AN10583

Realizing an MP3 player with the LPC2148, using libmad and EFSL

Rev. 01 — 18 April 2007 Application note

Document information

Info	Content
Keywords	MP3, player, codec, decoder MP3, Layer I, Layer II, Layer III, libmad, MAD, EFSL, Embedded File System Library, MMC, card, SD card
Abstract	This document describes how to design an MP3 player using the LPC2148

Revision history

Rev	Date	Description
01	20070418	Initial version

Contact information

For additional information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

1. Introduction

Thank you for choosing the NXP LPC2000 family for your application!

This Application Note describes how to integrate the libmad MPEG 1/2 layer 3 library and the Embedded File System Library EFSL in a LPC2148 device to realize MP3 player. The Keil MCB2140 board is used to test the application and the Rowley CrossStudio for ARM has been chosen as a reference tool-chain.

The MP3 player software package is available for download at http://www.nxp.com/standardics/support/documents/?type=software under the title *LPC2148-Based MP3 Player Software Package*.

1.1 About libmad

The MAD is a high-quality MPEG audio decoder library. It currently supports MPEG-1 and the MPEG-2 extension to lower sampling frequencies, as well as the de facto MPEG 2.5 format. All three audio layers — Layer I, Layer II, and Layer III (i.e. MP3) — are fully implemented. MAD has the following special features:

- 24-bit PCM output
- 100 % fixed-point (integer) computation
- completely new implementation based on the ISO/IEC standards
- available under the terms of the GNU General Public License (GPL)

For more information about *libmad* please refer to http://www.nxp.com/redirect/underbit.com/products/mad

Although *libmad* has a GPL license it doesn't mean that the MP3 algorithm can be used in any application without paying a fee. Royalties must be paid. For more info please refer to http://www.nxp.com/redirect/mp3licensing.com/royalty

If your application needs a fully open, non-proprietary, patent-and-royalty-free, general-purpose codec please consider the Vorbis and Speex solutions. For more info please refer to http://www.nxp.com/redirect/xiph.org/vorbis, and http://www.nxp.com/redirect/speex.org.

In this demo *libmad* version *0.15.1.b* library is used. Some specific modifications of the standard *libmad* files have been done to reach the following goals:

1. To reduce the amount of memory RAM used by *libmad* to match the LPC2148 RAM memory constrain.
2. To eliminate dynamic malloc()/free() functions

1.2 About EFSL

The EFSL project aims to create a library for filesystems, to be used on various embedded systems. Currently EFSL supports the Microsoft FAT filesystems family. The library currently supports FAT12/16/32 reading and writing on SD-cards. It is EFSL's intent to create pure ANSI C code that compiles on anything that bears the name 'C compiler'. Adding code for your specific hardware is straightforward. Just add code that fetches or writes a 512-byte sector, and the library will do the rest. Existing code can of course be used, own code is only required when you have hardware for which no target exists. For example, we support secure digital cards in SPI mode. EFSL has been ported to the LPC2000 family. For more info about EFSL please refer to http://www.nxp.com/redirect/efsl.be or http://www.nxp.com/redirect/sourceforge.net/projects/efsl. EFSL is available under GNU General Public License with an exception clause.

1.3 About LPC2148

The 16/32-bit LPC2000 family is based on an ARM7TDMI-S core operating up to 72 MHz together with a wide range of peripherals including serial interfaces, 10-bit ADC/DAC converter and external bus options. At the moment in which the MP3 example is written the LPC2148 was the only device suitable to realize a low cost MP3 player based on free GNU libraries. However, today, a new powerful device, the LPC2888 is appearing in the market. This device can be used to realize a low cost, high quality MP3 player due to the integration of a stereo high quality 16-bit codec. For those applications in which the audio quality is not the main point, the LPC2148 can be used to create a mono, low cost, single chip MP3 player. This example, of course, can be used as a reference to start the development on the LPC2888 device.

For more detail on the LPC214x family please refer to NXP website
http://www.nxp.com/pip/LPC2141_42_44_46_48_2.html

For more detail on the LPC2888 please refer to NXP website
http://www.nxp.com/pip/LPC2880FET180.html

1.4 About Keil MCB2140

The Keil MCB2140 is the evaluation board designed by Keil, an ARM company. The Keil MCB2140 evaluation board introduces you to the NXP LPC2140 ARM family and allows you to create and test working programs for this advanced architecture. The Keil MCB2140 evaluation board connects to your PC using the serial port (the utility can be downloaded at http://www.nxp.com/standardics/support/documents, under the title *LPC2000 Flash ISP Utility, V2.2.3*), or the JTAG interface. An on-chip USB interface, two serial interfaces, a speaker, an analog input (via potentiometer), and eight LEDs make this board a great starting point for your next ARM project. In this MP3 demo the speaker and the SD/MMC card connector, interfaced to the SSP serial port, have been used to design the player. For more detail on the MCB2140 board please refer to the Keil web site http://www.nxp.com/redirect/keil.com/arm/mcb2140

1.5 About Rowley CrossStudio for ARM

CrossStudio for ARM is a complete C/C++ and assembly code development system for ARM7, ARM9, XScale, and Cortex-M3 micro-controllers. It is based on a C, C++ and Assembler toolchain from the GNU Compiler Collection. In this demo the Wiggler low cost emulator is used. The schematic of the emulator is available in the file section of the Yahoo LPC2000 public web site.

For more info about wiggler commercial emulators refer to:
http://www.nxp.com/redirect/macraigor.com/wiggler

For more info about Rowley CrossStudio for ARM refer to:
http://www.nxp.com/redirect/rowley.co.uk/arm/index

For more info about Yahoo LPC2000 newsgroup refer to:

http://www.nxp.com/redirect/tech.groups.yahoo.com/group/lpc2000/

In this demo the Rowley CrossStudio for ARM v1.6 (with gcc v4.1.0 GNU compiler behind the IDE) has been chosen.

1.6 Limitations of the MP3 demo

The LPC2148 has a 10-bit DAC converter without any digital interpolator or any analog output reconstruction filter. The DAC is used as it is as a rendering audio output device. It

means that the sample rate of the 10-bit DAC is equal to the sample rate of the output stream generated by the MP3 decoder. With this approach the audio quality is low and only a mono channel is supported because the LPC2148 has only one DAC, so only a mono audio channel can be supported.

