

AN11178

MP3 player solution on NXP LPC1700 series

Rev. 1 — 1 April 2012

Application note

Information

Info	Content
Keywords	LPC1759FBD80; LPC1758FBD80; LPC1756FBD80; LPC1754FBD80; LPC1752FBD80; LPC1751FBD80; LPC1769FBD100; LPC1768FBD100; LPC1768FET100; LPC1767FBD100; LPC1766FBD100; LPC1765FBD100; LPC1765FET100; LPC1764FBD100; LPC1763FBD100 MP3, LPC1700, Helix MP3 decoder, I2S, File System, USBHostLite
Abstract	This application note describes how to playback MP3 files on NXP's LPC1700 series with the Helix MP3 decoder. The Helix MP3 library has an integer fixed-point decoder which is optimized and suitable for ARM processors. This solution takes full advantage of LPC1700's high performance and rich peripherals to make it possible to playback most common MP3 files. A USB Host stack and a file system are also integrated to support accessing files on SD/MMC card and USB flash disk.



Revision history

Rev	Date	Description
1	20120401	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

The LPC1700 is a series of ARM Cortex-M3 based microcontrollers for embedded applications requiring a high level of integration and low power dissipation. Operating at speeds up to 120 MHz, they have up to 512 kB of flash, up to 64 kB of SRAM and rich peripherals such as Ethernet, USB 2.0 Host/OTG/Device, CAN 2.0B, 12-bit ADC, 10-bit DAC, SPI/SSP, I2S, DMA, etc.

The Helix MP3 decoder provides MPEG-compliant decoding of MP3 content. Both floating-point and fixed-point decoder implementations are available. The fixed-point decoder is optimized especially for ARM processors but can run on any 32-bit fixed-point processor. In this solution it has been ported to Thumb-2 code.

This solution takes full advantage of LPC1700's high performance and rich peripherals to make it possible to playback most common MP3 files.

To be able to access the MP3 files in SD/MMC card and USB flash disk, this solution also integrates the FatFs library, a widely used FAT12/16/32 file system, and integrates the USBHostLite library, a stripped-down USB host stack.

This application note describes how to playback MP3 files on the NXP LPC1700 series using the Helix MP3 decoder. It includes:

- General description of the system.
- Hardware design and schematic.
- Software design, block diagram and flowchart.
- Demonstration and test result.

The sample software is tested on Code Red's RDB1768 evaluation board, with a 2 GB SanDisk Micro SD card and a 1 GB USB flash disk. Software is available for three IDE's/toolchains:

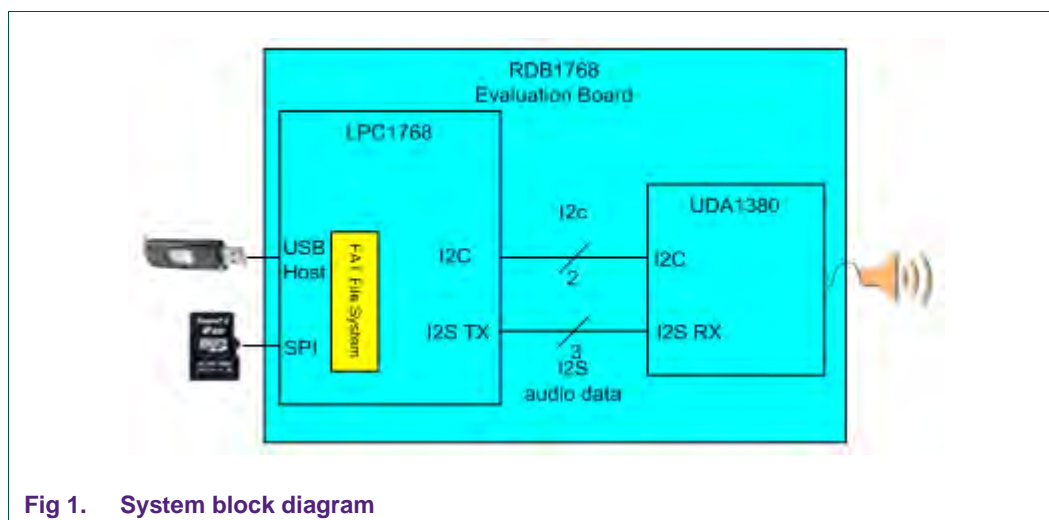
- LPCXpresso.
- Keil μ Vision.
- IAR EWARM.

2. System overview

2.1 Overview

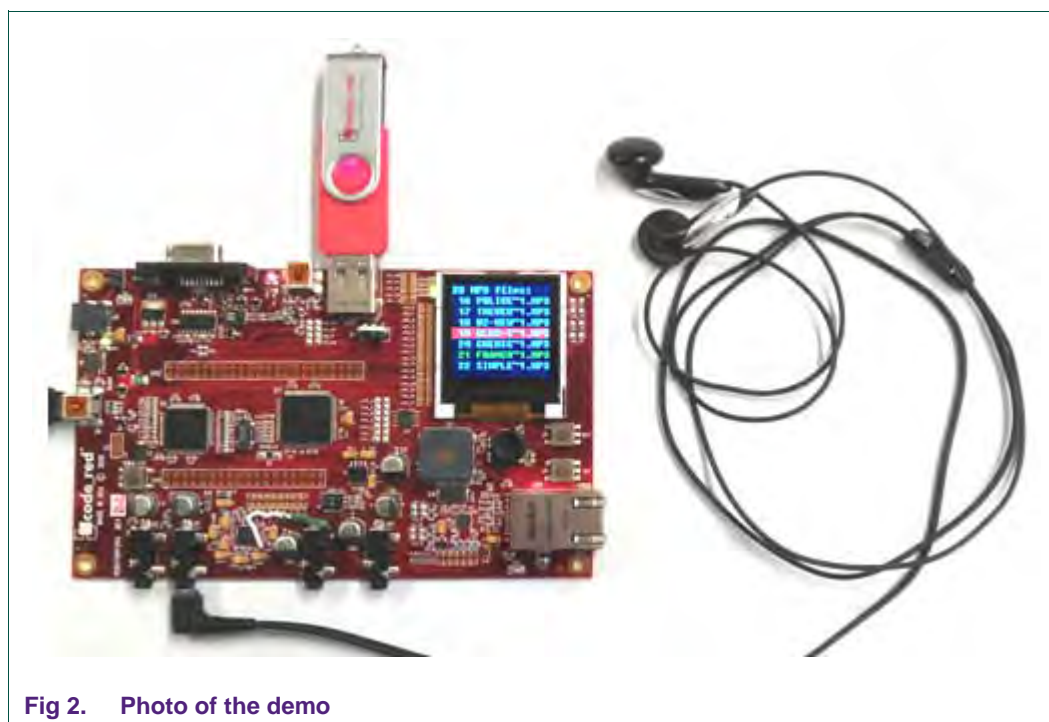
The solution uses the Code Red RDB1768 evaluation board, which features the two main components for this application: The LPC1768 microcontroller and the UDA1380 audio codec.

