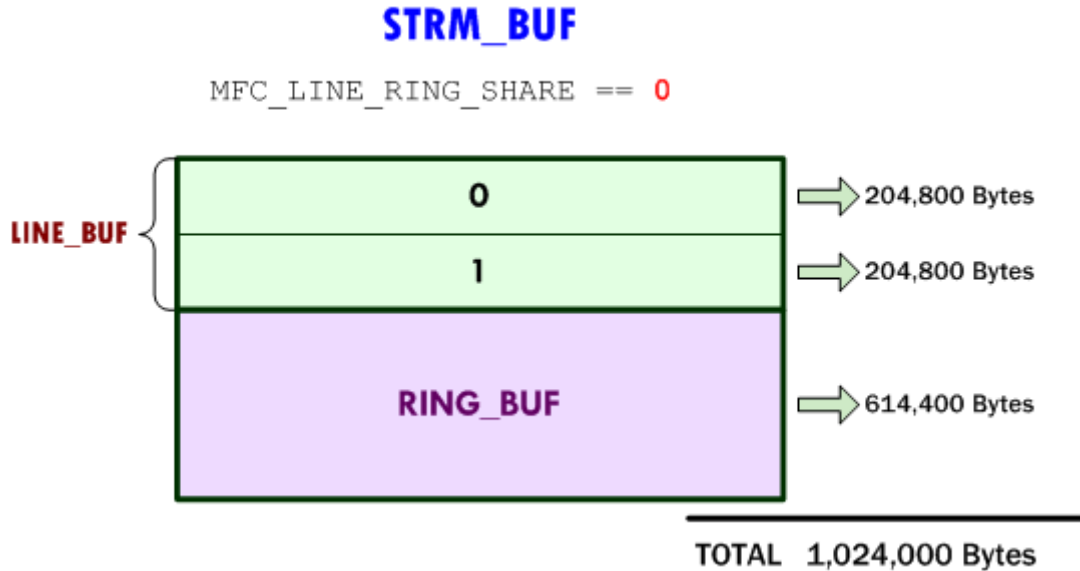


Fig - Size of STRM\_BUF with LINE\_BUF and RING\_BUF not-shared (Example)

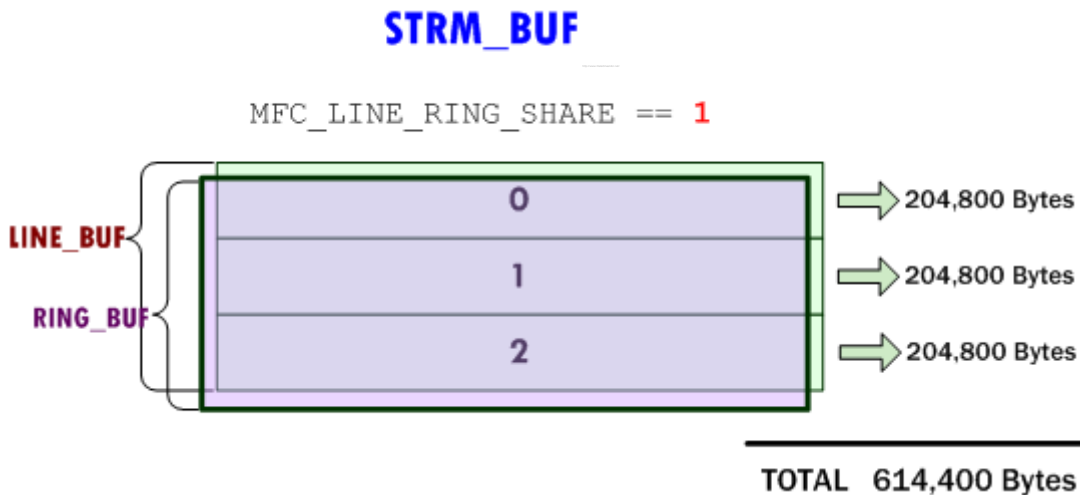


Fig - Size of STRM\_BUF with LINE\_BUF and RING\_BUF shared (Example)

③ FRAM\_BUF size calculation.

