CONTENTS

INTRODUCTION

This LPC4300 demo implements an internet radio, which is able to receive and play an MP3 music stream from a SHOUTcast radio station. Throughout the code and the documentation you also find the name LAN Radio.

It is based on the following hardware and software components:

- LPC4357 on a KEIL MCB4300 evaluation board
- NXP's LPCOpen platform v1.00
- FreeRTOS with LwIP TCP/IP stack
- emWin graphics library (must be downloaded separately, see below)
- Helix MP3 decoder (must be downloaded separately, see below)
- A SHOUTcast radio implementation (initially done for the LPC1768, found at http://code.google.com/p/stream-radio-v3/)
- KEIL µVision 4.70 + ULINK Pro

The minimum you need to do to get music is described in chapter Programming the demo software to MCB4357. The download package from lpcware.com includes binaries for both Cortex-M4 and M0, containing the whole project except the MP3 decoder part and the emWin libraries. These components can be loaded separately.

Without MP3 decoder the player can take WAV files from an SD card, the Internet Radio can connect to a SHOUTcast radio station and receive an MP3 stream, but will not play it.

If you want to get more functionality than just a WAV player, and this is assumed, just continue to read.

The chapters are pretty short, sometimes one-pagers. Please read them carefully before you raise any questions to the LPCWare forum. However, if there are questions please issue them in the LPC43xx forum.

INSTALLATION

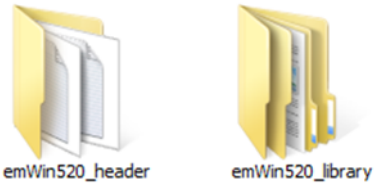
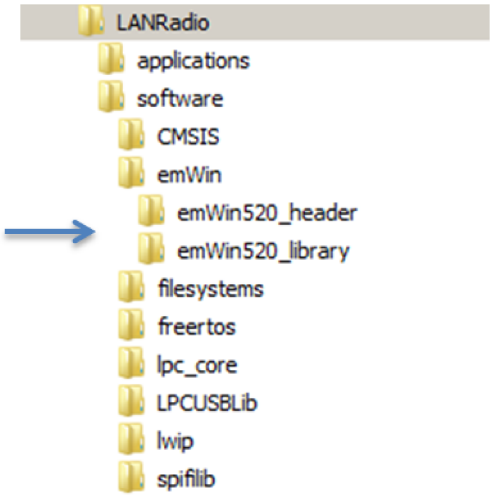
The demo package can be extracted to any location on your hard disk, however, it should not be placed too deep in the directory tree (< 40 characters is recommended). Otherwise the path length could exceed the maximum path length for some of the µVision build tools.

ADDING EMWIN LIBRARIES

The emWin graphics library components are not part of the demo software release, but can be directly downloaded using the link below:

http://www.lpcware.com/system/files/NXP_emWin520_libraries.exe__0.zip

- Extract the ZIP file and execute the installer.
- Install the emWin package to wherever you like.
- Overtake the following components into the Internet Radio demo project as shown below.

emWin installation directory	to	LAN Radio demo package
	→	

ADDING MP3 DECODER

The MP3 decoder component is not part of the demo software release due to a source code license conflict between the NXP software and the available open source MP3 decoders. Without MP3 decoder the project can be used as WAV Player, the Internet Radio will indeed connect to a station and load MP3 streaming data but will of course not play it.

One well known open source MP3 decoder is the Helix decoder, it can be downloaded from the Helix Community website using the link below:

https://helixcommunity.org/viewvc.cgi/datatype/mp3/codec/fixpt/

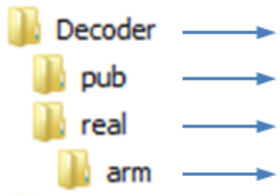
To get access to the MP3 decoder CVS project you would need to create a user account for the Helix Community. As an alternative to that you could download all required files as single files from the CVS view. This can be scripted using a command-line download tool.

The folder *Decoder* contains the batch file *Helix_dld.bat*, which downloads all required files from the CVS view. It uses the command-line tool *aria2c.exe*, which can be downloaded here:

http://aria2.sourceforge.net/

Copy the tool *aria2c.exe* into the folder *Decoder* and execute the batch file. It will take a few minutes to download the required MP3 decoder source files.

