ZLS38100 Build Guide on Omnivision 788B SDK

Version 1.0
Jun 2016

Table of Contents

Contents

List of Tables

List of Figures

No table of figures entries found.

Document Owner	Shally Verma
Version	1.0
Date	June 2016

Document Change History			
Date	**Version**	**Changed by**	**Change Description**

Abbreviations:

SDK	Software Development Kit
OV	OmniVision
VPROC	Voice Processor
HBI	Host Bus Interface
SSL	System Service Layer

Table 1 Abbreviations

1. Purpose

This document describes the steps for integrating ZLS38100 SDK with Omnivision SDK `OV788_SDK_FW.49165.pdk.tgz`.

2. Disclaimer

Please note that this document describes the steps that were followed to build ZLS38100 SDK with Omnivision SDK and therefore must be considered only as a reference, or example implementation, or as getting started guide. Different users may choose to integrate it differently as per their development setup.

3. OmniVision Development System

SW SDK	`OV788_SDK_FW.49165.pdk.tgz`
HW Platform	`OV 788 B V2 board, OV 788 MP EVB 1V1`
OV SDK toolchain	`OV78x_toolchain_V10201.bin`

Table 2 OmniVision Development Platform

4. Assumptions

The build instruction in this document is only tested for OmniVision SDK mentioned in section `Omnivision development system`. It may not be relevant for any other development platform which is incompatible to the mentioned SDK.

This document assumes that you have OV SDK untarred and toolchain built.

5. ZLS38100 SDK Release Package

ZLS38100 SDK Release Package consists of following main files.

Files	Description
Makefile.globals	System hardware and software configurations are defined here. Example, host endian can be big or little. Maximum number of vproc devices in a systems etc. User may omit the options not relevant to their system. User may port this file as per their development system.
config.mk	Converts the Makefile.globals variables to ‘C’ compiler options and add relevant include paths. User may port this file as per their development system
drivers/hbi	VPROC HBI Driver.
platform/omnivision/driver/	VPROC I2S and SPI/I2C Driver specific to development platform.
platform/omnivision/apps	Sample applications for OV platform

Table 3 Release Package

6. Building ZLS38100 SDK with OV SDK

Make sure to define relevant variables in Makefile.globals. Important ones include:

```
HOST_ENDIAN=big
HBI=I2C (if using i2c interface. If not mentioned, by default driver
        will be compiled for SPI)
```

6.1 Copy `config.mk` and `Makefile.globals` in OV SDK make folder OR import those defines into OV SDK makefiles so that they are visible to `ssl` and `hbi` driver

6.2 Make sure you have PATH variable properly set to OV toolchain

6.3 Building HBI Driver

6.3.1 Make a directory in `OV_SDK/share` with any name ex. “`zls380xx or zls38100`” and copy followings from ZLS38100 package to this new created directory. For example sake, we create `OV_SDK/share/zl380xx` directory

6.3.1.1 `drivers/hbi/*.c/*.h`
6.3.1.2 `include/*.h`
6.3.1.3 `platform/omnivision/driver/ssl/*.c` and `*.h`
6.3.1.4 `platform/omnivision/include/*.h`
6.3.1.5 Make sure you have `ssl, hbi and all dependent .h` file inside `zl380xx directory`

6.3.2 Add `Makefile` to compile `OV_SDK/share/zl380xx` driver with OV SDK. `Makefile` should have `config.mk` and `Makefile.globals` path included. See Appendix for an example zl380xx Makefile

6.4 Building Audio Codec Driver

6.4.1 Copy `platform/omnivision/driver/sound/codec_zl380xx.c` in to `OV SDK/share/acodec_settings`. Modify `kconfig` to add zl380xx in menu listing. See Appendix for an example kconfig file.

6.4.2 Modify `OV SDK/make/config.env` to include `zl380xx`(directory name containing HBI driver).

```
ifeq ($(AUDIO_CODEC), zl380xx)
IN_CFLAGS += -I$(CODEDIR)/share/zl380xx/
endif
```

6.5 Building ZL380xx Application

This folder contains a sample application with example reference code to test HBI driver and its feature.