[Fig 1](#) shows the system block diagram.



Besides the peripherals shown in the block diagram, the solution also uses other peripherals of the LPC1768, such as the UART, the System Tick Timer, the General Purpose DMA (GPDMA) module, the ADC, etc.

[Fig 2](#) shows a photo of the demo.



MP3 files should be stored in the root directory of the SD/MMC card and/or the USB flash disk. The SD/MMC card is inserted into the SD card slot (connected to the LPC1768 via SPI/SSP) and the USB flash disk is inserted into the USB host port of the RDB1768 board.

After decoding the MP3 data to PCM data, the output frame is sent to the UDA1380 via the I2S interface for playback. The audio can be heard by connecting a headphone to the headphone-out connector, and by the connecting the on-board speaker.

The playback volume can be controlled by the potentiometer on the RDB1768 board.

The whole system is powered by USB (+5 V).

2.2 Features

Below lists the features and some known limitations of the solution:

- Supports most common variants of MP3 (MPEG1, MPEG2, and MPEG2.5).
- Supports constant bit rate, variable bit rate, and free bit rate modes.
- Supports mono and all stereo modes (normal stereo, joint stereo, dual-mono).
- Supports playback MP3 files in SD/SDHC/MMC card and USB flash drive with FAT12/16/32 file system and long file name.
- Memory consumption for whole solution: ROM ~60 kB and RAM 50 kB.

3. Hardware description

3.1 Overview

The solution uses the Code Red RDB1768 board. The two main components used by this application are the LPC1768 and the UDA1380. These two components are connected to each other by two busses:

- I2C bus for control.
- I2S bus for data.

A block diagram of the hardware can be found in [Fig 3](#).

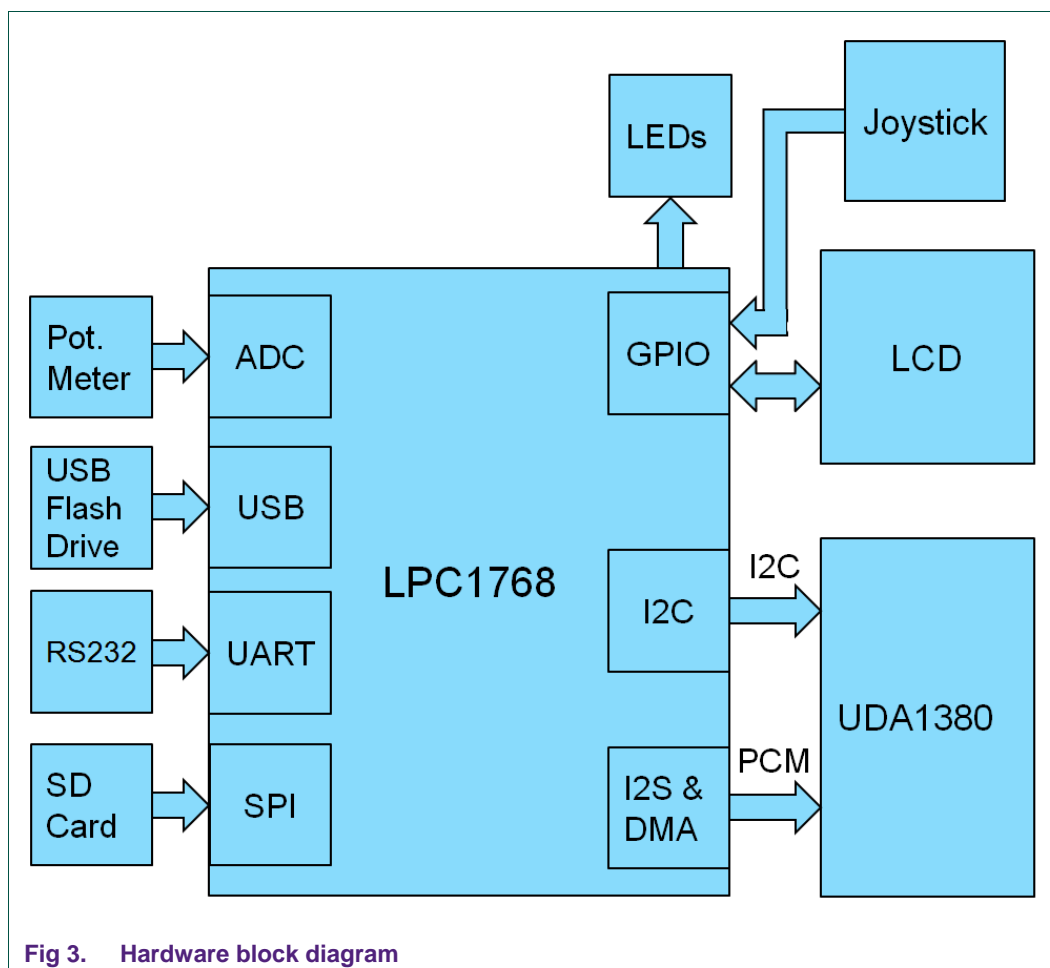


Fig 3. Hardware block diagram

Besides the LPC1768 and the UDA1380, other components of the RDB1768 board are also used:

- LCD and joystick, connected to GPIO, for the GUI.
- RS232, connected to UART, for a simple User Interface.
- LEDs, connected to GPIO, for indicating the system status.
- Potentiometer, connected to ADC, for setting the volume.
- USB, connected to USB peripheral, for MP3 storage.
- SD Card, connected to SPI, for MP3 storage.