The demo application is well prepared to work together with the API from the Helix decoder, nevertheless you need to apply the following changes to the original MP3 decoder project:

File	Changes to apply to the Helix MP3 decoder source
	mp3dec.c mp3dec.h assembly.h , buffers.c asmmisc.s

mp3dec.c	Comment out line 44 and 46:
	`//#include "hlxclib/string.h"`
	`//#include "hxthreadyield.h"`
	and add the following line:
	`#include <string.h>`
	
	These changes result in a conflict further on in the code:
	1) at around line 288:
	`ULONG32 ulTime;`
	`StartYield(&ulTime);`
	2) at around line 400
	`YieldIfRequired(&ulTime);`
	
	Simply comment out these code lines.

File	Changes to apply to the Helix MP3 decoder source

mp3dec.c	In the internet radio application there is a small problem with the buffer management. When the MP3 stream buffer wraps around, one or more bytes get lost and result in an error in the Huffcode check and the MP3 decoder stops working. Ignoring this error keeps the decoder running and causes every 6Mbytes a short "blip".
	
	Insert the following code section instead of the highlighted one at around line 391:
	
	`#ifdef IGNORE_HUFFCODES_ERROR`
	`  if (offset < 0) {`
	`  // Suppress this error, it will cause just a short "blip"`
	`  MP3ClearBadFrame(mp3DecInfo, outbuf);`
	`  }`
	`#else`
	`  if (offset < 0) {`
	`  MP3ClearBadFrame(mp3DecInfo, outbuf);`
	`  return ERR_MP3_INVALID_HUFFCODES;`
	`  }`
	`#endif`

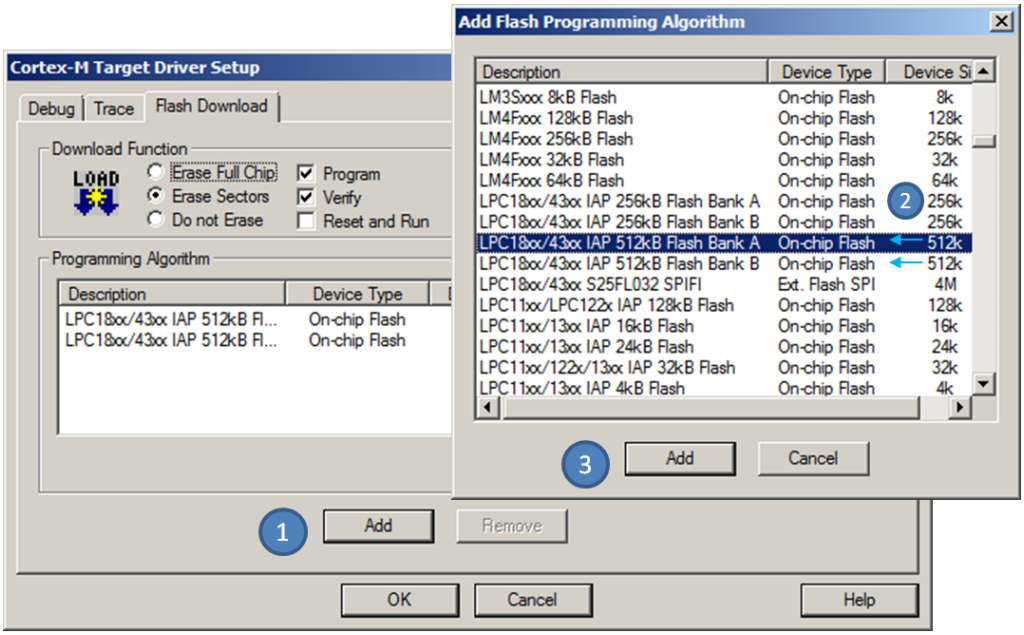
mp3dec.h	Add following code at line 46 (before the platform checks):
	`#define ARM_CM3_KEIL`
	
	In the platform check add at line 54 the following code:
	`#elif defined(ARM_CM3_KEIL)`

buffers.c	Comment out line 48:
	`//#include "hlxclib/stdlib.h"  /* for malloc, free */`
	
	Add the following lines en block instead:
	`#include <stdlib.h>`
	`//#define USE_LPC  /* local heap control in lpc_heap.c */`
	`#undef USE_LPC`
	
	`#ifdef USE_LPC`
	`  #include "lpc_heap.h"`
	`  #include "board.h"`
	`#endif`
	
	`#ifdef USE_LPC`
	`  #define ALLOC_FUNC  lpc_new`
	`  #define FREE_FUNC  lpc_free`
	`#else`
	`  #define ALLOC_FUNC  malloc`
	`  #define FREE_FUNC  free`
	`#endif`