MFC requires that the output buffer size is three or four times bigger than the YUV frame size. The number is determined by the return value (frame count in MFC DEC\_PIC\_RUN command.) The number is commonly 3 (three times bigger).

For the SD-sized image, the required size of FRAM\_BUF is shown in .

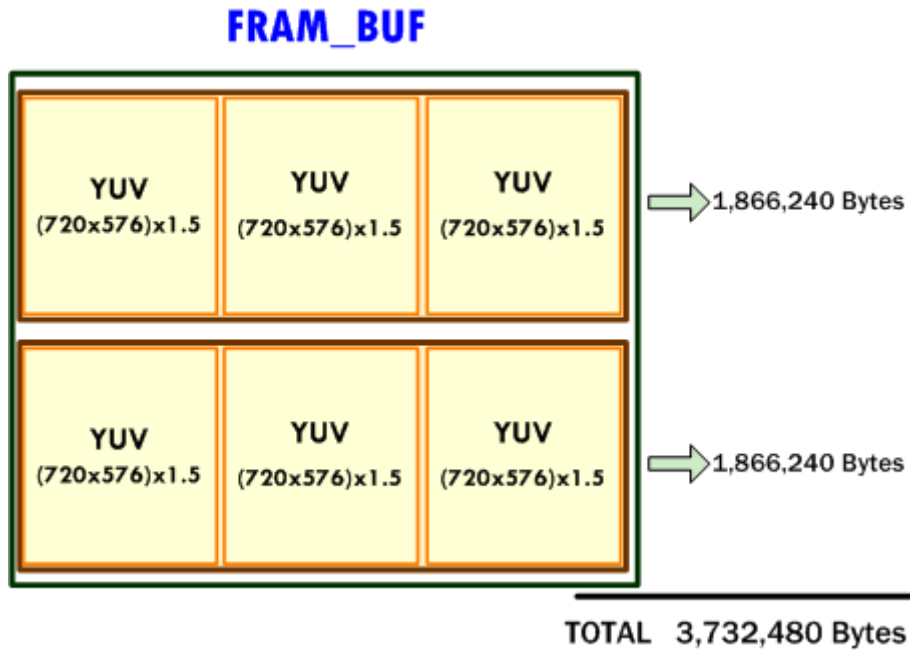


Fig - Size of FRAM\_BUF (Example)

The following tables, Table 2-3 and Table 2-4, are showing required memory size examples for common situations. If we set the MFC to support 1 SD image, it requires 3.3 MB. Since it is greater than the case of 2 CIF images, 1 SD image setting is supporting the 2CIF image setting as well.

(Unit : Bytes)

	2 SD images		1 SD image	
	LINE_RING split	LINE_RING share	LINE_RING split	LINE_RING share
BITPROC_BUF	1,138,688	1,138,688	1,138,688	1,138,688
STRM_BUF	1,024,000	614,400	512,000	307,200
FRAM_BUF	3,732,480	3,732,480	1,866,240	1,866,240
<b>TOTAL</b>	<b>5,895,168</b> 0x0059 F400	<b>5,485,568</b> 0x0053 B400	<b>3,516,928</b> 0x0035 AA00	<b>3,312,128</b> 0x0032 8A00

Table - Required Memory Size (1 or 2 SD images)

(Unit : Bytes)

	2 CIF images		1 CIF image	
	LINE_RING split	LINE_RING share	LINE_RING split	LINE_RING share
BITPROC_BUF	1,138,688	1,138,688	1,138,688	1,138,688
STRM_BUF	512,000	307,200	512,000	307,200
FRAM_BUF	912,384	912,384	912,384	912,384
<b>TOTAL</b>	<b>2,563,072</b> 0x0027 1C00	<b>2,358,272</b> 0x0023 FC00	<b>2,563,072</b> 0x0027 1C00	<b>2,358,272</b> 0x0023 FC00

Table - Required Memory Size (1 or 2 CIF images)

