

PR7261 FI1 Software Reference

Publication: CL/PR7261/UM/1 January
2016

PR7261 C Project

Contents

1	Introduction.....	4
1.1	History.....	4
2	File Index	5
2.1	File List	5
3	Code Introduction	6
4	Run Modes	6
5	Requirements	7
6	Basics	7
6.1	Executing using the EC0003 GUI	9
7	Settings Configuration	10
7.1	CONFIG 1 Encoder G729A Packed - PCM 16Bits signed.....	12
7.2	CONFIG 2 Decoder PCM 16Bits signed – G729A Packed.....	12
7.3	CONFIG 3 Transcoder G729A Packed - CVSD 32KHz Packed.....	12
8	File Documentation.....	14
8.1	inc/PR7261_Tools.h File Reference	14
8.1.1	Functions.....	14
8.1.2	Function Documentation	14
8.2	inc/ProgTools.h File Reference	16
8.2.1	Functions.....	16
8.2.2	Function Documentation	17
8.3	inc/Registers.h File Reference	24
8.3.1	Macros	24
8.3.2	Macro Definition Documentation	28
8.4	inc/Settings.h File Reference	41
8.4.1	Macros	41
8.4.2	Macro Definition Documentation	42
9	Supporting Documentation	45

1 Introduction

This document provides a description of the PR7261 C Project to create an application for CMX7261 evaluation. The project provides some typical CMX7261 tests and guidelines for alternative configuration of the PR72161 Project. This document has been created using Doxygen – an automatic documentation generation tool used to produce software reference documents. Content is created from within the code itself and therefore offers intuitive cross referencing between the document and code and provides an easy path to future updating.

1.1 History

Version	Changes	Date
1	First Release	8 January 2016

2 File Index

2.1 File List

The following is a list of all files. Brief descriptions are given for each in the relevant sections.

inc/PR7261_Tools.h

inc/ProgTools.h

inc/Registers.h

inc/Settings.h

3 Code Introduction

The PR7261 Project uses `lpc_chip_43xx`, `PE0003_DriverLib` and `CML_UtilsLib` libraries. These libraries divide the PR7261 Project into layers with the following purposes:

`lpc_chip_43xx` – This is the chip level library that controls the LPC4330 microcontroller peripherals (used by the PE0003 Universal Interface Card).

`PE0003_DriverLib` – This is the board level layer that provides drivers to control the PE0003 peripherals. This library uses the services of the `lpc_chip_43xx` library.

`CML_UtilsLib` – This library provides a set of functions to allow the PR7261 Project to connect with a PC. It also provides services like file writing and reading, loading FIs, etc.

The PR7261 Project contains the following files:

`Inc/PR7261_Tools.h` – A library of utilities to configure the PE0003 and manage the application.

`Inc/ProgTools.h` – A library of CMX7261 voice coder configuration functions and a user-defined configuration that takes its parameters from `settings.h`.

`Inc/Register.h` – A suite of macros that define the CMX7261 registers, register configuration settings and register flags.

`Inc/Settings.h` – A configuration file to set the default application parameters or configure the user defined mode. FOR USER-DEFINED OPERATION, THIS FILE SHOULD BE MODIFIED AND THE PROJECT RE-COMPILED.

`Cr_startup_lpc43xx.c`

`PR7261_tools.c` – Code for its `.h` library.

`ProgTools.c` – Code for its `.h` library.

`PR7261.c` – Main program and some helpful utilities.

To operate the PR7261 check the following configurations for `Settings.h` and how to manage the software.

4 Run Modes

There are two ways of running the PR7261 Project:

- Using a pre-compiled version with EC0003.
- Using LPCXpresso to modify the user-defined configuration and re-compile the PR7261 Project.

Pre-COMPILED MODE

Using a pre-compiled version; the EC0003 application allows the .bin file to be loaded and run on the PE0003. In this mode the custom operations should not be used because the configuration has been pre-set.

User COMPILED MODE

In this mode the PR7261 Project is configured in Settings.h and compiled. The resulting .bin is then run as in the pre-compiled mode above.

5 Requirements

To use the PR7261 Project the following tools and programs are required:

- PE0003 board.
- PE0601-7261.
- Power supply +/-5V

Programs:

- LPCXpresso (Required for User Compiled Mode)
- This Project PR7261_x.zip
- EC0003 (EC0003InstallerV1_3 or higher version)
- Drivers for PE0003

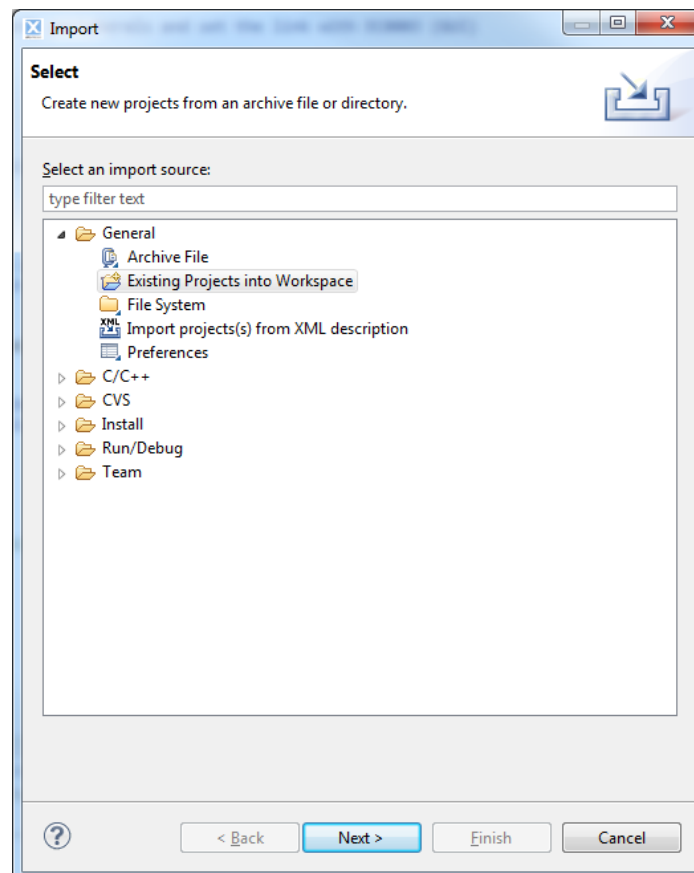
6 Basics

The description that follows is for User Compiled mode. To run the default project or to run user code that has already been compiled go to 6.1 Executing using the EC0003 GUI.

Install LPCXpresso.