3.2 About RDB1768 board

The Code Red RDB1768 evaluation board is designed to be a very flexible evaluation board for the NXP LPC1700 family of microcontrollers.

[Fig 4](#) shows the board and its components.

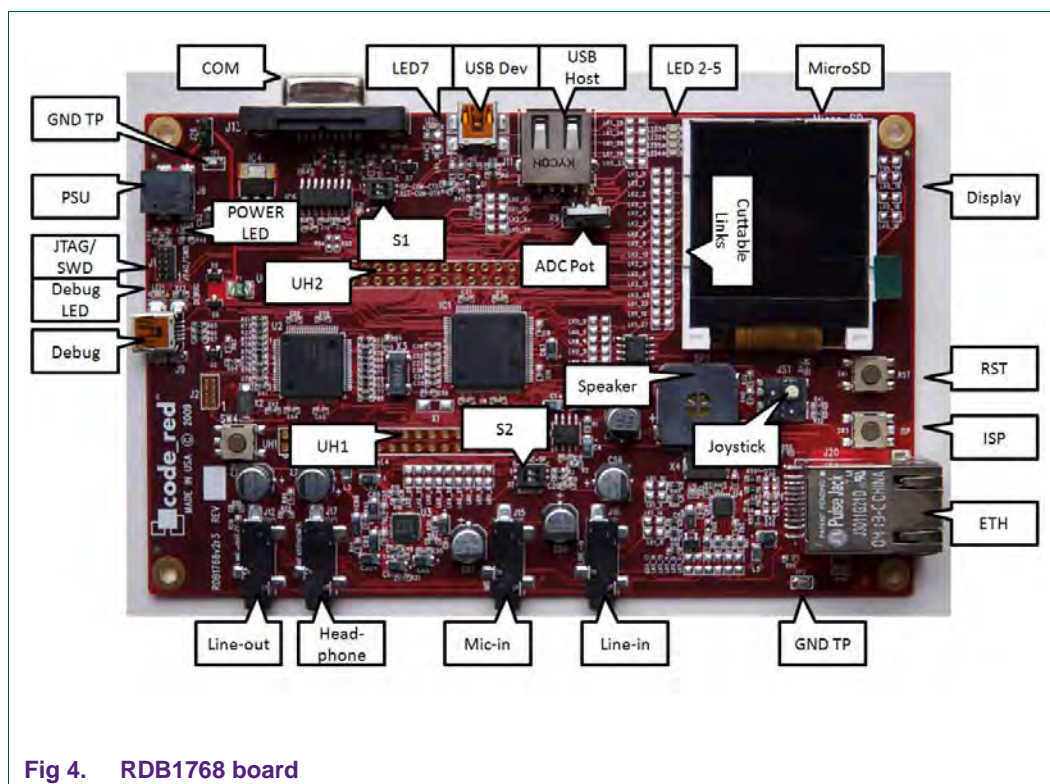


Fig 4. RDB1768 board

For more information, please refer to:

<http://support.code-red-tech.com/CodeRedWiki/RDB1768Support>.

3.3 About the UDA1380

The UDA1380 is a stereo audio coder-decoder which is suitable for home and portable applications like MD, CD and MP3 players. Its main features:

- 24-bit ADC and DAC.
- Selectable control via L3-bus microcontroller interface or I2C-bus interface.
- Supports sample frequencies from 8 kHz to 55 kHz for the ADC part, and 8 kHz to 100 kHz for the DAC part.
- ADC part and DAC part can run at different frequencies, either system clock or Word Select PLL (WSPLL).

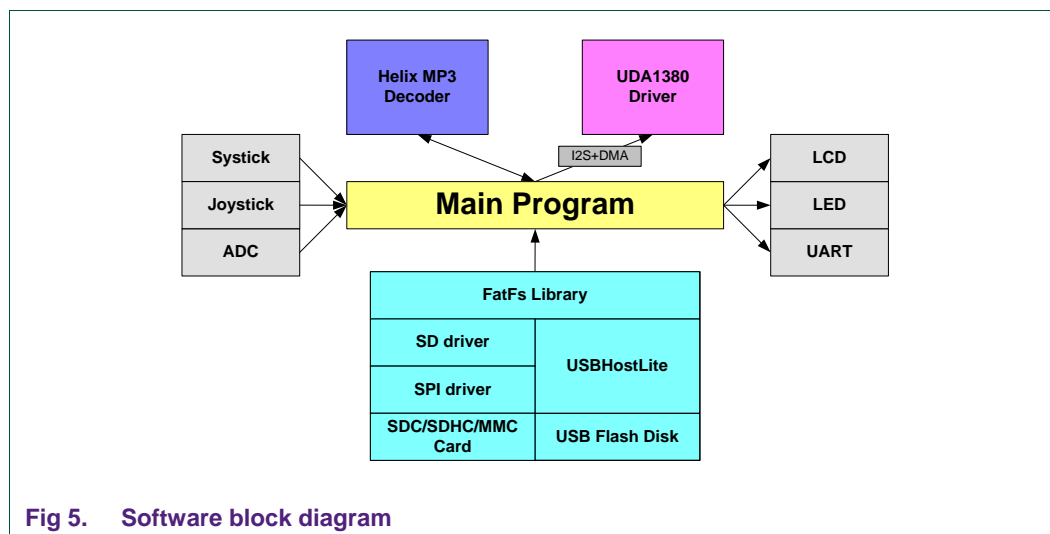
For more information, please refer

to http://www.nxp.com/documents/data_sheet/UDA1380.pdf.