File	Changes to apply to the Helix MP3 decoder source

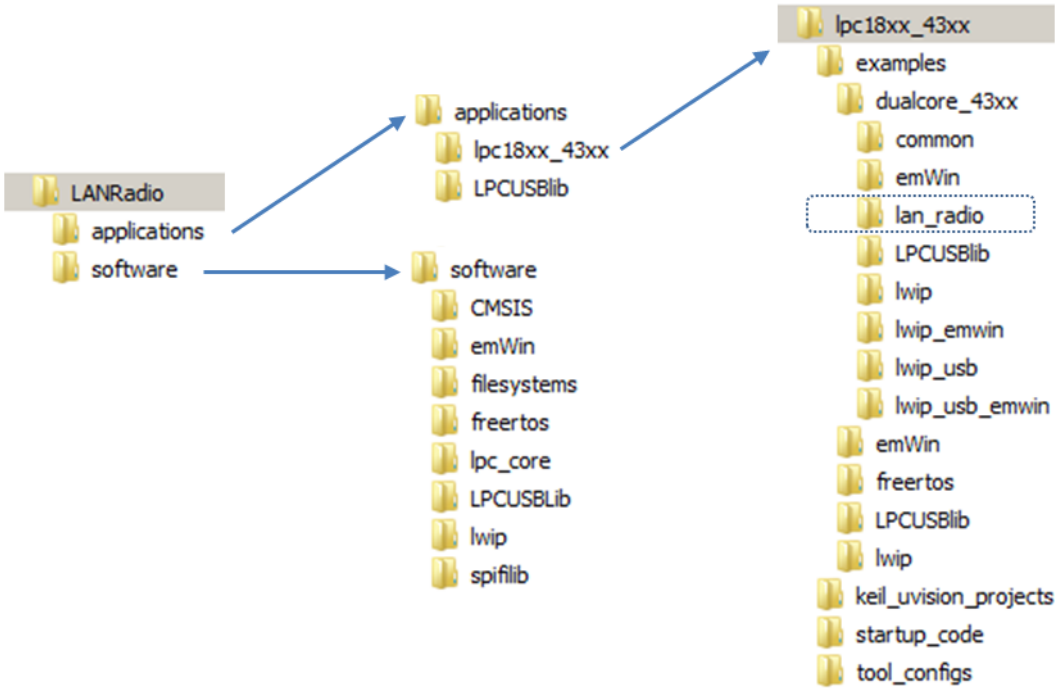
assembly.h	Add the following code at line 319 before the last `#else` directive in this file:
	
	`#elif defined ARM_CM3_KEIL`
	`extern int MULSHIFT32 (int x, int y);`
	`extern int FASTABS(int x) ;`
	`extern int CLZ(int x);`

asmmisc.s	Replace the existing xmp3_MULSHIFT32 implementation with the following functions:
	
	`;  int MULSHIFT32(int x, int y)`
	`EXPORT MULSHIFT32`
	`MULSHIFT32`
	`smull  r2, r0, r1, r0`
	`bx lr`
	
	`;  int FASTABS(int x)`
	`EXPORT  FASTABS`
	`FASTABS`
	`eor  r1, r0, r0, asr #31`
	`sub  r0, r1, r0, asr #31`
	`bx lr`
	
	`; int CLZ(int x)`
	`EXPORT CLZ`
	`CLZ`
	`clz r0, r0`
	`bx lr`

#	Changes to apply to the µVision project settings
1	C / C++ settings: • Change compiler define **NO**MP3_DECODER → MP3_DECODER • Change compiler define **NO**IGNORE_HUFFCODES_ERROR → IGNORE_HUFFCODES_ERROR • Add the following directory to the c/C++ Include Path setting: `.\Decoder\pub;`
2	Utility settings: • It could happen that the flash programming drivers are not loaded from your local µVision installation when you open the project. In this case locate them in your µVision installation. • Flash programming drivers need to be in place for both targets, but you can add both driver for both targets if you like. - iflash_keil_mcb_4357 (flash bank #A driver) - iflash_keil_mcb_4357_m0 (flash bank #B driver)  • If you use the qSPI memory as well (use it in combination with compiler define LOCATE_IN_SPIFI) then the driver "LPC18xx/43xx S25FL032 SPIFI" must be added as well.
3	After downloading all required files from the Helix MP3 decoder project with the aria2c.exe tool, these files need to be added to the µVision project. Simply double click on the folder Decoder and add all C-files and S-files in the folders Decoder, real and arm to the project. **Don't add the file polyphase.c** ! There is an assembler version instead. 14 C-files: `mp3dec, mp3tabs, bitstream, buffers, dct32, dequant, dqchan, huffman, hufftabs, imdct, scalfact, stproc, subband, trigtabs_fixpt` 2 S-files: `asmmisc, asmpoly`