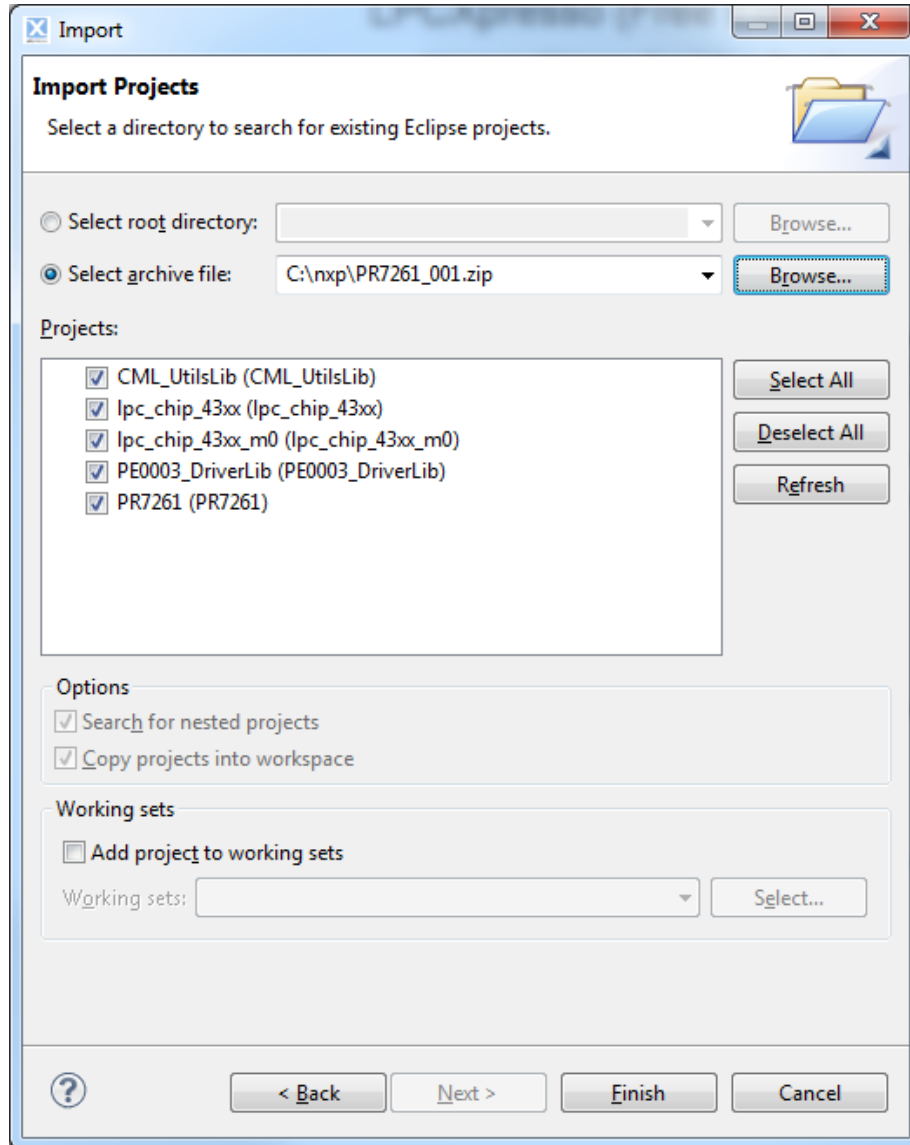
Run LPCXpresso and import project to your workspace.

File→Import. Select Existing Projects into Workspace and click next.



Check “Select archive file” and use the .zip file with the full project.

Finish.



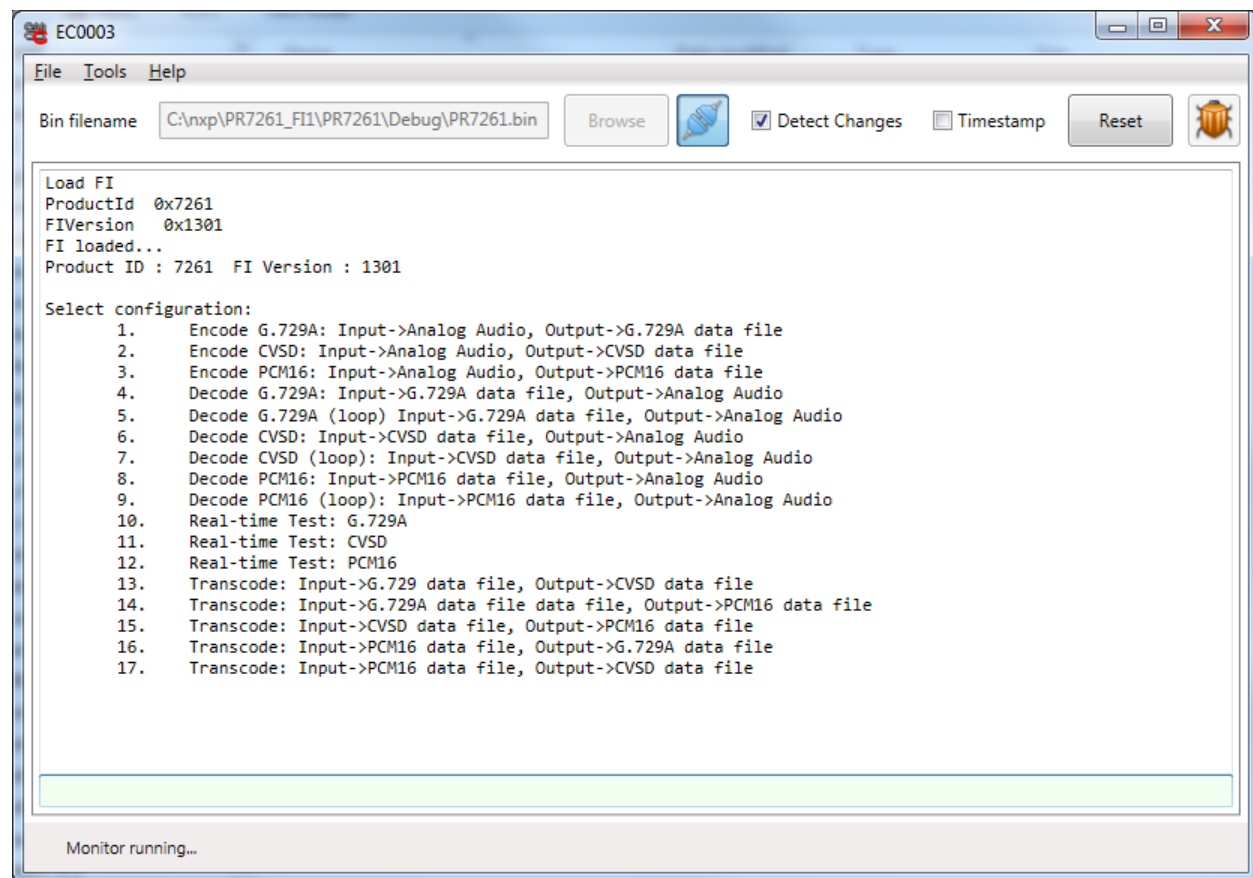
Configure the required operation in Settings.h See 7 Settings Configuration.

Compile the project . Project→Build All. This generates a .bin file for the application.

The .bin file should be located in the project folder “\YourWorkSpace\PR7261\Debug” or “\YourWorkSpace\PR7261\Release”/

6.1 Executing using the EC0003 GUI

To load the bin file, run the EC0003 GUI application, power up the board then browse to and select the bin.



For information about using the GUI for debugging with JTAG or other debugger tools, refer to the document [2]

NOTE: Data is stored as a text file by default using the formatting parameter “%u”, one unsigned integer per line.

It is possible to re-configure Settings.h for WAV or BIN file types.

7 Settings Configuration

The Settings.h file allows the run time operation to be customised using the definitions that follow.

FI Load Configuration

There are two macros that set the main behaviour when loading FIs.

To display a pop-up window requesting the FI at runtime;

```
#define FI_OPENDIALOG_EN      TRUE
```

To load the FI from a specific location on your computer without any interaction;

```
#define FI_OPENDIALOG_EN      FALSE
```

```
#define FI_PC_PATH            "c:\\Data\\fi\\7261-X.X.X.X.h"
```

Menu configuration

To receive a pop-up menu with operation mode options;

```
#define DEFAULT_OPT           0
```

To fix the operation mode at runtime set to the option number required. For example, to select “G729A packed Encoder Operation CBus”;

```
#define DEFAULT_OPT           6
```

General Configuration

To set a different PE0003 C-Bus port than the default CBUS1;

```
#define CBUSPORT              CBUS2
```

To display the sample number on the screen;

```
#define DEBUG_DISP            TRUE
```

Displaying messages can place a large burden on the USB pipe. If speed problems are observed such messages indicating underflows or overflows then disable reporting;

```
#define DEBUG_DISP            FALSE
```

To set the number of samples to process (transcoding is limited to the memory available);

```
#define NSAMPLES              15000      ///
```

To use different file types, set

```
#define CFG_IN_FILE           CFG_TEXT_FILE
#define CFG_OUT_FILE          CFG_TEXT_FILE
```

Possible values are; CFG_WAV_FILE, CFG_TEXT_FILE and CFG_BIN_FILE for Wav, text and binary files respectively. Text file handling is set by default to use unsigned integer values with one value per line. If a different text format is required, the open file function code must be modified with the appropriate formatting parameter.

