SW Coding Style

(For C/C++)

Revision History

Rev.	Date	Modifier	Description
0.1	2009/04/20	Charley.Lee	Preliminary Draft for Non-OS TV Application
0.2	2009/04/30	Ata.Chen	For All TV solution
0.3	2009/05/21	Carrie.Wu	Add MUF/C++ coding style
1.0	2009/06/01	Carrie.Wu	SSG members approved
1.1	2009/10/09	Wenjing.Tien	MStar SDK design rule

File Organization

A file consists of various sections that should be separated by several blank lines. Although there is no maximum length limit for source files, files with more than about 3000 lines are cumbersome to deal with. The editor may not have enough temp space to edit the file; compilations will go more slowly, etc. Many rows of asterisks, for example, present little information compared to the time it takes to scroll past, and are discouraged. Lines longer than 79 columns are not handled well by all terminals and should be avoided if possible. Excessively long lines which result from deep indenting are often a symptom of poorly-organized code.

- **< 3000 lines per file**
- **< 80 character per line**

File Naming Conventions

MMI part

File names are made up of a prefix, “MApp”, module item, extend item and suffix. The first character of the name in module item and extend item should be a capital letter and all characters should be lower-case letters and numbers. The module item should be use the abbreviation of functional module; the extend item is optional description. The module name (abbreviation) and extend item name should be twenty or fewer characters and suffix should three or fewer characters.

- **MApp**_InputSource.c

MW part

Middleware’s application layer file name are all lower case, mw app file are made up of a prefix, “mapp”, ”_”, module name. Middleware API layer file name are ,ade up of a prefix, “msapi”, ”_”, module name!

- **mapp**_mheg5.c
- **msAPI**_MPEG_Subtitle.c → **msapi**_mpeg_subtitle.c

Drv part

API File names are made up of a prefix, “api”, function abbreviations, “_”, sub-function abbreviations, HAL File names are made up of a prefix, “hal”, HWIP, module name, Drv File names are made up of a prefix, “drv”, HWIP, module name.

- **api**DMX.c
- **hal**MVD.c
- **drv**GE.c
- **fw**MAD_ac3.bin

MUF：

File names are made up of Class name and suffix.

(File will be generated by UML tool automatically)

The Class name should be Pascal Casting (The first character of every word is in upper case)

- **Definition：XXXX.cpp**
- **Declaration：XXXX.h**
- **Forward Declaration：XXXX_fdef.h**

MSrv：

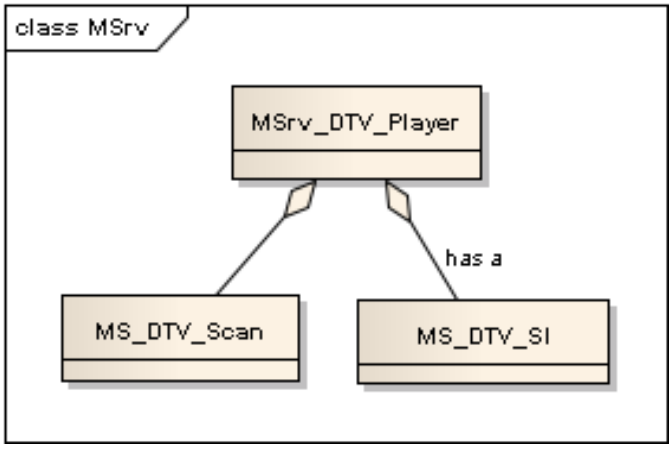
File names are made up of Class name and suffix. The Class name should be Pascal Casting (The first character of every word is in upper case). The name of interface class is MSrv_Xxxxx. The name of internal class or implement class is MS_Xxxxxx.

