Linux Kernel Drivers for I^{2}C-manageable High Precision Power Source Based on ISL22317 and PCA9536 Chips

Evgeniy Kravtsunov, Andrey Kuyan, Sergey Radchenko, Andrey Kozlov
Moscow Center of SPARC Technologies (ZAO MCST)
Moscow, Russia
{kravtsunov_e, kuyan_a, radch_s, kozlov_a}@mcst.ru

Abstract

Within the bounds of verification test bench development for multicore Elbrus microprocessor ZAO MCST [1] designed precision single I^{2}C-controlled power source based on Intersil ISL22317 digital control potentiometer and NXP PCA9536 4-bit SMBus I/O port chip. Power source was built on four potentiomenters and one SMBus I/O port. Linux kernel driver for ISL22317, implemented by the authors, allowed to support software-manageable power supply for test microprocessor and to select a set of optimal supply voltages for test microprocessor cores precisely. As the solution has been admitted to be useful for developers, it was decided to contribute implemented driver of ISL22317 to linux kernel community. Suggested driver implementation is universal and it can turn to be useful for other hardware devices built on ISL22317. Current article contains a generalized scheme of power source, description of ISL22317 driver's sysfs interfaces. The article reports on the suggested algorithm of voltage supply management from userspace programms. Much attention is payed to details of potentiometer driver implementaion. Readers attention is drawn to the driver initialization and device instantiating methods for ISL22317 and peculiarities of interaction between potentiometers and PCA9536 chip. Suggested software support makes possible to manage up to four potentiometers bound to common I^{2}C bus. Article also informs on algorithm of management up to 4 power sources bound to the same I^{2}C bus. Proposed algorithm is based on I^{2}C-bus multiplexing by Linear LTC4306 chip. The source code of linux kernel driver of LTC4306 multiplexer, imlemented by authors, is covered by GNU General Public License version 2 and is available in public git repository along with ISL22317 driver. Current article reports on project in progress. In the conclusion authors carry out drawbacks analyze of suggested power source design and propose a simplified variant of I^{2}C-manageable power source implementaion based on a single chip.

Index Terms: I^{2}C-bus, ISL22317, LTC4306, Linux Kernel Driver.

I. Introduction

IIC (Inter-Integrated Circuit or I^{2}C) is a serial bus which was invented and proposed for support of integrated peripherals by Phillips at the beginning of 1980 [2]. Currently I^{2}C is one of de-facto standards of interaction with integrated peripherals in personal computers, embedded systems and mobile devices. Integrated peripherals is a variety of sensors and controls located on motherboard of personal computer or on board of embedded system. The physical layer of I^{2}C is a pair of wires: serial clock (SCL) wire and serial data (SDA) wire. A rate of data exchange is determined by the frequency of SCL. Modern I^{2}C controllers usually support three frequency modes: 100 KHz, 400 KHz and 1MHz. Data exchange over I^{2}C bus is performed according to master-slave protocol. It means that data exchange over I^{2}C bus with up to 128 slave chips (sensors and controls) is managed by master-I^{2}C controller. Every slave chip owns unique 7-bit length address. I^{2}C specification provides support for 10-bit addresses,

but currently a very few number of controllers support 10-bit addressing. Core support of I^2C protocol is implemented in linux kernel. Protocol is based on sequential writing of bytes to bus and reading the reply bytes from bus. Exchange is initiated by controller, which is either a standalone chip or an integrated part of SOC. Controller has a number of registers on board that are software available. These registers are used by driver of controller (software driver of controller is named adapter in linux kernel) for implementing software I^2C interfaces. There is a wide variety of I^2C controllers by different vendors and each owns a set of registers that differs from other controllers. Most popular controllers have their adapter implementations in linux kernel. Every adapter realizes the unified set of I^2C interfaces, which are declared in linux kernel header file `include/linux/i2c.h`. Every I^2C slave chip has SCL and SDA pins for binding to I^2C bus, vendor assigned 7-bit address that can not be updated from software, and a set of registers available from software. Software available registers are used by linux kernel driver of slave chip. To work with registers slave chip driver uses interface functions declared in `include/linux/i2c.h`, that are implemented in adapter of I^2C controller. Slave chip driver has no information about internals of I^2C controller and can use only unified interface functions provided by adapter. As a rule, adapters implement a subset of I^2C protocol named SMBus. SMBus is a set of read/write interfaces listed in table 1. Using