When using a wav file, set the required sample ;rate;

```
#define WAV_SAMPLERATE      8000
```

CUSTOM OPERATION SETTINGS

Check the following configurations. For transcoding operations the memory size for the buffers is set by INSTATICBUFFER_SIZE and OUTSTATICBUFFER_SIZE. The maximum size used by encoder and decoder must not exceed the 34000 samples together.

7.1 CONFIG 1 Encoder G729A Packed - PCM 16Bits signed

```
#define MODE_AUDIO_DST_VAL    MODE_AUDIO_DST_ANA

#define INPUT_TYPE_VAL        INPUT_TYPE_16BIT_SIGNED_PCM
#define OUTPUT_TYPE_VAL       OUTPUT_TYPE_PACKED_G729A

#define INPUT_RATE_VAL        RATE_8KHZ
#define OUTPUT_RATE_VAL       RATE_8KHZ

#define ANAOUT_CONF_VAL       ANAOUT_DAC_PWR | ANAOUT_DRVPWR | SPKR2_PWR
#define ANAOUT_COARSE_GAIN_VAL 0

#define ANAIN_CONF_VAL        ADC_PWR | ANAIN_PWR
#define ANAIN_COARSE_GAIN_VAL 0
```

7.2 CONFIG 2 Decoder PCM 16Bits signed – G729A Packed

```
#define MODE_AUDIO_SRC_VAL    MODE_AUDIO_SRC_ANA

#define INPUT_TYPE_VAL        INPUT_TYPE_PACKED_G729A
#define OUTPUT_TYPE_VAL       OUTPUT_TYPE_16BIT_SIGNED_PCM

#define INPUT_RATE_VAL        RATE_8KHZ
#define OUTPUT_RATE_VAL       RATE_8KHZ

#define ANAOUT_CONF_VAL       ANAOUT_DAC_PWR | ANAOUT_DRVPWR | SPKR2_PWR
#define ANAOUT_COARSE_GAIN_VAL 0
#define ANAIN_CONF_VAL        ADC_PWR | ANAIN_PWR
#define ANAIN_COARSE_GAIN_VAL 0
```

7.3 CONFIG 3 Transcoder G729A Packed - CVSD 32KHz Packed

```
#define MODE_AUDIO_SRC_VAL          MODE_AUDIO_SRC_CBUS (Set in transcoder by
default)
#define MODE_AUDIO_DST_VAL          MODE_AUDIO_SRC_CBUS (Set in transcoder by
default)

#define INPUT_TYPE_VAL              INPUT_TYPE_PACKED_G729A
#define OUTPUT_TYPE_VAL             OUTPUT_TYPE_PACKED_CVSD

#define INPUT_RATE_VAL              RATE_8KHZ
#define OUTPUT_RATE_VAL             RATE_32KHZ

#define ANAOUT_CONF_VAL             ANAOUT_DAC_PWR | ANAOUT_DRVPWR |
SPKR2_PWR
#define ANAOUT_COARSE_GAIN_VAL      0
#define ANAIN_CONF_VAL              ADC_PWR | ANAIN_PWR
#define ANAIN_COARSE_GAIN_VAL 0
```

8 File Documentation

8.1 inc/PR7261_Tools.h File Reference

8.1.1 Functions

- `uint8_t BoardInit ()`
Initialise the board, peripherals and set the link with EC0003 (GUI)
 - `uint8_t LoadFI ()`
 - `void SetDelay (uint32_t delay_us)`
Use Timer1 to make a timeout trigger. Poll DelayReached for timeout.
 - `uint8_t DelayReached (void)`
Poll check for SetDelay().
 - `uint32_t GetNumberOfSamples ()`
-

8.1.2 Function Documentation

`uint8_t BoardInit ()`

Initialise the board, peripherals and set the link with EC0003 (GUI)

Returns:

TRUE

`uint8_t DelayReached (void )`

Poll check for **SetDelay()**.

Returns:

True - if timer value reached, FALSE otherwise

`uint32_t GetNumberOfSamples ()`

Ask to the user for the number of samples

Returns:

Number of samples

`uint8_t LoadFI ()`

Loads a FI

Returns:

TRUE if success, FALSE if fail

`void SetDelay (uint32_t delay_us)`

Use Timer1 to make a timeout trigger. Poll DelayReached for timeout.

Parameters:

<i>delay_us</i>	- delay in microseconds
-----------------	-------------------------

Returns:

None

8.2 inc/ProgTools.h File Reference

8.2.1 Functions

- uint8_t **Menu_DefaultSelection** (uint8_t defOpt)
Selection menu.
- uint8_t **Custom_Encoder_Op** (uint32_t nSamples)
*Custom encoding application, using configuration set from **Settings.h** Set to CBus destination by default.*
- uint8_t **Custom_Decoder_Op** (uint32_t nSamples)
*Custom decoding application, using configuration set from **Settings.h** Set to CBus source by default.*
- uint8_t **Custom_Decoder_Loop_Op** (uint32_t nSamples)
*Custom decoding in a infinite loop, using configuration from **Settings.h** Set to CBus source by default.*
- uint8_t **Custom_Transcoder_Op** (uint32_t nSamples)
*Transcoding Custom Operation, using configuration from **Settings.h** Set to CBus source and CBus destination by default.*
- uint8_t **Custom_EncoderDecoder_Op** ()
Encoder/Decoder mode.
- uint8_t **G729A_Encoder_Op** (uint32_t nSamples)
The purpose of this function is to configure the PE0601-7261/CMX7261 for G729A encoding.
- uint8_t **G729A_Decoder_Op** (uint32_t nSamples)
The purpose of this function is to configure the PE0601-7261/CMX7261 for G729A decoding.
- uint8_t **G729A_Decoder_Op_Loop** (uint32_t nSamples)
Decode G729A samples. But in this case the samples to decode are stored into a buffer and pass to the fifo in an infinite loop. Uses static Buffers that are hold in the internal RAM memory of the PE0003 (memory size limited)
- uint8_t **Unpacked_CVSD_Encoder_Op** (uint32_t nSamples)
The purpose of this function is to configure the PE0601-7261/CMX7261 for CVSD encoding.
- uint8_t **Unpacked_CVSD_Decoder_Op** (uint32_t nSamples)
The purpose of this function is to configure the PE0601-7261/CMX7261 for unpacked CVSD decoding.
- uint8_t **Unpacked_CVSD_Decoder_Op_Loop** (uint32_t nSamples)
The purpose of this funtion is to configure the PE0601-7261/CMX7261 for CVSD decoding. Takes the samples from a file and stores them in a static buffer for decoding. The data is passed one at the time.
- uint8_t **G729A_CVSD_Transcoder** (uint32_t nSamples)
The purpose of this script is to configure the PE0601-7261/CMX7261 for G729A to CVSD transcoding.
- uint8_t **PCM16_Encoder_Op** (uint32_t nSamples)
The purpose of this function is to configure the PE0601-7261/CMX7261 for PCM16 encoding.
- uint8_t **PCM16_Decoder_Op** (uint32_t nSamples)
The purpose of this function is to configure the PE0601-7261/CMX7261 for PCM16 decoding.
- uint8_t **PCM16_Decoder_Op_Loop** (uint32_t nSamples)
PCM16 decoding using a loop.
- uint8_t **G729Packed_EncoderDecoder_Op** ()
G729 Packed Encoder/Decoder mode.
- uint8_t **CVSD_EncoderDecoder_Op** ()
CVSD Unpacked 32KHz Encoder/Decoder mode.
- uint8_t **PCM16_EncoderDecoder_Op** ()
PCM16 Encoder/Decoder mode.
- uint8_t **PCM16_G729_Transcoder** (uint32_t nSamples)
The purpose of this script is to configure the PE0601-7261/CMX7261 for PCM16 to PCM16 transcoding.