6.5.1 Copy `platform/omnivision/apps/zl380xx_util` into `OV SDK/ram/`
6.5.2 Copy `zl380xx_util.config` to `OV SDK/prj/r2/zl380xx_util/`

Below table summarizes the different sample application inside `zl380xx_util` and configuration options

Apps	Description	
hbi_test	Contains basic HBI register read/write tests. Tests can be enabled / disabled through #define macros in `hbi_test.c` file. **Table-5** summarizes current test macros. User can add	Enabled by defining `CONFIG_ZL380XX_TEST_HBI=y` in `zl380xx_util.config`

	more based on their test requirements.	
hbi_ldfwcfg	Contains firmware and configuration record loading through HBI. User can boot voice processor device by downloading an S-record based boot image or a C-style based firmware image. Test code can be compiled to boot load either of the image format type. **Table-6** summarizes the option influencing test behavior which can be enabled/disabled through `zl380xx_util.config`	Enabled by defining `CONFIG_ZL380XX_HBI_LOAD_FWRCFG=y in zl380xx_util.config`

Table 4 ZL380xx_util Sample Applications

Macro	Description
`TEST_RST`	Tests hbi_reset(). Issues hard reset to device
`TEST_LOAD_FWRCFG_FROM_FLASH`	Tests HBI_CMD_LOAD_FWRCFG_FROM_FLASH. Read firmware and configuration record from flash.
`TEST_ERASE_FLASH`	Erases whole flash

Table 5 hbi_test Case Configure Options

Macro	Description
`CONFIG_ZL380XX_HBI_BOOT`	If defined, boot image load code is enabled
`CONFIG_ZL380XX_HBI_BOOT_STATIC`	If defined, test code uses the C-style based firmware image. User need to set `FWR_C_FILE` variable in `zl380xx_util/Makefile` to base file name of 'C' firmware image as generated using firmware conversion tool in firmware/tool. If this config option is disabled, firmware will read `*.s3` image from sd-card. User need to modify test case with *.s3 filename in file open call.
`CONFIG_ZL380XX_HBI_LOAD_CFGREC`	If defined, configuration record loading code is enabled
`CONFIG_ZL380XX_HBI_LOAD_CFGREC_STATIC`	If defined, test code uses C-style based configuration record file. User need to set `CFGREC_C_FILE` variable in test `Makefile` to base file name of C-based configuration record file generated using conversion tool in firmware/tool. If this config options is disabled, test code

	would read a `*.cr2` file from SD-Card
`SAVE_IMAGE_TO_FLASH (defined in hbi_ldfwcfg.c)`	If defined, loaded firmware and configuration record will be saved to flash.
`CONFIG_ZL380XX_HBI_I2C`	If defined, test code will open I2C as HBI else SPI. User need to update bus number and device id variable in test file as per their system.

Table 6 hbi_ldfwcfg Configure Options

6.6 Building hbi_ldfwcfg

6.6.1 Define `CONFIG_ZL380XX_HBI_LDFWRCFG=y` in `zl380xx_util.config` file
6.6.2 Define `CONFIG_ZL380XX_HBI_BOOT=y` for boot image load
6.6.3 Define `CONFIG_ZL380XX_CFGREC_LOAD`=y for configuration record loading
6.6.4 If using C-based image, define `CONFIG_ZL380XX_HBI_BOOT_STATIC=y` and copy `firmware/*_firmware.c and *.h` into `OV SDK/ram/zl380xx_util` else copy `firmware/*.s3 image in to sd-card`
6.6.5 If using C-based configuration record file, define `CONFIG_ZL380XX_CFGREC_LOAD_STATIC=y`, then copy `firmware/*.config.c and .h` into `OV SDK/ram/zl380xx_util` else copy `.cr2 file in sd-card`
6.6.6 Modify `zl380xx_util/Makefile` to define proper path to `Makefile.globals and config.mk and SUBDIRS,CFLAGS and IN_CFLAGS` variable pointing to directory as described in `section 6.3 Building HBI Driver.`
6.6.7 Initiate `zl380xx_util` build from root by giving
`./autobuild_r2mp.sh zl380xx_util zl380xx_util`

6.7 Building hbi_test

6.7.1 Define `CONFIG_ZL380XX_HBI_TEST=y` in `zl380xx_util config` file
6.7.2 Modify `zl380xx_util/Makefile` to define proper path to `Makefile.globals and config.mk and SUBDIRS,CFLAGS and IN_CFLAGS` variable pointing to directory as described in `section 6.3 Building HBI Driver`
6.7.3 Initiate sample app build from root by `giving`
`./autobuild_r2mp.sh zl380xx_util zl380xx_util`