4. Software description

4.1 Software block diagram

[Fig 5](#) shows the software block diagram.



From a software point-of-view, the whole system can be divided into several major parts:

- Drivers to read MP3 files from SD/MMC card and USB flash disk.
- MP3 software decoder to decode the MP3 encoded data.
- Drivers to transfer output frames after decoding to the UDA1380.
- Main program to manage the whole application running and control other auxiliary modules (System tick timer, ADC, LCD, etc).

4.1.1 Main program

The main program is the most important module, which functions as the scheduler for the whole system. It is responsible for the following major tasks:

- Initialize all necessary modules shown in [Fig 5](#).
- Read MP3 files from SD/MMC card and USB flash disk and store the data into the read buffer.
- Decode MP3 data and store the output frame into the audio buffer.
- Transfer the output frame to the UDA1380 via I2S interface by DMA.
- Print debug messages and information.
- Display a menu on the LCD for selecting which file to play.

To ensure a smooth playback of the MP3 files, multiple output buffers are used. Due to the RAM size of the LPC1700, the number of buffers is limited to a maximum of 3. One buffer will be used for decoding MP3 data, while the other one or two buffers are used for playback.

4.1.2 About Helix MP3 decoder

The Helix MP3 decoder provides MPEG-compliant decoding of MP3 content. Both floating-point and fixed-point decoder implementations are available. The fixed-point

decoder is optimized especially for ARM processors but can run on any 32-bit fixed-point processor which can perform a long multiply operation (two 32-bit inputs generating a 64-bit result) and long multiply-accumulate (long multiply with 64-bit accumulator).

It has the following main features:

- Pure 32-bit fixed-point implementation.
- Designed for high performance and low power consumption in handheld and mobile devices.
- Full layer 3 support for:
 - MPEG1 layer 3 - sampling frequencies: 48 kHz, 44.1 kHz, and 32 kHz.
 - MPEG2 layer 3 - sampling frequencies: 24 kHz, 22.05 kHz, and 16 kHz.
 - MPEG2.5 layer 3 - sampling frequencies: 12 kHz, 11.025 kHz, and 8 kHz.
- Supports constant bit rate, variable bit rate, and free bit rate modes.
- Supports mono and all stereo modes (normal stereo, joint stereo, dual-mono).

For more information, please refer to <https://datatype.helixcommunity.org/Mp3dec>.

4.1.3 About USBHostLite

USBHostLite is a stripped-down USB host stack that includes only USB mass storage class support with the bare minimum code to run in an OS-less environment.

USBHostLite provides a simple solution for accessing the files on USB mass storage devices such as USB Pen Drives, USB Hard Disk Drives, etc., connected to the USB Host port.

It has the following main features and limitations:

- Supports control and bulk transfers.
- Classes other than the Mass Storage are not supported.
- USBHostLite integrated file system library supports only FAT16 file system.
- Long filenames are not supported.
- Files located in folders other than root directory cannot be accessed.

For more information, please refer to <http://ics.nxp.com/support/software/usb.host.msc/>.

4.1.4 About FatFs

FatFs is a generic FAT file system module for small embedded systems. The FatFs is written in compliance with ANSI C and completely separated from the disk I/O layer. Therefore it is independent of hardware architecture.

The FatFs has the following features:

- Windows compatible FAT12/16/32 file system.
- Various configuration options:
 - Multiple volumes (physical drives and partitions).
 - Long file name (LFN) support in OEM code or Unicode.
 - RTOS support.
 - Multiple sector size support.

For more information, please refer to http://elm-chan.org/fsw/ff/00index_e.html.

Note that the LFN feature on the FAT file system is a patent of Microsoft Corporation. When LFN is enabled on commercial products, a license from Microsoft may be required depending on the final destination.

4.1.5 UDA1380 driver

The UDA1380 driver provides some simple APIs to configure and control the UDA1380 to allow it to receive the audio data from LPC1700's I2S interface.

It includes the following APIs which the main program uses:

- **UDA1380_Init ()**: Initialize the device which includes a sequence of commands to initialize related registers.
- **UDA1380_MasterVolCtrl ()**: Adjust the volume.

After initialization, the UDA1380 is ready to receive the PCM audio data from the LPC1700 via the I2S interface.

4.1.6 LCD driver

The LCD driver provides some simple APIs to display text and graphics to the LCD of the RDB1768 board. It includes the following APIs which the main program uses:

- **LCDdriver_initialisation ()**: Initialize the LCD which includes a sequence of commands to initialize related registers.
- **LCD_PrintString ()**: Prints a string with a specified color and specified background color to the LCD, at a certain specified location.
- **LCD_ClearScreen ()**: Clears the total LCD screen to a specified background color.

4.1.7 Other modules

Besides the major modules mentioned above, there are also some other modules to support the whole system to work efficiently:

- **System Tick Timer**
 - Functions as the system timer (1000 Hz) to evaluate the performance such as file read speed, MP3 decoding speed, etc.
 - Every 10 ms, it will call function disk_timerproc() which is necessary for SD driver to maintain a timer.
 - Checks every 1 ms if a button of the joystick has been pushed. If so, appropriate action will be executed (e.g. update LCD, play next song).
- **ADC**: Every 500 ms, the main program will read the analog input value and adjust the UDA1380 volume accordingly.
- **LED**: Indicates different system status.
- **UART**: Display some debug messages, such as init process, frame information, statistics and some necessary data. Also songs can be selected by sending a number to the UART, followed by a new line (enter).
- **I2S and DMA**: Transfer an output frame after decoding to the UDA1380 under the control of DMA.