- **uint8_t G729_PCM16_Transcoder** (uint32_t nSamples)
The purpose of this script is to configure the PE0601-7261/CMX7261 for G729A to PCM16 transcoding.
- **uint8_t CVSD_G729_Transcoder** (uint32_t nSamples)
The purpose of this script is to configure the PE0601-7261/CMX7261 for CVSD to G729A transcoding.
- **uint8_t CVSD_PCM16_Transcoder** (uint32_t nSamples)
The purpose of this script is to configure the PE0601-7261/CMX7261 for CVSD to PCM16 transcoding.
- **uint8_t PCM16_CVSD_Transcoder** (uint32_t nSamples)
The purpose of this script is to configure the PE0601-7261/CMX7261 for PCM16 to CVSD Unpacked 32KHz transcoding.

8.2.2 Function Documentation

uint8_t Custom_Decoder_Loop_Op (uint32_t nSamples)

Custom decoding in a infinite loop, using configuration from **Settings.h** Set to CBus source by default.

Parameters:

<i>nSamples</i>	- Number of samples
-----------------	---------------------

Returns:

TRUE if success, FALSE if fails

uint8_t Custom_Decoder_Op (uint32_t nSamples)

Custom decoding application, using configuration set from **Settings.h** Set to CBus source by default.

Parameters:

<i>nSamples</i>	- Number of samples
-----------------	---------------------

Returns:

TRUE if success, FALSE if failed

uint8_t Custom_Encoder_Op (uint32_t nSamples)

Custom encoding application, using configuration set from **Settings.h** Set to CBus destination by default.

Parameters:

<i>nSamples</i>	- Number of samples
-----------------	---------------------

Returns:

TRUE if success, FALSE if failed

uint8_t Custom_EncoderDecoder_Op ()

Encoder/Decoder mode.

Returns:

TRUE if success, FALSE if fails

uint8_t Custom_Transcoder_Op (uint32_t nSamples)

Transcoding Custom Operation, using configuration from **Settings.h** Set to CBus source and CBus destination by default.

Parameters:

<i>nSamples</i>	- Number of samples
-----------------	---------------------

Returns:

TRUE if success, FALSE if fails

uint8_t CVSD_EncoderDecoder_Op ()

CVSD Unpacked 32KHz Encoder/Decoder mode.

Returns:

TRUE if success, FALSE if fails

uint8_t CVSD_G729_Transcoder (uint32_t nSamples)

The purpose of this script is to configure the PE0601-7261/CMX7261 for CVSD to G729A transcoding.

Parameters:

<i>nSamples</i>	- Number of samples
-----------------	---------------------

Returns:

TRUE if success, FALSE if fails

uint8_t CVSD_PCM16_Transcoder (uint32_t nSamples)

The purpose of this script is to configure the PE0601-7261/CMX7261 for CVSD to PCM16 transcoding.

Parameters:

<i>nSamples</i>	- Number of samples
-----------------	---------------------

Returns:

TRUE if success, FALSE if fails

uint8_t G729_PCM16_Transcoder (uint32_t *nSamples*)

The purpose of this script is to configure the PE0601-7261/CMX7261 for G729A to PCM16 transcoding.

Parameters:

<i>nSamples</i>	- Number of samples
-----------------	---------------------

Returns:

TRUE if success, FALSE if fails

uint8_t G729A_CVSD_Transcoder (uint32_t *nSamples*)

The purpose of this script is to configure the PE0601-7261/CMX7261 for G729A to CVSD transcoding.

Parameters:

<i>nSamples</i>	- Number of samples
-----------------	---------------------

Returns:

TRUE if success, FALSE if fails

uint8_t G729A_Decoder_Op (uint32_t *nSamples*)

The purpose of this function is to configure the PE0601-7261/CMX7261 for G729A decoding.

Parameters:

<i>nSamples</i>	- Number of samples
-----------------	---------------------

Returns:

TRUE if success, FALSE if fails

uint8_t G729A_Decoder_Op_Loop (uint32_t *nSamples*)

Decode G729A samples. But in this case the samples to decode are stored into a buffer and pass to the fifo in an infinite loop. Uses static Buffers that are hold in the internal RAM memory of the PE0003 (memory size limited)

Parameters:

<i>nSamples</i>	- Number of samples
-----------------	---------------------

Returns:

TRUE of success, FALSE if fails

uint8_t G729A_Encoder_Op (uint32_t *nSamples*)

The purpose of this function is to configure the PE0601-7261/CMX7261 for G729A encoding.

Parameters:

<i>nSamples</i>	- Number of samples
-----------------	---------------------

Returns:

TRUE if success, FALSE if fails

uint8_t G729Packed_EncoderDecoder_Op ()

G729 Packed Encoder/Decoder mode.

Returns:

TRUE if success, FALSE if fails

uint8_t Menu_DefaultSelection (uint8_t *defOpt*)

Selection menu.

Parameters:

<i>defOpt</i>	- Default option. If different than 0, executes the number given
---------------	--

Returns:

TRUE if success, false if fails

uint8_t PCM16_CVSD_Transcoder (uint32_t *nSamples*)

The purpose of this script is to configure the PE0601-7261/CMX7261 for PCM16 to CVSD Unpacked 32KHz transcoding.

Parameters:

<i>nSamples</i>	- Number of samples
-----------------	---------------------

Returns:

TRUE if success, FALSE if fails

uint8_t PCM16_Decoder_Op (uint32_t nSamples)

The purpose of this function is to configure the PE0601-7261/CMX7261 for PCM16 decoding.

Parameters:

<i>nSamples</i>	- Number of samples
-----------------	---------------------

Returns:

TRUE if success, FALSE if fails

uint8_t PCM16_Decoder_Op_Loop (uint32_t nSamples)

PCM16 decoding using a loop.

Parameters:

<i>nSamples</i>	- Number of samples
-----------------	---------------------

Returns:

TRUE if success, FALSE if fails

uint8_t PCM16_Encoder_Op (uint32_t nSamples)

The purpose of this function is to configure the PE0601-7261/CMX7261 for PCM16 encoding.

Parameters:

<i>nSamples</i>	- Number of samples
-----------------	---------------------

Returns:

TRUE if success, FALSE if fails

uint8_t PCM16_EncoderDecoder_Op ()

PCM16 Encoder/Decoder mode.

Returns:

TRUE if success, FALSE if fails

uint8_t PCM16_G729_Transcoder (uint32_t *nSamples*)

The purpose of this script is to configure the PE0601-7261/CMX7261 for PCM16 to PCM16 transcoding.

Parameters:

<i>nSamples</i>	- Number of samples
-----------------	---------------------

Returns:

TRUE if success, FALSE if fails

uint8_t Unpacked_CVSD_Decoder_Op (uint32_t *nSamples*)

The purpose of this function is to configure the PE0601-7261/CMX7261 for unpacked CVSD decoding.

Parameters:

<i>nSamples</i>	- Number of samples
-----------------	---------------------

Returns:

TRUE if success, FALSE if fails

uint8_t Unpacked_CVSD_Decoder_Op_Loop (uint32_t *nSamples*)

The purpose of this function is to configure the PE0601-7261/CMX7261 for CVSD decoding. Takes the samples from a file and stores them in a static buffer for decoding. The data is passed one at the time.

Parameters:

<i>nSamples</i>	- Number of samples
-----------------	---------------------

Returns:

TRUE if success, FALSE if fails

uint8_t Unpacked_CVSD_Encoder_Op (uint32_t *nSamples*)

The purpose of this function is to configure the PE0601-7261/CMX7261 for CVSD encoding.