6.8 Building dual_audio app for zl380xx_codec

6.8.1 Modify `dual_audio.config` in `OV_SDK/prj/r2/` with option

```
CONFIG_AUDIO_CODEC_EN=y
CONFIG_AUDIO_CODEC_EN_zl380xx=y
```

6.8.2 Modify dual_audio `Makefile` to include `zl380xx` into OBJS. See Appendix for example `dual_audio makefile`

Appendix

A. Example Makefile for zl380xx (HBI/SSL) Driver

```
include $(ROOTDIR)/Makefile.globals
include $(ROOTDIR)/config.mk
help:
  echo "ROOTDIR"$(ROOTDIR)
  echo "HBI"$(HBI)

IN_CFLAGS += $(EXTRA_CFLAGS)
ifeq ($(HBI),I2C)
  OBJS += hal_i2c.o
else
  OBJS += hal_spi.o
endif

OBJS += ssl.o
OBJS += hbi_tw.o hbi.o
```

B. Example kconfig file for zl380xx acodec driver

```
menu "Audio Codec Support"

config AUDIO_CODEC_EN
  bool "Audio Codec enable"
  help
    add audio codec support

if AUDIO_CODEC_EN
choice
  prompt "Audio external CODEC to link"
  help
    select the CODEC

  config AUDIO_CODEC_EN_wm8960
    bool "wm8960 DAC&ADC"
  config AUDIO_CODEC_EN_wm8960_kd
    bool "wm8960_kd DAC&ADC"
  config AUDIO_CODEC_EN_da7323
    bool "da7323 DAC&ADC"
     config AUDIO_CODEC_EN_zl380xx
    bool "zl380xx DAC&ADC"
endchoice
endif



config AUDIO_CODEC_DAC_EN
  bool "Audio Codec DAC enable"
  default y if AUDIO_DEC_EN
  default n if AUDIO_ENC_EN
  help
    add audio codec DAC support

if AUDIO_CODEC_DAC_EN
```

```
choice
  prompt "Audio external DAC to link"
  help
    select the DAC

  config AUDIO_DAC_EN_wm8955
    bool "wm8955 DAC"
  config AUDIO_DAC_EN_wm8978
    bool "wm8978 DAC"
  config AUDIO_DAC_EN_ti3100
    bool "ti3100 DAC"
  config AUDIO_DAC_EN_aic3256
    bool "aic3256 DAC"
  config AUDIO_DAC_EN_nau8814
    bool "nau8814 DAC"
  config AUDIO_DAC_EN_wm8960
    bool "wm8960 DAC"
  config AUDIO_DAC_EN_ht82v731
    bool "ht82v731 DAC"
  config AUDIO_DAC_EN_hdmi_sii9022
    bool "hdmi sii9022 DAC"
  config AUDIO_DAC_EN_aic3262
    bool "TI AIC3262 DAC"
  config AUDIO_DAC_EN_da7323
    bool "da7323 DAC"
  config AUDIO_DAC_EN_da7323_halfduplex
    bool "da7323 HalfDuplex"
endchoice

endif

config AUDIO_CODEC_ADC_EN
  bool "Audio Codec ADC enable"
  default y if AUDIO_ENC_EN
  default n if AUDIO_DEC_EN
  help
    add audio codec ADC support

if AUDIO_CODEC_ADC_EN

choice
  prompt "Audio external ADC to link"
  help
    select the external ADC

  config AUDIO_ADC_EN_wm8978
    bool "wm8978 ADC"
  config AUDIO_ADC_EN_nau8814
    bool "nau8814 ADC"
  config AUDIO_ADC_EN_wm8960
    bool "wm8960 ADC"
  config AUDIO_ADC_EN_aic3256
    bool "aic3256 ADC"
  config AUDIO_ADC_EN_dm
    bool "digital mic ADC"
  config AUDIO_ADC_EN_aic3262
    bool "TI AIC3262 ADC"
  config AUDIO_ADC_EN_da7323
    bool "da7323 ADC"
endchoice

endif

if AUDIO_DAC_EN_da7323 || AUDIO_ADC_EN_da7323
||AUDIO_CODEC_EN_da7323

config DA7323_DIAGNOSTICS
  bool "enable DA7323 diagnostics"
  help
    print out more information during calls to codec API

config DA7323_DSP_BYPASS
  bool "DA7323 ignore the DSP totally"
  help
    the bypass mode uses a smaller footprint

if ! DA7323_DSP_BYPASS

choice
  prompt "DSP Graph to be used"
  default DA7323_DSP_GRAPH_0D

  config DA7323_DSP_GRAPH_0B
    bool "0B == Generic mixer with no audio processing"

  config DA7323_DSP_GRAPH_0D
    bool "0D == Acustic Echo Cancellation at 16k and Noise
Suppresion DSP function"

  config DA7323_DSP_GRAPH_0E
    bool "0E == Acustic Echo Cancellation at 16k and Noise
Suppresion DSP function"

  config DA7323_DSP_GRAPH_10
    bool "10 == Acustic Echo Cancellation at 11k and Noise
Suppresion DSP function"

  config DA7323_DSP_GRAPH_11
