Improved LIRC driver for the Raspberry Pi

Table of contents

Date	Description
2014-01-12	Initial version.

Table 1: Revision history

1 Introduction

Using the Raspberry Pi, with its flexible GPIO pins, for LIRC IR control and reception is a natural wish. Sending IR diodes and IR receivers can easily be attached to the GPIO pins. Aron Szabo wrote a LIRC driver for the Raspberry, as a development of the LIRC serial driver. This article describes a further development of Aron's driver.

The present article assumes fundamental knowledge about LIRC and the Raspberry Pi. It is not a tutorial.

1.1 Copyright

2 Features

Next the features introduced will be described.

2.1 Multiple transmitter support

The RPi has at least 17 usable GPIO pins, so why should a LIRC driver only support using one? Furthermore, the LIRC driver API has `LIRC_SET_TRANSMITTER_MASK` ioctl. For the user, the `irsend` command the directive `SET-TRANSMITTERS` (taking as argument a list of numbers), so also LIRC can handle several transmitters (although very few drivers have implemented it). I, somewhat randomly, selected to support up to 8 transmitters. Should more be needed, only the compilation constant `LIRC_RPI_MAX_TRANSMITTER`. As a consequence, the module parameter `tx_mask`, which states the initial state of the transmitters selected, has been introduced.

I could not find any recommendation on the numbering of the transmitters in the LIRC documentation. Since the only driver implementing multiple transmitters (CommandIR) numbered the transmitters starting from 1, I have decided to do likewise.

2.2 Support for "modulation frequency" 0

Almost all modern IR protocols use a modulation frequency, however, some do not (Revox, Barco, Archer). Possibly more interesting, sending RF devices such as the RX433 are not designed to be fed with modulated signals. Many LIRC drivers, like Aron's, have a boolean parameter `softcarrier`, that govern the generation of the modulation. If turned off, no modulation will be generated. However, being a module

parameter, it cannot be changed in a running module, and it can also not have a different value for different transmitters.

The modification presented here simply allows the frequency parameter in the ioctr call LIRC_SET_SEND_CARRIER to have the value 0, in which case no carrier will be generated.

To my knowledge, other LIRC drivers no not support non-modulated IR signals in the sense of frequency = 0.

2.3 Inversion of outputs

The module parameter `invert`, which inverts the on/off-status of the outputs (for all transmitters) is new.

3 Installation

This module is no different from other LIRC modules when it comes to compiling and installation. There are a number of LIRC pages on the Internet.

Just copying the compiled binary module `lirc_rpi.ko` to the target system may work.

4 Module arguments

The module arguments are described in this section. These are parameters that are passed to the module on startup.

`gpio_out_pins` (array of integers)
GPIO output/transmitter pins of the BCM processor as array. The first is called transmitter #1 (not 0). Valid pin numbers are: 0, 1, 4, 8, 7, 9, 10, 11, 14, 15, 17, 18, 21, 22, 23, 24, 25. Default is none.

`gpio_in_pin` (integer)
GPIO input pin number of the BCM processor. Valid pin numbers are: 0, 1, 4, 8, 7, 9, 10, 11, 14, 15, 17, 18, 21, 22, 23, 24, 25. Default is none.

`sense` (bool)
Override auto detection of IR receiver circuit (0 = active high, 1 = active low).

`softcarrier` (bool)
Software carrier (0 = off, 1 = on, default on)

`invert` (bool)
Invert output (0 = off, 1 = on, default off)

`tx_mask` (integer)
Transmitter mask (default: 0x01)

`debug` (bool)
Enable debugging messages.

The name `gpio_out_pins` is incompatible with Aron's version, using the name `gpio_out_pin`. This also goes for the default values: Aron uses 18 and 17 for

`gpio_in_pin` and `gpio_out_pin`, my version has both undefined as default. To minimize confusion, I have decided just to publish one version of the driver.

5 Further work

I would be very interested in getting the driver to work with a non-demodulation sensor, like the QSE159 (see this article), to arrive at an estimate of the modulation frequency.

Except for this, I do not plan to continue the development, or to "maintain" the driver.

6 Downloads

- lirc_rpi.c. The source.
- lirc_rpi.ko. Compiled kernel module. Sometimes it suffices just to copy this to the RPi.

7 Resources

- Aron's site.
- Sort-of announcement on LIRC mailing list.
- Diskussion thread on the driver. This driver is discussed on page 6.
- Another guide on LIRC on the RPi.