Parameters:

<i>nSamples</i>	- Number of samples
-----------------	---------------------

Returns:

TRUE if success, FALSE if fails

8.3 inc/Registers.h File Reference

8.3.1 Macros

- `#define FIFO_IN_BYTE 0x48`
- `#define FIFO_IN_WORD 0x49`
- `#define FIFO_IN_LEVEL 0x4B`
- `#define FIFO_OUT_BYTE 0x4C`
- `#define FIFO_OUT_WORD 0x4D`
- `#define FIFO_OUT_LEVEL 0x4F`
- `#define FIFO_CONTROL 0x50`
- `#define FIFO_COUNT_INT 0x51`
- `#define INPUT_TYPE 0x54`
- `#define OUTPUT_TYPE 0x56`
- `#define SIGNAL_CONTROL 0x58`
- `#define VAD_THRSHLD_CH1 0x59`
- `#define VAD_THRSHLD_CH2 0x5A`
- `#define VAD_LEVEL_CH1 0x76`
- `#define VAD_LEVEL_CH2 0x77`
- `#define VAD_SNTD_CH1 0x7A`
- `#define VAD_SNTD_CH2 0x7B`
- `#define VAD_NSTD_CH1 0x7C`
- `#define VAD_NSTD_CH2 0x7D`
- `#define FINE_GAIN_CH1 0x5B`
- `#define FINE_GAIN_CH2 0x5C`
- `#define PCM_TALKTHROUGH_DATA 0x63`
- `#define GPIO_CONTROL 0x64`
- `#define GPIO_INPUT 0x79`
- `#define REG_DONE_SELECT 0x69`
- `#define PROG_REG 0x6A`
- `#define MODE_REG 0x6B`
- `#define IRQ_ENABLE 0x6C`
- `#define STATUS 0x7E`
- `#define MODE_REG_READBACK 0x7F`
- `#define ANAIN_CONF 0xB0`
- `#define ANAIN_COARSE_GAIN 0xB1`
- `#define ANAOUT_CONF 0xB3`
- `#define ANAOUT_COARSE_GAIN 0xB4`
- `#define MONOUT_COARSE_GAIN 0xB5`
- `#define SPKR_COARSE_GAIN 0xB6`
- `#define BIAS_CONTROL 0xB7`
- `#define MODE_IDLE 0x00`
- `#define MODE_HALF_TRANSC 0x01`
Transcode (Half-Duplex Transcode)
- `#define MODE_ENCDEC 0x02`
EncDec.
- `#define MODE_FULL_TRANSC 0x03`
Full Duplex Transcode.
- `#define MODE_AUDIO_SRC_CBUS 0x01`
C-BUS Audio In FIFO.
- `#define MODE_AUDIO_SRC_ANA 0x02`
Analogue audio input.

- **#define MODE_AUDIO_SRC_EXT_PCM 0x04**
External PCM.
- **#define MODE_AUDIO_DST_CBUS 0x01**
C-BUS Audio Out FIFO.
- **#define MODE_AUDIO_DST_ANA 0x02**
Analogue audio output, no monitor output.
- **#define MODE_AUDIO_DST_CBUSANA 0x03**
C-BUS analogue audio output, no monitor output.
- **#define MODE_AUDIO_DST_PCM 0x04**
External PCM, no monitor output.
- **#define MODE_AUDIO_DST_PCMCBUS 0x05**
External PCM and C-BUS no monitor.
- **#define MODE_AUDIO_DST_PCMANA 0x06**
External PCM and analogue audio output, no monitor output.
- **#define MODE_AUDIO_DST_PCMANACBUS 0x07**
External PCM, analogue audio output and C-BUS, no monitor output.
- **#define MODE_AUDIO_DST_MON 0x08**
Monitor output enabled, all other audio destinations disabled.
- **#define MODE_AUDIO_DST_MONCBUS 0x09**
C-Bus - Audio outFIFO, monitor enabled.
- **#define MODE_AUDIO_DST_MONANA 0x0A**
Analogue Output, monitor enabled.
- **#define MODE_AUDIO_DST_MONCBUSANA 0x0B**
C-BUS, Analogue output, monitor enabled.
- **#define MODE_AUDIO_DST_MONPCM 0x0C**
External PCM, monitor enabled.
- **#define MODE_AUDIO_DST_MONPCMCBUS 0x0D**
External PCM and C-BUS, monitor enabled.
- **#define MODE_AUDIO_DST_MONPCMANA 0x0E**
External PCM and Analogue output, monitor enabled.
- **#define MODE_AUDIO_DST_MONPCMANACBUS 0x0F**
PCM, Analogue, C-BUS, monitor enabled.
- **#define INPUT_TYPE_UNPACKED_G729A 0x00**
- **#define INPUT_TYPE_PACKED_G729A 0x10**
Packed G729A.
- **#define INPUT_TYPE_UNPACKED_G711A 0x01**
Unpacked G711A.
- **#define INPUT_TYPE_PACKED_G711A 0x11**
Packed G711A.
- **#define INPUT_TYPE_UNPACKED_G711A_XOR 0x41**
Unpacked G711A with XOR.
- **#define INPUT_TYPE_PACKED_G711A_XOR 0x51**
Packed G711A with XOR.
- **#define INPUT_TYPE_UNPACKED_G711U 0x02**
Unpacked G711U.

- **#define INPUT_TYPE_PACKED_G711U 0x12**
Packed G711U.
- **#define INPUT_TYPE_UNPACKED_G711U_XOR 0x42**
Unpacked G711U with XOR.
- **#define INPUT_TYPE_PACKED_G711U_XOR 0x52**
Packed G711U with XOR.
- **#define INPUT_TYPE_UNPACKED_CVSD 0x03**
Unpacked CVSD.
- **#define INPUT_TYPE_PACKED_CVSD 0x13**
Packed CVSD.
- **#define INPUT_TYPE_16BIT_SIGNED_PCM 0x04**
16 bit signed linear PCM
- **#define INPUT_TYPE_16BIT_UNSIGNED_PCM 0x24**
16 bit unsigned linear PCM
- **#define INPUT_TYPE_16BIT_SIGNED_PCM_REV 0x84**
16 bit signed linear PCM Bit reversed
- **#define INPUT_TYPE_16BIT_UNSIGNED_PCM_REV 0xA4**
16 bit unsigned linear PCM Bit reversed
- **#define OUTPUT_TYPE_UNPACKED_G729A 0x00**
- **#define OUTPUT_TYPE_PACKED_G729A 0x10**
Packed G729A.
- **#define OUTPUT_TYPE_UNPACKED_G711A 0x01**
Unpacked G711A.
- **#define OUTPUT_TYPE_PACKED_G711A 0x11**
Packed G711A.
- **#define OUTPUT_TYPE_UNPACKED_G711A_XOR 0x41**
Unpacked G711A with XOR.
- **#define OUTPUT_TYPE_PACKED_G711A_XOR 0x51**
Packed G711A with XOR.
- **#define OUTPUT_TYPE_UNPACKED_G711U 0x02**
Unpacked G711U.
- **#define OUTPUT_TYPE_PACKED_G711U 0x12**
Packed G711U.
- **#define OUTPUT_TYPE_UNPACKED_G711U_XOR 0x42**
Unpacked G711U with XOR.
- **#define OUTPUT_TYPE_PACKED_G711U_XOR 0x52**
Packed G711U with XOR.
- **#define OUTPUT_TYPE_UNPACKED_CVSD 0x03**
Unpacked CVSD.
- **#define OUTPUT_TYPE_PACKED_CVSD 0x13**
Packed CVSD.
- **#define OUTPUT_TYPE_16BIT_SIGNED_PCM 0x04**
16 bit signed linear PCM
- **#define OUTPUT_TYPE_16BIT_UNSIGNED_PCM 0x24**
16 bit unsigned linear PCM

