MTK 平台进行驱动调试

1. makefile

MXX_GPRS.mak(mcu\make)

Option.mak(mcu\make)

对于同一个平台，不同的项目有着不同的功能配置。对于驱动调试来说，需要修改和添加一些宏开关控制；

基本格式和步骤：

MXX_GPRS.mak ：

XXX_XXX = XXX

XXX_XXX = NONE

XXX_XXX = TRUE

XXX_XXX = FALSE

Option.mak:

```
ifdef XXX_XXX
ifneq ($(strip $( XXX_XXX)),FALSE)
    COM_DEFS    += XXX_XXX
endif
endif
```

代码中就使用 XXX_XXX 来控制相关代码；

```
＃if defined(XXX_XXX)
//add code here
#endif
```

以 MT6226(05c)平台为例：列举一些修改比较频繁的开关：

	MT6226A	MT6226B	MT6226M	MT6227A	MT6227B
PLATFORM	MT6226	MT6226	MT6226M	MT6227A	MT6227B
CHIP_VER	S00	S01	S01	S00	S01
LCD_MODULE	用于 LCM Module 控制；详细见 No.2				
CUSTOM_CFLAGS	如用 JTAG 进行 DEBUG 需打开此开关；注意关闭 Watch Dog				
EXT_CAM_MODULE	如用外部 DSP 来控制 Sensor；需用到此开关；详见 No.3				
ISP_SUPPORT	如 Camera 没有外挂 DSP 需打开此开关；				
CMOS_SENSOR	用以区分 Sensor 的类型；如 OV7660;OV9650 等				
NOR_FLASH_TYPE	所用 NOR Flash 的类型；现一般是 AMD Series；				
MSDC_CARD_SUPPORT_TYPE	用以是否支持 T 卡的开关；不支持为 NONE；支持为				

	MSDC_SD_MMC
BLUETOOTH_SUPPORT	用以是否支持蓝牙的开关；不支持为 NONE；支持为所用芯片的类型；
TOUCH_PANEL_SUPPORT	用以是否支持触摸屏的开关；不支持为 FALSE；支持为 TRUE；
MOTION_SENSOR_SUPPORT	用以是否支持 Motion sensor 的开关；不支持为 NONE；支持为所用芯片类型
MAIN_LCD_SIZE	所用 LCD 的大小；如 240X320；176X220;120X160 等
FM_RADIO_CHIP	用以是否支持 FM 功能；不支持为 NONE；支持为所用芯片类型；
NAND_SUPPORT	用以是否支持 NAND FLASH；不支持为 FASLE；支持为 TRUE
PHONE_TYPE	滑盖机: SLIDE；翻盖机：CLAMSHELL；平板机：BAR；旋转：SPIN
PLATFORM_NAME	项目名称；
CAMERA_PIXEL	所用 Sensor 象素；默认 30 万；130 万：ONE_MEGA_PIXEL; 200 万：TWO_MEGA_PIXEL 等等
WEBCAM_SUPPORT	用以是否支持 Web Camera 功能开关；支持为 TRUE;不支持为 FALSE
ect…….	
	其他一些开关视项目而定；可以会修改；可能会添加一些开关。

Custominfo.pl(mcu\tools)
这个文件是当在 mcu\custom\drv 下添加与 LCD，image_sensor 等类似的模块时，需要修改此文件以便 ADS 编译系统能编译到此模块下的文件；
可以参考 image_sensor。

```
if (($project ne "basic") && ($project ne "l1s") && defined($cmos_sensor)) {
  if ($cmos_sensor ne "NONE") {
    push(@thatdirs, "drv\\image_sensor\\$cmos_sensor");
  }
}
```

2. LCM；Backlight；Vibrator
以 SUNRISE_0255_LCM 为例
步骤 1：
MXX_GPRS.mak 中配置 LCD_MODULE 和 MAIN_LCD_SIZE；
LCD_MODULE ＝ SUNRISE_0255_LCM
MAIN_LCD_SIZE = 240X320