4.2 Software implementation

[Fig 6](#) shows the basic software flowchart.

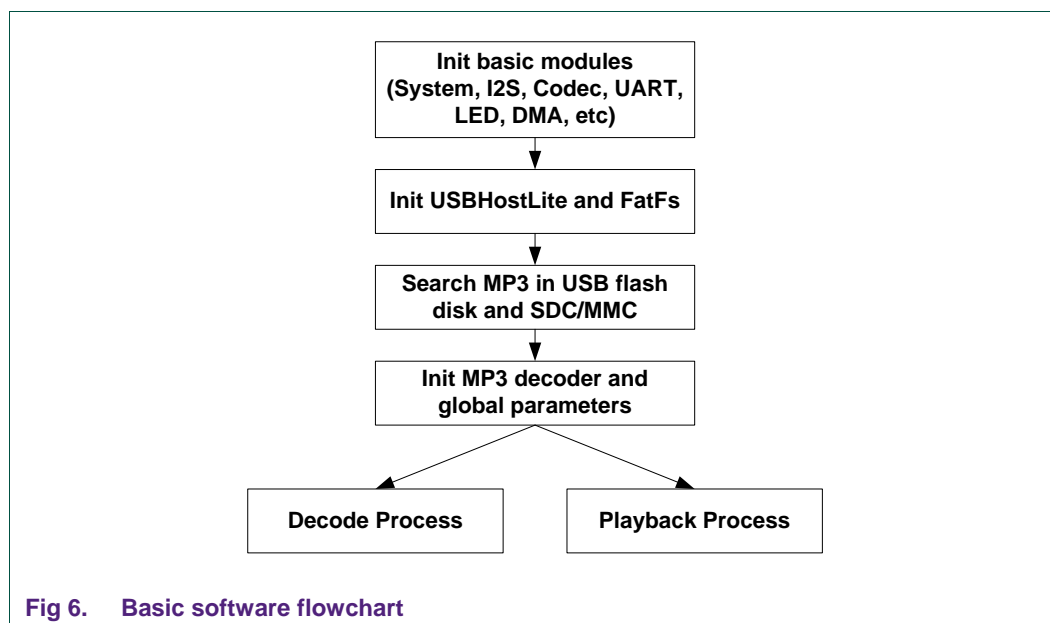


Fig 6. Basic software flowchart

After initialization of some basic modules (system, I2S, UDA1380, UART, etc) as well as USBHostLite and FatFs, the main program will search MP3 files in the root directory of USB flash disk and SD/MMC card, and then start to decode and playback these MP3 files one by one.

There are two major processes (tasks or threads) that work in parallel to decode a file and playback the decoded data at the same time

It is important to synchronize these two processes to make playback as smooth and continuous as possible.

4.2.1 Audio buffer

The Audio Buffer is a data buffer which is 4608 (4.5k) bytes in length and can hold a complete output frame.

After decoding, an output frame will be placed in the audio buffer. When playing, the data in this audio buffer will be sent to the I2S TX channel, which will transfer the data to the UDA1380. Transferring the data from the buffer over I2S is controlled by the DMA module.

[Fig 7](#) shows the audio buffer data structure definition.

```
#define AUDIOBUF_BASEADDR 0x20080000
#define AUDIOBUF_SIZE      (0x1200)    /* 4608 byte (4.5KB) */

/* Allocate 2 or 3 audio buffer to store PCM after decoding */
#define MAX_AUDIOBUF_NUM 3
#if (MAX_AUDIOBUF_NUM < 2) || (MAX_AUDIOBUF_NUM > 3)
    #error Invalid MAX_AUDIOBUF_NUM setting.
#endif

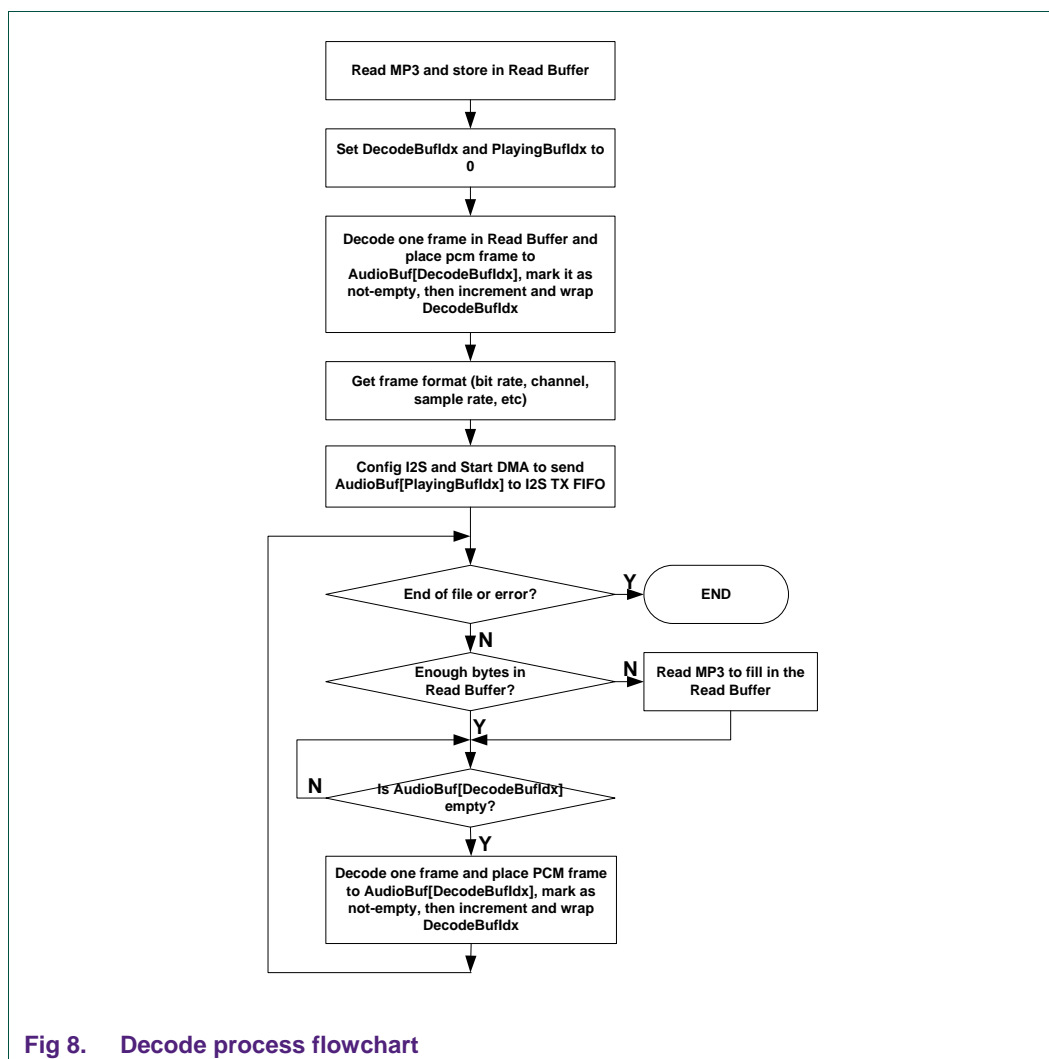
enum {AUDIOBUF0=0, AUDIOBUF1=1, AUDIOBUF2};
typedef struct
{
    uint32_t BaseAddr; /* base address for the audio buf */
    int32_t Size;      /* size of the audio buf */
    uint8_t Empty;     /* buf status, 1-> empty */
} AUDIOBUF;

AUDIOBUF audiobuf[MAX_AUDIOBUF_NUM] = {
    {AUDIOBUF_BASEADDR, -1, 1},
    {AUDIOBUF_BASEADDR+AUDIOBUF_SIZE, -1, 1},
    #if MAX_AUDIOBUF_NUM==3
    {AUDIOBUF_BASEADDR+AUDIOBUF_SIZE*2, -1, 1},
    #endif
};
```