Because the MP3 files are often created in the stereo format, you need to convert them in the mono version before storing in the MMC memory card. A free GPL license tool MediaCoder is a possible choice to realize this conversion activity. For more info on the MediaCoder please refer to http://www.nxp.com/redirect/mediacoder.sourceforge.net. The MMC must be formatted in one of the FAT32 or FAT16 filesystems.

To increase the audio quality we suggest connecting an additional external audio DAC to the LPC2148.

1.7 Memory requirements

The EFSL and the *libmad* libraries consume memory to work. The amount of the memory is about:

Table 1. Memory requirements for the MP3 player demo

	RAM [kB]	Code [kB]
Libmad + EFSL	33	109

Because the LPC2148 has only 32 kB of user RAM and the demo requires more than 32 kB of RAM, the additional 8 kB of USB/user RAM, is fully allocated for the stack and for the libraries data structures. Therefore 25 kB of the 32 kB of user RAM and 8 kB of the additional USB RAM (25K User + 8 kB of USB RAM = 33 kB total RAM used) is used in this demo. The rest of the user RAM (7 kB) is reserved for the user application.

To get these memory values the Rowley CrossStudio for ARM v1.6 (with gcc v 4.1.0 GNU compiler behind the IDE) is used and the –O option is selected in the Additional Compiler Options tab. All C modules have been compiled in the 32-bit ARM code mode.

[Select Project->Proprietary then chose Compiler tab and finally put -O in the Additional Compiler Options field in the Rowley CrossStudio for ARM]

1.8 General notice

This document is intended as an application guideline, this means that in some cases depending on the requirements of the application there is room for change in some software modules by the application designer.

This program (the module of this application that are not part of Libmad and of the EFSL) is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version. This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY and WITHOUT ANY SUPPORT; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. For the rest of the modules (*Libmad* and EFSL) we referred to the related licenses.

See the GNU General Public License for more details. You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

2. MP3 player components

Before describing the configuration of the EFSL and of the *libmad* decoder libraries, the following block diagram shows the hardware structure of the MP3 player

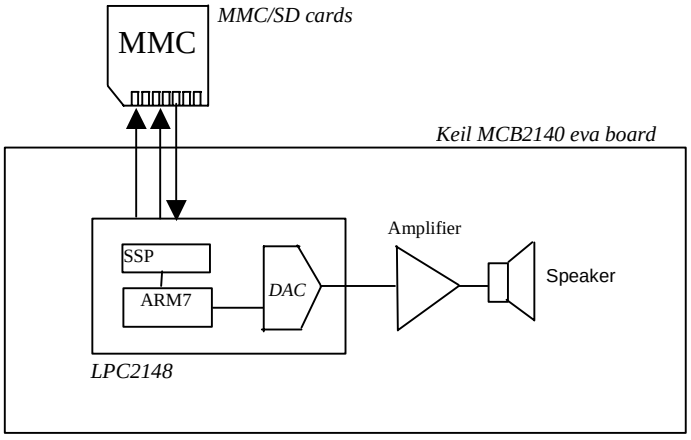


Fig 1. Keil MCB2140 board

The MCB2140 is populated by a small speaker and an MMC/SD card connector, and can be used as the hardware platform to realize the player. An additional MMC memory card is inserted in the MMC/SD card connector for the MP3 file storing. The firmware of the MP3 player is stored in the LPC2148 embedded flash memory. The LPC2148 single channel DAC is used to convert the decompressed MP3 stream in an analog audible mono signal.

The EFSL is a library of functions dedicated to open, read, and write files (and much more) stored in a memory card that has been formatted in a FAT32 or FAT16 filesystems. The user can format the card using a PC or other appliance, store the MP3 files (mono version) in the memory card and finally place the memory card in the slot of the MCB2140 board. A specific software driver for the LPC2000 family is available in the public http://www.nxp.com/redirect/sourceforge.net/projects/efsl web site.

The MP3 file is read out from the memory card in frame pieces (frame length is in the range of 1500 B, but this value can change depending on the content of the MP3 file) and delivered to the MP3 decoder. The output of the MP3 decoder is a decompressed frame that is pushed in a software FIFO buffer. The software FIFO is fetched periodically by an interrupt service routine that sends the raw samples to the LPC2148 DAC.

3. Setup environment

The Rowley Cross Studio IDE environment has been chosen to compile the *libmad* and EFSL. The Rowley Crosswork V1.6 environment uses the GCC compiler V4.1.0. This paragraph shows how to configure the environment and also how to compile the libraries using the GCC command line compiler.

After installing and launching the Rowley CrossStudio IDE the following window appears:

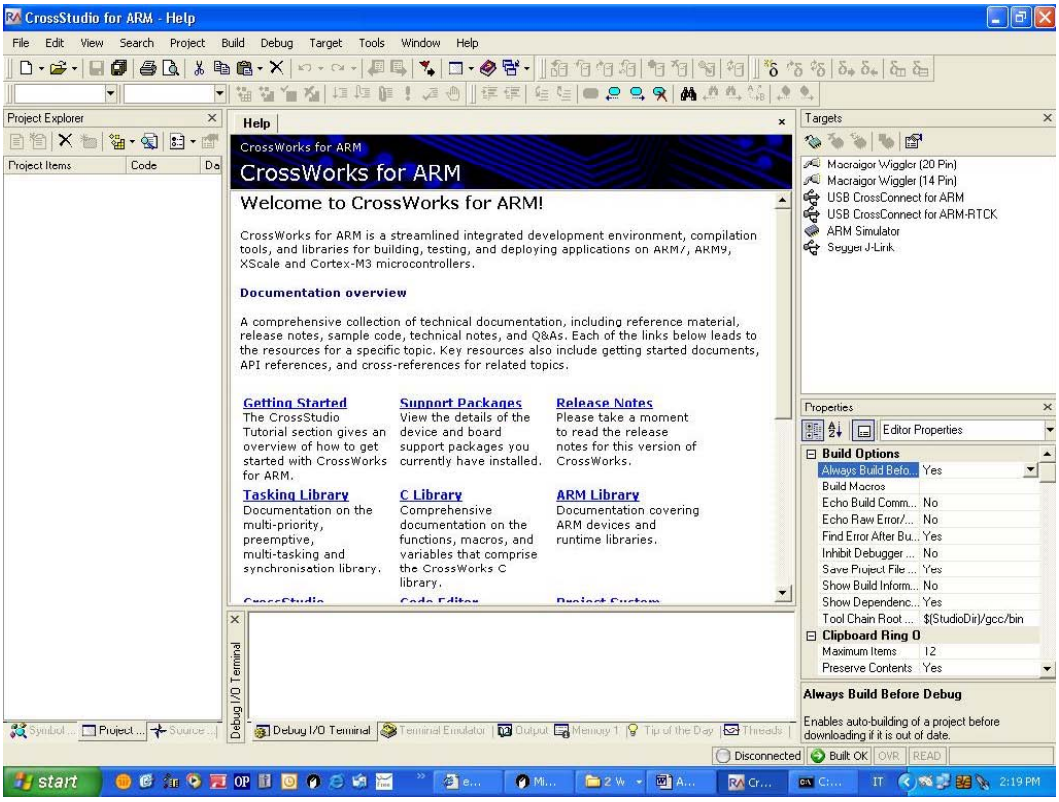


Fig 2. Rowley Cross Studio IDE

The project file that includes all libraries and device settings for the LPC2148 and for the Keil MCB2140 board has been created for you in the file:

- MP3Player.hzp

Please select File->Open Solution from the Rowley CrossStudio IDE and select the *MP3Player.hzp* in the project directory. The IDE loads all files used in the MP3Player project. You can see what the IDE looks like in Fig 3.

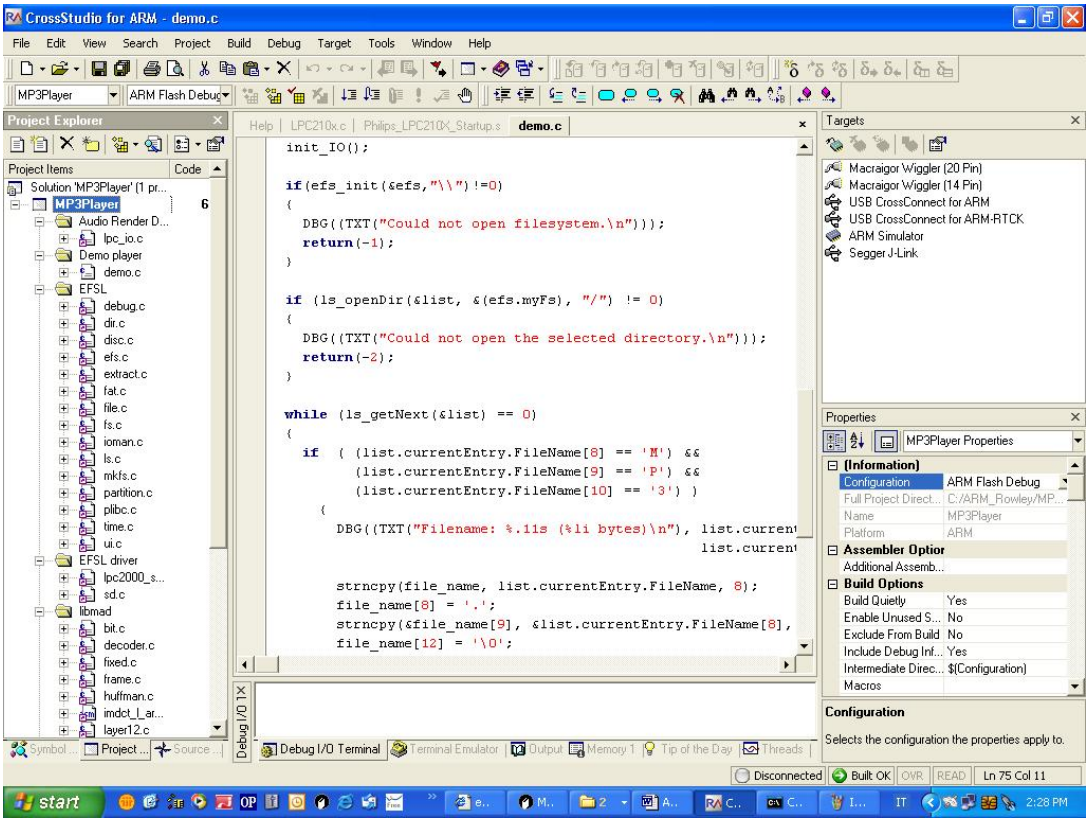
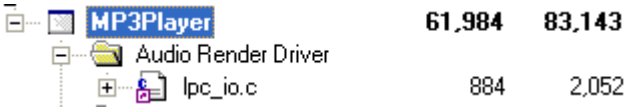


Fig 3. MP3 player demo project loaded in the CrossStudio IDE

Source files have been grouped in different sections. The audio Render Driver section contains the code related to the DAC audio

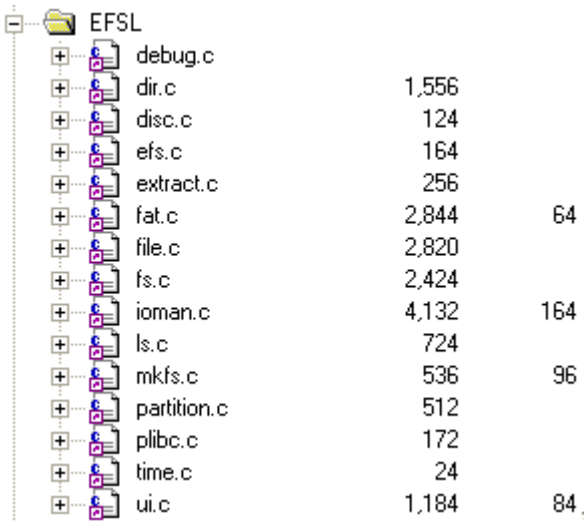


render device. You can modify the code, for example, to include the support from an external high-quality audio DAC.

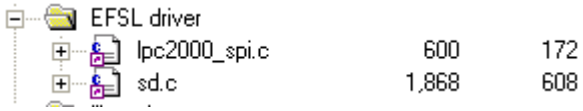
The **section Demo Player**:

shows the main module that we describe in detail later on.