TABLE I
SMBUS INTERFACES

Function	Description
`i2c_smbus_read_byte`	Reads one byte from the chip. Address of register is not specified. Function is useful for chips containing the only 8-bit register.
`i2c_smbus_write_byte`	Writes one byte to the chip. Address of register is not specified. Function is useful for chips containing the only 8-bit register.
`i2c_smbus_read_byte_data`	Reads one byte from specified register of chip.
`i2c_smbus_write_byte_data`	Writes one byte to specified register of chip.
`i2c_smbus_read_word_data`	Reads word (2 bytes) from specified register of chip.
`i2c_smbus_write_word_data`	Writes word (2 bytes) to specified register of chip.
`i2c_smbus_read_block_data`	Reads block of data up to 64 bytes length from specified register of chip.
`i2c_smbus_write_block_data`	Writes block of data up to 64 bytes to specified register of chip.

SMBus subset in slave chip driver is optimal from the point of view of driver portability. For the majority of I^2C slave chips information about vendor assigned address of I^2C slave chip, a set of registers and recommended operational conditions is open and available in datasheet. Datasheet for ISL22317 is available here [3].

II. GENERALIZED SCHEME OF POWER SOURCE AND INTERACTION BETWEEN CHIPS

Authors were given a task of implementation software support of I^2C-manageable power source that is able to provide four different supply voltage values on different outputs. Power source was constructed by engineers of ZAO MCST. Power source is built on four I^2C-managable potentiometers ISL22317. In conformity with technical documentation[3] ISL22317 chip realizes the following functionalities:

1) supply voltage management in the range of [0; Vcc] by setting the position of wiper (from 0 till 127 in conditional values);

2) storing the default wiper position in EEPROM;
3) ability to manage precision regimes ;
4) support of low power consumpion state of chip and ability to move chip from active to low power state and back;
5) support of two modes of chip: potentiometer and resistor, and ability to switch between them.

According to datasheet ISL22317 chip can have one of two addresses on bus: either `0x2a` or `0x28`. Value of address is determined by the logic value at pin A1 of chip: if A1 is true - the address is `0x2a`, else the address is `0x28`. Thereby only two ISL22317 chips can be bound to the same I^2C bus. To support four chips on the same bus a decision was taken to use 4-bit PCA9536 SMBus I/O chip for switching logic value 1 between A1 pins of ISL22317 chips. PCA9536 realizes 4 gpio pins that can be managed through I^2C (Fig1.). Switching between

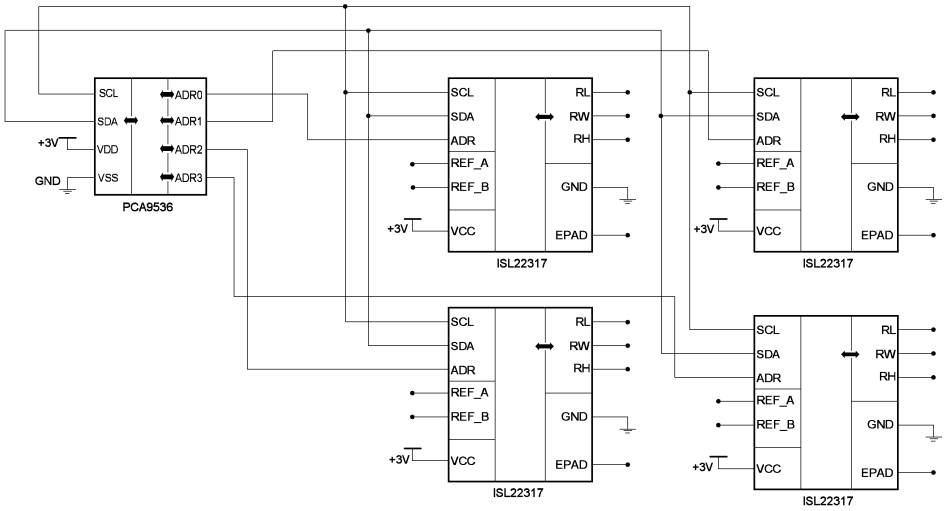

Fig. 1. Generalized scheme of power source

ISL22317 chips is performed by setting "1" to the corresponding "addr" output of PCA9536 chip. Setting is carried out through interface of PCA9536 I^2C slave chip driver. Thereby from the point of view of kernel, power source looks like two slave chips on the same I^2C bus: one PCA9536 chip with address `0x41` [4], and one ISL22317 chip with address `0x2a`. From the software, switching between potentiometers ISL22317 is performed in three steps:

1) setting the "1" logic value in the corresponding "addr" output pin of PCA9536 chip;
2) reinitialization of data structures in memory that describe ISL22317 slave chip;
3) writing the value of wiper position in conditional values to ISL22317.

PCA9536 slave chip driver is implemented in linux kernel (`drivers/gpio/pca953x.c`). Outputs of PCA9536 are managed through sysfs interfaces, that are realized in driver according to gpiolib convention [5]. Thereby to provide support for power source authors had to implement only ISL22317 slave chip driver. Details of potentiometer driver implementation are considered below.

III. Test bench, interaction with I^2C controller

Figure 2 illustrates the test bench (installation) that was used for testing power source. As control host was used an x86 machine with IntelPIIX4 [6] south bridge chip on board.

IntelPIIX4 chip contains embedded I^2C controller, that was used by authors for managing I^2C bus. Power source is located on the prototype of Elbrus computer's motherboard. IntelPIIX4 was connected to power source by two wires. Control host was running linux with latest kernel

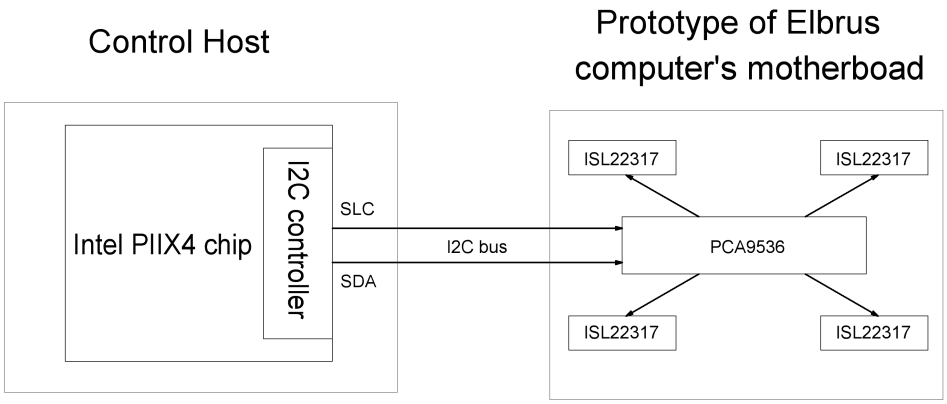

Fig. 2. Scheme of test bench

built with enabled support of I^2C `(CONFIG_I2C)`, IntelPIIX4 `(CONFIG_I2C_PIIX4)`, PCA9536 `(CONFIG_GPIO_PCA953X)` and ISL22317 `(CONFIG_ISL22317)`. The goal of experiment was to select optimal values of voltage supply for Elbrus microprocessors.

IV. Driver internals

Consider some details of ISL22317 slave chip driver implementation. Source code of ISL22317 driver is available in linux kernel 3.6 tree in public git repository `git.mcst.ru`. Driver is implemented as a kernel module, that can be enabled by setting `CONFIG_ISL22317` config option to "m". Driver implements a set of read/write functions for accessing registers of potentiometer. Realization of sysfs interfaces to userspace is based on these readwrite functions. Read/write functions use only two SMBus interfaces from table 1: `i2c_smbus_read_byte_data` and `i2c_smbus_write_byte_data`. It is enough because all the registers of potentiometer are 8-bit ones according to datasheet. Driver implements `isl22317_probe` function that is registered as probe callback for driver model while loading module `isl22317.ko`. Function `isl22317_probe` carries out the following initialization steps:

1) checks the adapter functionality for support `i2c_smbus_readwrite_byte_data` interfaces;
2) allocates internal structure `isl22317_data` that describers mode, state and registers of ISL22317 chip;
3) initializes chip by default values using smbus interfaces;
4) creates a group of sysfs files for managing ISL22317 from userspace.