    bool "11 == Generic mixer with no audio processing for 1
MIC(Mono-Stereo)"

  config DA7323_DSP_GRAPH_12
    bool "12 == AEC/NS @11k for MIC1 (MonoToStereo)"


endchoice

choice
  prompt "I2C block write mode used"
  default DA7323_I2C_SINGLE_MODE
  help
```

```
     There are three possible modes supported

   config DA7323_I2C_SINGLE_MODE
     bool "single I2C byte writes"
     help
       There are 3 ways that multiple DA7323 codec registers can be
written.
       The simplest (and slowest) is to just write each register by
itself
       in a single I2C transaction.
       The standard block mode can be used to transfer a sequence of
contigueous
       registers. Typically 24 or 32 bit DSP registers are transfered
in this way.
       The fast transfer mode is burst mode where multiple sets of 3
or 4 byte
       contiguous registers byte are chained together and the codec
itself keeps
       track of which DSP address is actually being targetted.
       Note that OmniVision's SCCB is an implementation of the I2C
protocol.
   config DA7323_I2C_BLOCK_MODE
     bool "standard multiple I2C byte block writes"
     help
       There are 3 ways that multiple DA7323 codec registers can be
written.
       The simplest (and slowest) is to just write each register by
itself
       in a single I2C transaction.
       The standard block mode can be used to transfer a sequence of
contigueous
       registers. Typically 24 or 32 bit DSP registers are transfered
in this way.
       The fast transfer mode is burst mode where multiple sets of 3
or 4 byte
       contiguous registers byte are chained together and the codec
itself keeps
       track of which DSP address is actually being targetted.
       Note that OmniVision's SCCB is an implementation of the I2C
protocol.
   config DA7323_I2C_BURST_MODE
     bool "special multiple I2C byte burst writes"
     help
       There are 3 ways that multiple DA7323 codec registers can be
written.
       The simplest (and slowest) is to just write each register by
itself
       in a single I2C transaction.
       The standard block mode can be used to transfer a sequence of
contigueous
       registers. Typically 24 or 32 bit DSP registers are transfered
in this way.
       The fast transfer mode is burst mode where multiple sets of 3
or 4 byte
       contiguous registers byte are chained together and the codec
itself keeps
```

```
      track of which DSP address is actually being targetted.
      Note that OmniVision's SCCB is an implementation of the I2C
protocol.

endchoice

endif
endif
if AUDIO_DAC_EN_aic3262 || AUDIO_ADC_EN_aic3262
choice
  prompt "Choose control interface for AIC3262"
  config AIC3262_USE_SCCB
    bool "sccb"
  config AIC3262_USE_SIF
    bool "SIF"
endchoice

if AIC3262_USE_SIF
choice
  prompt "Choose SIF interface for AIC3262"
  help
    select the correct SIP interface, choose one of the four SIF
interface
  config AIC3262_SIF0_GPIO_NORMAL
    bool "SIF0 GPIO shared pins"
  config AIC3262_SIF0_SC_NORMAL
    bool "SIF0 SC shared pins"
  config AIC3262_SIF1_NORMAL
    bool "SIF1 NORMAL pins"
  config AIC3262_SIF1_SI_NORMAL
    bool "SIF1 SI Shared pins"
endchoice
choice
  prompt "Choose CS PIN"
  help
    select cs0 or cs1 for current SIF
  config SIF_CS0
    bool "select cs0"
  config SIF_CS1
    bool "selct cs1"
endchoice
endif
choice
  prompt "Choose input channel "
  config AIC3262_USE_IN1
    bool "IN1L IN1R"
  config AIC3262_USE_IN2
    bool "IN2L IN2R"
endchoice
endif

endmenu
```