The **section EFSL**

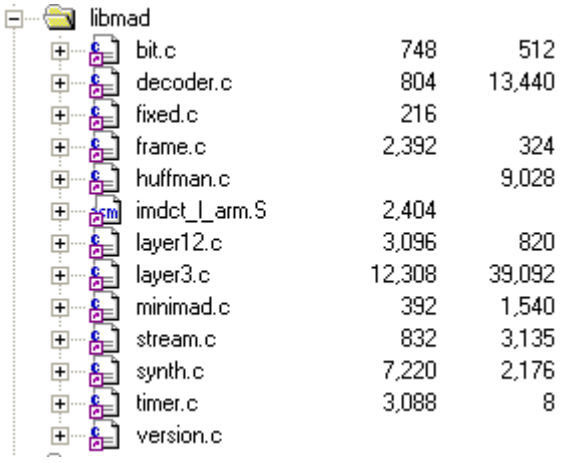


includes all *.c* modules used to manage the Embedded File System Library. This library enables the demo.c module to open and read MP3 files from the external MMC memory card. The driver that manage the MMC low level functions has been confined in the **EFSL driver section**



Excluding the configuration settings of the library EFSL no changes have been made in the file contents of this library.

The **libmad section** groups the .c module related to the *libmad* library



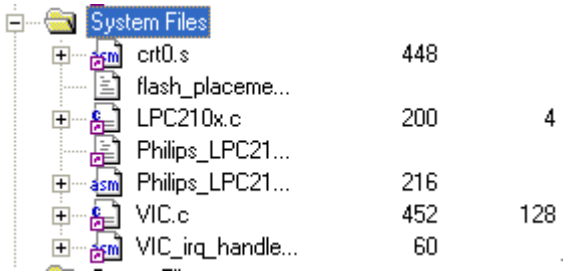
mp3_play(&file) is the only function in the *demo.c* main program that access to the *libmad* functionalities. *Libmad* source code is changed to match the goals described in

the paragraph 1.1 “About Libmad”. According to the GPL license constraints the modifications are described as follows:

1. The module *synth.c* the functions *synth_full* and *synth_half* have been modified to reduce the RAM memory requirement cutting the output *pcm.samples* buffer. The buffer is replaced with the direct access to the audio render FIFO declared in the module *io_lpc.c*. This FIFO is defined to manage only a mono-audio signal. Because the *pcm.samples[ch]* buffer store 9216 bytes for a stereo channel, this change saves 7168 bytes for the application in the case of a mono audio signal is managed. To reach this target the order in which the two *”for”* cycles are executed in these functions has been swapped.
2. Because the *synth.c* module access directly to the audio render, the function *output* in the module *midmad* became unnecessary, but is still there because the presence of the output LED toggle command TOGGLE_LIVE_LED0().
3. In the module *synth.c* the function *mad_synth_frame* has been changed to insert the call to the set_dac_sample_rate(synth->pcm.samplerate). When a new output sample frequency has been detected by the MP3 decoder, suddenly this information is sent to the audio render.
4. The *malloc()/free()* functions in the modules *decoder.c stream.c and layer3.c* have been cancelled and replaced with a reference to a static structures always allocated in RAM memory.
5. In the module *midmad*.c an additional mp3_play() function has been created to hide all details (software encapsulation) of the *libmad* implementation for a *libmad* user.

The reader should also consider that only the SYNC mode feature of the *libmad* has been implemented in this demo.

System files are used to initiate the LPC2148 device in a proper way. System Files are grouped in the **System Files section** of the Project Explorer window



3.1 Configuring compiler and linker for the proper operation

Although all setting options have been included in the ***MP3Player.hzp*** project file, a short walk through the IDE settings can help you in the developing/porting phase.

3.1.1 Global directories settings

Command line tools must find the project files and include files. The following additional paths:

$(ProjectDir)/render-driver
$(ProjectDir)/libmad
$(ProjectDir)/efsl/inc
$(ProjectDir)/efsl/conf
$(ProjectDir)/efsl/inc/interfaces.

Fig 4. Configuring the project directories paths

have been added in the User Include Directories field of the Preprocessor tab of the Project Options window in both *ARM Flash Release* and *ARM Flash Debug* profiles. The additional DEBUG #undef directive (Preprocessor tab), the .hex file creation option (Linker tab), and the Include Debug Information option (Build tab), have been added only in the *ARM Flash Release* profile. The DEBUG #undef option blocks the JTAG message sending channel when the code is deployed in production (semi-hosting feature of the debugger disabled).

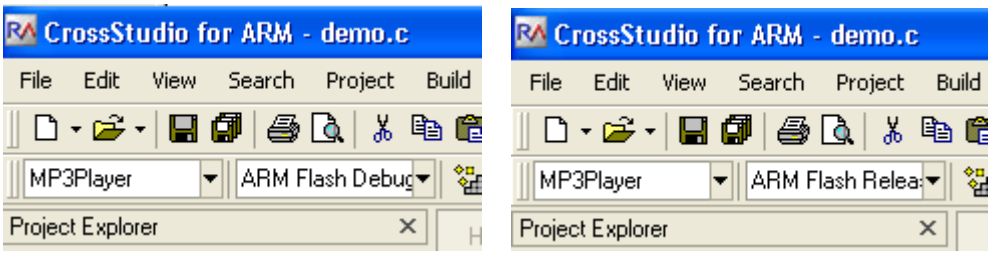


Fig 5. Project Profiles of the Rowley CrossWork IDE

Moving between the two profiles is possible selecting *ARM Flash Debug* or the *ARM Flash Release* options from upper left combo-box in the CrossStudio IDE.

3.1.2 Configuring the compiler

The code is compiled in 32-bit mode because the amount of fixed-point math calculation is very high and we need the extended math instruction set only available in 32-bit mode. USB Memory RAM is used and the stack is allocated there. USB RAM is not working by default in the LPC214x so it must be switched on. The standard Philips_LPC210x_startup.s that is provided by the Rowley CrossStudio IDE has been changed to take in account of this additional hardware initialization and a new Philips_LPC2148_startup.s file has been created and placed in the MP3Player project directory. The Philips_LPC2148_startup.s includes a small piece of code that enables the internal USB RAM, however, the programmer should also inform the linker to add the stack section to the USB DMA RAM memory section placement. Please refer to the Rowley CrossStudio documentation for more information about memory segments placement.

Compiler options are accessible by selecting the Proprietary option of the MP3Player project in the Project Explorer Window. Fig 6 indicates the options set in both *ARM Flash Release* and *ARM Flash Debug* profiles.