Callback function probe for driver is called on module loading only when the device is instantiated. There are 4 methods suggested by linux kernel driver model that can be used for instantiating devices. The most popular method based on autodetection of bus clients can not be used for ISL22317. This is because of two reasons: ISL22317 does not have registers where vendor id and revision id of chip are stored, and ISL22317 doesn't support `SMBUS_QUICK` command, that is used by I^2C core while probing an I^2C bus for certain devices. Instead autodetection authors have tested two alternative methods proposed by I^2C core: 1) declaring the I^2C devices by bus number; 2) instantiating the devices explicitly. Both methods are

appropriate for ISL22317. First method can be implemented by initializing the array of struct `i2c_board_info` which is registered by calling `i2c_register_board_info` function during the kernel init process. This method is preferable for linux kernels for embedded or mobile devices. Second method can be implemented by calling `i2c_new_device()` for initialized `i2c_board_info` structure. Function `i2c_new_device()` can be called from the slave chip driver initialization path. Second method is appropriate when more then one adapter is registered in the system and number of I^2C adapter is unknown in advance. Authors used first method on the control host of test bench (Fig. 2).

V. Sysfs interfaces for ISL22317 potentiometer

Clients for I^2C-controlled potentiometer weren't found in linux kernel by authors while developing driver, therefore original sysfs interface for ISL22317 was developed according to linux kernel driver model and sysfs standard [7]. But now driver for AD525x digital potentiometer in latest versions of kernel, so authors below will be demonstrated a comparison of the ISL22317 an AD525x interfaces. Sysfs interfaces [8] for I^2C core subsystem can be created by using `i2c-dev` kernel module [9]. While loading `i2c-dev` module detects registered I^2C adapters by scanning the list of adapters. If adapters were found, `i2c-dev` creates `adapter name` folder in `/sys/class/i2c-dev`. Folder `adapter` name contains nested folders which correspond to slave chips bound to the I^2C bus, controlled by adapter. On the control host (figure 2) of test bench folder `i2c-0` was created by `i2c-dev` module for IntelPIIX4 I2C controller. The following nested folders were created in `i2c-0`: `devices/0-0041` - for PCA9536 slave chip and `devices/0-002a`-for ISL22317 slave chip. While loading modules `pca953x.ko` and `isl22317.ko`, folders `0-0041` and `0-002a` are filling with sets of files. As for `isl22317.ko` module - it's loading causes creating following file structure (Fig.3). All listed files are available for reading and writing, except `reinit_chip`, which is available only for writing. Writing "1" value to `reinit_chip` file causes reinitialization of `i2c_client` structure that describes ISL22317. File `wiper` contains current decimal value of wiper position in conditional values (from 0 till 127). Writing to `wiper` file causes to setting corresponding voltage on the ISL22317 output. Absolute value of voltage depends on the peculiarities of power source implementation, therefore power source should be calibrated before using. File `ivalue` contains the initial value of wiper position that is to be set by default on power on. This value is stored in EEPROM on board of ISL22317 when the power is off. Initial value can be updated by writing updated value to `ivalue` file. File `precision` having only two valid contents: `1` or `0`, allows to manage precision regimes of ISL22317. Writing value "1" to `precision` file causes switching on the precision regime, value "0" switched off precision regime. File `mode` is used for switching ISL22317 chip from "potentiometer" mode (corresponds value "1") to "resistor" mode (corresponds value "0"). File `power_state` is used for moving chip from active state (value "0") to low power consumption state (value "0") and back. While loading module `pca953x.ko` a symbolic link `gpiochip0` appears in `/sys/class/i2c-dev/i2c-0/devices/1-0041` folder. This is a link to the path in sysfs where files for gpio are placed according to gpiolib convention [5]. For example, for setting "1" to adr0 output of PCA9536 (Fig. 1) and values of "0" to other outputs addr1, addr2, addr3 the following actions should be carried out (Tab.2).