SOFTWARE ARCHITECTURE

The overall project is based on the LPCOpen v1.00 release from January 2013. The rather huge package has been reduced to a reasonable size, keeping all required and optional components , and of course adding the Internet Radio and MP3 Player application. As of November 2013 there is already LPCOpen v2.02 available which has a similar structure, but is not fully compatible to the v1.00 used in this demo. It's under investigation if it makes sense to port the demo to the latest release of LPCOpen.



The LPCOpen v1.0 structure is based on 2 main folders, *application* and *software*. This is also the case in the current version 2.xx	The *applications* folder contains example projects based on components available in the *software* folder	Inside the section for the LPC1800/4300 you find different examples, this is where the lan_radio project has been added.

Inside the *lan_radio* folder the demo is sorted into several subfolders with intuitive names. Important there is the folder *Decoder*, which is empty besides the file Helix_dld.bat and aria2c.txt.

The source code contains a lot of comments and short tutorials which are worthwhile to read if you want to understand the application setup.

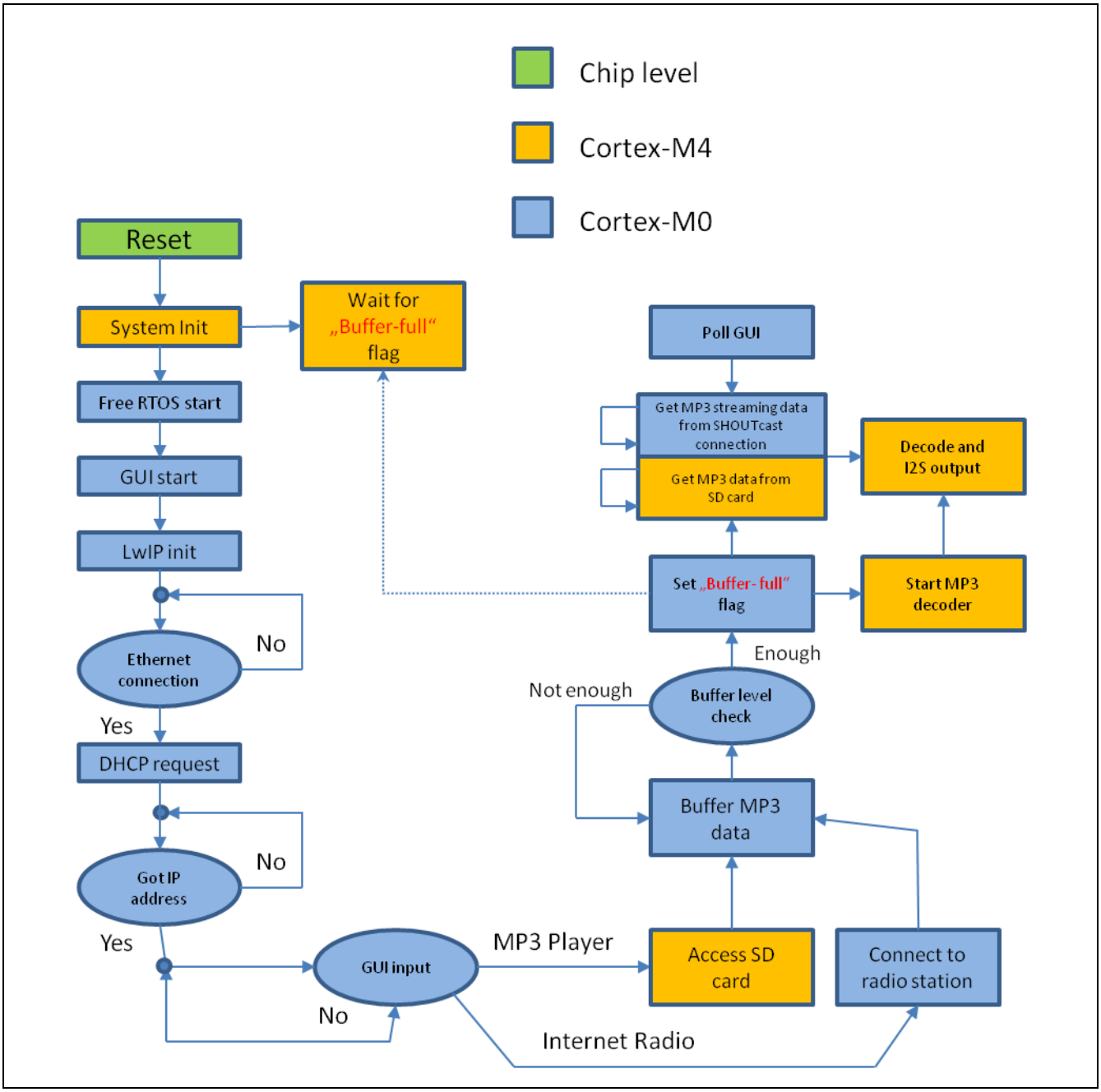
INTERNET RADIO & MP3 PLAYER APPLICATION SETUP

The LPC4300 is a dual core architecture with a Cortex-M4 and a Cortex-M0. In this demo application both cores are used to handle various tasks. The Internet Radio / MP3 Player application as is does not necessarily require the two cores, the Cortex-M4 would have enough horsepower to deal with all tasks. But in case more features are added the additional horsepower of the second core is simply needed. Running a feature like DOLBY Surround 5.1 on one core in addition to a GUI, MP3 decoder etc requires more processing performance than a typical single core 180MHz MCU can provide.

On the other hand even in the current setup it is an advantage to have e.g. the TCP/IP and GUI task running on a different core as the MP3 decoder task. It is possible to develop and change setups on both sides, without directly affecting the runtime behavior on the other side.

Feature/Resource	Cortex-M0	Cortex-M4
CPU speed	180 MHz	180 MHz