-
- **#define OUTPUT_TYPE_16BIT_SIGNED_PCM_REV** 0x84
16 bit signed linear PCM Bit reversed
 - **#define OUTPUT_TYPE_16BIT_UNSIGNED_PCM_REV** 0xA4
16 bit unsigned linear PCM Bit reversed
 - **#define RATE_NONE** 0
 - **#define RATE_8KHZ** 0
 - **#define RATE_16KHZ** (1 << 8)
 - **#define RATE_32KHZ** (1 << 9)
 - **#define ANAOUT_MUTE** (1 << 0)
 - **#define ANAOUT_DRVPWR** (1 << 1)
Powers up the output driver to ANAOUT, 0 - Off, 1 - On.
 - **#define MONOOUT_MUTE** (1 << 2)
Mutes MONOUT 0 - Active, 1 - Mute.
 - **#define MONOOUT_DRVPWR** (1 << 3)
Powers up the output driver to MONOUT.
 - **#define SPKR2_MUTE** (1 << 4)
Mutes the speaker driver signal.
 - **#define SPKR2_PWR** (1 << 5)
Powers up the SPKR2 output internally.
 - **#define OP_BIAS_PWR** (1 << 6)
Powers up output bias buffer for SPKR2 output.
 - **#define SPKR1_PWR** (1 << 7)
Powers up SPKR1 output.
 - **#define SPKR_INPUT_SEL** (1 << 8)
0 - Connects ANAOUT 1 - MONOUT to the speaker drivers
 - **#define LINE_OUT_SW** (1 << 9)
0 - ANOUT through the MONOUT pins, when 1 - Monitor signal is presented on MONOUT
 - **#define MONOOUT_DAC_PW** (1 << 10)
Powers up MONOUT DAC internally.
 - **#define ANAOUT_DAC_PWR** (1 << 11)
Powers up ANAOUT DAC internally.
 - **#define ANAIN_MUTE** (1 << 0)
 - **#define ANAIN_PWR** (1 << 1)
Powers up the input driver to the ANAIN.
 - **#define ANAIN2_MUTE** (1 << 2)
Mutes ANAIN2 0 - Active, 1 - Mute.
 - **#define ANAIN2_PWR** (1 << 3)
Powers up the input driver to the ANAIN2.
 - **#define ANA_SW** (1 << 9)
Switch to select which analogue input is connected to the ADC 0 - ANAIN, 1 - ANAIN2.
 - **#define ADC_PWR** (1 << 11)
Powers up ADC internally.
 - **#define ST_REG_DONE** 0x2000
 - **#define ST_PRG** 0x4000
 - **#define ST_IRQ** 0x8000
 - **#define REG_DONE_FIFO** (1 << 1)

-
- **#define REG_DONEL_GPIO** (1 << 2)
Handshake GPIO.
 - **#define REG_DONE_MODE** (1 << 3)
Handshake MODE.
 - **#define RIRQ_VAD_N_TO_N_CH2** (1 << 1)
 - **#define IRQ_VAD_SIG_TO_SIG_CH2** (1 << 2)
Set when the VAD output transitions from noise to signal, for ch2.
 - **#define IRQ_COUNT_IN** (1 << 3)
Set when CMX7261 reads 'N' input samples from the Audio In FIFO, so that the host can write 'N' new input samples at once.
 - **#define IRQ_COUNT_OUT** (1 << 4)
Set when there are 'N' new samples in the Audio Out FIFO, so that the host can read 'N' samples at once.
 - **#define IRQ_PCM_THRU** (1 << 5)
Set when a PCM pass through read or write is completed.
 - **#define IRQ_AUDIO_IN_FIFO** (1 << 6)
Audio In FIFO empties to the level specified by the host.
 - **#define IRQ_AUDIO_OUT_FIFO** (1 << 7)
Audio Out FIFO fills to the level specified by the host.
 - **#define IRQ_VAD_SIG_TO_N_CH1** (1 << 8)
VAD output transitions from signal to noise, for ch1.
 - **#define IRQ_VAD_N_TO_SIG_CH1** (1 << 9)
VAD output transitions from noise to signal, for ch1.
 - **#define IRQ_OFLOW** (1 << 10)
Overflow.
 - **#define IRQ_UFLOW** (1 << 11)
Underflow.
 - **#define IRQ_MON_UFLOW** (1 << 12)
Monitor Underflow.
 - **#define IRQ_REG_DONE** (1 << 13)
Register access Done.
 - **#define IRQ_PRG** (1 << 14)
Programming register ready.
 - **#define IRQ_EN** (1 << 15)
Enable IRQ1.
-

8.3.2 Macro Definition Documentation

#define ADC_PWR (1 << 11)

Powers up ADC internally.

#define ANA_SW (1 << 9)

Switch to select which analogue input is connected to the ADC 0 - ANAIN, 1 - ANAIN2.

#define ANAIN2_MUTE (1 << 2)

Mutes ANAIN2 0 - Active, 1 - Mute.

#define ANAIN2_PWR (1 << 3)

Powers up the input driver to the ANAIN2.

#define ANAIN_COARSE_GAIN 0xB1

#define ANAIN_CONF 0xB0

#define ANAIN_MUTE (1 << 0)

Mutes ANAIN 0 - Active, 1 - Mute

#define ANAIN_PWR (1 << 1)

Powers up the input driver to the ANAIN.

#define ANAOUT_COARSE_GAIN 0xB4

#define ANAOUT_CONF 0xB3

#define ANAOUT_DAC_PWR (1 << 11)

Powers up ANAOUT DAC internally.

#define ANAOUT_DRVPWR (1 << 1)

Powers up the output driver to ANAOUT, 0 - Off, 1 - On.

#define ANAOUT_MUTE (1 << 0)

Mutes ANAOUT 0 - Active, 1 - Mute

#define BIAS_CONTROL 0xB7

#define FIFO_CONTROL 0x50

#define FIFO_COUNT_INT 0x51

#define FIFO_IN_BYTE 0x48

#define FIFO_IN_LEVEL 0x4B

#define FIFO_IN_WORD 0x49

#define FIFO_OUT_BYTE 0x4C

#define FIFO_OUT_LEVEL 0x4F

#define FIFO_OUT_WORD 0x4D

#define FINE_GAIN_CH1 0x5B

#define FINE_GAIN_CH2 0x5C

#define GPIO_CONTROL 0x64

#define GPIO_INPUT 0x79

#define INPUT_TYPE 0x54

#define INPUT_TYPE_16BIT_SIGNED_PCM 0x04

16 bit signed linear PCM

#define INPUT_TYPE_16BIT_SIGNED_PCM_REV 0x84

16 bit signed linear PCM Bit reversed

#define INPUT_TYPE_16BIT_UNSIGNED_PCM 0x24

16 bit unsigned linear PCM

#define INPUT_TYPE_16BIT_UNSIGNED_PCM_REV 0xA4

16 bit unsigned linear PCM Bit reversed

#define INPUT_TYPE_PACKED_CVSD 0x13

Packed CVSD.

#define INPUT_TYPE_PACKED_G711A 0x11

Packed G711A.

#define INPUT_TYPE_PACKED_G711A_XOR 0x51

Packed G711A with XOR.

#define INPUT_TYPE_PACKED_G711U 0x12

Packed G711U.

#define INPUT_TYPE_PACKED_G711U_XOR 0x52

Packed G711U with XOR.

#define INPUT_TYPE_PACKED_G729A 0x10

Packed G729A.

#define INPUT_TYPE_UNPACKED_CVSD 0x03

Unpacked CVSD.

#define INPUT_TYPE_UNPACKED_G711A 0x01

Unpacked G711A.

#define INPUT_TYPE_UNPACKED_G711A_XOR 0x41

Unpacked G711A with XOR.

#define INPUT_TYPE_UNPACKED_G711U 0x02

Unpacked G711U.

#define INPUT_TYPE_UNPACKED_G711U_XOR 0x42

Unpacked G711U with XOR.

#define INPUT_TYPE_UNPACKED_G729A 0x00

Unpacked G729A

#define IRQ_AUDIO_IN_FIFO (1 << 6)

Audio In FIFO empties to the level specified by the host.

#define IRQ_AUDIO_OUT_FIFO (1 << 7)

Audio Out FIFO fills to the level specified by the host.