VI. Algorithm of managing power source from userspace

Linux kernel built with enabled support of: I^2C (`CONFIG_I2C`), IntelPIIX4 (`CONFIG_I2C_PIIX4`), ISL22317 (`CONFIG_ISL22317`), PCA9536 (`CONFIG_GPIO_PCA953X`),

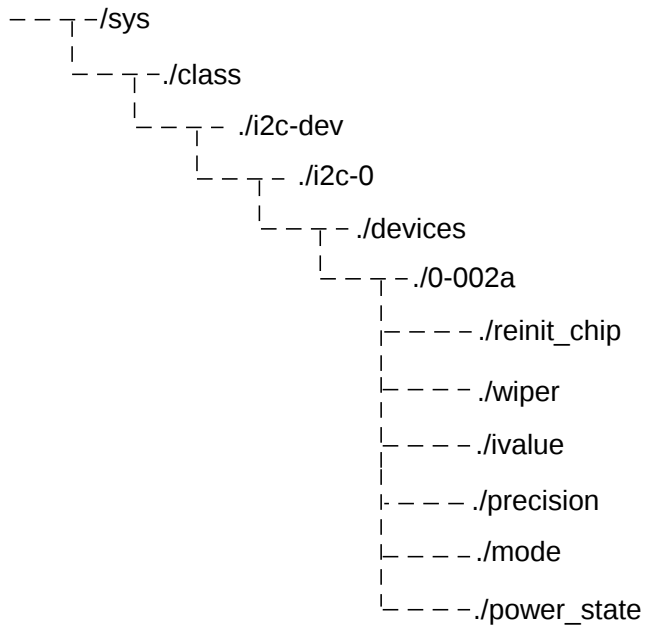

Fig. 3. File structure for ISL22317 interface

TABLE II
EXAMPLE OF USING PCA9536 THROUGH SYSFS INTERFACE

Action	Corresponding shell commands

Export all gpio pins of PCA9536:	`$ cd /sys/class/i2c-dev/i2c-0`
	`$ echo "0" > ./devices/0-0041/export`
	`$ echo "1" > ./devices/0-0041/export`
	`$ echo "2" > ./devices/0-0041/export`
	`$ echo "3" > ./devices/0-0041/export`

Set the direction of all gpio pins to ”output”:	`$ cd /sys/class/i2c-dev/i2c-0`
	`$ echo "out" > ./devices/0-0041/gpio0/direction`
	`$ echo "out" > ./devices/0-0041/gpio1/direction`
	`$ echo "out" > ./devices/0-0041/gpio2/direction`
	`$ echo "out" > ./devices/0-0041/gpio3/direction`

Set ”1” to gpio0 output and values of ”0”	`$ cd /sys/class/i2c-dev/i2c-0`
	`$ echo "1" > ./devices/0-0041/gpio0/value`
	`$ echo "0" > ./devices/0-0041/gpio1/value`
	`$ echo "0" > ./devices/0-0041/gpio2/value`
	`$ echo "0" > ./devices/0-0041/gpio3/value`

I^2C-DEV (`CONFIG_I2C_CHARDEV`), SYSFS (`CONFIG_SYSFS`) allows to use the following algorithm for power source management from userspace (Tab. 3). For updating output voltage on potentiometers number 1, 2, 3 steps 3 and 4 should be repeated with setting setting the ”1” to corresponding gpio output and ”0” to other gpio outputs of PCA9536 chip. ISL22317 driver interfaces allow also to perform the following optional actions (Tab. 4). While switching to low power state driver saves the context of ISL22317 registers to memory and restores registers from memory during switching back to active state.

TABLE III
ALGORITHM OF POWER SOURCE MANAGMENT FROM USERSPACE

Action	Corresponding shell commands

Load necessary modules:	`$ modprobe i2c-dev`
	`$ modprobe pca953x`
	`$ modprobe isl22317`

Export gpio pins and set their directions to ”output”:	`$ cd /sys/class/i2c-dev/i2c-0/devices`
	`$ echo "0" > ./0-0041/export`
	`$ echo "1" > ./0-0041/export`
	`$ echo "2" > ./0-0041/export`
	`$ echo "3" > ./0-0041/export`
	`$ echo "out" > ./0-0041/gpio0/direction`
	`$ echo "out" > ./0-0041/gpio1/direction`
	`$ echo "out" > ./0-0041/gpio2/direction`
	`$ echo "out" > ./0-0041/gpio3/direction`