Interface:

- **Definition： MSrv_Xxxxx.cpp**
- **Declaration：MSrv_Xxxxx.h**
- **Forward Declaration：MSrv_Xxxxx_fdef.h**

Internal/implement:

- **Definition： MS_Xxxxx.cpp**
- **Declaration：MS_Xxxxx.h**
- **Forward Declaration：MS_Xxxxx_fdef.h**



MStarSDK:

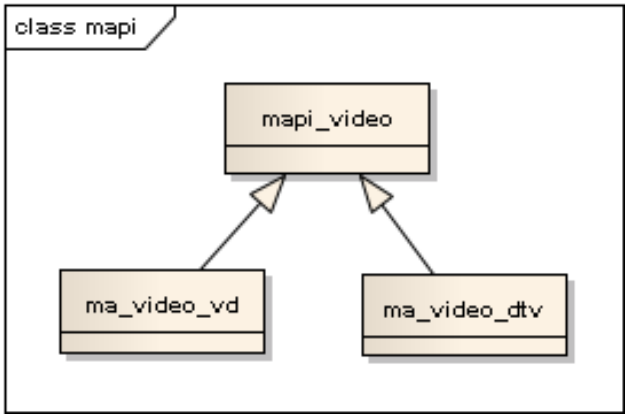
The file name and class name in SDK layer are all lower case. File name are makeup of class name and suffix. The name of interface class (which will release header file to customer) is mapi_xxxxx. The name of internal class or implement class is ma_xxxxx

Interface:

- **Definition： mapi_xxxxx.cpp**
- **Declaration：mapi_xxxxx.h**
- **Forward Declaration：mapi_xxxxx_fdef.h**

Internal/implement:

- **Definition： ma_xxxxx.cpp**
- **Declaration：ma_xxxxx.h**
- **Forward Declaration：ma_xxxxx_fdef.h**



File structure

The suffix of source file should be ".c" in ANSI C, ".ccp" in C++; header file should be ".h". In each source file (.c/.cpp) and header file (.h), the MStar legal declaration and file purpose description should be added in the beginning. In each source file and header file, one specific definition is used as itself identification.

Example:

```
////////////////////////////////////////////////////////////////////////////////
//
// Copyright (c) 2006-2009 MStar Semiconductor, Inc.
// All rights reserved.
//
```

```
// Unless otherwise stipulated in writing, any and all information contained
// herein regardless in any format shall remain the sole proprietary of
// MStar Semiconductor Inc. and be kept in strict confidence
// (“MStar Confidential Information”) by the recipient.
// Any unauthorized act including without limitation unauthorized disclosure,
// copying, use, reproduction, sale, distribution, modification, disassembling,
// reverse engineering and compiling of the contents of MStar Confidential
// Information is unlawful and strictly prohibited. MStar hereby reserves the
// rights to any and all damages, losses, costs and expenses resulting therefrom.
//
//////////////////////////////////////////////////////////////////////////////////

//////////////////////////////////////////////////////////////////////////////////
/// @file map_mheg5.c
/// @author MStar Semiconductor, Inc.
/// @brief MHEG5 application
//////////////////////////////////////////////////////////////////////////////////

#define _MAPP_MHEG5_C_
( or _MAPP_MHEG5_CPP_ )
```

Header files should not be nested. The prologue for a header file should, therefore, describe what other headers need to be #included for the header to be functional. In extreme cases, where a large number of header files are to be included in several different source files, it is acceptable to put all common #includes in one include file.

It is common to put the following into each *.h* file to prevent accidental double-inclusion.

```
#ifndef _MAPP_MHEG5_H_
#define _MAPP_MHEG5_H_

#ifdef _MAPP_MHEG5_C_
// Version define Only for module entry function file
#define mapp_MHEG5_Ver() 'M','S','I','F',              /* Version ID Header */\
                         '0','3',                        /* Info Class Code */\
                         0x66,0x66,                          /* Customer ID */\
                         0x66,0x66,                            /* Module ID */\
```

```
                0xFF,0xFF,                          /* Chip ID */\
                '0',                                /* CPU */\
                'M','5','1','5',                    /* Library Code */\
                '0','0',                            /* Library Ver */\
                '0','0','0','1',                    /* Build Number */\
                '0','0','1','2','3','4','5','5', /* P4 Change Number */\
                '2'                                 /* OS */\
#define EXTCALL
#else
#define EXTCALL extern
#endif
....
//------------------------------------------------------------------------------
// External Function, if this one is library, external function
// Must be define in interface include file
//------------------------------------------------------------------------------
extern void MApi_XC_FreezeImg(MS_BOOL bEnable, SCALER_WIN eWindow);
...
//------------------------------------------------------------------------------
// Function
//------------------------------------------------------------------------------
EXTCALL void MApp_MHEG5_ChannelProcessMailbox(U8 *u8MailBox)
;
....
#undef EXTCALL
#endif    //_MSAPI_MHEG5_H_
```