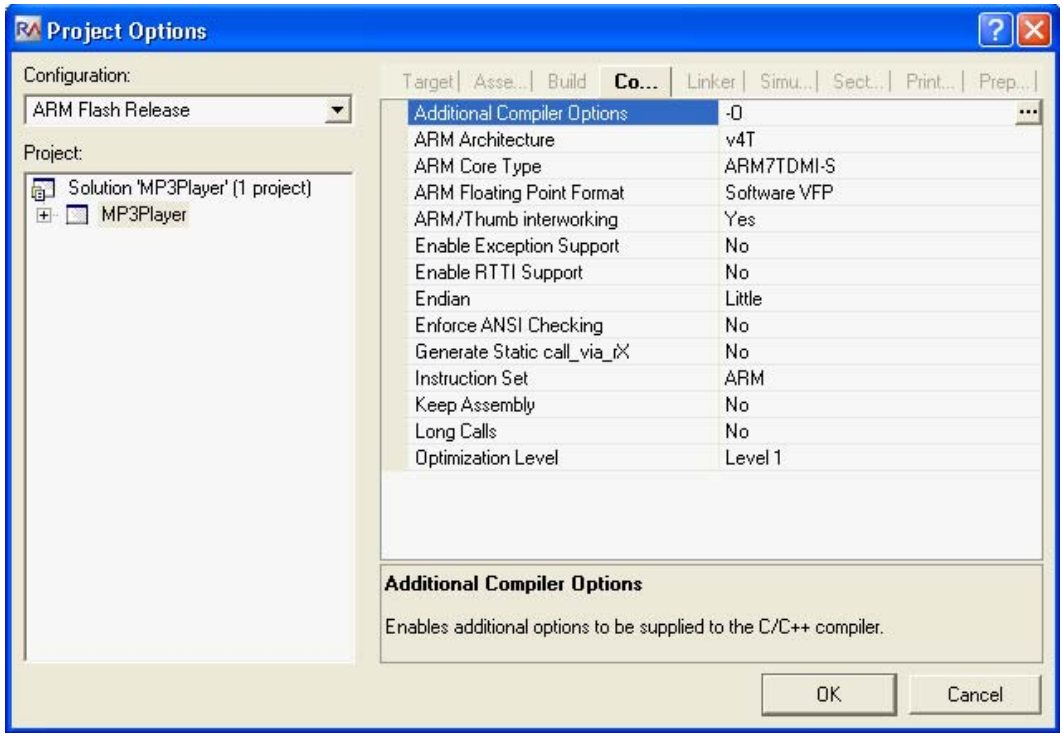


Fig 6. Configuration settings of the compiler

3.1.3 Configuring the linker

The linker tab is accessible in the Project Option window. The only difference between the options of the *ARM Flash Release* and *ARM Flash Debug* profiles is the additional *.hex* file created in the *ARM Flash Release* section. Please note that the stack size is 8 kB. The placement of the stack is defined in the *flash_placement.xml* file. Because the standard Rowley *flash_placement.xml* file is changed to move the stack from the Internal SRAM to the USB DMA RAM section a specific *flash_placement.xml* has been defined and imported in the MP3Player project directory

The final parameter set for the linker is shown in Fig 7.

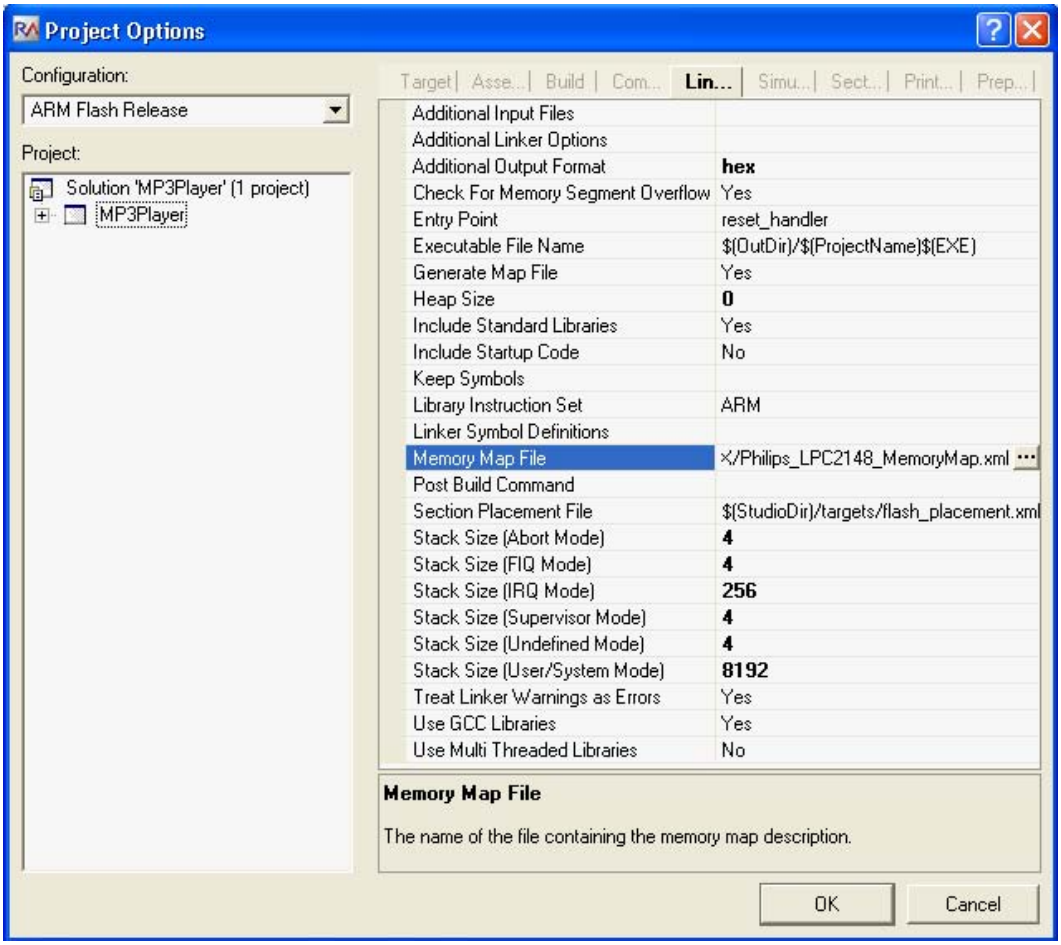


Fig 7. Linker settings

3.2 Configuring EFSL and libmad libraries

Libraries must be configured for the proper operation with the ARM7 LPC2148.

3.2.1 Configuring libmad

The files related to the *libmad* library have been grouped in the *libmad* section of the Project Explorer window.