Select and reinitialize potentiometer:	`$ cd /sys/class/i2c-dev/i2c-0`
	`$ echo "1" > ./devices/0-0041/gpio0/value`
	`$ echo "0" > ./devices/0-0041/gpio1/value`
	`$ echo "0" > ./devices/0-0041/gpio2/value`
	`$ echo "1" > ./devices/0-002a/reinit_chip`

Set the value of wiper position X : (X belongs to the range [0,127])	`$ cd /sys/class/i2c-dev/i2c-0` `$ echo X > ./devices/0-002a/wiper`

TABLE IV
ALGORITHM OF UPDATING OUTPUT VOLTAGE ON POTENTIOMETRS USING SYSFS INTARFACE

Action	Corresponding shell commands
Read the initial value of wiper position from EEPROM:	`$ cd /sys/class/i2c-dev/i2c-0` `$ cat ./devices/0-002a/ivalue`
Write the initial value X of wiper position to EEPROM:	`$ cd /sys/class/i2c-dev/i2c-0` `$ echo x > ./devices/0-002a/ivalue`
Read current wiper position:	`$ cd /sys/class/i2c-dev` `$ cat ./i2c-0/devices/0-002a/wiper`

Switch to ”precision” regime and back:	`$ cd /sys/class/i2c-dev/i2c-0`
	`$ echo 1 > ./devices/0-002a/precision`
	`$ echo 0 > ./devices/0-002a/precision`

Switch to ”resistor” mode and back to ”potentiometer” mode:	`$ cd /sys/class/i2c-dev/i2c-0`
	`$ echo 0 > ./devices/0-002a/mode`
	`$ echo 1 > ./devices/0-002a/mode`

Switch from ”active” state to ”low power” state and back:	`$ cd /sys/class/i2c-dev/i2c-0`
	`$ echo 1 > ./devices/0-002a/power_state`
	`$ echo 0 > ./devices/0-002a/power_state`

VII. MANAGING SEVERAL POWER SOURCES BOUND TO THE SAME I^2C BUS BY USING MULTIPLEXER LINEAR LTC4306

During the experiments with test bench authors have encountered additional problem. One of target motherboards contained four power sources instead of one and I^2C multiplexer chip Linear LTC4306 [10]. Multiplexer allows to transmit I^2C data from upstream bus to one of downstream busses (Fig. 3). Downstream bus can be selected by I^2C commands from upstream bus as LTC4306 itself is I^2C slave chip. Upstream bus was the bus that connects master (IntelPIIX4) and target board, every downstream bus connected one power source based on ISL22317 and PCA9536 chips with multiplexer. Problem has been successfully solved by authors by implementing linux kernel driver for LTC4306. Source code of driver is available in public git repository `git.mcst.ru` along with ISL22317 source code.

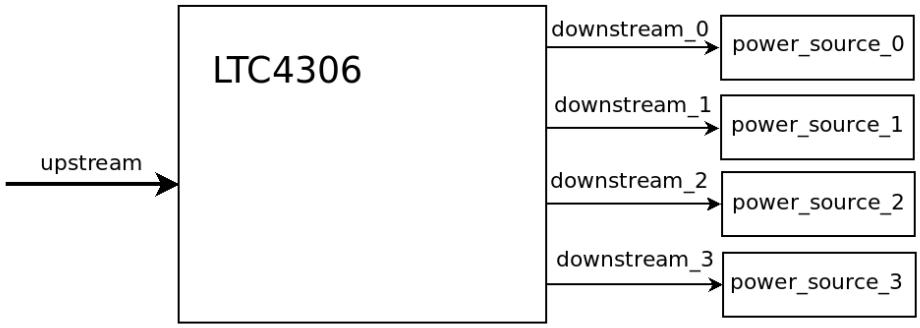

Fig. 4. Using LTC4306 multiplexer for accessing 4 power sources on target board

VIII. SYSFS INTERFACE FOR AD525X

The `ad525x_dpot` driver allows to work with the immediate resistance settings as well as update the saved startup settings. Access to the factory programmed tolerance is also provided, but interpretation of this settings is required by the end application according to the specific part in use. Each dpot device will have a set of `eeprom`, `rdac`, and `tolerance`

