Trilby Sample Firmware

Version 1.02
20 April 2016

Licensing Information

Please Note

This document is not a tutorial. It assumes that the reader is familiar with the principles and terminology of computer communications, digital signal processing and/or programming in general.
In order to modify the firmware for your own purposes, you will also need to install and be familiar with the Lattice Diamond software.

General Description

The sample firmware consists of a Lattice Diamond project with a top-level schematic and several VHDL modules. These implement an example radio receiver and also provide an Application Programming Interface (API) allowing the user to read and write various internal control registers within the firmware. By default the firmware flashes the Red LED once per second to indicate that it is correctly running.
There is an I/O control module, which provides registers allowing the user to configure the majority of the external connector pins as either inputs or outputs and enable or override specific functionality associated with the pins (such as SPI and I2C).
There is also a tuner control module, which initialises the tuner chip and can then retune to a specified input (RF) frequency. The tuner is set up to give an intermediate frequency (IF) of 6.1 MHz,

The tuner input can be switched between the VHF/UHF antenna socket and the HF antenna socket. When HF is selected, the signal is up-converted by mixing with 96 MHz so that for example to receive a station at 25 MHz the tuner is set to 121 MHz.

Radio Receiver

A sample AM/FM mono radio receiver is included in the project. This can be used to listen to AM, narrow band FM (e.g. marine channels) or wideband (broadcast) FM stations, by connecting a suitable antenna and an audio output device.
The audio output device may be connected to the PWM audio output pins. Audio amplification is not provided on the Trilby board, so an external audio amplifier or suitable powered speakers will be required.
Alternatively digital audio is available to send to the Raspberry Pi using the SPI interface. The audio device is then connected to the 3.5 mm audio jack socket on the Raspberry Pi and suitable software must be running on the Raspberry Pi to handle the audio data. The ttune tuning utility (see below) has an option for producing this audio output using the Raspberry Pi's playback channel. Please ensure that the appropriate audio device (PWM or HDMI) is selected on the Pi - The raspi-config utility can be used for this.

Technical Overview of the Radio Receiver's Operation

The RF output from tuner is sampled at 24MHz using the Analog to Digital Converter chip (ADC). The digital data is read in by the firmware and contains the 6.1 MHz IF signal. However for narrow band reception, the tuning is offset by 25kHz so that the desired carrier is at 6.125 MHz. This signal is mixed with a 6 MHz quadrature square wave to yield I and Q data, which is then passed through two decimating low pass filters to reduce the sample rate to 960 kHz (for wide band) or 480 kHz for narrow band. Then the data is low-pass filtered again and mixed with either 240 or 120 kHz for image rejection, and there is a band pass channel filter (140 kHz for wide band, 45 kHz for narrow band) before the demodulator. Finally there is an audio low-pass filter before the data is pulse-width modulated (PWM) at 48 kHz to drive the audio output pins, and is also made available in digital form to the Raspberry Pi interface.

Tuning the Radio

The tuner frequency can be changed using one of the control interfaces (SPI, I2C or Serial) as described in the API document. For convenience, a sample Raspberry Pi command line application (ttune) is provided which does this using the SPI interface. The C source code for this application is also supplied as a NetBeans project, and requires the bcm2835 library to be installed.

The command line usage is:

ttune [options] n

where n is the frequency in kHz.

The options available are:
-a Selects AM demodulation

-n Selects narrow band FM demodulation (the default)
-w Selects wide band FM demodulation
-h Selects the HF antenna
-v Selects the VHF/UHF antenna (the default)
-s Turns on audio playback over the Raspberry Pi audio output device
-r Displays received signal strength (RSSI) information

Options can be combined, for example:
ttune –ws 93500
Tunes to 93.5 MHz wideband FM and outputs the sound

ttune –ah 909
selects the HF antenna and tunes to 909 kHz AM

Notes on building the firmware under Lattice Diamond

The sample firmware project as supplied can be loaded under Lattice Diamond and built by right-clicking on Export Files on the Process tab and selecting Run.

The stand-alone Diamond programming utility can then be used to program the resulting .BIT file directly into the FPGA using a USB lead. The .BIT file can be found in the impl1 subfolder. Ensure that the jumpers are set correctly (J450 installed but not J451). Please also ensure that the FTDI drivers supplied by Lattice are installed, the cable type is set to B (FTDI) within the programming utility and that the programming delay is set to the maximum value (10).

To produce a VME file that can be uploaded from the Raspberry Pi, the deployment tool provided by Lattice must be used. This can be launched from the stand-alone programmer.

Tip: when modifying the project or starting a new project, ensure that the Synplify Pro synthesis engine remains selected and that “Disable I/O insertion” is set to True (to access this option, right click on Strategy1 in the file list and select Edit). Also ensure that the correct Lattice part is selected for the project (LFE5U-45F-6BG381).
