

Re: [PATCH 1/1] : hwmon – new chip driver for TI ADS7828 A–D

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2008-01/msg01234.html>

- *From:* Steve Hardy <steve@xxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Fri, 4 Jan 2008 07:34:39 +0000
-

Andrew/Jean,

Sorry for the delay – christmas & getting mutt working delayed my revised patch. Here is an updated patch against 2.6.24-rc6, which hopefully addresses all comments made so far.

Jean – you mentioned declaring the devices as platform data & writing a new-style I2C driver – are there any examples of this approach I can refer to? I simply copied what is/was currently in-tree for this driver, which seems to work well with the hardware I'm working with (a COTS CPCI carrier)

Since I've dropped Thunderbird I hope the patch mangling will be fixed.

Any problem, please let me know.

Regards,
Steve

Signed-Off by Steve Hardy <steve@xxxxxxxxxxxxxxxxxxxxxx>

```
diff -uprN -X linux-2.6.24-rc6/Documentation/dontdiff linux-2.6.24-rc6/Documentation/hwmon/ads7828
linux-2.6.24-rc6-ads7828/Documentation/hwmon/ads7828
--- linux-2.6.24-rc6/Documentation/hwmon/ads7828 1970-01-01 01:00:00.000000000 +0100
+++ linux-2.6.24-rc6-ads7828/Documentation/hwmon/ads7828 2008-01-02 16:30:39.000000000 +0000
@@ -0,0 +1,38 @@
+Kernel driver ads7828
+=====
+
+Supported chips:
+ * Texas Instruments/Burr-Brown ADS7828
+ Prefix: 'ads7828'
+ Addresses scanned: I2C 0x48, 0x49, 0x4a, 0x4b
+ Datasheet: Publicly available at the Texas Instruments website :
```

Re: [PATCH 1/1] : hwmon – new chip driver for TI ADS7828 A–D

+ <http://focus.ti.com/lit/ds/symlink/ads7828.pdf>

+

+Authors:

+ Steve Hardy <steve@xxxxxxxxxxxxxxxxxxxx>

+

+Module Parameters

+-----

+

+* se_input: bool (default Y)

+ Single ended operation – set to N for differential mode

+* int_vref: bool (default Y)

+ Operate with the internal 2.5v reference – set to N for external reference

+* vref_mv: int (default 2500)

+ If using an external reference, set this to the reference voltage in mv

+

+Description

+-----

+

+This driver implements support for the Texas Instruments ADS7828.

+

+This device is a 12bit 8ch A–D converter.

+

+It can operate in single ended mode (8 +ve inputs) or in differential mode,

+when 4 differential pairs can be measured.

+

+The chip also has the facility to use an external voltage reference. This

+may be required if your hardware supplies the ADS7828 from a 5v supply, see

+the datasheet for more details.

+

+

diff –uprN –X linux–2.6.24–rc6/Documentation/dontdiff linux–2.6.24–rc6/drivers/hwmon/Kconfig

linux–2.6.24–rc6–ads7828/drivers/hwmon/Kconfig

--- linux–2.6.24–rc6/drivers/hwmon/Kconfig 2008–01–02 16:37:43.000000000 +0000

+++ linux–2.6.24–rc6–ads7828/drivers/hwmon/Kconfig 2008–01–02 16:49:42.000000000 +0000

@@ –588,6 +588,16 @@ config SENSORS_SMSC47B397

This driver can also be built as a module. If so, the module

will be called smsc47b397.

+config SENSORS_ADS7828

+ tristate "Texas Instruments ADS7828"

+ depends on I2C

+ help

+ If you say yes here you get support for Texas Instruments ADS7828

+ 12–bit 8–channel ADC device.

+

+ This driver can also be built as a module. If so, the module

+ will be called ads7828.