Option.mak 中加入：
COM_DEFS_FOR_SUNRISE_0255_LCM = SUNRISE_0255_LCM HX8312 COLOR_LCD

TFT_MAINLCD QVGA_MAINLCD
其中 HX8312 为 LCD 的型号；
COLOR_LCD，TFT_MAINLCD，QVGA_MAINLCD 为 LCD 的类型
QVGA_MAINLCD：240X320
QCIF_MAINLCD: 176X220
QQVGA_MAINLCD: 120X160

如在 SUNRISE_0255_LCM 的基础上添加兼容屏；则在 HX8312 后面顺序加入 LCD 的型号
如果有 Sub LCD；则需要顺序加入 DUAL_LCD；COLOR_SUBLCD 以及型号名称；
步骤 2：
Mcu\custom\drv\LCD 目录下添加 SUNRISE_0255_LCM 模块；目录以 SUNRISE_0255_LCM 为名。
可以参考其他 LCM；加入和修改 5 个文件分别为:
lcd.c
lcd_hw.h
lcd_sw.h
lcd_sw_inc.h
lcd_sw_rnd.h

lcd.c：实现 LCD 的驱动接口
一般有 init； sleep in；sleep out; block write 等
lcd_hw.h： 一般不需要修改；
lcd_sw.h: 配置 data address；command address 和 output format；
lcd_sw_inc.h：配置 LCD WIDTH 和 HEIGHT;
lcd_sw_rnd.h: 一般不需修改；

步骤 3：
加入背光控制代码；
Mcu\custom\drv\misc_drv\custom_equipment.c
```
kal_bool custom_cfg_gpio_set_level(kal_uint8 gpio_dev_type, kal_uint8 gpio_dev_level )
{
   //用 SUNRISE_0255_LCM 来控制代码
}
```

GPIO 模式和 PWM 模式；（函数默认是 PWM 模式）
如果是 GPIO 模式；则在上述函数中加入模拟代码；
如果是 PWM 模式；则需要根据实际情况在 Mcu\custom\drv\misc_drv\custom_hw_default.c 文件中修改 PWM1_Level_Info；即频率和占空比；

另外在 mcu\plutommi\mmi\gpio\gpioSrc\GeneralDeviceInterface.c 中有详细的 MMI 对背光的控制代码；

另外还需熟悉 lcd_if.c（mcu\drv\src）对上述接口的调用；

3．Camera module；Sensor
外挂 DSP 暂不讨论；以 OV9650 为例：
步骤 1:
MXX_GPRS.mak 中配置 ISP_SUPPORT 和 CMOS_SENSOR；
ISP_SUPPORT ＝ TRUE
CMOS_SENSOR = OV9650
CAMERA_PIXEL = ONE_MEGA_PIXEL
注：(一般 30 万象素可以插值到 100 万；130 万可以插值到 200 万象素)

步骤 2:
Mcu\custom\drv\image_sensor 下添加 OV9650 目录；
添加或修改文件：
camera_hw.c：Module Power on/off 控制；（GPIO 分配要看硬件的资源分配表）
camera_para.c： ISP、Sensor 相关寄存器配置；在实际调试中对比效果来修改；
image_sensor.c：timing；sensor init；power on/off；preview；capture 等接口实现；
image_sensor.h：有关 sensor 一些属性设置（一些宏定义）；
camera_info.c： 一般不需修改；

在完成所有的功能之后；效果调试是主要工作；
主要工具：META；CCT 等；
关于 ISP Address 可参考 MTK 平台的 DataSheet；

步骤 3:
需对以下目录的文件有一定的了解！
media\camera\；
media\video\；
plutommi\mtkapp\Camera\
plutommi\mtkapp\Video\
plutommi\mtkapp\MDI\

4．Keypad
按键分布图(可对照特定项目的硬件资源分配表)