Fig 7. Audio buffer definition

4.2.2 Decode process

[Fig 8](#) shows the decode process flowchart.



The MP3 file is loaded into the Read Buffer, which is the working buffer for the decoder.

When compared to a small buffer, a large buffer will result in less numbers of transfers, but will also consume more time per transfer. The size of this buffer is a trade-off to keep the amount of transfers as low as possible, but also to ensure that reading per transfer does not take too much time in order to gain a continuous playback. In this application, the Read Buffer size can be set separately for USB and SD/MMC playback. By default they are both 8 kB.

After loading the MP3 data to the Read Buffer, the decoder will be executed:

First it tries to find the start of the first frame, after which it decodes this frame and places the data in the first audio buffer. If the number of unprocessed bytes in the Read Buffer is less than $2 \times \text{MAINBUF_SIZE}$ ($= 2 \times 1940 = 3880$ bytes), the Read Buffer will be filled again; all the remaining bytes will be moved to the beginning of the buffer and new data from the MP3 file will fill up the remaining space. If necessary, the buffer is padded with zero's to avoid finding a false sync word after last frame.

After decoding the first frame, the frame info is extracted, such as the bit rate, number of channels, sample rate, bits per sample and number of output samples. The number of

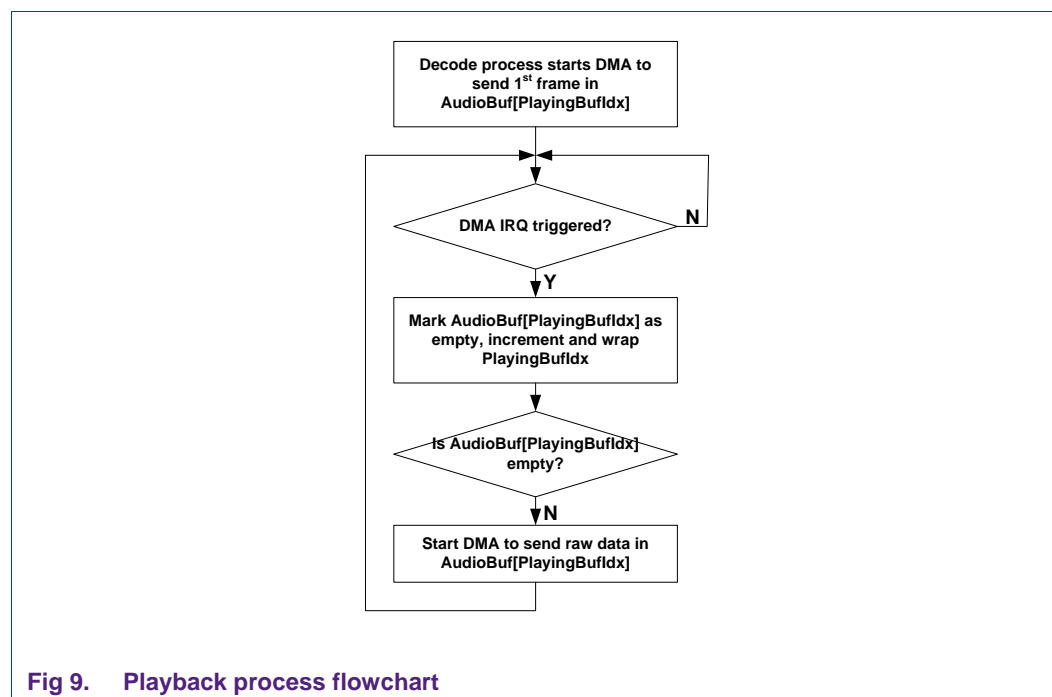
channels, the sample rate and the number of bits per sample are used to configure the LPC1700 I2S TX channel.

