Универсальный драйвер контроллера программируемой кнопки LTC2954.

А. Е. Козлов, Е. М. Кравцунов

УДК 004.454

Аннотация.

В докладе описывается опыт авторов в разработке драйвера для контроллера *LTC2954*. Авторами приведена краткая информация о самом контроллере и его технические характеристики. Рассмотрен пример работы драйвера. В пункте «Детали реализации драйвера» описывается взаимодействие драйвера, находящегося в пространстве ядра, с приложениями уровня пользователя, а так же описаны режимы работа драйвера. В пункте «Результаты работы» перечислены написанные программы и способ получения их исходных текстов.

Постановка задачи.

Авторы доклада решают задачу программной поддержки контроллера *LTC2954*. Программная поддержка включает в себя модуль ядра *Linux* - драйвер для контроллера *LTC295*, и программы уровня пользователя. Контроллер *LTC2954*, являющийся периферийным устройством на материнской плате, позволяет решить 2 задачи:

1. Реализовать функциональность кнопки *Sleep/Power* стандарта *ACPI* [1] при отсутствии в центральном процессоре и контроллере южного моста аппаратной поддержки *ACPI*.
2. Реализовать возможность отключения питания и перезапуска вычислительного комплекса при отсутствии такой поддержки в контроллере южного моста.

Контроллер *LTC2954* предоставляет операционной системе возможность доступа к двум контактам (пинам): пин прерывания типа «*input*» и пин отключения типа «*output*». Если в реализации материнской платы предусмотрена свободная линия прерывания и она может быть соединена с пином прерывания контроллера *LTC2954*, то при обработке события нажатия кнопки возможно использование обработчика прерывания, реализованного в драйвере *LTC2954*. Если линию прерывания выделить невозможно – драйвер позволяет использовать метод опроса значения пина прерывания.

Предлагаемое решение построено на взаимодействии 3-х подсистем ядра *Linux*: подсистема управления выводами общего назначения (*gpio*), подсистема ядра *input layer* для взаимодействия со служебными потоками (демонами) и сервисами пользователя, подсистема прерываний. Передаваемые при загрузке модуля параметры позволяют сделать драйвер универсальным, работающим для любых архитектур. С помощью параметров можно задать способ работы с пином прерывания – опрос или обработчик прерывания, а также задать номера контактов *gpio*, соединенных с пинами контроллера *LTC2954*.

Технические характеристики контроллера *LTC2954*.

LTC2954 – это контроллер включения/выключения, который управляет системой питания через интерфейс кнопки [2]. Пин отключения переключает систему питания, в то время как пин прерывания обеспечивает подавление дребезга контактов кнопки. Пин прерывания может быть использован в управляемых из меню приложениях, для выполнения запроса на отключение системы. Независимо настраиваемые таймеры включения и отключения обеспечивают надежный контроль над пином отключения и устойчивость к случайным переключениям системы питания.

Особенности *LTC2954*:

1. Настраиваемые таймеры включения/отключения.
2. Низкий ток питания: 6μA.
3. Широкий диапазон рабочего напряжения: 2.7V - 26.4V.
4. Простой интерфейс обеспечивает корректное завершение работы микропроцессора.
5. Высокое входное напряжение с внутренним подтягивающим резистором.
6. 8-контактный 3 мм × 2 мм.

Детали реализации драйвера ядра.

Рассмотрим работу драйвера на примере отключения системы по нажатию кнопки (Рисунок 1.Схема работы драйвера.):

1. Сначала получаем сигнал с выхода контроллера (*out*) и передаем его модулю (драйверу), находящемуся в пространстве ядра.

2. Далее модуль обрабатывает сигнал и отправляет его пространство пользователя демону, через подсистему *Input Layer*.

3. Демон запускает заложенный в него скрипт, в данном случае это команда *halt* [3], которая отключает сначала программы пользователя, а затем передается ядру.

4. Ядро отправляет сигнал отключения на вход контроллера (*in*).

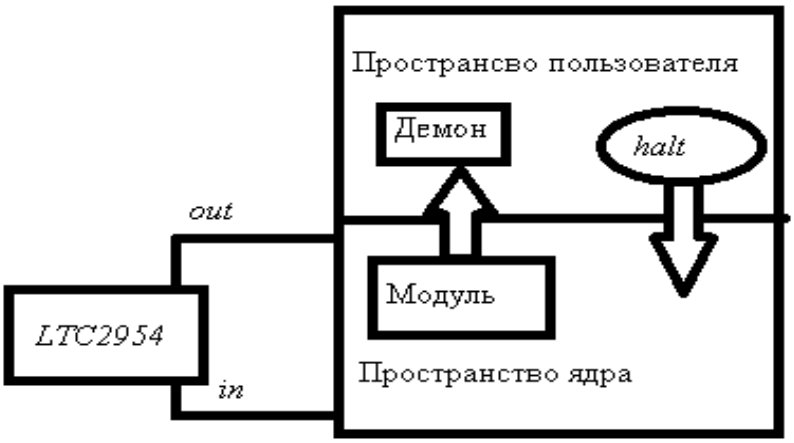

Рисунок 1. Схема работы драйвера.

Как было сказано ранее, драйвер устройства написан с поддержкой двух режимов:

- С использованием линии прерываний (*irq*).
- С использованием механизма опроса (*poll*).