#define IRQ_COUNT_IN (1 << 3)

Set when CMX7261 reads ‘N’ input samples from the Audio In FIFO, so that the host can write ‘N’ new input samples at once.

#define IRQ_COUNT_OUT (1 << 4)

Set when there are ‘N’ new samples in the Audio Out FIFO, so that the host can read ‘N’ samples at once.

#define IRQ_EN (1 << 15)

Enable IRQ1.

#define IRQ_ENABLE 0x6C

#define IRQ_MON_UFLOW (1 << 12)

Monitor Underflow.

#define IRQ_OFLOW (1 << 10)

Overflow.

#define IRQ_PCM_THRU (1 << 5)

Set when a PCM pass through read or write is completed.

#define IRQ_PRG (1 << 14)

Programming register ready.

#define IRQ_REG_DONE (1 << 13)

Register access Done.

#define IRQ_UFLOW (1 << 11)

Underflow.

#define IRQ_VAD_N_TO_SIG_CH1 (1 << 9)

VAD output transitions from noise to signal, for ch1.

#define IRQ_VAD_SIG_TO_N_CH1 (1 << 8)

VAD output transitions from signal to noise, for ch1.

#define IRQ_VAD_SIG_TO_SIG_CH2 (1 << 2)

Set when the VAD output transitions from noise to signal, for ch2.

#define LINE_OUT_SW (1 << 9)

0 - ANOUT through the MONOUT pins, when 1 - Monitor signal is presented on MONOUT

#define MODE_AUDIO_DST_ANA 0x02

Analogue audio output, no monitor output.

#define MODE_AUDIO_DST_CBUS 0x01

C-BUS Audio Out FIFO.

#define MODE_AUDIO_DST_CBUSANA 0x03

C-BUS analogue audio output, no monitor output.

#define MODE_AUDIO_DST_MON 0x08

Monitor output enabled, all other audio destinations disabled.

#define MODE_AUDIO_DST_MONANA 0x0A

Analogue Output, monitor enabled.

#define MODE_AUDIO_DST_MONCBUS 0x09

C-Bus - Audio outFIFO, monitor enabled.

#define MODE_AUDIO_DST_MONCBUSANA 0x0B

C-BUS, Analogue output, monitor enabled.

#define MODE_AUDIO_DST_MONPCM 0x0C

External PCM, monitor enabled.

#define MODE_AUDIO_DST_MONPCMANA 0x0E

External PCM and Analogue output, monitor enabled.

#define MODE_AUDIO_DST_MONPCMANACBUS 0x0F

PCM, Analogue, C-BUS, monitor enabled.

#define MODE_AUDIO_DST_MONPCMCBUS 0x0D

External PCM and C-BUS, monitor enabled.

#define MODE_AUDIO_DST_PCM 0x04

External PCM, no monitor output.

#define MODE_AUDIO_DST_PCMANA 0x06

External PCM and analogue audio output, no monitor output.

#define MODE_AUDIO_DST_PCMANACBUS 0x07

External PCM, analogue audio output and C-BUS, no monitor output.

#define MODE_AUDIO_DST_PCMCBUS 0x05

External PCM and C-BUS no monitor.

#define MODE_AUDIO_SRC_ANA 0x02

Analogue audio input.

#define MODE_AUDIO_SRC_CBUS 0x01

C-BUS Audio In FIFO.

#define MODE_AUDIO_SRC_EXT_PCM 0x04

External PCM.

#define MODE_ENCDEC 0x02

EncDec.

#define MODE_FULL_TRANSC 0x03

Full Duplex Transcode.

#define MODE_HALF_TRANSC 0x01

Transcode (Half-Duplex Transcode)

#define MODE_IDLE 0x00

Idle-Low Power mode

#define MODE_REG 0x6B

#define MODE_REG_READBACK 0x7F

#define MONOOUT_DAC_PW (1 << 10)

Powers up MONOUT DAC internally.

#define MONOOUT_DRVPWR (1 << 3)

Powers up the output driver to MONOUT.

#define MONOOUT_MUTE (1 << 2)

Mutes MONOUT 0 - Active, 1 - Mute.

#define MONOUT_COARSE_GAIN 0xB5

#define OP_BIAS_PWR (1 << 6)

Powers up output bias buffer for SPKR2 output.

#define OUTPUT_TYPE 0x56

#define OUTPUT_TYPE_16BIT_SIGNED_PCM 0x04

16 bit signed linear PCM

#define OUTPUT_TYPE_16BIT_SIGNED_PCM_REV 0x84

16 bit signed linear PCM Bit reversed

#define OUTPUT_TYPE_16BIT_UNSIGNED_PCM 0x24

16 bit unsigned linear PCM

#define OUTPUT_TYPE_16BIT_UNSIGNED_PCM_REV 0xA4

16 bit unsigned linear PCM Bit reversed

#define OUTPUT_TYPE_PACKED_CVSD 0x13

Packed CVSD.

#define OUTPUT_TYPE_PACKED_G711A 0x11

Packed G711A.

#define OUTPUT_TYPE_PACKED_G711A_XOR 0x51

Packed G711A with XOR.

#define OUTPUT_TYPE_PACKED_G711U 0x12

Packed G711U.

#define OUTPUT_TYPE_PACKED_G711U_XOR 0x52

Packed G711U with XOR.

#define OUTPUT_TYPE_PACKED_G729A 0x10

Packed G729A.

#define OUTPUT_TYPE_UNPACKED_CVSD 0x03

Unpacked CVSD.

#define OUTPUT_TYPE_UNPACKED_G711A 0x01

Unpacked G711A.

#define OUTPUT_TYPE_UNPACKED_G711A_XOR 0x41

Unpacked G711A with XOR.

#define OUTPUT_TYPE_UNPACKED_G711U 0x02

Unpacked G711U.

#define OUTPUT_TYPE_UNPACKED_G711U_XOR 0x42

Unpacked G711U with XOR.

#define OUTPUT_TYPE_UNPACKED_G729A 0x00

Unpacked G729A

#define PCM_TALKTHROUGH_DATA 0x63

#define PROG_REG 0x6A

#define RATE_16KHZ (1 << 8)

#define RATE_32KHZ (1 << 9)

#define RATE_8KHZ 0

#define RATE_NONE 0

#define REG_DONE_FIFO (1 << 1)

Handshake FIFO

#define REG_DONE_MODE (1 << 3)

Handshake MODE.

#define REG_DONE_SELECT 0x69

#define REG_DONEL_GPIO (1 << 2)

Handshake GPIO.

#define RIRQ_VAD_N_TO_N_CH2 (1 << 1)

Set when the VAD output transitions from signal to noise, for ch2

#define SIGNAL_CONTROL 0x58

#define SPKR1_PWR (1 << 7)

Powers up SPKR1 output.

#define SPKR2_MUTE (1 << 4)

Mutes the speaker driver signal.

#define SPKR2_PWR (1 << 5)

Powers up the SPKR2 output internally.