+

config SENSORS_THMC50

tristate "Texas Instruments THMC50 / Analog Devices ADM1022"

depends on I2C && EXPERIMENTAL

Re: [PATCH 1/1] : hwmon – new chip driver for TI ADS7828 A–D

Re: [PATCH 1/1] : hwmon – new chip driver for TI ADS7828 A–D

```
diff –uprN –X linux–2.6.24–rc6/Documentation/dontdiff linux–2.6.24–rc6/drivers/hwmon/Makefile
linux–2.6.24–rc6–ads7828/drivers/hwmon/Makefile
--- linux–2.6.24–rc6/drivers/hwmon/Makefile 2008–01–02 16:37:43.000000000 +0000
+++ linux–2.6.24–rc6–ads7828/drivers/hwmon/Makefile 2008–01–02 16:35:27.000000000 +0000
@@ –22,6 +22,7 @@ obj–$(CONFIG_SENSORS_ADM1026) += adm1026
obj–$(CONFIG_SENSORS_ADM1029) += adm1029.o
obj–$(CONFIG_SENSORS_ADM1031) += adm1031.o
obj–$(CONFIG_SENSORS_ADM9240) += adm9240.o
+obj–$(CONFIG_SENSORS_ADS7828) += ads7828.o
obj–$(CONFIG_SENSORS_ADT7470) += adt7470.o
obj–$(CONFIG_SENSORS_APPLESMC) += applesmc.o
obj–$(CONFIG_SENSORS_AMS) += ams/
diff –uprN –X linux–2.6.24–rc6/Documentation/dontdiff linux–2.6.24–rc6/drivers/hwmon/ads7828.c
linux–2.6.24–rc6–ads7828/drivers/hwmon/ads7828.c
--- linux–2.6.24–rc6/drivers/hwmon/ads7828.c 1970–01–01 01:00:00.000000000 +0100
+++ linux–2.6.24–rc6–ads7828/drivers/hwmon/ads7828.c 2008–01–02 16:35:27.000000000 +0000
@@ –0,0 +1,296 @@
+/*
+ ads7828.c – lm_sensors driver for ads7828 12bit 8ch ADC
+ (C) 2007 EADS Astrium
+
+ This driver is based on the lm75 and other lm_sensors/hwmon drivers
+
+ Written by Steve Hardy <steve@xxxxxxxxxxxxxxxxxxxxxx>
+
+ Datasheet available at : http://focus.ti.com/lit/ds/symlink/ads7828.pdf
+
+ This program is free software; you can redistribute it and/or modify
+ it under the terms of the GNU General Public License as published by
+ the Free Software Foundation; either version 2 of the License, or
+ (at your option) any later version.
+
+ This program is distributed in the hope that it will be useful,
+ but WITHOUT ANY WARRANTY; without even the implied warranty of
+ MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
+ GNU General Public License for more details.
+
+ You should have received a copy of the GNU General Public License
+ along with this program; if not, write to the Free Software
+ Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
+*/
+
+#include <linux/module.h>
+#include <linux/init.h>
+#include <linux/slab.h>
+#include <linux/jiffies.h>
+#include <linux/i2c.h>
+#include <linux/hwmon.h>
+#include <linux/hwmon–sysfs.h>
+#include <linux/err.h>
+#include <linux/mutex.h>
```

```

+
+/* The ADS7828 registers */
+#define ADS7828_NCH 8 /* 8 channels of 12bit A–D supported */
+#define ADS7828_CMD_SD_SE 0x80 /* Single ended inputs */
+#define ADS7828_CMD_SD_DIFF 0x00 /* Differential inputs */
+#define ADS7828_CMD_PD0 0x0 /* Power Down between A–D conversions */
+#define ADS7828_CMD_PD1 0x04 /* Internal ref OFF && A–D ON */
+#define ADS7828_CMD_PD2 0x08 /* Internal ref ON && A–D OFF */
+#define ADS7828_CMD_PD3 0x0C /* Internal ref ON && A–D ON */
+#define ADS7828_INT_VREF_MV 2500 /* Internal vref is 2.5v, 2500mV */
+
+
+/* Addresses to scan */
+static unsigned short normal_i2c[] = { 0x48, 0x49, 0x4a, 0x4b,
+ I2C_CLIENT_END };
+
+
+/* Insmod parameters */
+I2C_CLIENT_INSMOD_1(ads7828);
+
+
+/* Other module parameters */
+static int se_input = 1; /* Default is SE, 0 == diff */
+static int int_vref = 1; /* Default is internal ref ON */
+static int vref_mv = ADS7828_INT_VREF_MV; /* set if vref != 2.5v */
+module_param(se_input, bool, S_IRUGO);
+module_param(int_vref, bool, S_IRUGO);
+module_param(vref_mv, int, S_IRUGO);
+
+
+/* Global Variables */
+static u8 ads7828_cmd_byte; /* cmd byte without channel bits */
+static unsigned int ads7828_lsb_resol; /* resolution of the ADC sample lsb */
+
+
+/* Each client has this additional data */
+struct ads7828_data {
+ struct i2c_client client;
+ struct device *hwmon_dev;
+ struct mutex update_lock; /* mutex protect updates */
+ char valid; /* !=0 if following fields are valid */
+ unsigned long last_updated; /* In jiffies */
+ u16 adc_input[ADS7828_NCH]; /* ADS7828_NCH 12bit samples */
+};
+
+
+/* Function declaration – necessary due to function dependencies */
+static int ads7828_detect(struct i2c_adapter *adapter, int address, int kind);
+
+
+/* The ADS7828 returns the 12bit sample in two bytes,
+ these are read as a word then byte–swapped */
+static u16 ads7828_read_value(struct i2c_client *client, u8 reg)
+{
+ return swab16(i2c_smbus_read_word_data(client, reg));
+}
+
+
+static inline u8 channel_cmd_byte(int ch)