	KCOL0	KCOL1	KCOL2	KCOL3	KCOL4	KCOL5	KCOL6
KROW0	KEY_XX	KEY	KEY	KEY	KEY	KEY	KEY
KROW1	KEY	KEY	KEY	KEY	KEY	KEY	KEY
KROW2	KEY	KEY	KEY	KEY	KEY	KEY	KEY
KROW3	KEY	KEY	KEY	KEY	KEY	KEY	KEY
KROW4	KEY	KEY	KEY	KEY	KEY	KEY	KEY
KROW5	KEY	KEY	KEY	KEY	KEY	KEY	KEY

对于特定项目的键盘的定义和映射以及其他一些特殊的处理；代码用 PHONE_TYPE 和 PLATFORM_NAME 的组合来控制；
如 PHONE_TYPE ＝ SLIDE
PLATFORM_NAME ＝ M678

则代码控制如下：
custom\drv\misc_drv\M678_BB\keypad_def.c
const keypad_struct keypad_custom_def = {
#if defined（__PHONE_SLIDE__） //滑盖机型
    #if defined(__SLIDE_M678)
    //add m678 keypad define here
    //ROW 0(第一列)
DEVICE_KEY_XX,
……
//ROW 1(第二列)
…….
//
…….
    #else
    #endif
#elif defined(__PHONE_BAR__) //直板机型
    //
#elif defined(__PHONE_SPIN) //旋转机型
    //
#elif defined(__PHONE_CLAMSHELL) //翻盖机型
    //
#endif
};

如果除了平台提供的基本的按键外还需添加特定的按键定义；则除了上述外还需修改以下几处：
interface\hwdrv\kbd_table.h
定义 DEVICE_KEY_XX；注意顺序
#define DEVICE_KEY_XX 顺序值

plutommi\mmi\Framework\Osl\OslSrc\KeyBrd.c
假设 DEVICE_KEY_XX 为特定的按键
//定义特定的按键
const U16 PresentAllKeys[]=
{
    KEY_0,
    KEY_1,
    KEY_2,
    KEY_3,

KEY_4,
KEY_5,
……
//定义特定的按键
#if defined(__SLIDE_M678)
,KEY_XX
＃endif
};
//键盘映射（注意加入顺序要与 PresentAllKeys[]一致）
static const KeyPadMap nKeyPadMap[] = {
{DEVICE_KEY_0, KEY_0, KEY_TIMER_ID0, TONE_DTMF_0, DEVICE_AUDIO_PLAY_INFINITE},
{DEVICE_KEY_1, KEY_1, KEY_TIMER_ID1, TONE_DTMF_1, DEVICE_AUDIO_PLAY_INFINITE},
……
……
//映射特定的按键
#if defined(__SLIDE_M678)
{DEVICE_KEY_XX, KEY_XX, KEY_TIMER_IDXX, TONE_DTMF_1, DEVICE_AUDIO_PLAY_INFINITE},
＃endif
};
//定义 TIMER ID;即 KEY_TIMER_IDXX
plutommi\mmi\Inc\TimerEvents.h
顺序加入 KEY_TIMER_IDXX；

另外在 drv\src\kbdmain.c
中可以修改 debounce time； Long press Time； Repeat Time 等参数来满足特殊的需要；

5．耳机检测；线控
custom\drv\misc_drv\M678_BB\auxmain.c
对于耳机检测；一般只需关心 AUX_EINT_NO（一般平台默认）和 SENDKEY_ADC（见 auxmain.c）；
可根据硬件的实际情况作一些相关的调整；

对于线控来说；需要配置 REMOTE_EINT_NO、 REMOTE_ADC 以及按键 Press/Release 对应的 ADC 值；
具体的处理过程可参考 26 平台__LINE_CONTROL_EARPHONE_SUPPORT__控制的代码；