C. Example dual_audio.cfg

```
#
# Automatically generated make config: don't edit
# Configuration
#

#
# APP(dual_audio) configuration
#
CONFIG_SAMPLE_RATE=11025

#
# Hardware configuration
#

#
# System Hardware
#
CONFIG_PADCLK_24M=y
# CONFIG_PADCLK_12M is not set
# CONFIG_SYSCLK_PADCLK is not set
# CONFIG_SYSCLK_PLL60M is not set
CONFIG_SYSCLK_PLL144M=y
# CONFIG_SYSCLK_PLL180M is not set
# CONFIG_USBDEBUG is not set
# CONFIG_TICK2_EN is not set

#
# Image sensor support
#
# CONFIG_S_EN is not set

#
# serial flash support
#
# CONFIG_HAS_SF is not set

#
# Nandflash support
#
# CONFIG_HAS_NAND is not set

#
# display support
#
# CONFIG_DISPLAY_OUT_EN is not set

#
# 2.4G rf module support
#
# CONFIG_RF_EN is not set

#
# RTC support
#
```

```
# CONFIG_RTC_EN is not set

#
# GSensor support
#
# CONFIG_GSENSOR_EN is not set

#
# WIFI support
#
# CONFIG_WIFI_EN is not set

#
# WiFi work mode
#

#
# StorageCard support
#
CONFIG_FS_SCIF=y
# CONFIG_APP_USE_SCIF02 is not set
# CONFIG_SCIF_DETECT_REVERSE is not set
# CONFIG_SCIF_USE_INTR is not set

#
# Audio configuration
#
CONFIG_AUDIO_EN=y
#CONFIG_AUDIO_DEC_EN=y
#CONFIG_AUDIO_DFMT_F1=y
CONFIG_AUDIO_ENC_EN=y
CONFIG_AUDIO_EFMT_F1=y
# CONFIG_AUDIO_EFMT_F2 is not set
# CONFIG_AUDIO_EFMT_F3 is not set
# CONFIG_AUDIO_EFMT_F4 is not set
# CONFIG_AUDIO_EFMT_F5 is not set
# CONFIG_AUDIO_EFMT_F6 is not set
# CONFIG_AUDIO_EFMT_F7 is not set
# CONFIG_AUDIO_EHW_AI1 is not set
CONFIG_AUDIO_EHW_AI2=y
# CONFIG_AUDIO_EPRE_EN is not set

#
# Audio Codec Support
#
CONFIG_AUDIO_CODEC_EN=y
#CONFIG_AUDIO_CODEC_EN_wm8960=y
CONFIG_AUDIO_CODEC_EN_zl380xx=y
#CONFIG_ZL380XX_HBI_BOOT=y
#CONFIG_ZL380XX_LOAD_CFGREC=y

# CONFIG_AUDIO_CODEC_EN_wm8960_kd is not set
# CONFIG_AUDIO_CODEC_EN_da7323 is not set
# CONFIG_AUDIO_CODEC_DAC_EN is not set
# CONFIG_AUDIO_CODEC_ADC_EN is not set
# CONFIG_DA7323_DSP_GRAPH_03 is not set
# CONFIG_DA7323_DSP_GRAPH_0B is not set
# CONFIG_DA7323_DSP_GRAPH_10 is not set
# CONFIG_DA7323_I2C_SINGLE_MODE is not set