Coding Style

Indentation and Spacing:

Tabs should be used to indent code. Owing to complex hierarchical code structures, tabs should be represented four spaces. To simplicity and reading reasons, we have to use unifying font and indent to avoid nesting statements

Comment

Comments, if used improperly, can make code very difficult to understand. Use comments sparingly, to document important parts of the code, and let the code speak for itself at other times. A comment at the beginning of each function is a good idea, and within the code comments are best kept short and used to document unusual things.

Comments that describe data structure, algorithm, disabling unuseful code etc., should be comment form with opening /* and closing */. Comments that describe the function call, should be comment form with "///, command provided from Doxygen. Comments that describe any reminding information in one-line comment statement, should be comment form with "//".

Variable and function name

A variable or function name should be sufficient to understand the meaning, while remaining concise. The shorter the lifespan or scope of a variable, the shorter its name should be; conversely, the more important a variable, and the more places it is used, the more descriptive that name should be.

Internal and external function

Function format for C source projects should be

MM:I

"<Internal / external> <Return Type> MApp_<Module Abbr>_<Function Purpose>(…)"

- void MApp_ZUI_ACT_AtvManualTuningChangeCurrentCH(U16 Value)

MW:

"<Internal / external> <Return Type> MApp_<Module Abbr>_<Function Purpose>(…)"

- void MApp_MHEG5_ChannelProcessMailbox(U8 *u8MailBox);

"<Internal / external> <Return Type> msAPI_<Module Abbr>_<Function Purpose>(…)"

- void msAPI_MHEG5_SetKeyRegisterGroup(U8 u8RegisterGroup)

Drv:

"<Internal / external> <Return Type> HAL_<HWIP>_<Function Purpose> (…)"

- void MDrv_GE_BitBlt(GEBitBltInfo *BitbltInfo, GEPitBaseInfo *PitBaseInfo)

"<Internal / external> <Return Type> MDrv_<HWIP>_<Function Purpose> (…)"

- GE_Result HAL_GE_GetFmtCaps(GE_BufFmt buf_fmt, GE_BufType buf_type, GE_FmtCaps *caps)

"<Internal / external> <Return Type> MApi_<Module Abbr>_<Function Purpose> (…)"

- BOOLEAN MApi_DMX_Init(void)

In ANSI C standard, "static" specific command and "_MApp" should be used to identify the internal function.

Ex：static void _MApp_ZUI_ACT_AtvManualTuningChangeCurrentCH(U16 Value)

In C++, for "PRIVATE" / "PROTECTED" field, please use Camel Casting (Except the first character of the first word is in lower case, the first character of every other words is in upper case.), for PUBLIC, please use Pascal Casting (The first character of every word is in upper case).

Function input parameters should be four or less for clarifies. If the input parameter is a pointer which transmits the data messages, Read-only, the const specific word should be used to protect the pointer value and that pointed by a pointer. Call by address is rather than call by value.

- **void functionA(para1, para2, para3, para4, ~~para5, ….~~)**

void functionSetWindow(const U8 *pWindowInfo)

In C++, use const whenever possible.

EX:

```
class A
{
public:
    //this function will not change any member variable
    void FunA(U32 u32Par) const;

    //pu32Data point to a const
    void Func2(const U32 *pu32Data);
```

```
    void Func3(U32 const *pu32Data);

    //u32Data reference to a const
    void Func4(const U32 &u32Data);
};
```

C++ Naming：

Casting format description：

Pascal Casing：The first character of every word is in upper case.

Ex：**P**ascal**C**asting**F**orma**E**xample

Camel Casing：Except the first character of the first word is in lower case, the first character of every other words is in upper case.