6．ADC

MT6226 平台可用资源 ADC0-ADC6；

以线控为例，介绍一下修改或者添加流程：

interface\hwdrv\Bmt.h

```
typedef enum {
   vbat_adc_channel=0,
   visense_adc_channel,
   vbattmp_adc_channel,
……
#if defined(__LINE_CONTROL_EARPHONE_SUPPORT__)
remote_adc_channel,
   #endif
……
} adc_channel_type;
```

custom\drv\misc_drv\M678_BB\adc_channel.c

```
#if defined(__LINE_CONTROL_EARPHONE_SUPPORT__)
const kal_uint8 ADC_REMOTE=6;  //假设硬件接 ADC6
#endif

kal_uint8 custom_adc_get_channel(adc_channel_type type)
{
    Switch(type)
{
    ……
#if defined(__LINE_CONTROL_EARPHONE_SUPPORT__)
        case remote_adc_channel:
            return ((kal_uint8)ADC_REMOTE);
        #endif
    ……
}
#endif
};
```

custom\drv\misc_drv\M678_BB\auxmain.c

```
void aux_task_main( task_entry_struct * task_entry_ptr )
{
#if defined(__LINE_CONTROL_EARPHONE_SUPPORT__)
kal_uint8           remote_adc_logic_id;
kal_uint8          remote_adc_no
#endif
……
……
```

```
//创建
#if defined(__LINE_CONTROL_EARPHONE_SUPPORT__)
remote_adc_no   = custom_adc_get_channel(remote_adc_channel)
          remote_adc_logic_id    =    adc_sche_create_object(MOD_AUX,    remote_adc_no,40,1,
KAL_TRUE);
    #endif
    ……
    while(1)
{
receive_msg_ext_q(task_info_g[task_entry_ptr->task_indx].task_ext_qid, &current_ilm);
     switch(current_ilm.msg_id)
     {
    ……
    //读 ADC
    case MSG_ID_READ_ALL_ADC_CHANNEL_REQ:
    ……
       #if defined(__LINE_CONTROL_EARPHONE_SUPPORT__)
        aux_read_adc_channel(remote_adc_logic_id);
        #endif
……
break;
    ……
    //销毁
    case MSG_ID_BMT_ADC_MEASURE_DONE_CONF:
       ……
        #if defined(__LINE_CONTROL_EARPHONE_SUPPORT__)
        else if (mea_done_ptr->adc_sche_id == remote_adc_logic_id)
       {
           adc_measure_count++;
           remote_value = (kal_int32)mea_done_ptr->volt;
           aux_remove_adc_channel(remote_adc_logic_id);
       }
       #endif
       ……
}
}
}
```

CLAMSHELL 中断的 REGISTOR 可以在上述函数中添加

7．EINT

关于 EINT 的描述请参考平台的 datasheet;
以线控为例，介绍一下修改或者添加流程：

interface\hwdrv\Eint.h:

```
typedef enum
{
    ……
     #if defined(__LINE_CONTROL_EARPHONE_SUPPORT__)
    remote_eint_chann,
    #endif
    ……
} eint_channel_type;
```

custom\drv\misc_drv\M678_BB\Eint_def.c

```
#if defined(__LINE_CONTROL_EARPHONE_SUPPORT__)
    const kal_uint8 REMOTE_EINT_NO=3;   //假设硬件配置为 EINT3
#endif

kal_uint8 custom_eint_get_channel(eint_channel_type type)
{
    switch(type)
    {
       ……
       #if defined(__LINE_CONTROL_EARPHONE_SUPPORT__)
       case remote_eint_chann:
           return ((kal_uint8)REMOTE_EINT_NO);
       #endif
       ……
}
}
```

custom\drv\misc_drv\M678_BB\auxmain.c

```
//中断处理函数
void REMOTE_EINT_HISR(void)
{
    if (remote_state) //高电平
    {
       //相关处理代码
    }
    else//低电平
    {
       //相关处理代码
    }
}
```