```



```

+in_reg(3);
+in_reg(4);
+in_reg(5);
+in_reg(6);
+in_reg(7);
+
+static struct attribute *ads7828_attributes[] = {
+ &sensor_dev_attr_in0_input.dev_attr.attr,
+ &sensor_dev_attr_in1_input.dev_attr.attr,
+ &sensor_dev_attr_in2_input.dev_attr.attr,
+ &sensor_dev_attr_in3_input.dev_attr.attr,
+ &sensor_dev_attr_in4_input.dev_attr.attr,
+ &sensor_dev_attr_in5_input.dev_attr.attr,
+ &sensor_dev_attr_in6_input.dev_attr.attr,
+ &sensor_dev_attr_in7_input.dev_attr.attr,
+ NULL
+};
+
+static const struct attribute_group ads7828_group = {
+ .attrs = ads7828_attributes,
+};
+
+static int ads7828_attach_adapter(struct i2c_adapter *adapter)
+{
+ if (!(adapter->class & I2C_CLASS_HWMON))
+ return 0;
+ return i2c_probe(adapter, &addr_data, ads7828_detect);
+}
+
+static int ads7828_detach_client(struct i2c_client *client)
+{
+ struct ads7828_data *data = i2c_get_clientdata(client);
+ hwmon_device_unregister(data->hwmon_dev);
+ sysfs_remove_group(&client->dev.kobj, &ads7828_group);
+ i2c_detach_client(client);
+ kfree(i2c_get_clientdata(client));
+ return 0;
+}
+
+/* This is the driver that will be inserted */
+static struct i2c_driver ads7828_driver = {
+ .driver = {
+ .name = "ads7828",
+ },
+ .attach_adapter = ads7828_attach_adapter,
+ .detach_client = ads7828_detach_client,
+};
+
+/* This function is called by i2c_detect */
+static int ads7828_detect(struct i2c_adapter *adapter, int address, int kind)
+{