Ex：camelCastingFormatExample

Upper Casing：All Words are in upper case with "_" for separation.

Ex：UPPER_CASTING_FORMAT_EXAMPLE

Naming：

Class：Use Nouns for naming in Pascal Casting.

- Class for Application with a prefix "MApp_" ➔ MApp_MainMenu
- Class for Main Frame with a prefix "MFrm_" ➔ MFrm_MainMenu
- Class for service with a prefix "MSrv_" ➔ MSrv_VBI

Methods：Use Verb for naming in Pascal casting

Ex：Run, Init, Set, Close

Member variable: add prefix m_ in member variable

EX:

```
Class A
{
private:
    U32 m_u32VarB;
    U32 *m_pu32PointerB;
    Struct VDEC_CCCfg *m_pstCfgB;
};
```

Events：Add “On” prefix and Verb with present/past tense is recommended to be used for naming to clarify the event timeframe. (Pascal casting)

Data Types and Typedef

The typedef instruction in C allows you to define new words in the language. Such definitions are best placed in a header file so that all source code files which rely on that header file have access to the same set of definitions. Using typedef is a good idea in case you need to modify your code later, since it is simply a matter of changing one line in the header file and all code which uses that definition will work differently after being recompiled. This makes modifications much easier than having to use an editor to replace all instances of a type throughout your code.

Table 1: variable data type

Language key word	New type definition	Desclaration

Bit	BIT (HK51)
	not allow (32 bit Risc)
		BOOL bFlag; // TRUE or FALSE
		not allow (32 bit Risc)

boolean or bool	BOOL	BOOL bFlag; // TRUE or FALSE
unsigned char (1 Byte)	U8	U8 u8Var;
unsigned short (2 Bytes)	U16	U16 u16Var;
unsigned int (2 Bytes, HK51)	U16	U16 u16Var;
unsigned int (4 Bytes, 32bit Risc)	U32	U32 u32Var;
signed char (1 Byte)	S8	S8 s8Var;
signed short (2 Bytes)	S16	S16 s16Var;
signed int (2 bytes, HK51)	S16	S16 s16Var;
signed int (4 bytes, 32 bit Risc)	S32	S32 s32Var;
Pointer	<Type> *	U32 *pU32Var; void *pVar;
String	<Type> Var[];	U8 u8Var[size];
function pointer	fn<Function>	void * fnFunctionName
boolean array	not allowed	x
Float	not allowed	x
Double	not allowed	x
long long	not allowed	x
double double	not allowed	x

Structures are best defined using the typedef mechanism. It is possible to define a name using typedef before the structure is actually defined, which allows recursive definitions. Structures should be used for all compound data types where related information needs to be kept together. Structures can be passed to functions or returned from functions, and even assigned to each other

	New type definition	Desclaration
enum	typedef enum { E_<MODULE>_<FUNCTION>_<ACTION> } EN_<CAPITAL DESCRIPTION>	typedef enum { E_STATE_ANALOGINPUTS_INIT =0, …… } EN_ANALOGINPUTS_STATE; EN_ANALOGINPUTS_STATE enAnalogInputState;
struct	typedef struct { …. } ST_<CAPITAL DESCRIPTION>	ST_MS_SYSINFO stSystemInfo;

class	class MSrv_Foo
	{
	
	};
		MSrv_Foo ***c***Foo, *p***c***Foo;

Don’t use any program language specific key word as needed definition.
Example: INTERFACE (used in original design). “EXTCALL” is to substitute “INTERFACE”.

- ~~INTERFACE~~ EXTCALL void msAPI_Key_Transmit(U8 u8Key, U8 Status);

Programmer should use the “cast” method to read / write data structure to avoid Big-Endian / Little-Endian.

Programmer should use different “variable / structure name” to avoid shadow warning for local / global usage.

Statement

- If – else statement: must have “else” condition, and { } for code area.

```
If (condition-A)
{

}
else
{

}
```

- Switch statement:

```
Switch (control variable)
{
    case “A”:
    {

    }
    break;
    default:
    {

    }
    break;
}
```

- Others constrained coding rule should be followed MSTAR MISRA-C / MISRA-C++ rule.