## 3 File I/O Operation

### 3.1 Linux Device Driver's File I/O Operation

API Functions	Description
open	Create the 6400 MFC instance.
ioctl	IOCTL_MFC_MPEG4_DEC_INIT IOCTL_MFC_MPEG4_ENC_INIT IOCTL_MFC_MPEG4_DEC_EXE IOCTL_MFC_MPEG4_ENC_EXE  IOCTL_MFC_H264_DEC_INIT IOCTL_MFC_H264_ENC_INIT IOCTL_MFC_H264_DEC_EXE IOCTL_MFC_H264_ENC_EXE  IOCTL_MFC_H263_DEC_INIT IOCTL_MFC_H263_ENC_INIT IOCTL_MFC_H263_DEC_EXE IOCTL_MFC_H263_ENC_EXE  IOCTL_MFC_VC1_DEC_INIT IOCTL_MFC_VC1_DEC_EXE  IOCTL_MFC_GET_LINE_BUF_ADDR IOCTL_MFC_GET_RING_BUF_ADDR IOCTL_MFC_GET_FRAM_BUF_ADDR IOCTL_MFC_GET_PHY_FRAM_BUF_ADDR IOCTL_MFC_GET_CONFIG  IOCTL_MFC_SET_CONFIG IOCTL_MFC_SET_H263_MULTIPLE_SLICE
mmap	Map reserved memory for application
close	Close the 6400 MFC instance.

#### 3.1.1 open

open	
Syntax	int open(const char * path, int oflag);
Description	This function creates the 6400 MFC instance. Several MFC instance can be made simultaneously. This means that open function can be called several times in a process(task).
Parameters	path [IN] : path of the MFC device driver's node oflag[IN] : flags of MFC.
Returns	HANDLE(file descriptor) of the MFC instance. If it fails, it returns INVALID_HANDLE_VALUE.

### 3.1.2 ioctl

ioctl	
Syntax	int ioctl(int fd, int cmd, int arg);
Description	Most of functions are developed in ioctl. This system call has many functions which is separated by cmd
Parameters	fd [IN] : HANDLE returned by open() function. cmd [IN] : Control codes for the operation. Detailed information of cmd will explain below. arg[IN] : Structure of the MFC argument.
Returns	If the operation completes successfully, the return value is nonzero. If the operation fails or is pending, the return value is zero.

### 3.1.3 mmap

mmap	
Syntax	Void *mmap(void *addr, size_t len, int prot, int flags, int fd, off_t off );
Description	This function maps physically continuous memory. This memory can share user and device driver. It is used as Stream buffer and frame buffer of the MFC
Parameters	addr[IN] : none len[IN] : mapped memory size prot[IN] : memory access permission(PROT_READ, PROT_WRITE, etc) flag[IN] : attribute of memory (MAP_SHARED, etc) fd [IN] : HANDLE of the MFC off[IN] : none
Returns	Base address of stream buffer. This address can use in user application.

### 3.1.4 close

Close	
Syntax	int close(int fd);
Description	Closes an open MFC's handle.
Parameters	fd [IN] : HANDLE returned by open() function
Returns	If the function succeeds, the return value is nonzero. If the function fails, the return value is zero

### 3.2 Control Codes for ioctl()

IOCTL_MFC_MPEG4_DEC_INIT IOCTL_MFC_H263_DEC_INIT IOCTL_MFC_H264_DEC_INIT IOCTL_MFC_VC1_DEC_INIT	
Syntax	See 3.1.2
Description	It initializes the MFC's instance with the configure stream.
Parameters	fd [IN] : HANDLE returned by open() function cmd [IN] : IOCTL_MFC_MPEG4_DEC_INIT, IOCTL_MFC_H263_DEC_INIT, IOCTL_MFC_H264_DEC_INIT, IOCTL_MFC_VC1_DEC_INIT arg [IN/OUT] : Pointer to MFC_DEC_INIT_ARG structure.
Returns	If the operation completes successfully, the return value is nonzero. If the operation fails or is pending, the return value is zero.