Memory	Code execution: flash bank #B External SDRAM	Code execution: flash bank #A Internal SRAM
Software components	- Free RTOS - LwIP Ethernet stack - emWin GUI - SHOUTcast application - UART debug output	- MP3 decoder - WAV/MP3 player application - I2S interface handling - UART debug output
Used peripherals	- Ethernet - LCD - UART - GPIO	- I2S - UART
Possible add-on features	USB device for connection to a PC host More sophisticated GUI	Implementation of AAC decoder USB host for playing MP3 from a USB memory stick Audio enhancements (equalizer, 5.1 surround etc)

See below a simplified flow diagram for the Internet Radio / MP3 Player application:



SOFTWARE PROJECT

The project has been developed with Keil µVision version 4.70. You need the licensed version of µVision to be able to compile the software. It is intended to port this over to the free version of LPCXpresso, however the first public version of this demo is just available for µVision.

The (potential) differences in the setup between µVision and LPCXpresso are shown below:

Keil µVision 4.70	LPCXpresso 6
1 project including all files for both cores, selection between Cortex-M4 and M0 project is done with the target selection	2 projects for each of the cores in the same Eclipse workspace
Start environment based on startup.s + sysinit.c	Start environment based on cr_startup.c and system.c
*.s files of the Helix decoder based on ARM-MDK assembler	*.s files of the Helix decoder based on GNU assembler
Linker setup based on memory layout settings done in the Target tab of the project settings	Linker setup based on settings done with memory config editor, which will in turn generate the linker script

MEMORY CONFIGURATION

In order to achieve the best performance the Cortex-M0 and the M4 should execute their code from different memory areas which do not (directly) affect each other.

The best solution is clearly the distribution of the code to the two flash banks.

Cortex-M0	Cortex-M4	CPU Speed	Note
Flash Bank #B	Flash Bank #A	180MHz	No runtime problems, MP3 files with 256kbps are played correctly
Flash Bank #B	SPIFI @ 96MHz	192MHz	Decoder runtime problems with files >128kbps

You could think of various different RAM setups for this dual core application, the current setup is shown below:

Memory	Cortex-M4	Cortex-M0
SRAM	Free for linker Address: 0x10000000 Size: 0x8000	-
SRAM	Free for linker (parameters, stack, heap) Address: 0x10080000 Size: 0x9F00	-
SRAM	Free for linker (parameters, stack, heap) Address: 0x10089F00 Size: 0x100	-
SRAM	Free for linker (parameters, stack, heap) Address: 0x20008000 Size: 0x8000	-
SDRAM	-	LCD display buffer Address: 0x28050000 Size: 1024*1024*2 (2MByte)
SDRAM	-	LAN data buffer for Ethernet block Address: 0x28800000 Size: 6144
SDRAM	-	MP3 streaming data buffer Address: 0x28802000 Size: 0x600000 (6MByte)
SDRAM	-	Free for linker (parameters, stack, heap) Address: 0x28FD0000 Size: 0x30000

USER INTERFACE DESCRIPTION

After a reset of the board the following GUI is presented:

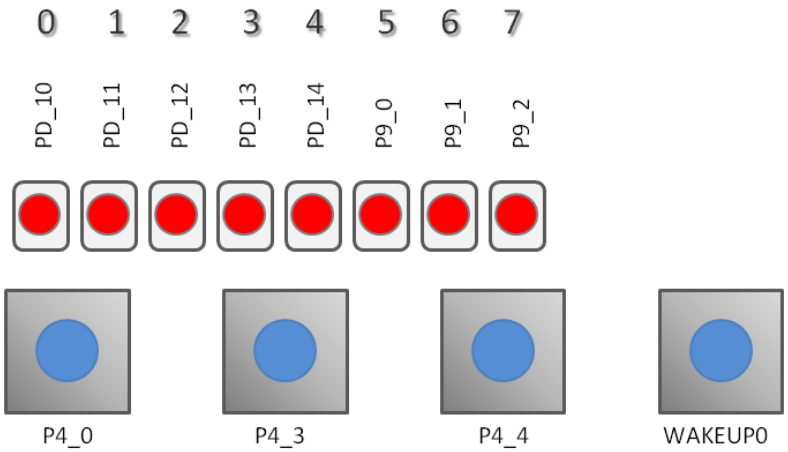
- Here you do the basic selection between the MP3 Player and the Internet Radio.
- In case you did not add an MP3 decoder to the Cortex-M4 project, the upper button will show “WAV Player”.

- If you don’t have a valid TCP/IP connection the station buttons will not be shown. They appear as soon as the system detects a network connection
- 4 radio stations: playing starts by touching the respective station button, text color of the active station changes to blue
- Audio settings:
  - Bass boost and Treble boost: a touch enables it (text color changes to green), another touch disables it.
  - volume plus and minus
- STOP button: stops the player for selection of another station
- The recently touched button gets the focus and changes to red color

- The WAV/MP3 Player looks for WAV or MP3 files on the SD card
- After touching the green Play button the player starts with the first detected file and displays the ID3 information (or file name for WAV files)
- With Forward or Backward buttons you can jump through the list of detected files
- Another PLAY button touch replays the current file
- Blue STOP button: stops the player for selection of another station

LED Status Indicators on the Keil MCB4300 board

LEDs and buttons on MCB4300



LED PD_10: TCP/IP connection hang-up

LED PD_11: Indicates that MP3 data buffers are filled with data

LED PD_12: DHCP was successful (board got IP address

LED PD_13: Physical Ethernet connection exists

LED PD_14: System started, waiting for decoder data

Example for controlling a LED from C code:

```
Board_LED_Set(0, false);    // switch LED PD_10 off
Board_LED_Set(0, true);     // switch LED PD_10 on
```

PROGRAMMING THE DEMO SOFTWARE TO MCB4357

With µVision4: You need µVision 4.70 or later and ULINK Pro or ULINK2 or ULINK ME. UV5 should work accordingly.

The project as is uses ULINK2/ME, if you have a ULINK Pro you need to change it in the Debug and Utility tab of the project settings.

1. Program from **source release**: `.\IRD_Src_v1.0\applications\lpc18xx_43xx\examples\dualcore_43xx\lan_radio`
   a. Open the µVision4 project *lan_radio.uvproj*
   b. Select Cortex-M4 target *iflash_keil_mcb_4357,* compile and flash it
   c. Select Cortex-M0 target *iflash_keil_mcb_4357_m0,* compile and flash it
   d. Select target *KEIL_Demo_mcb_4357* and flash it
   (in case the software has been compiled with C/C++ define KEILDEMO_SELECTION in the Cortex-M0 project and ASM define DEMO_COMBO in the Cortex-M4 project)