//中断注册
```
void aux_task_main( task_entry_struct * task_entry_ptr )
{
    kal_uint8 remote_eint_no;
    ……
#if defined(__LINE_CONTROL_EARPHONE_SUPPORT__)
    remote_eint_no = custom_eint_get_channel(remote_eint_chann);
            EINT_Registration(remote_eint_no,KAL_TRUE,remote_state,REMOTE_EINT_HISR,
KAL_TRUE);
    #endif
……
}
```
在注册中断时要注意：
中断触发方式：电平触发/边沿触发？
debounce: Enable/Disable？
电平触发高电平有效还是低电平有效等问题。

对于 Eint0 － Eint3；可以通过下面的文件来修改 debounce time；而 Eint4 － Eint7 没有 debounce 机制
custom\drv\misc_drv\M678_BB\Eint_def.c
```
kal_uint8 custom_eint_sw_debounce_time_delay[EINT_MAX_CHANNEL] =
{
   50,    /*EINT0*/
   25,    /*EINT 1*/
   50,    /*EINT2*/
   50     /*EINT3*/
};
```
单位：50ms

也可通过下述方法来修改
EINTaddr()
EINT_Set_HW_Debounce()
EINT_SW_Debounce_Modify()
等函数来修改

8. Charger/USB
可参考 Customer_BMT_V0.1.pdf
相关文件：
custom\drv\misc_drv\M678_BB\chr_parameter.c　关注点：
bmt_customized_struct　bmt_custom_chr_def[] = {};
此为充电相关电压和电流的设置；

static const kal_int32 chr_usb_detect_volt;
此为 CHARGER/USB 检测的电压值；

Mcu\bmt*.*
Mcu\drv\src\pwic.c

9．Touch panel
custom\drv\misc_drv\M678_BB\touch_panel_custom.c
custom\drv\misc_drv\M678_BB\touch_panel_custom.h

custom\drv\misc_drv\M678_BB\touch_panel_spi.c
custom\drv\misc_drv\M678_BB\touch_panel_spi.h
drv\src\touch_panel.c
drv\src\touch_panel_main.c
修改点包括：
GPIO 分配；ADC 值；坐标值；中断配置；压力检测等
具体修改可参考代码；

10. AFE
custom\audio\M678_BB\afe.c（模拟开关、PA 切换等）
custom\audio\M678_BB\audcoeff.c（FIR Input/Output 参数）
custom\audio\M678_BB\nvram_default_audio.c（GAIN 值）
l1audio\afe2.c (AFE 管理代码)

以上根据硬件给出的数据来进行调整！

11．Task
如何在 MTK 平台使用 Task？
以 DMB 项目为例；请参照下述文件；DMB_SUPPPORT 控制
custom\system\M678_BB\custom_config.c
custom\system\M678_BB\custom_config.h
custom\drv\DMB\TCC78X\dmb_hw.c

12 Nor/Nand Flash; T-Flash
添加新的 NAND：
Drv\src\NAND_MTD.c
static const flash_list NAND_ID_Table[] =
{

//{ ID, planesize in MB, blocksize in KB, pagesize in B, address cycle, IO bus width, mtd sub driver}

//添加新的 NAND 的相关信息（参考芯片 Spec）
}
DA（Download Agent）部分代码（一般不需修改）
DA_SRC\src\nand_dev_tbl.c
Const NAND_Device_S g_NandFlashDevTbl[] =
{
    //加入新的 NAND 信息（参考芯片 Spec）
};

添加新的 Nor Flash
DA_SRC\src\flash_dev_tbl.c
Const Nor_Device_S g_FlashDevTbl[] =
{
    //加入新的 Nor 信息（参考芯片 Spec）
};

对于 DA 的修改；需重新生成 DA 文件用于 Flash Download Tool

13. 其他功能调试
    其他的调试可根据实际情况参考本平台或者其他平台。