IOCTL_MFC_MPEG4_DEC_EXE IOCTL_MFC_H263_DEC_EXE IOCTL_MFC_H264_DEC_EXE IOCTL_MFC_VC1_DEC_EXE	
Syntax	See 3.1.2
Description	It decodes the stream in the LINE_BUF or RING_BUF.
Parameters	fd [IN] : HANDLE returned by open() function cmd [IN] : IOCTL_MFC_MPEG4_DEC_EXE, IOCTL_MFC_H263_DEC_EXE, IOCTL_MFC_H264_DEC_EXE, IOCTL_MFC_VC1_DEC_EXE arg [IN/OUT] : Pointer to MFC_DEC_EXE_ARG structure.
Returns	If the operation completes successfully, the return value is nonzero. If the operation fails or is pending, the return value is zero.

IOCTL_MFC_MPEG4_ENC_INIT IOCTL_MFC_H263_ENC_INIT IOCTL_MFC_H264_ENC_INIT IOCTL_MFC_VC1_ENC_INIT	
Syntax	See 3.1.2
Description	It initializes the MFC's encoding information.

Parameters	fd [IN] : HANDLE returned by open() function cmd [IN] : IOCTL_MFC_MPEG4_ENC_INIT, IOCTL_MFC_H263_ENC_INIT, IOCTL_MFC_H264_ENC_INIT, IOCTL_MFC_VC1_ENC_INIT arg [IN/OUT] : Pointer to MFC_ENC_INIT_ARG structure.
Returns	If the operation completes successfully, the return value is nonzero. If the operation fails or is pending, the return value is zero.

<b>IOCTL_MFC_MPEG4_ENC_EXE</b> <b>IOCTL_MFC_H263_ENC_EXE</b> <b>IOCTL_MFC_H264_ENC_EXE</b> <b>IOCTL_MFC_VC1_ENC_EXE</b>	
Syntax	See 3.1.2
Description	It encodes the stream in the FRAM_BUF.
Parameters	fd [IN] : HANDLE returned by open() function cmd [IN] : IOCTL_MFC_MPEG4_ENC_EXE, IOCTL_MFC_H263_ENC_EXE, IOCTL_MFC_H264_ENC_EXE, IOCTL_MFC_VC1_ENC_EXE arg [IN/OUT] : Pointer to MFC_ENC_EXE_ARG structure.
Returns	If the operation completes successfully, the return value is nonzero. If the operation fails or is pending, the return value is zero.

<b>IOCTL_MFC_GET_LINE_BUF_ADDR</b> <b>IOCTL_MFC_GET_RING_BUF_ADDR</b> <b>IOCTL_MFC_GET_FRAM_BUF_ADDR</b>	
Syntax	See 3.1.2
Description	It obtains the address of the LINE_BUF, RING_BUF or FRAM_BUF.
Parameters	fd [IN] : HANDLE returned by open() function cmd [IN] : IOCTL_MFC_GET_LINE_BUF_ADDR, IOCTL_MFC_GET_RING_BUF_ADDR, IOCTL_MFC_GET_FRAM_BUF_ADDR arg [IN/OUT] : Pointer to MFC_GET_BUF_ADDR_ARG structure.
Returns	If the operation completes successfully, the return value is nonzero. If the operation fails or is pending, the return value is zero.

<b>IOCTL_MFC_GET_PHY_FRAM_BUF_ADDR</b>	
Syntax	See 3.1.2
Description	It obtains the physical address of the FRAM_BUF.



Parameters	fd [IN] : HANDLE returned by open() function cmd [IN] : IOCTL_MFC_GET_PHY_FRAM_BUF_ADDR arg [IN/OUT] : Pointer to MFC_GET_BUF_ADDR_ARG structure.
Returns	If the operation completes successfully, the return value is nonzero. If the operation fails or is pending, the return value is zero.

IOCTL_MFC_GET_CONFIG	
Syntax	See 3.1.2
Description	It get the configurable parameters. The list of configurable parameters is described below.
Parameters	fd [IN] : HANDLE returned by open() function cmd [IN] : IOCTL_MFC_GET_CONFIG arg [IN/OUT] : Pointer to MFC_GET_CONFIG_ARG structure.
Returns	If the operation completes successfully, the return value is nonzero. If the operation fails or is pending, the return value is zero.