Libmad is a library that can run on different platforms, and is optimized for ARM7. The complier must be aware about the environment in which the library operates. This is done using the following compiler directives:

```
FPM_ARM           /* An ARM architecture is used */
SIZEOF_INT=4      /* Integers data types are 4 bytes long */
SIZE_LONG=4       /* Long are data types 4 bytes long */
SIZE_LONG_LONG=8  /* Long long data types are 8 bytes long */

ASO_IMDCT  /* An optimized ARM7 assembly module is used to
calculate the DCT*/
```

To set this compiler directive in the IDE just select the *libmad* folder in the Project Explorer windows, then right mouse click and select Properties menu option, select the Preprocessor tab, and fill in the Preprocessor Definitions field.

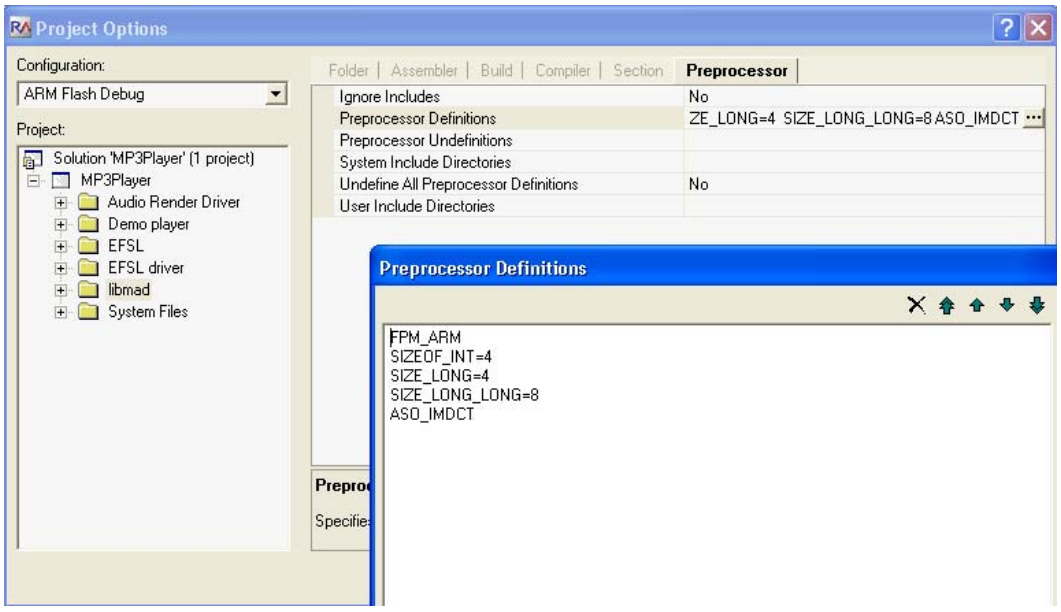


Fig 8. Configuring libmad

3.2.2 Configuring the EFS Library

The files related to the EFS library have been grouped in the EFSL section of the Project Explorer window.

The *debug.h* file stored in the .\ *MP3Player\efsl\inc* directory has been changed to support the semi-hosting feature of the debugger. The following piece of code shows that all debug messages will be sent, via JTAG, to the I/O terminal windows of the debugger. *debug_printf* is the “c” function provided by the Rowley library to realize that function:

```
#ifdef HW_ENDPOINT_LPC2000_SD
#include <__cross_studio_io.h>
#define TXT(x) x
#define DBG(x) debug_printf x
#define FUNC_IN(x) ;
#define FUNC_OUT(x) ;
#define debug debug_printf
```

To keep the RAM memory usage at the minimum level, the number of buffers in the EFSL has been reduced at the minimum and some definitions in the file *config.h* stored in the \.*MP3Player\efsl\conf* directory have been changed:

```
#define IOMAN_NUMBUFFER 1 /* 512 byte RAM used on the LPC2148 */
#define IOMAN_NUMITERATIONS 3
#define IOMAN_DO_MEMALLOC
```

The following memory cards are used in the application tests:

SunDisk 256 MB RS-MMC
Transcend MMCplus 128 MB
SunDisk SD card 128 MB

The SPI interface connected to the MMC/SD card slot is set to generate a data clock of about 15 MHz in the normal data transfer mode and less that 400 kHz during the initialization phase (see MMC specs).

3.3 Configuring system files

The system files initialize the LPC2148 for the proper operating condition. The MP3player demo needs the full power of the LPC2148. Therefore the LPC2148 must operate at the maximum speed (60 MHz) to switch on and configuring the PLL. The USB RAM must be powered, the IRQ interrupt must be handled to manage the timer 0 and the VPB divider is set to 1. A small piece of assembly code has been added to the *Philips_LPC2148_Startup.*s assembly module:

```
/* Activate Additional USB AHB RAM */
#if defined(USE_USB_RAM)
  ldr     R0, =PCONP
  ldr     R1, [R0]
  orr     R1,  R1, #PCONP_Val
  str     R1, [R0]
#endif
```

Philips_LPC2148_Startup.s must be compiled including the following preprocessor directives:

```
USE_USB_RAM               /* USB RAM is used by the application */
VECTORED_IRQ_INTERRUPTS   /* Fast IRQ management */
VPBDIV_VAL=1              /* Timer 0 runs with 60 MHz CPU clock */
STARTUP_FROM_RESET
PINSEL1_VAL=0x000080000   /* Connect DAC output */
```

To set this compiler directives in the IDE just select the *Philips_LPC2148_ Startup.s* file in the Project Explorer windows, then right mouse click and select Properties menu option, select the Preprocessor tab and fill in the Preprocessor Definitions field with the above values as shown in Fig 9.