# CONFIG_DA7323_I2C_BLOCK_MODE is not set
# CONFIG_DA7323_I2C_BURST_MODE is not set

#
# USB configuration
#

#
# USB Mass-Storage support
#
# CONFIG_USB_UMASS is not set

#
# USB Printer Support
#
# CONFIG_USB_PRINTER is not set

#
# USB UVC/UAC support
#
# CONFIG_USB_UVC is not set
# CONFIG_USB_UAC is not set

#
# USB Host support
#
# CONFIG_USB_HOST is not set

#
# StillImage support
#
# CONFIG_LIBCAPTURE_PREVIEW_EN is not set
# CONFIG_LIBCAPTURE_NOPREVIEW_EN is not set
# CONFIG_LIBIMGPLAY_EN is not set

#
# Video Engine support
#
# CONFIG_LIBVENC_EN is not set
# CONFIG_MIMG_EN is not set
# CONFIG_SINGLE_MIMG is not set
# CONFIG_LIBVDEC_EN is not set
# CONFIG_LIBMIMG_PLAY_EN is not set
# CONFIG_LIBVENC_YUVCPY_BA2 is not set

#
# FACEDETECT support
#
# CONFIG_LIBFACEDETECT_EN is not set

#
# Net Mac Support
#
```

```
# CONFIG_LIBMAC_EN is not set

#
# Networking Support
#

#
# uIP supprot
#
# CONFIG_LIBUIP_EN is not set
# CONFIG_DNS_EN is not set

#
# LWIP support
#
# CONFIG_LIBLWIP_EN is not set

#
# WVTP support
#
# CONFIG_LIBWVTP_EN is not set

#
# MJPGSTREAM support
#
# CONFIG_LIBMJPGSTREAM_EN is not set

#
# HTTPD support
#
# CONFIG_LIBHTTPD_EN is not set

#
# ALDG support
#
# CONFIG_BROADCAST_PUSH is not set
CONFIG_BROADCAST_PULL=y
# CONFIG_AV_FDIST is not set

#
# WPS support
#
# CONFIG_LIBWPS_EN is not set

#
# RTP support
#
# CONFIG_LIBRTP_EN is not set

#
# STUN support
#
# CONFIG_LIBSTUN_EN is not set

#
# Bluetooth support
```

```
#
# CONFIG_LIBBT_EN is not set

#
# 2.4G RF FHSS transfer protocol select
#
# CONFIG_RF_FHSS_EN is not set

#
# firmware general configuration
#
CONFIG_NEWOS_EN=y
# CONFIG_FILETASK_EN is not set
# CONFIG_UAVTASK_EN is not set
# CONFIG_AVTASK_EN is not set
# CONFIG_NOOS_EN is not set
CONFIG_CLEAR_SHARE_DATA=y
CONFIG_UART_PRINT=y

#
# BOARD and prj_share configuration
#
CONFIG_BOARD_OBJS=""
# CONFIG_NO_LIBPRJSHARE is not set
CONFIG_PRJSHARE=""
# CONFIG_PRJSHARE_NOT_LINK is not set
CONFIG_BOARD_EXTRA_HEADER=""
CONFIG_APP_LDFLAGS=""
```