```

```

+ struct i2c_client *client;
+ struct ads7828_data *data;
+ int err=0;
+ const char *name = "";
+
+ /* Check we have a valid client */
+ if (!i2c_check_functionality(adapter, I2C_FUNC_SMBUS_READ_WORD_DATA))
+ goto exit;
+
+ /* OK. For now, we presume we have a valid client. We now create the
+ client structure, even though we cannot fill it completely yet.
+ But it allows us to access ads7828_read_value. */
+ data = kzalloc(sizeof(struct ads7828_data), GFP_KERNEL);
+ if (!data) {
+ err = -ENOMEM;
+ goto exit;
+ }
+
+ client = &data->client;
+ i2c_set_clientdata(client, data);
+ client->addr = address;
+ client->adapter = adapter;
+ client->driver = &ads7828_driver;
+
+ /* Now, we do the remaining detection. There is no identification
+ dedicated register so attempt to sanity check using knowledge of the chip
+ – Read from the 8 channel addresses
+ – Check the top 4 bits of each result are not set (12 data bits)
+ */
+ if (kind < 0) {
+ int ch;
+ for (ch = 0; ch < ADS7828_NCH; ch++) {
+ u16 in_data;
+ u8 cmd = channel_cmd_byte(ch);
+ in_data = ads7828_read_value(client, cmd);
+ if (in_data & 0xF000) {
+ printk(KERN_DEBUG
+ "%s : Doesn't look like an ads7828 device\n",
+ __FUNCTION__);
+ goto exit_free;
+ }
+ }
+ }
+
+ /* Determine the chip type – only one kind supported! */
+ if (kind <= 0)
+ kind = ads7828;
+
+ if (kind == ads7828)
+ name = "ads7828";
+
+

```

```

+ /* Fill in the remaining client fields, put it into the global list */
+ strncpy(client->name, name, I2C_NAME_SIZE);
+
+ mutex_init(&data->update_lock);
+
+ /* Tell the I2C layer a new client has arrived */
+ err = i2c_attach_client(client);
+ if (err)
+ goto exit_free;
+
+ /* Register sysfs hooks */
+ err = sysfs_create_group(&client->dev.kobj, &ads7828_group);
+ if (err)
+ goto exit_detach;
+
+ data->hwmon_dev = hwmon_device_register(&client->dev);
+ if (IS_ERR(data->hwmon_dev)) {
+ err = PTR_ERR(data->hwmon_dev);
+ goto exit_remove;
+ }
+
+ return 0;
+
+exit_remove:
+ sysfs_remove_group(&client->dev.kobj, &ads7828_group);
+exit_detach:
+ i2c_detach_client(client);
+exit_free:
+ kfree(data);
+exit:
+ return err;
+}
+
+static int __init sensors_ads7828_init(void)
+{
+ /* Initialise the command byte according to module parameters */
+ ads7828_cmd_byte = se_input ?
+ ADS7828_CMD_SD_SE : ADS7828_CMD_SD_DIFF;
+ ads7828_cmd_byte |= int_vref ?
+ ADS7828_CMD_PD3 : ADS7828_CMD_PD1;
+
+ /* Calculate the LSB resolution */
+ ads7828_lsb_resol = (vref_mv*1000)/4096;
+
+ return i2c_add_driver(&ads7828_driver);
+}
+
+static void __exit sensors_ads7828_exit(void)
+{
+ i2c_del_driver(&ads7828_driver);
+}

```

Re: [PATCH 1/1] : hwmon – new chip driver for TI ADS7828 A–D

```
+  
+MODULE_AUTHOR("Steve Hardy <steve@xxxxxxxxxxxxxxxxxxxx>");  
+MODULE_DESCRIPTION("ADS7828 driver");  
+MODULE_LICENSE("GPL");  
+  
+module_init(sensors_ads7828_init);  
+module_exit(sensors_ads7828_exit);
```

--
To unsubscribe from this list: send the line "unsubscribe linux–kernel" in
the body of a message to majordomo@xxxxxxxxxxxxxxxx
More majordomo info at <http://vger.kernel.org/majordomo–info.html>
Please read the FAQ at <http://www.tux.org/lkml/>