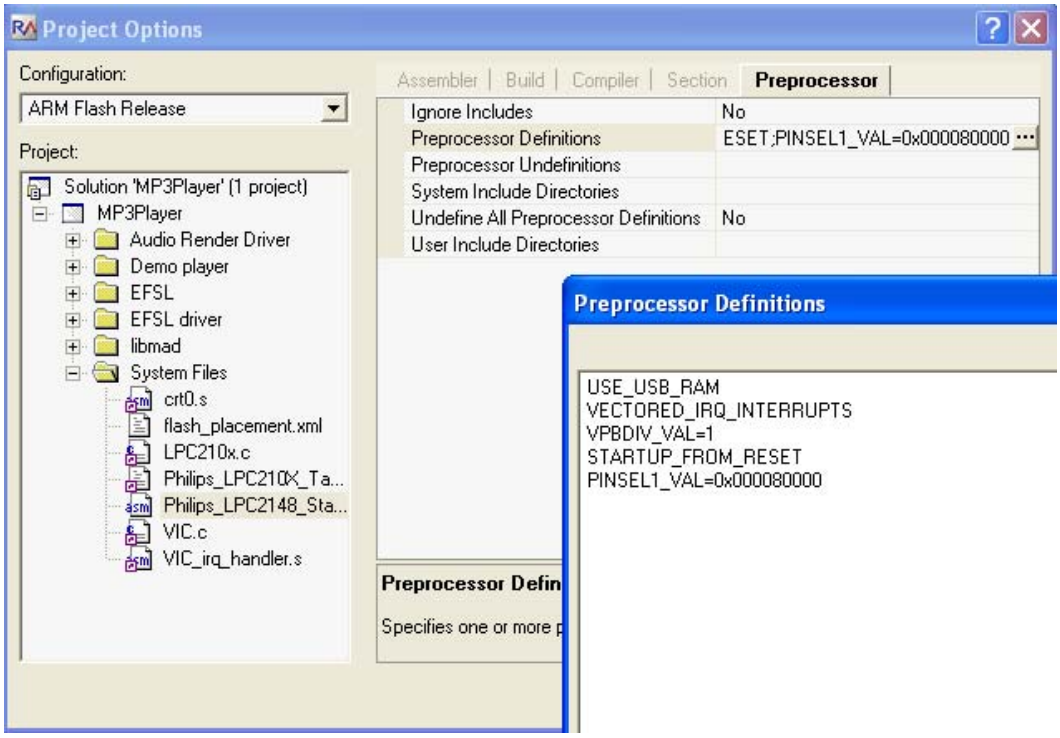


Fig 9. Configuring the system startup file

4. Compiler batch files

The compiler batch files have been created for you and are available in the project directory MP3Player. To work properly, these batch files assume that the following directory structure has been created for the project files and for the Rowley files.

Project file directory ->

"C:\ARM_Rowley\MP3Player"

Rowley CrossStudio for ARM directory ->

"C:\Program Files\Rowley Associates Limited\CrossWorks for ARM 1.6"

There are two batch files:

build_all_debug.bat /* Creates the JTAG and LED message oriented release */

build_all_release.bat /* Produces the release version with the .hex file */

The release version creates the production version of the project with the *MP3Player.hex* file in the *"C:\ARM_Rowley\MP3Player\ARM Flash Release"* directory. If your directory structure is different you can always create the batch files from the Rowley CrossStudio IDE.

[Select the *MP3Player Folder* in the *Project Explorer Windows*, then mouse right click, then select *Build Information* menu option, and finally copy and past the content of the *Build Steps windows* in a .bat file replacing the "rm" command with "del" DOS command.]

5. MP3 player demo operation

The main.c file shows how the MP3 demo example works:

```
# include <stdio.h>
# include <efs.h>
# include <ls.h>
# include <string.h>
# include <targets/lpc214x.h>
# include "lpc_io.h"
# include "mad.h"
# include "debug.h"
# include "midmad.h"

EmbeddedFileSystem efs;
EmbeddedFile file;
DirList list;
unsigned char file_name[13];
unsigned int size;

int main()
{
  init_IO();
  if(efs_init(&efs,"\\")!=0)
  {
    DBG((TXT("Could not open filesystem.\n")));
    return(-1);
  }
  if (ls_openDir(&list, &(efs.myFs), "/") != 0)
  {
    DBG((TXT("Could not open the selected directory.\n")));
    return(-2);
  }

  while (ls_getNext(&list) == 0)
  {
    if (   (list.currentEntry.FileName[8] == 'M') &&
           (list.currentEntry.FileName[9] == 'P') &&
           (list.currentEntry.FileName[10] == '3') )
    {
      DBG((TXT("Filename: %.11s (%li bytes)\n"),
               list.currentEntry.FileName,
               list.currentEntry.FileSize));

      format_file_name(file_name, list.currentEntry.FileName);

      if(file_fopen(&file,&efs.myFs,file_name,'r')==0)
      {
        mp3_play(&file);
        file_fclose(&file);
        TOGGLE_LIVE_LED1();
      }
      else
        DBG((TXT("Could not open file.\n")));

    } /* if */
  } /* while */

  fs_umount(&(efs.myFs));
  return 0;
```

The `init_IO()` initiates the LPC2148 Timer 0 to generate periodic interrupts and initiates some data-structures. The rate of the periodic interrupts is the same as the decoded stream sample rate. This data is stored in the header of the MP3 file. Then the EFSL is initiated invoking the `efs_init(&efs,"\\")` function. If an error is encountered the demo stops its execution and sends the message `"Could not open filesystem"` to the CrossStudio IDE via the JTAG interface. Because the JTAG interface couldn't be connected in the *release* standalone version of this demo, two different profiles have been created.

The root directory of the MMC memory card is explored in the

```
while (ls_getNext(&list) == 0)
{
…
} /* while */
```

cycle. The file name and file size of each file that has the MP3 extension are printed out through the JTAG interface in the Debug I/O terminal window of the CrossStudio Debugger as shown in Fig 10.

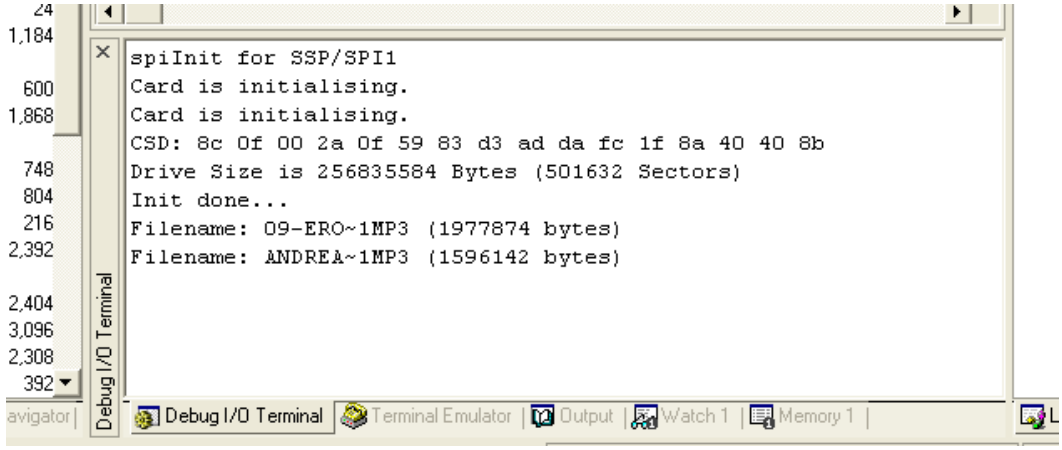


Fig 10. Debug I/O Terminal window of the Rowley CrossStudio IDE

The messages appear only if the *ARM Flash Debug* profile has been selected.