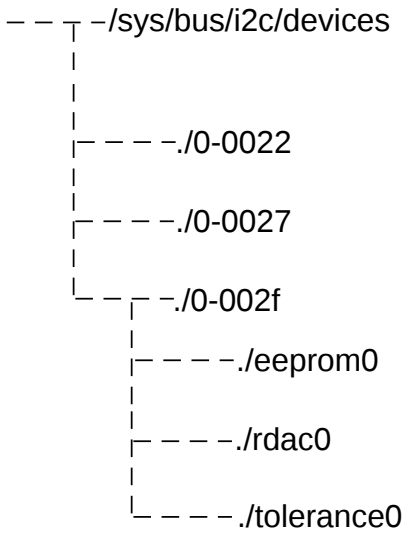

Fig. 5. File structure of AD525x interface

files (Fig.5). How many depends on the actual part you have, as will the range of allowed values. The eeprom files are used to program the startup value of the device. The `rdac` files are used to program the immediate value of the device. The tolerance files are the read-only factory programmed tolerance settings and may vary greatly on a part-by-part basis. For exact interpretation of this field, please consult the datasheet for your part. This is presented as a hex file for easier parsing. The above description of both interfaces indicates the similarity in the implementation of the drivers. This is a consequence of following sysfs standard and I^2C-core interface. The difference is that the support of devices functionality is implemented in

TABLE V
ALGORITHM OF UPDATING OUTPUT VOLTAGE ON POTENTIOMETRS USING SYSFS INTARFACE

Action	Corresponding shell commands
Locate the device in your sysfs tree.This is probably easiest by going into `032` the common I^2C directory and locating the device by the I^2C slave address:	`$ ls /sys/bus/i2c/devices/` `0-0022 0-0027 0-002f`
So assuming the device in question is on the first I^2C-bus and has the slave `038` address of `0x2f`, we descend (unrelated sysfs entries have been trimmed).	`$ ls /sys/bus/i2c/devices/0-002f/` `eeprom0 rdac0 tolerance0`

You can use simple reads/writes to access these files:	`$ cd /sys/bus/i2c/devices/0-002f/`
	`$ cat eeprom0`
	`0`
	`$ echo 10 > eeprom0`
	`$ cat eeprom0`
	`$ cat rdac0`
	`5`
	`$ echo 3 > rdac0`
	`$ cat rdac0`
	`3`

accordance with techical documentation, which describe internal structure of devices. Several recommendations for driver development can be given by authors: driver should strongly follow `i2c-core` interfaces and sysfs standard, particular realisation of sysfs interface should be inplemented according to datasheet and take into account the subsequent usage of the device. Below will be shown why following these guidelines will allow to develop cross-platform Linux kernel driver.

IX. DRIVER AS A CROSS-PLATFORM SOFTWARE

Process of Linux kernel driver development, which has been described, demonstrates the general scheme of interaction user space programs with hardware through several kernel layers. The way suggested by authors may be used as an example in design of new drivers, because there are strong similarities between drivers for I^2C-devices. As you can see (Fig.6), developed drivers are situated in architecture independent layer of Linux kernel, so there are possibilities to use it on an machine with any microprocessor architecture running Linux. Adapter is placed on architecture dependent layer in the sense that hardware realisation of I^2C-controller hinges on manufacturer. Each hardware realization of controller should have its own implementation of driver in kernel. Adapter should support all necessary `i2c-core` interfaces to interract with I^2C-client's driver. According to driver's place in kernel, it can be considered as a cross-platform software. Authors hope, that the above description will be useful for novice developers, who want to start write drivers to use it in their embeded systems or peripheral devices and send patches to community, and for all people working with I^2C kernel drivers.

X. NEXT MILESTONE FOR DEVELOPMENT - POWER SOURCE BASED ON LINEAR LTC2970 CHIP

Power source described in current article provided effective solution for the task of experimental selecting voltages for processor cores of mutlicore microprocessor. Precision mode of ISL22317 is a significant advantage of considered power source. But for the tasks of runtime power management more effective and fast solution can be found. Next step in designing of power sources is to build a power source on a single chip Linear LTC2970 [11]. Engineers