IOCTL_MFC_SET_CONFIG	
Syntax	See 3.1.2
Description	It set the configurable parameters with new value. The list of configurable parameters is described below.
Parameters	fd [IN] : HANDLE returned by open() function cmd [IN] : IOCTL_MFC_SET_CONFIG arg [IN/OUT] : Pointer to MFC_SET_CONFIG_ARG structure.
Returns	If the operation completes successfully, the return value is nonzero. If the operation fails or is pending, the return value is zero.

IOCTL_MFC_SET_H263_MULTIPLE_SLICE	
Syntax	See 3.1.2
Description	This command supports multiple slice mode in H.263 encoding case
Parameters	fd [IN] : HANDLE returned by open() function cmd [IN] : IOCTL_MFC_SET_H263_MULTIPLE_SLICE
Returns	None

### 3.3 Data Structure for Passing the IOCTL Arguments

#### 3.3.1 MFC\_ENC\_INIT\_ARG

MFC_ENC_INIT_ARG
------------------

int ret_code	[OUT] Return code
int in_width	[IN] width of YUV420 frame to be encoded
int in_height	[IN] height of YUV420 frame to be encoded
int in_bitrate	[IN] Encoding parameter: Bitrate (kbps)
int in_gopNum	[IN] Encoding parameter: GOP Number (interval of I-frame)
int in_frameRateRes	[IN] Encoding parameter: Frame rate (Res)
int in_frameRateDiv	[IN] Encoding parameter: Frame rate (Divider)

### 3.3.2 MFC\_ENC\_EXE\_ARG

MFC_ENC_EXE_ARG	
int ret_code	[OUT] Return code
int out_encoded_size	[OUT] Length of Encoded video stream
int out_header_size	[OUT] Length of Encoded header size

### 3.3.3 MFC\_DEC\_INIT\_ARG

MFC_DEC_INIT_ARG	
int ret_code	[OUT] Return code
int in_strmSize	[IN] Size of video stream filled in STRM_BUF
int out_width	[OUT] width of YUV420 frame
int out_height	[OUT] height of YUV420 frame

### 3.3.4 MFC\_DEC\_EXE\_ARG

MFC_DEC_EXE_ARG	
int ret_code	[OUT] Return code
int in_strmSize	[IN] Size of video stream filled in STRM_BUF

### 3.3.5 MFC\_GET\_BUF\_ADDR\_ARG

MFC_GET_BUF_ADDR_ARG	
int ret_code	[OUT] Return code

int in_usr_data	[IN] address returned by mmap() function
int out_buf_addr	[OUT] Buffer address
int out_buf_size	[OUT] Size of buffer address

### 3.3.6 MFC\_GET\_CONFIG\_ARG

MFC_GET_CONFIG_ARG	
int ret_code	[OUT] Return code
int in_config_param	[IN] Configurable parameter type
int out_config_value[2]	[IN] Values to get for the configurable parameter. Maximum two integer values can be obtained.

### 3.3.7 MFC\_SET\_CONFIG\_ARG

MFC_SET_CONFIG_ARG	
int ret_code	[OUT] Return code
int in_config_param	[IN] Configurable parameter type
int in_config_value[2]	[IN] Values to be set for the configurable parameter. Maximum two integer values can be set
int out_config_value_old[2]	[OUT] Old values of the configurable parameter.

## 3.4 Obtainable Parameters for MFC

In this section, the obtainable parameters for MFC are explained. They are 'obtainable parameter types' which are obtained to out\_config\_param in MFC\_GET\_CONFIG\_ARG that is an argument of IOCTL\_MFC\_GET\_CONFIG command

Obtainable Paramter	Parameter Value	
MFC_GET_CONFIG_DEC_FRAME_NEED_COUNT	out_config_value_old[0]	Count of frame buffers
	out_config_value_old[1]	Not Used

## 3.5 Configurable Parameters for MFC

In this section, the configurable parameters for MFC are explained. They are 'configurable parameter types' which are assigned to in\_config\_param in MFC\_SET\_CONFIG\_ARG that is an argument of IOCTL\_MFC\_SET\_CONFIG command