#define SPKR_COARSE_GAIN 0xB6

#define SPKR_INPUT_SEL (1 << 8)

0 - Connects ANAOUT 1 - MONOUT to the speaker drivers

#define ST_IRQ 0x8000

#define ST_PRG 0x4000

#define ST_REG_DONE 0x2000

#define STATUS 0x7E

#define VAD_LEVEL_CH1 0x76

#define VAD_LEVEL_CH2 0x77

#define VAD_NSTD_CH1 0x7C

#define VAD_NSTD_CH2 0x7D

#define VAD_SNTD_CH1 0x7A

#define VAD_SNTD_CH2 0x7B

#define VAD_THRSHLD_CH1 0x59

#define VAD_THRSHLD_CH2 0x5A

8.4 inc/Settings.h File Reference

```
#include "Registers.h"
```

8.4.1 Macros

- **#define ENABLE_CUSTOM_MODES 0**
Enable the Custom modes in the menu.
- **#define P1_2_VAL 40**
- **#define P1_3_VAL 208**
- **#define P1_4_VAL 0x41**
- **#define P1_5_VAL 78**
- **#define FI_OPENDIALOG_EN TRUE**
- **#define FI_PC_PATH "c:\\Data\\fi\\7xxx-1.X.X.X.h"**
Set to FALSE to load the FI from a PC location, specified by FI_PC_PATH.
- **#define DEFAULT_OPT 0**
Value 0 for enabling the menu selection, otherwise runs the selection with that value.
- **#define DEBUG_DISP TRUE**
Disable the display if the update frequency cause overflows or underflows.
- **#define CBUSPORT CBUS1**
CBUS port to use.
- **#define NSAMPLES 18000**
- **#define CFG_WAV_FILE 1**
- **#define CFG_TEXT_FILE 2**
- **#define CFG_BIN_FILE 3**
- **#define CFG_IN_FILE CFG_TEXT_FILE**
- **#define CFG_OUT_FILE CFG_TEXT_FILE**
- **#define USE_WAV_FILES TRUE**
- **#define WAV_CHANNELS 1**
Use 1 mono, 2 stereo //DEFAULT 1 DO NOT CHANGE IT.
- **#define WAV_BITSPERSAMPLE 16**
Use 8 or 16 bits per sample //DEFAULT 16 DO NOT CHANGE IT.
- **#define WAV_SAMPLERATE 8000**
Set the sample rate of the wav file.
- **#define MODE_AUDIO_SRC_VAL MODE_AUDIO_SRC_ANA**
FOR CUSTOM OPERATIONS.
- **#define MODE_AUDIO_DST_VAL MODE_AUDIO_DST_ANA**
Set the audio destination. Check mode register values.
- **#define INPUT_TYPE_VAL INPUT_TYPE_PACKED_G729A**
- **#define OUTPUT_TYPE_VAL OUTPUT_TYPE_16BIT_SIGNED_PCM**
Set the output type for custom operations. Check output register values.
- **#define INPUT_RATE_VAL RATE_8KHZ**
Set input frequency rate if applicable. Check input register values.
- **#define OUTPUT_RATE_VAL RATE_8KHZ**
Set output frequency rate if applicable. Check output register values.
- **#define ANAOUT_CONF_VAL ANAOUT_DAC_PWR | ANAOUT_DRVPWR | SPKR2_PWR**
- **#define ANAOUT_COARSE_GAIN_VAL 0**
Set the output coarse gain. Check ANAOUT_COARSE_GAIN register.
- **#define ANAIN_CONF_VAL ADC_PWR | ANAIN2_PWR | ANA_SW**
- **#define ANAIN_COARSE_GAIN_VAL 0**
Set the input coarse gain. Check ANAIN_COARSE_GAIN register.
- **#define INBUFFER_SIZE 100**
FIX VALUES.

-
- `#define OUTBUFFER_SIZE 100`
Size small bufferOut to hold the samples.
 - `#define MAX_FIFO_SIZE 127`
Size of the FIFO.
 - `#define INSTATICBUFFER_SIZE 18000`
 - `#define OUTSTATICBUFFER_SIZE 18000`
-

8.4.2 Macro Definition Documentation

`#define ANAIN_COARSE_GAIN_VAL 0`

Set the input coarse gain. Check ANAIN_COARSE_GAIN register.

`#define ANAIN_CONF_VAL ADC_PWR | ANAIN2_PWR | ANA_SW`

`#define ANAOUT_COARSE_GAIN_VAL 0`

Set the output coarse gain. Check ANAOUT_COARSE_GAIN register.

`#define ANAOUT_CONF_VAL ANAOUT_DAC_PWR | ANAOUT_DRVPWR | SPKR2_PWR`

`#define CBUSPORT CBUS1`

CBUS port to use.

`#define CFG_BIN_FILE 3`

`#define CFG_IN_FILE CFG_TEXT_FILE`

`#define CFG_OUT_FILE CFG_TEXT_FILE`

`#define CFG_TEXT_FILE 2`

`#define CFG_WAV_FILE 1`

`#define DEBUG_DISP TRUE`

Disable the display if the update frequency cause overflows or underflows.

`#define DEFAULT_OPT 0`

Value 0 for enabling the menu selection, otherwise runs the selection with that value.

#define ENABLE_CUSTOM_MODES 0

Enable the Custom modes in the menu.

#define FI_OPENDIALOG_EN TRUE

#define FI_PC_PATH "c:\\Data\\fi\\7xxx-1.X.X.X.h"

Set to FALSE to load the FI from a PC location, specified by FI_PC_PATH.

Full path of the location of the FI in the PC, should not use Program Files(it requires administrative privileges)

#define INBUFFER_SIZE 100

Size small bufferIn to hold the samples

#define INPUT_RATE_VAL RATE_8KHZ

Set input frequency rate if applicable. Check input register values.

#define INPUT_TYPE_VAL INPUT_TYPE_PACKED_G729A

#define INSTATICBUFFER_SIZE 18000

#define MAX_FIFO_SIZE 127

Size of the FIFO.

#define MODE_AUDIO_DST_VAL MODE_AUDIO_DST_ANA

Set the audio destination. Check mode register values.

#define MODE_AUDIO_SRC_VAL MODE_AUDIO_SRC_ANA

FOR CUSTOM OPERATIONS.

Set the audio source. Check mode register values

#define NSAMPLES 18000

#define OUTBUFFER_SIZE 100

Size small bufferOut to hold the samples.

#define OUTPUT_RATE_VAL RATE_8KHZ

Set output frequency rate if applicable. Check output register values.

#define OUTPUT_TYPE_VAL OUTPUT_TYPE_16BIT_SIGNED_PCM

Set the output type for custom operations. Check output register values.

#define OUTSTATICBUFFER_SIZE 18000

#define P1_2_VAL 40

CLOCK CONTROL Set to 19.2MHz

#define P1_3_VAL 208

#define P1_4_VAL 0x41

#define P1_5_VAL 78

#define USE_WAV_FILES TRUE

#define WAV_BITSPERSAMPLE 16

Use 8 or 16 bits per sample //DEFAULT 16 DO NOT CHANGE IT.

#define WAV_CHANNELS 1

Use 1 mono, 2 stereo //DEFAULT 1 DO NOT CHANGE IT.

#define WAV_SAMPLERATE 8000

Set the sample rate of the wav file.

9 Supporting Documentation

Supporting Documentation

[1] CMX7261 Voice Multi-transcoder User manual and Data Sheet. Available from the CML Portal – requires authorisation.

[2] Getting Started With The EC0003 Application. Available from the CML website.

[3] EC0003 C Driver Libraries CML_UtilsLib. Available from the CML website.

[4] EC0003 C Driver Libraries PE0003_DriverLib. Available from the CML website.

CML does not assume any responsibility for the use of any algorithms, methods or circuitry described. No IPR or circuit patent licenses are implied. CML reserves the right at any time without notice to change the said algorithms, methods and circuitry and this product specification. CML has a policy of testing every product shipped using calibrated test equipment to ensure compliance with this product specification. Specific testing of all circuit parameters is not necessarily performed.

 CML Microcircuits (UK) Ltd COMMUNICATION SEMICONDUCTORS	 CML Microcircuits (USA) Inc. COMMUNICATION SEMICONDUCTORS	 CML Microcircuits (Singapore) Pte Ltd COMMUNICATION SEMICONDUCTORS
Tel: +44 (0)1621 875500 Fax: +44 (0)1621 875600 Sales: sales@cmlmicro.com Tech Support: techsupport@cmlmicro.com	Tel: +1 336 744 5050 800 638 5577 Fax: +1 336 744 5054 Sales: us.sales@cmlmicro.com Tech Support: us.techsupport@cmlmicro.com	Tel: +65 62 888129 Fax: +65 62 888230 Sales: sg.sales@cmlmicro.com Tech Support: sg.techsupport@cmlmicro.com
- www.cmlmicro.com -		