Finally each MP3 file found in the root directory is opened and then played invoking the `mp3_play(&file)` function:

```
if(file_fopen(&file,&efs.myFs,file_name,'r')==0)
{
  mp3_play(&file);
  file_fclose(&file);
  TOGGLE_LIVE_LED1();
}
```

5.1 Output LEDs

MCB2140 is populated by eight LEDs, five of which are used to display status information of the MP3 player. The LEDs are connected to the P1 port pins of the LPC2148 and are driven, only in the *ARM Flash Debug* profile, by the following C macros:

```
#define TOGGLE_LIVE_LED0()   toggles the LED P1.16
#define TOGGLE_LIVE_LED1()   toggles the LED P1.17
#define TOGGLE_LIVE_LED2()   toggles the LED P1.18
#define TOGGLE_LIVE_LED3()   toggles the LED P1.19
#define TOGGLE_LIVE_LED4()   toggles the LED P1.20
#define SET_LIVE_LED4()      switch on the LED P1.20
#define CLR_LIVE_LED4()      switch off the LED P1.20
```

P1.16 or LED0 toggles every decoded output frame. The function TOGGLE_LIVE _LED0() is placed in the module *midmad.c*, in the *mad_flow_output* function of the decoder.

P1.17 or LED1 toggles after the execution of each MP3 excerpt. The function TOGGLE_LIVE _LED1() is placed in the module *demo.c*, in the *main()* function of the demo.

P1.18 or LED2 toggles if the FIFO buffer that receives the decoded output stream is at a certain time empty. This means that the CPU power is not enough to decode, in real time, a specific MP3 file. The function TOGGLE_LIVE_LED2() is placed in the *lpc_io.c* module in the function *tc0.*

P1.19 or LED3 toggles every time the DAC update interrupt service routine *tc0* is called. Because the rate of the interrupt service routine calls is equal to the frequency of the decoded output stream, for example 44 kHz for a MP3 file encoded at this frequency, LED2 toggles at that frequency. The function TOGGLE_LIVE_LED3() is placed in the *lpc_io.c* module in the function *tc0.*

P1.20 or LED4 is not used in this demo.

5.2 Average CPU usage

The CPU usage depends on the parameters of the MP3 encoded file. One of the most important parameters is the decoded output stream rate. Furthermore, the length of the decoded output frame block is also important:

If **Td** [ms] is the maximum block decoding time of an MP3 frame, **nb** [samples] is the decoder output block length (usually 1152 or 576 samples), and **fd** [Ksamples/s] is the decoded stream rate, **the CPU load is**

CPUload[%] = **Td**[ms] * **fd**[Ksamples/s] /**nb**[samples] * 100

We achieve a maximum **CPU load of 73 %** and a minimum **CPU load of 41 %** testing MP3 files of different input stream rate and output row sample rate. In this testing the following combination input/output stream frequency are used:

Table 2. Input/output stream frequency

Combination of input and output stream rate during performance tests	
56 Kbit/s input -> 32 Ksample/s output	56 Kbit/s input -> 44 Ksample/s output
64 Kbit/s input -> 32 Ksample/s output	64 Kbit/s input -> 44 Ksample/s output
80 Kbit/s input -> 32 Ksample/s output	80 Kbit/s input -> 44 Ksample/s output
96 Kbit/s input -> 32 Ksample/s output	96 Kbit/s input -> 44 Ksample/s output
112 Kbit/s input -> 32 Ksample/s output	112 Kbit/s input -> 44 Ksample/s output
128 Kbit/s input -> 32 Ksample/s output	128 Kbit/s input -> 44 Ksample/s output
Variable bit rate input (avg 92kbit/s) > 44 Ksample/s output	Variable bit rate input (avg 57kbit/s) > 44 Ksample/s output
Variable bit rate input (avg 71kbit/s) > 44 Ksample/s output	

Measured conditions are:

CPU operating frequency = 60 MHz, VPB divisor = 1, Memory Accelerator Module switched on. To measure the maximum decoding time, the output FIFO length has been set to infinite. At those conditions the decoded output time **Td** is the maximum toggle time of the LED0 output signal. To calculate the CPU load the following equation is used: CPU load [%] = Td * fd/nb * 100. The TOGGLE_LIVE_LED3(); macro in the tc0() function has been deleted. Of course, an audio mono signal is used in all tests.

5.3 Further development steps

Some optimization activities can start from this demo. For example, the tc0() interrupt service routine can be rewritten in assembly language because it is a very critical routine, as well as some math intensive functions of the libmad package like synth_half, synth_full, and dct32.

Libmad is a well-known library to decode MP3 files. Libmad is also used in the popular free VLC player, however, as you can see in the last table, the porting of the actual version of libmad on an ARM7 (@60 MHz) loads the CPU from the 40 % to 70 % of its power, so only one audio channel can be decoded in real time. Browsing the web we found one interesting alternative represented by the Helix MP3 Decoder suitable to decode, in real time, a stereo MP3 stream. Additional audio codec is also available, like AAC. All these codecs have a specific RCSL and RPSL license. For more info about this codec please refer to:

http://www.nxp.com/redirect/datatype.helixcommunity.org/mp3dec

6. MP3 patents

The list of MP3 patents and related Ids can be found here:

http://www.nxp.com/redirect/mp3licensing/patents/index

7. Legal information

7.1 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

7.2 Disclaimers

General — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in medical, military, aircraft, space or life support equipment, nor in applications where failure or malfunction of a NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors accepts no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is for the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

7.3 Trademarks

Notice: All referenced brands, product names, service names and trademarks are property of their respective owners.

8. Contents

For more information, please visit: http://www.nxp.com
For sales office addresses, email to: salesaddresses@nxp.com

Date of release: 18 April 2007
Document identifier: AN10583_1