The decoder will not decode the next frame if all the audio buffer status' are marked as "Not EMPTY", which means all audio buffers contain valid output frames and should not be overwritten.

The status will be set to "EMPTY" after the data in the audio buffer has been played back.

4.2.3 Playback process

[Fig 9](#) shows the playback process flowchart.



The playback process starts when the main program has decoded the first MP3 frame, extracted the frame format and started the DMA engine to transfer the first decoded frame. Each time, triggered by the DMA interrupt, it will send the data from the next Audio Buffer to the LPC1700 I2S TX channel, assuming that the next audio buffer has already been filled with decoded data before the end of playing the previous output frame.

4.2.4 Design considerations

There are two key considerations:

- How to keep the decode process and the playback process synchronized?
- How to playback MP3 files smoothly and continuously without delay or interrupt?

To address the first issue, a global shared parameter is introduced: Audio Buffer. There exist two (or three) audio buffers in the application; one for decoding and the other for playback. If an audio buffer is held by one process, it cannot be used by the other process. The decoder will start when the current audio buffer's status is "EMPTY". After decoding, its status will be set to "Not EMPTY". The playback will start when the current audio buffer's status is "Not EMPTY". After playback, its status will be set to "EMPTY".

The second issue is affected by several factors: decoding speed, file read speed, Read Buffer size and playing speed. For MP3, the duration of playing a single frame is fixed at ~26 ms. Given the fact that the CPU is running at 100 MHz, decoding a frame will cost around 8 ms with the Helix MP3 library. This means that there is 18 ms left for the CPU to handle other tasks. Saving the data from the MP3 file stored on SD/MMC/USB flash disk to the Read Buffer is another time-consuming task that has to be performed. If this takes too long, gaps in the audio will be audible. Therefore, it is important to choose an appropriate size of the Read Buffer. To offer maximum flexibility, the size of the Read Buffer can be set separately for reading files from SD/MMC and for reading files from the USB flash disk.

5. Demonstration

5.1 Setup

This demo is tested on the RDB1768 evaluation board as mentioned above.

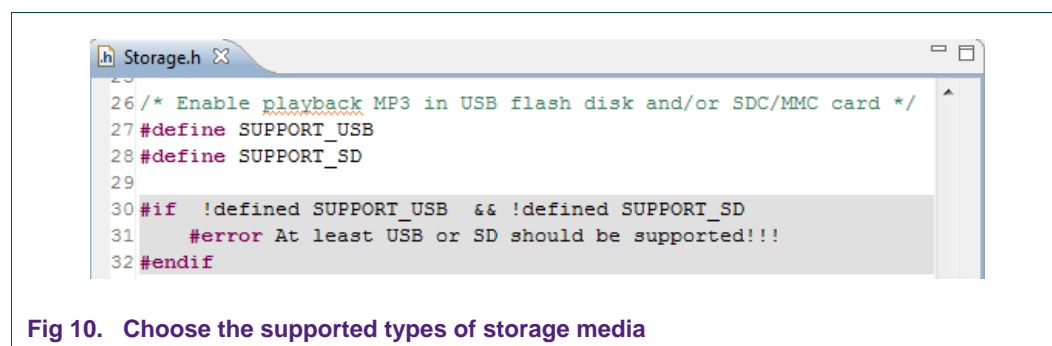
PuTTY (or any other terminal emulator) is used for serial communication between the PC terminal and the LPC1768's UART0. The UART is configured at 115200 baud, 8-bits, no parity and 1 stop bit.

With the default build configuration, a USB flash drive should be connected to the RDB1768 board in order to playback files. Optionally, a micro SD card can be inserted into the micro SD socket to play files from.

If playback from USB is not required, the need for connecting a USB flash drive to the RDB1768 board can be removed by removing support for USB storage. This can be done by removing/commenting the following line in the Storage.h file:

```
#define SUPPORT_USB
```

See [Fig 10](#) for more info on this file.



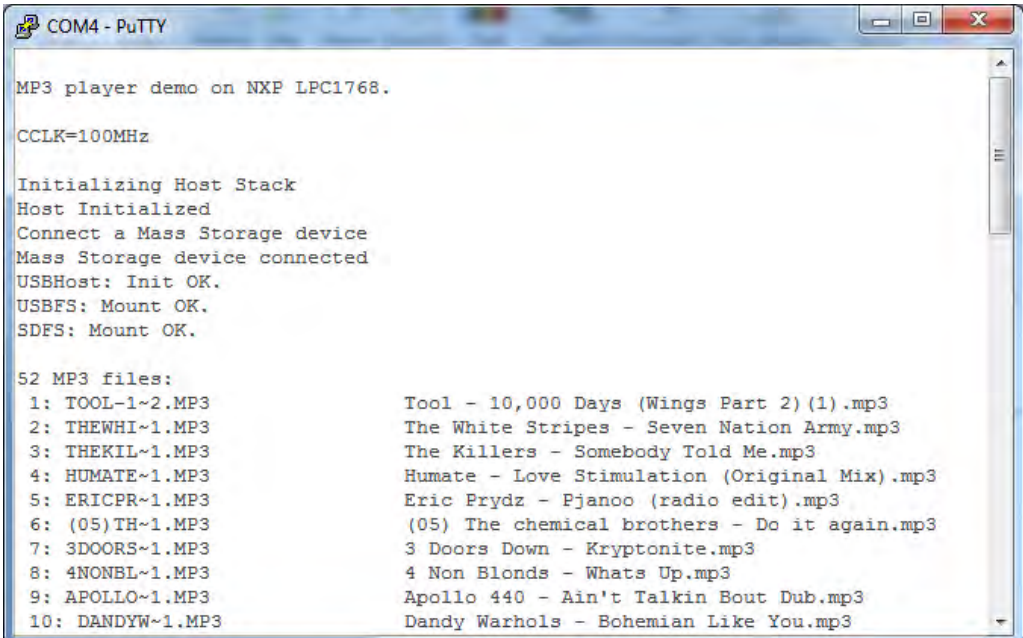
MP3 files should always be placed on the root directory of the storage device.