D. Example OV_SDK/ram/dual_audio/Makefile to make zl380xx acodec and zl380xx HBI driver

```
OBJS = main.o except_int.o task_one.o task_two.o

SUBDIRS += $(CODEDIR)/share/zls38100/
IN_CFLAGS +=

$(CODEDIR)/share/zls38100

ifeq ($(CONFIG_AUDIO_ENC_EN), 1)

#EXTRA_OBJS += $(BLDDIR)/share/liba_enc.o
#EXTRA_OBJS += $(BLDDIR)/share/liba_ehw2.o
#EXTRA_OBJS += $(BLDDIR)/share/liba_ehw_codec.o
ifeq ($(CONFIG_AUDIO_EFMT_F1), 1)
#EXTRA_OBJS += $(BLDDIR)/share/liba_epost_f111k.o
IN_CFLAGS += -DAUDIO_REC_PCM
#OBJS += 8ks_1s.o
endif
ifeq ($(CONFIG_AUDIO_EFMT_F4), 1)
#EXTRA_OBJS += $(BLDDIR)/share/liba_epost_f4.o
IN_CFLAGS += -DAUDIO_REC_AAC
#OBJS += 44ks_aac.o
endif
ifeq ($(CONFIG_AUDIO_EFMT_F3), 1)
#EXTRA_OBJS += $(BLDDIR)/share/liba_epost_f3.o
IN_CFLAGS += -DAUDIO_REC_AF3
```

```
#OBJS += 44ks_af3.o
endif
ifeq ($(CONFIG_AUDIO_EFMT_F2), 1)
#EXTRA_OBJS += $(BLDDIR)/share/liba_epost_f2.o
IN_CFLAGS += -DAUDIO_REC_ADPCM
#OBJS += audio_adpcm.o
endif
ifeq ($(CONFIG_AUDIO_EFMT_F5), 1)
#EXTRA_OBJS += $(BLDDIR)/share/liba_epost_f5.o
IN_CFLAGS += -DAUDIO_REC_G711
endif
ifeq ($(CONFIG_AUDIO_EFMT_F6), 1)
#EXTRA_OBJS += $(BLDDIR)/share/liba_epost_f2.o
IN_CFLAGS += -DAUDIO_REC_ADPCM
#OBJS += audio_adpcm.o
endif
ifeq ($(CONFIG_AUDIO_EFMT_F7), 1)
#EXTRA_OBJS += $(BLDDIR)/share/liba_epost_f111k.o
IN_CFLAGS += -DAUDIO_REC_PCM
#OBJS += 8ks_1s.o
endif

endif

ifeq ($(CONFIG_AUDIO_DEC_EN), 1)
#EXTRA_OBJS += $(BLDDIR)/share/liba_dec.o
#EXTRA_OBJS += $(BLDDIR)/share/liba_dhw_codec.o
#EXTRA_OBJS += $(BLDDIR)/share/liba_dhw2.o
ifeq ($(CONFIG_AUDIO_DFMT_F1), 1)
IN_CFLAGS += -DAUDIO_PLAY_PCM
#EXTRA_OBJS += $(BLDDIR)/share/liba_dec_f1.o
#OBJS += 16km_pcm.o
endif
ifeq ($(CONFIG_AUDIO_DFMT_F2), 1)
IN_CFLAGS += -DAUDIO_PLAY_ADPCM
OBJS += audio_adpcm.o
#EXTRA_OBJS += $(BLDDIR)/share/liba_dec_f2.o
endif
ifeq ($(CONFIG_AUDIO_DFMT_F4), 1)
IN_CFLAGS += -DAUDIO_PLAY_AAC
OBJS += 44ks_aac.o
#EXTRA_OBJS += $(BLDDIR)/share/liba_dec_f4.o
endif
ifeq ($(CONFIG_AUDIO_DFMT_F3), 1)
IN_CFLAGS += -DAUDIO_PLAY_AF3
OBJS += 44ks_af3.o
#EXTRA_OBJS += $(BLDDIR)/share/liba_dec_f3.o
endif
ifeq ($(CONFIG_AUDIO_DFMT_F5), 1)
IN_CFLAGS += -DAUDIO_PLAY_G711
#EXTRA_OBJS += $(BLDDIR)/share/liba_dec_f5.o
endif
ifeq ($(CONFIG_AUDIO_DFMT_F6), 1)
IN_CFLAGS += -DAUDIO_PLAY_ADPCM_IMA
endif
ifeq ($(CONFIG_AUDIO_DFMT_F7), 1)
```

```
IN_CFLAGS += -DAUDIO_PLAY_PCM
endif
endif


ifeq ($(CONFIG_AUDIO_ENC_EN), 1)
ifneq ($(CONFIG_AUDIO_DEC_EN), 1)
IN_CFLAGS += -DAUDIO_REC
OBJS += arec_test.o
endif
endif

ifeq ($(CONFIG_AUDIO_DEC_EN), 1)
ifneq ($(CONFIG_AUDIO_ENC_EN), 1)
IN_CFLAGS += -DAUDIO_PLAY
OBJS += aplay_test.o
endif
endif

halfdulex:=1
#halfdulex:=0
ifeq ($(CONFIG_AUDIO_DEC_EN), 1)
ifeq ($(CONFIG_AUDIO_ENC_EN), 1)
IN_CFLAGS += -DAUDIO_REC_PLAY
ifeq ($(halfdulex), 1)
OBJS += half_aduplex_test.o
endif
ifeq ($(halfdulex), 0)
OBJS += aduplex_test.o
endif
ifeq ($(CONFIG_AUDIO_EFMT_F4), 1)
IN_CFLAGS += -DAUDIO_REC_AAC
endif
ifeq ($(CONFIG_AUDIO_EFMT_F1), 1)
IN_CFLAGS += -DAUDIO_REC_PCM
endif
endif
endif


EXTRA_OBJS += $(BLDDIR)/share/libfs_scif.o
```