Configurable Paramter	Parameter Value	
MFC_SET_CONFIG_DEC_ROTATE	in_config_value[0]	PostRotMode
	in_config_value[1]	Not Used
	out_config_value_old[0]	Old value of PostRotMode

	during the decoding.	out_config_value_old[1]	Not Used
MFC_SET_CONFIG_DEC_H264_REORDER (Not implemented)		In_config_value[0]	ReorderEn
		In_config_value[1]	Not used
		Out_config_value_old[0]	Old value of ReorderEn
		Out_config_value_old[1]	Not Used
MFC_SET_CONFIG_ENC_H263_PARAM	'H.263 Encode Parameters'	In_config_value[0]	H.263 EncParam
		In_config_value[1]	Not Used
		Out_config_value_old[0]	Old value of H.263 EncParam
		Out_config_value_old[1]	Not Used
MFC_SET_CONFIG_ENC_SLICE_MODE	MFC 'EncSliceMode'	In_config_value[0]	0: Single, 1: Multi
		In_config_value[1]	Number of Slices
		Out_config_value_old[0]	Old value of Single/Multi
		Out_config_value_old[1]	Old value of Number Of Slices
MFC_SET_CONFIG_ENC_PARAM_CHANGE	'EncParamChange' for Changing encode Parameter during the Encoding	In_config_value[0]	Parameter to be Changed
		In_config_value[1]	New value of the Parameter
		Out_config_value_old[0]	Parameter to be Changed
		Out_config_value_old[1]	Old value of the Parameter
MFC_SET_CONFIG_ENC_CUR_PIC_OPT		In_config_value[0]	Option Name
		In_config_value[1]	Option Value
		Out_config_value_old[0]	Not used
		Out_config_value_old[1]	Not used

Table - Configurable parameter types for IOCTL\_MFC\_SET\_CONFIG

### 3.5.1 MFC\_SET\_CONFIG\_DEC\_ROTATE

'Post rotation mode' configures the MFC to rotate and/or mirror the output YUV image during the decoding.

in\_config\_value[0] = {HorMir, VerMir, RotAng[1:0]}

HorMir : Horizontal mirroring

VerMir : Vertical mirroring

RotAng[1:0]

0 : 0 degree counterclockwise rotate

1 : 90 degree counterclockwise rotate

2 : 180 degree counterclockwise rotate

3 : 270 degree counterclockwise rotate

in\_config\_value[1] = Not Used

out\_config\_value[0] = Old value of PostRotMode

out\_config\_value[1] = Not Used

In_config_value[0]	HorMir	VerMir	RotAng
0x0000	X	X	X
0x0010	X	X	X
0x0011	X	X	90° rotate
0x0012	X	X	180° rotate
0x0013	X	X	270° rotate
0x0014	X	O	X
0x0015	X	O	90° rotate

0x0016	X	O	180° rotate
0x0017	X	O	270° rotate
0x0018	O	X	X
0x0019	O	X	90° rotate
0x001A	O	X	180° rotate
0x001B	O	X	270° rotate
0x001C	O	O	X
0x001D	O	O	90° rotate
0x001E	O	O	180° rotate
0x001F	O	O	270° rotate

### 3.5.2 MFC\_SET\_CONFIG\_ENC\_H263\_PARAM

'H.263 Encode Annex' sets the MFC to produce H.263 stream with the PLUSTYPE features as defined in H.263 standard specification document. The S3C6400 MFC supports four Annexes, Annex T, Annex J, Annex K and Annex I. They are enabled by bitwise-OR operation.

in\_config\_value[0] = {ANNEX\_T\_ON/OFF | ANNEX\_K\_ON/OFF | ANNEX\_J\_ON/OFF }  
 ANNEX\_T\_ON, ANNEX\_T\_OFF: Modified Quantization mode  
 ANNEX\_K\_ON, ANNEX\_K\_OFF : Slice Structured mode  
 ANNEX\_J\_ON, ANNEX\_J\_OFF : Deblocking Filter mode

in\_config\_value[1] = Not Used  
 out\_config\_value[0] = Old value of H.263 Annex Setting  
 out\_config\_value[1] = Not Used

in_config_value[0]	Annex T	Annex K	Annex J
0x0000	X	X	X
0x0001	O	X	X
0x0002	X	O	X
0x0003	O	O	X
0x0004	X	X	O
0x0005	O	X	O
0x0006	X	O	O
0x0007	O	O	O