В программе реализован запуск с параметрами, что позволяет выбрать режим работы (Рисунок 2. Запуск драйвера с параметрами). Параметр *use_irq* отвечает за запуск драйвера в режиме работы по прерыванию, если он выставлен в 1, если в 0, то необходимо задать параметр *gpionum_in*, значение в котором указывает на конкретный *GPIO*, который необходимо использовать.

Рисунок 2. Запуск драйвера с параметрами.

В режиме работы по прерыванию мы используем текущую линию прерывания, отлавливаем само прерывание затем находим соответствующий ему номер *gpio* и передаем обработчику прерываний ядра(ОПЯ) (Рисунок 3. Работа по прерыванию.).

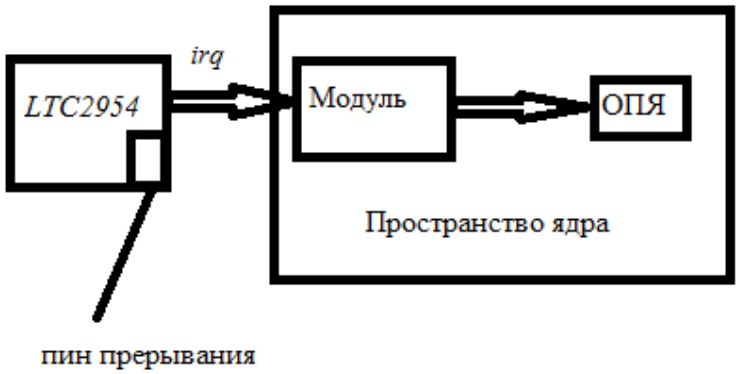

Рисунок 3. Работа по прерыванию.

В режиме работы с механизмом опроса может возникнуть проблема использования линии прерывания – она может быть занята (например сторожевым таймером). Решение проблемы – циклический опрос портов *gpio* (Рисунок 4. Опрос портов.).

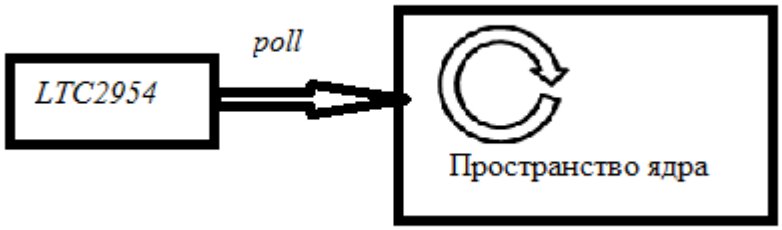

Рисунок 4. Опрос портов.

Программы уровня пользователя.

Подсистема *Input Layer(IL).* Для взаимодействия пользовательского пространства с пространством ядра существует подсистема *IL*. Для управления модулем ядра из пространства пользователя был написан демон который принимает информацию из *IL*.

Демон - опрашивает в бесконечном цикле устройство */dev/input/eventX*, которое занимает кнопка при подгрузке модуля. Демон должен запускаться из сервиса, на старте системы, поэтому он должен быть написан грамотно, то есть с использованием *daemonize*. *Daemonize* - это общие правила для написания пользовательских демонов, основные требования таковы:

- Вызов *fork()* [3] используется для создания отдельного процесса.
- Вызов *setsid()* [3] используется для отделения процесса от родителя.
- Изменить текущий каталог.
- Переоткрыть все *std*-файлы (стандартный ввод/вывод, стандартный вывод ошибок и т.д.).
- Маска файла должна быть изменена.

Невыполнение любого из этих шагов приведет к неправильной работе демона. Типичные ошибки:

- Запуск демона, а затем выход из системы приведет к зависанию терминала.
- Каталог, из которого запущен демон, будет заблокирован.
- Появляются паразитные сигналы.

При написании драйвера автороми использовался *daemonize* с расширенными функциями:

- Сборка сообщений в системный журнал (*syslog* [3])
- Создание файла, блокирующего повторный запуск демона.

- Об ошибках запуска сообщается в основной процесс.

Сервис подгружает модуль, находит номер устройства *eventX*, на котором оказалась кнопка и запускает демон, передавая ему в качестве параметра путь к устройству.

Результаты работы.

Реализована программная поддержка кнопки на базе контроллера LTC2954, состоящая из 3-х программ:

- Драйвер работающий в 2-х режимах: по прерыванию и с использованием механизма опроса.
- Демон обеспечивающий взаимодействие пространства пользователя с драйвером, работающим в пространстве ядра.
- Специальная программа сервис, отвечающая за запуск драйвера и модуля на старте системы.

Предложенное решение является универсальным и может быть использовано для любой архитектуры. Результат работы опубликован под лицензией GPL на общедоступном сервере git.mcst.ru.

Авторы надеются, что наш опыт написания драйвера сможет быть полезным как новым, так и опытным разработчикам драйверов устройств *Linux*.

Используемая литература.

[1] *Advanced Configuration and Power Interface Specification, Hewlett-Packard Corporation, Intel Corporation, Microsoft Corporation, Phoenix Technologies Ltd, Toshiba Corporation, Revision 4.0a, April 5, 2010*

[2] Техническая документация LTC2954,http://cds.linear.com/docs/en/datasheet/2954fb.pdf

[3] *Unix: In a Nutshell*, Арнольд Роббинс, КУДИЦ-Пресс, *ISBN* 5-91136-031-4; 2007