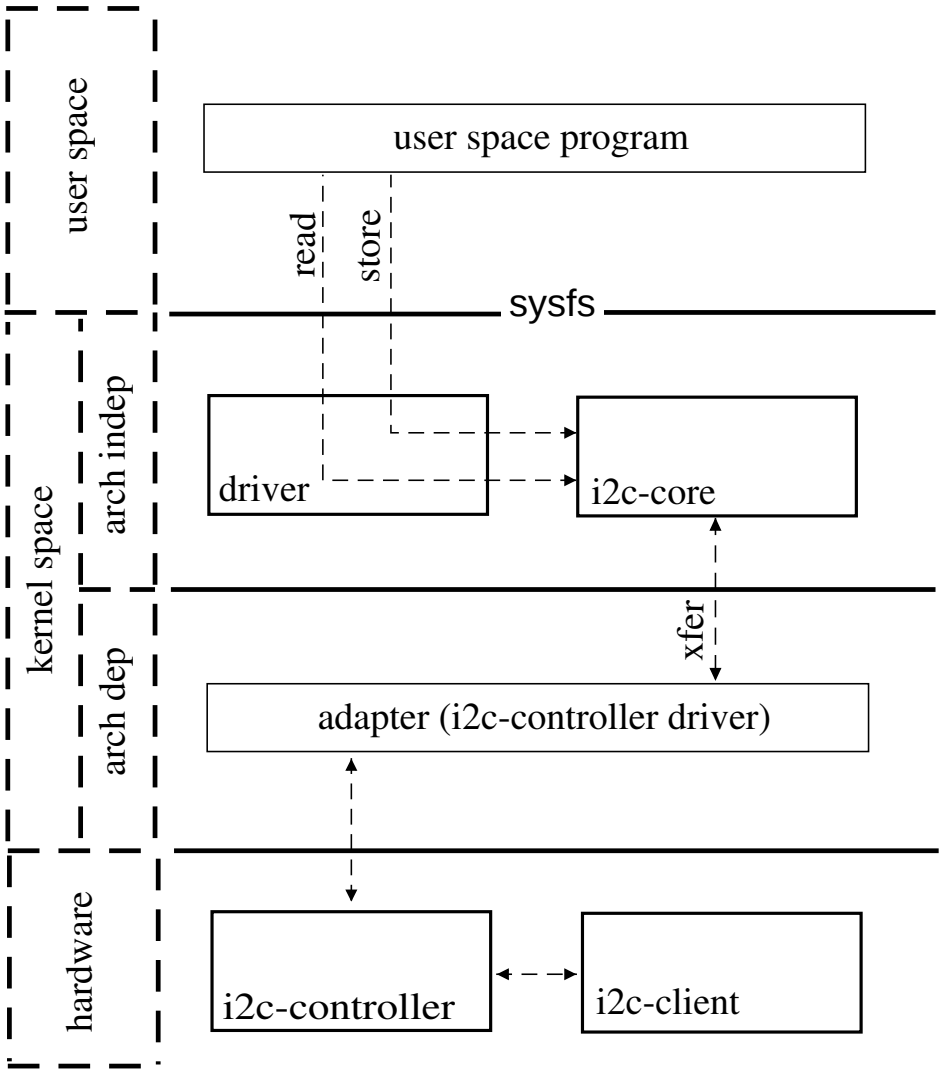

Fig. 6. Generalized scheme of interaction between user space program and hardware

of ZAO MCST are working on such new power source. As for authors - they are currently developing corresponding driver.

REFERENCES

[1] http://www.mcst.ru/
[2] http://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux-2.6.git Documentation/i2c/i2c-protocol
[3] http://www.intersil.com/content/dam/Intersil/documents/fn69/fn6912.pdf
[4] http://www.nxp.com/documents/data_sheet/PCA9536.pdf
[5] http://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux-2.6.git Documentation/gpio.txt
[6] http://www.intel.com/assets/pdf/specupdate/297738.pdf
[7] http://www.kernel.org/doc/Documentation/hwmon/sysfs-interface
[8] J. Corbet, A. Rubini, G. Kroah-Hartman , "Linux Device Drivers," *O'Reilly*, 3rd Edition, pp. 375-383, 2005.
[9] http://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux-2.6.git Documentation/i2c/dev-interface
[10] http://cds.linear.com/docs/Datasheet/4306.pdf

[11] http://cds.linear.com/docs/Datasheet/29067f.pdf