### 3.5.3 MFC\_SET\_CONFIG\_ENC\_SLICE\_MODE

'EncSliceMode' determines the encoded output stream format to be single or multiple slices.

in\_config\_value[0] = {0: single slice, 1: multiple slice}  
 in\_config\_value[1] = Number of slices per picture  
 out\_config\_value[0] = Old value of Single/Multiple  
 out\_config\_value[1] = Old value of NumSlices

in_config_value[0]	in_config_value[1]	Description
0	ignored	Single slice
1	1 ~ 256	Number of multiple slices

		per picture
--	--	-------------

### 3.5.4 MFC\_SET\_CONFIG\_ENC\_PARAM\_CHANGE

'EncParamChange' changes the encoding parameter during the encoding.

The encoding parameters such as frame rate and bitrate are set at the encoder initialization stage. Once the encoder is initialized, the parameters can be changed dynamically

in\_config\_value[0] = Encoding Parameter to be changed

in\_config\_value[1] = Parameter value to be set

out\_config\_value[0] = Encoding Parameter to be changed

out\_config\_value[1] = Old value of parameter

in_config_value[0]	in_config_value[1]	Description
ENC_PARAM_GOP_NUM	0~60	0 - I, P, P, P, ... 1 - I, I, I, I, ... 2 - I, P, I, P, ... 3 - I, P, P, I, P, P, I, ...
ENC_PARAM_INTRA_QP	1 ~ 31 (MPEG4/H.263), 0 ~ 51 (H.264)	Intra frame picture quantized step parameter
ENC_PARAM_BITRATE	1 ~ 32767	Target bitrate in kbps
ENC_PARAM_F_RATE	[FrameRateDiv-1] [FrameRateRes]	Bits 31~16 : FrameRateDiv-1 Bits 15~0 : FrameRateRes  F_RATE = [FrameRateRes] / [FrameRateDiv-1]
ENC_PARAM_INTRA_REF	0 ~ N	Intra MB refresh number. 0 - Intra MB refresh is not used N - At least N number of MBs are encoded as intra mode at every picture
ENC_PARAM_SLICE_MODE	0 ~ 256	Number of multiple slices per picture 0 - Single slice 1 ~ 256 - Multiple slices

### 3.5.5 MFC\_SET\_CONFIG\_ENC\_CUR\_PIC\_OPT

in\_config\_value[0] = Encoding Parameter to be changed

in\_config\_value[1] = Parameter value to be set

out\_config\_value[0] = Encoding Parameter to be changed

out\_config\_value[1] = Old value of parameter

in_config_value[0]	in_config_value[1]	Description
ENC_PIC_OPT_IDR	1	The current source image is encoded as H.264 - IDR picture MPEG4/H263 - I picture
ENC_PIC_OPT_SKIP	1	The current source image is ignored. (Encoding is

		skipped.)
ENC_PIC_OPT_RECOVERY	1	The current and several following images will be encoded as multiple slice for the gradual recovery. The SEI message which is containing the recovery point is generated. <b>H.264 only.</b>