Abbreviation

Common Abbreviation

Abbr	Descripition	Abbr	Descripition
Rect	Rectangle	Addr	Address
Blt	Bitblt	Idx	Index
Pos	Position	Hdl	Handle
Dir	Directory, Direction	Tbl	Table
Src	Source	Str	String
Dst	Destination	Char	Character
Win	Window	Def	Definition
Fmt	Format	Flt	Filter
Font	Font	Sym	Symbol
Bmp	Bitmap	Map	Mapping
Disp	Display	To/2	To
Up/Dn	Up/Down	For/4	For
Top/Botm	Top/Bottom	Ldr	Loader
Head/Tail	Head/Tail	St/Ed/Start/End	Start/End
Info	Information	Rec	Record
Caps	Capabilities	Func	Function
Ver	Version	Param	Paramemter
Hdr	Header	Attr	Attribute
Rst	Reset	Alloc	Allocate
Cmd	Command	Spd	Speed
CmdQ	Command Queue	Act	Action
Ack	Acknowledge	Trig	Trigger
Init	Initialization	Inv	Invert
Ins/Del	Inser/Delete	Conv	Convert
Ret	Return	Cfg/Config	Configuration

HW module abbreviation

Abbreviation	Full name	Note
GE	Graphics Engine	
G3D		
PE	Pixel Engine	
MVOP	MPEG Video Output Path	Designer request to "correct" the name from VOP to **MVOP**
GOP	Graphics Output Path	
TSP	Transport Stream Processor	
XC	Scaler	
MVD/RVD/SVD/HVD	MPEG/RM/H264/… Video Decoder	
MAD	MPEG Audio Decoder	
UART	UART	
GPIO	General Purpose IO	
MIU	Memory Interface Unit	
JPD	JPEG Decoder	
VBI	Vertical Blanking Interval	
CI/CI+	Common Interface	PCMCIA
CA	Conditional Access	
DSCMB	Descrambler	
PM	Power Manager	
SYS	System	
RLD	Subtitle	
VD	Video Decoder (TV decoder)	
VE	Video Encoder (TV Encoder)	
EMAC	Ethernet MAC	
FCIE	Card reader, SD/MMC	
HDMI Tx/Rx		
IIC		
MBOX	Mail box control	
MPIF	Master parallel interface	
NDS	NDS_HDI, NDS_RASP	
NFIE	Parallel NAND flash, FCIE2	
OTG	USB OTG	
PFSH	Parallel NOR flash	

PWM		
RLD	Subtitle run-length decoder	
RTC	Real time clock	
SAR		
SDIO		
SPI	SPI serial flash	
SYS	ChipTop(TOP), WatchDog(WDG), System related	
UART		
USB		
BDMA	Byte DMA	
PM	Power Management	
GDMA		

Middleware module abbreviation

Abbreviation	Full name	Note
TTX	Teletext	
CC	ClosedCaption	
MHEG5	Multimedia and Hypermedia Expert Group	
Subtitle	DVB or DVD subtitle	
SI	Service Information	
PSI	Public Service Information	
CM	Channel Manage	
DB	Database	
PVR	Personal Video Recorder	
EPG	Electronic Program Guide	
OAD OTA	On-Air-Download Over-The-Air	
BL	Bootloader	
MM	Multimedia	
MP	Multimedia Player	
FS	File System	

MUF/MIDE abbreviation

Abbreviation	Full name	Note
MUF	Mstar Unify Framework	
MIDE	Mstar Integrated Development Environment	
SRV	Service	
FRM	Frame	

Routinue Functions

- Get Library Version String:
  XXX_GetLibVer
- Show debug message
  XXXShowVerString(U8 *string)
- Get Static Information:
  XXX_GetInfo
- Get Runtime Status:
  XXX_GetStatus
- Show debug message
  XXXDBGprintf(U32 u32Module, U32 u32Level, char* fmt, ...)
- Set debug message Module
  XXXDebugSetModuleValue (U32 u32Module)
- Set debug message Level
  XXXDebugSetLevelValue (U32 u32Level)