2. From **binary release**: `.\IRD_Bin_v1.0\UV4_ULINK2-ME\lan_radio`
   a. Open the µVision4 project *lan_radio.uvproj*
   b. Select Cortex-M4 target *iflash_keil_mcb_4357* and flash it
   c. Select Cortex-M0 target *iflash_keil_mcb_4357_m0* and flash it
   d. Select target *KEIL_Demo_mcb_4357* and flash it

3. From **binary or source release** project using batch mode:
   Adjust the batch file *Flash_LAN_Radio_Demo.bat* in one of the folders *UV4_ULINK-Pro* or *UV4_ULINK2-ME* according to your path to the µVision executable, e.g.

```
C:\Keil\UV4\UV4.exe -f lan_radio.uvproj -t"iflash_keil_mcb_4357"
C:\Keil\UV4\UV4.exe -f lan_radio.uvproj -t"iflash_keil_mcb_4357_m0”
C:\Keil\UV4\UV4.exe -f lan_radio.uvproj -t"KEIL_Demo_mcb_4357"
```

With FlashMagic: The link to the Flash Magic utility can be found at the end of the document.

File	Changes
FlashMagic_LAN_Radio_Demo.bat	Modify the path to *FM.exe* in this batch file
ldr_m4.txt *ldr_m0.txt*	Adjust the following options according to your PC serial hardware: COM(1, 9600) HIGHSPEED(0, 115200)

Modify jumper settings on MCB4300:

- Set jumpers J16/J13 to UART0. Location is near to the JTAG connector.
- Set jumpers URST (J18) and UISP (J17). Location is near to the Reset button.
- Set all bootmode jumpers to the L side. Location is near to the joystick.

IDEAS, ADDITIONAL FEATURES, DESIGN CHALLENGES

- Implement a smarter way of adding and selecting radio stations
- Replace memory copy loops by DMA setups
- Replace the simple memory flag mechanism for M0 ← → M4 communication with an interrupt based mechanism
- Replace I2C implementation (polling and therefore blocking) with an interrupt driven solution
- Porting of the GUI to a bigger display, adding more elements, keypad for station address input, animations, MP3 player GUI, etc
- USB host for connection of a memory stick (playing MP3 from the stick) or for connection of a USB loudspeaker
- USB device for connection to a PC (e.g. for station list download, MP3 file download to SD Card etc
- Audio enhancements like equalizers or 5.1 surround (requires additional I2S interfaces emulated by SGPIO and also additional analog front ends)
- Add AAC+ and MP4 CODECs from Helix package
- Move const data for GUI elements (BMP data for background, buttons etc) into qSPI flash to get more application code space in internal flash bank #B

KNOWN LIMITATIONS AND ISSUES IN V1.0

1. The touch input has some limitation with regards to touch recognition. The current touch detection implementation does not evaluate the pressure, only the x/y components are analyzed. This means that a very smooth and short pressure can result in wrong x/y detection. However, a solid tap with a touch pen or a finger should work reliably enough.

2. The GUI or/and the application might hang up if various buttons are touched at the same time or in a short time frame.

3. A very short “blip” is audible every time the decoder read pointer on the MP3 buffer wraps around (every 6 Mbytes). It’s most likely a loss of samples due to incorrect counting of pointer positions (to be fixed in next revision).

4. The SHOUTcast application currently does not ask for the metadata (e.g. the current music title). It is foreseen in the software, but if the metadata is requested, then the received data is not sorted correctly into the buffers. The title for example is recognized, but the MP3 data is not correctly extracted from the stream to the buffer (to be corrected in future version of the demo).

5. On internet connections with not enough bandwidth for the currently used 128kbps stations the application hangs up. A reset gets the system out of the hang-up, but only to run into the next one. The solution is simple: use a connection with sufficient bandwidth.

INTERESTING WEB LINKS

http://www.lpcware.com/content/project/LPC4300-Internet-Radio-Demo

http://www.lpcware.com/system/files/NXP_emWin520_libraries.exe__0.zip

https://helixcommunity.org/viewvc.cgi/datatype/mp3/codec/fixpt/

http://aria2.sourceforge.net/

http://www.flashmagictool.com/

http://www.shoutcast.com/

http://thecodeartist.blogspot.com/2013/02/shoutcast-internet-radio-protocol.html

https://datatype.helixcommunity.org/Mp3dec

http://oreilly.com/catalog/mp3/chapter/ch02.html#80629

HISTORY

Date	Version	Comment
25. November 2013	1.0	First release of document and software project on www.lpcware.com