## 4 Annex A (H.264 Decoder Usage)

### H.264 Decoding Example

```

#include "test.h"
#include "MfcDrvParams.h"

#define DEVICE_FILENAME      "/dev/misc/s3c-mfc"
#define BUF_SIZE            0x459000

int Test_Decoder_Ring_Buffer(int argc, char **argv)
{
    int          dev_fd, in_fd, out_fd;
    char         *addr, *in_addr;
    int          cnt = 1;
    int          file_size;
    int          remain;
    struct stat  s;

    // arguments of ioctl
    MFC_DEC_INIT_ARG      dec_init;
    MFC_DEC_EXE_ARG       dec_exe;
    MFC_GET_BUF_ADDR_ARG  get_buf_addr;

    if (argc != 3) {
        printf("Usage : mfc <input file name> <output filename>\n");
        return -1;
    }

    ////////////////////////////////////////////////////////////////////
    // In/Out file open //
    ////////////////////////////////////////////////////////////////////
    in_fd  = open(argv[1], O_RDONLY);
    out_fd = open(argv[2], O_RDWR | O_CREAT | O_TRUNC, 0644);
    if( (in_fd < 0) || (out_fd < 0) ) {
        printf("input/output file open error\n");
        return -1;
    }

    // get input file size
    fstat(in_fd, &s);
    file_size = s.st_size;

    // mapping input file to memory
    in_addr = (char *)mmap(0, file_size, PROT_READ, MAP_SHARED, in_fd, 0);
    if(in_addr == NULL) {
        printf("input file memory mapping failed\n");
        return -1;
    }

    ////////////////////////////////////////////////////////////////////
    // MFC device driver open //
    ////////////////////////////////////////////////////////////////////
    dev_fd = open(DEVICE_FILENAME, O_RDWR|O_NDELAY);
    if (dev_fd < 0) {
        printf("MFC open error : %d\n", dev_fd);
        return -1;
    }
}

```



```

////////////////////////////////////
// mapping shared in/out buffer between App and D/D //
////////////////////////////////////
    addr = (char *) mmap(0,
                        BUF_SIZE,
                        PROT_READ | PROT_WRITE,
                        MAP_SHARED,
                        dev_fd,
                        0
                    );
    if (addr == NULL) {
        printf("MFC mmap failed\n");
        return -1;
    }

    // Get input buffer address in ring buffer mode //
    // When below ioctl function is called for the first time, It returns double buffer size.
    // So, Input buffer will be filled as 2 part unit size
    get_buf_addr.in_usr_data = (int)addr;
    ioctl(dev_fd, IOCTL_MFC_GET_RING_BUF_ADDR, &get_buf_addr);

    remain = file_size;

    // copy input stream to input buffer
    memcpy((char *)get_buf_addr.out_buf_addr, in_addr, get_buf_addr.out_buf_size);
    remain -= get_buf_addr.out_buf_size;
    printf("remain : %d\n", remain);

    //////////////////////////////////
    // MFC decoder initialization //
    //////////////////////////////////
    dec_init.in_strmSize = get_buf_addr.out_buf_size;
    ioctl(dev_fd, IOCTL_MFC_H264_DEC_INIT, &dec_init);
    printf("out_width : %d, out_height : %d\n", dec_init.out_width, dec_init.out_height);

    while(1)
    {
        //////////////////////////////////
        // Get input stream buffer address //
        //////////////////////////////////
        ioctl(dev_fd, IOCTL_MFC_GET_RING_BUF_ADDR, &get_buf_addr);

        if(get_buf_addr.out_buf_size > 0) {
            if(remain >= get_buf_addr.out_buf_size) {
                cnt++;
                memcpy((char *)get_buf_addr.out_buf_addr, in_addr + (cnt *
(get_buf_addr.out_buf_size)), get_buf_addr.out_buf_size);
                remain -= get_buf_addr.out_buf_size;
                dec_exe.in_strmSize = get_buf_addr.out_buf_size;
            } else {
                cnt++;
                memcpy((char *)get_buf_addr.out_buf_addr, in_addr + (cnt *
(get_buf_addr.out_buf_size)), remain);
                dec_exe.in_strmSize = remain;
                remain = 0;
            }
            printf("remain : %d\n", remain);
        }
    }
}

```

```
        else {
            dec_exe.in_strmSize = 0;
        }

        ////////////////////////////////////////////////////
        // MFC decoding //
        ////////////////////////////////////////////////////
        ioctl(dev_fd, IOCTL_MFC_H264_DEC_EXE, &dec_exe);
        if(dec_exe.ret_code < 0) {
            printf("ret code : %d\n", dec_exe.ret_code);
            break;
        }

        ////////////////////////////////////////////////////
        // Get output buffer address //
        ////////////////////////////////////////////////////
        ioctl(dev_fd, IOCTL_MFC_GET_FRAM_BUF_ADDR, &get_buf_addr);
        write(out_fd, (char *)get_buf_addr.out_buf_addr, get_buf_addr.out_buf_size);
    }

    close(dev_fd);
    close(in_fd);
    close(out_fd);

    return 0;
}
```