Power on the board and enjoy the music.

5.2 Test result

The solution is tested with a 2 GB Micro SD card and a 1 GB USB flash disk.

[Fig 11](#) shows the init process and the MP3 files found in the USB flash disk and SD/MMC card.



```

COM4 - PuTTY

MP3 player demo on NXP LPC1768.

CCLK=100MHz


Initializing Host Stack
Host Initialized
Connect a Mass Storage device
Mass Storage device connected
USBHost: Init OK.
USBFS: Mount OK.
SDFS: Mount OK.

52 MP3 files:
1: TOOL-1~1.MP3
2: THEWHI~1.MP3
3: THEKIL~1.MP3
4: HUMATE~1.MP3
5: ERICPR~1.MP3
6: (05)TH~1.MP3
7: 3DOORS~1.MP3
8: 4NONBL~1.MP3
9: APOLLO~1.MP3
10: DANDYW~1.MP3
Tool - 10,000 Days (Wings Part 2) (1).mp3
The White Stripes - Seven Nation Army.mp3
The Killers - Somebody Told Me.mp3
Humate - Love Stimulation (Original Mix).mp3
Eric Prydz - Pjanoo (radio edit).mp3
(05) The chemical brothers - Do it again.mp3
3 Doors Down - Kryptonite.mp3
4 Non Blonds - Whats Up.mp3
Apollo 440 - Ain't Talkin Bout Dub.mp3
Dandy Warhols - Bohemian Like You.mp3


```

Fig 11. Init process and list of MP3 files found on the storage devices

Fig 12 shows the frame information and some statistics which are displayed during and after playback.



a. Playback of a file from USB



b. Playback of a file from SD card

Fig 12. Frame information and statistics for a MP3 on a USB flash disk

6. Conclusion

This application note describes the solution which makes it possible to playback MP3 files on the Cortex-M3 based NXP LPC1700 series with the use of the Helix MP3 software decoder. The solution takes full advantage of LPC1700's high performance and rich peripherals.

Although there are some limitations, it is still a good reference solution to validate LPC1700's high performance and high level of integration.

With the introduction of the Cortex-M3 LPC1800 (with external memory bus) and Cortex-M4 LPC4300 (with DSP functions), NXP will expand its domain in digital audio processing applications.

7. References

- [1] NXP LPC17xx User Manual UM10360 (Rev. 2), NXP Semiconductors, 18 August 2010
- [2] The Helix MP3 Decoder, RealNetworks, Inc, <https://datatype.helixcommunity.org/Mp3dec>
- [3] FatFs Generic FAT File System, Chan, http://elm-chan.org/fsw/ff/00index_e.html
- [4] USB host mass storage class (MSC) example, NXP Semiconductors, <http://ics.nxp.com/support/software/usb.host.msc/>
- [5] UDA1380 DATA SHEET, NXP Semiconductors, http://www.nxp.com/documents/data_sheet/UDA1380.pdf.

8. Legal information

8.1 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.

8.2 Disclaimers

Limited warranty and liability — 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. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

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 life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at 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.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP

Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Evaluation products — This product is provided on an "as is" and "with all faults" basis for evaluation purposes only. NXP Semiconductors, its affiliates and their suppliers expressly disclaim all warranties, whether express, implied or statutory, including but not limited to the implied warranties of non-infringement, merchantability and fitness for a particular purpose. The entire risk as to the quality, or arising out of the use or performance, of this product remains with customer.

In no event shall NXP Semiconductors, its affiliates or their suppliers be liable to customer for any special, indirect, consequential, punitive or incidental damages (including without limitation damages for loss of business, business interruption, loss of use, loss of data or information, and the like) arising out of the use of or inability to use the product, whether or not based on tort (including negligence), strict liability, breach of contract, breach of warranty or any other theory, even if advised of the possibility of such damages.

Notwithstanding any damages that customer might incur for any reason whatsoever (including without limitation, all damages referenced above and all direct or general damages), the entire liability of NXP Semiconductors, its affiliates and their suppliers and customer's exclusive remedy for all of the foregoing shall be limited to actual damages incurred by customer based on reasonable reliance up to the greater of the amount actually paid by customer for the product or five dollars (US\$5.00). The foregoing limitations, exclusions and disclaimers shall apply to the maximum extent permitted by applicable law, even if any remedy fails of its essential purpose.

8.3 Trademarks

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

9. Contents

1.	Introduction	3
2.	System overview	3
2.1	Overview	3
2.2	Features	5
3.	Hardware description.....	5
3.1	Overview	5
3.2	About RDB1768 board	6
3.3	About the UDA1380	7
4.	Software description	8
4.1	Software block diagram.....	8
4.1.1	Main program	8
4.1.2	About Helix MP3 decoder.....	8
4.1.3	About USBHostLite	9
4.1.4	About FatFs.....	9
4.1.5	UDA1380 driver.....	10
4.1.6	LCD driver	10
4.1.7	Other modules.....	10
4.2	Software implementation.....	11
4.2.1	Audio buffer.....	11
4.2.2	Decode process	12
4.2.3	Playback process	14
4.2.4	Design considerations.....	14
5.	Demonstration	15
5.1	Setup.....	15
5.2	Test result	15
6.	Conclusion.....	17
7.	References	18
8.	Legal information	19
8.1	Definitions	19
8.2	Disclaimers.....	19
8.3	Trademarks	19
9.	Contents.....	20

Please be aware that important notices concerning this document and the product(s) described herein, have been included in the section 'Legal information'.
