Technische Informatik IV: Praktikum

Protokoll zu Aufgabe A17

Alexander Steen , Max Wisniewski

Vorbereitung

Als Vorbereitung haben wir uns mit der Doku von SMB380, sowie den Rahmenwerkdateien *smb380.c* bzw. *smb380.h* beschäftigt.

Aufgaben

1. Messbereich des Sensors auf $\pm$ *2g* stellen
2. Im Sekundentakt die Beschleunigung aller drei Achsen ausgeben
3. LEDs je nach Neigung (de-)aktivieren
4. Bei einer Grenzüberschreitung alle LEDs aktivieren

Dokumentation

irq_1_enable Meldet den Interrupt für den Beschleunigungssensor an bzw. ab.

smb380_acceleration Diese Funktion speichert die aktuellen Beschleunigungswerte in den übergebenen Referenzparameter. Der Aufruf passiert wie folgt:
`smb380_acceleration(smb380_data* data)`
Dabei ist `smb380_data` ein Struct, in dem drei Integer-Variablen stehen, die mit `data->x_axis`, `data->y_axis` bzw. `data->z_axis` angesprochen werden können.

SMB380_SET Ändert Einstellungen für die Erkennung von Erschütterungen am SMB380-Modul. Die Funktion nimmt drei Argumente, wobei die ersten zwei die Einstellung charakterisieren und der dritte Paramter den neuen Wert der Einstellung darstellt.

smb380_subscribe Registriert einen Callback-Handler, bei bei einem Interrupt des SMB380-Moduls ausgelöst wird.

Durchführung

Damit der Beschleunigungssensor des Moduls aktiviert ist, haben wir in der config.h die Flag `INIT_SMB380` auf 1 gesetzt und ein Clean-Build ausgeführt.

Zuerst haben wir eine globale Variable für einen Timer angelegt. Dieser wird im Laufe des Programms sekundenweise die Beschleunigungsdaten ausgeben. Außerdem haben erstellen wir uns einen Zeiger auf ein `smb_380_data`-Struct namens data_buffer, in dem dann die Beschleunigungswerte gespeichert werden. Das anlegen des Structs geschiet in der `main`-Funktion.

```
adl_tmr_t *timer_n;
smb380_data *data_buffer;
```

Der Timer-Handler ist ziemlich simpel: Wir lesen mit der Funktion `smb_380_acceleration` die Beschleunigungswerte nach `data_buffer` (siehe Dokumentation oben) und bauen damit einen Ausgabestring zusammen. Dieser wird dann auf der Konsole ausgegeben.

```
void timer_handler_n(u8 event, void *context) {
    // Auslesen der Daten und schreiben nach data_buffer
    smb380_acceleration(data_buffer);
    // Ausgabe zusammenbauen:
    ascii output_buffer[128];
    wm_sprintf(output_buffer, "x:%i\r\ny:%i\r\nz:%i\r\n\r\n",
        data_buffer->x_axis, data_buffer->y_axis, data_buffer->z_axis);

    adl_atSendResponse(ADL_AT_RSP, output_buffer);
}
```

In der main-Funktion erstellen wir uns zunächst einen ascii-Buffer der Größe sechs und casten diesen auf den smb380_data-Typ. So haben wir Speicherplatz auf dem Stack erzeugt und können die Messdaten speichern. Die sechs Byte ergeben sich aus drei zwei-Byte-Einträgen (integer) im Struct.

Nun setzen wir den Messbereich via SMB_380_SET auf 4g-Interval ($\pm$ 2g) und melden unseren timer wie gewohnt an.

```
void main_task(void) {
    // Hier erstellen wir uns Speicher auf dem Stack -
    // und zwar 6 Byte, genau wie viel der Struct braucht.
    ascii buffer[6];
    data_buffer = (smb380_data*) buffer;

    // Messbereich auf 4g-Interval (+/-2g) schalten.
    SMB380_SET (SMB380_C2, SMB380_C2_RANGE, 0x04);
    // Timer anmelden
    timer_n = adl_tmrSubscribe(
        TRUE,
        10, ADL_TMR_TYPE_100MS,
        timer_handler_n);
}
```

Testlauf:

Wie wir sehen, werden periodisch die Beschleunigungswerte der drei Achsen ausgegeben.

```
+GSM: Anmeldung im Netz abgeschlossen
x:0
y:-19
z:252
x:38
y:-19
z:251
...
```

Nun erweitern wir die Funktionalität so, dass wir je nach Neigung bestimmte LEDs ein- bzw. ausschalten.

Durch Testen konnten wir beobachten, dass die Beschleunigungsdaten relativ stark schwanken - selbst, wenn wir das Modul gar nicht bewegen. Also haben wir uns eine Art Hysterese bestimmt, bei denen wir von einer Neigung ausgehen (durch Testen). Für uns ergeben sich folgende Daten:

```
s16 x_Ruhe = 40; s16 y_Ruhe = -18;
s16 x_Hyst = 50; s16 y_Hyst = 20;
```

Dann passen wir unseren Timer einfach an, dass er nach dem Ausgeben der Daten zusätzlich die passenden LEDs an- bzw. ausschaltet. Dabei schauen wir einfach, ob die aktuelle Lage die bestimmte Ruhelage plus Hysterese über- bzw. unterschreitet.

```
void timer_handler_n(u8 event, void *context)   {
    // Auslesen der Daten und schreiben nach data_buffer
    smb380_acceleration(data_buffer);
```

```
    // Ausgabe zusammenbauen:
    ascii output_buffer[128];
    wm_sprintf(output_buffer, "x:%i\r\ny:%i\r\nz:%i\r\n\r\n",
        data_buffer->x_axis, data_buffer->y_axis, data_buffer->z_axis);

    adl_atSendResponse(ADL_AT_RSP, output_buffer);

    // x-Neigung bestimmen
    if (x_Ruhe + x_Hyst <= data_buffer->x_axis) {
        led_on(1); led_off(2);
    } else if (x_Ruhe - x_Hyst >= data_buffer->x_axis) {
        led_on(2); led_off(1);
    } else {
        led_off(1); led_off(2);
    }
    // y-Neigung bestimmen
    if (y_Ruhe + y_Hyst <= data_buffer->y_axis)         {
        led_on(0); led_off(3);
    } else if (y_Ruhe - y_Hyst >= data_buffer->y_axis) {
        led_on(3); led_off(0);
    } else {
        led_off(0); led_off(3);
    }
}
```

Um bei einer Überschreitung der Grenzwerte alle LEDs anzuschalten, legen wir uns einen weiteren Handler an, der in diesem Fall einfach ausgeführt wird:

```
void accHandler(smb380_data *data) {
    led_on(0); led_on(1);
    led_on(2); led_on(3);
}
```

Nun passen wir unsere `main`-Funktion an: Jetzt schalten wir zusätzlich den Interrupt bei Grenzüberschreitung an und registrieren den `accHandler`.

```
void main_task(void) {
    // Hier erstellen wir uns Speicher auf dem Stack -
    // und zwar 6 Byte, genau wie viel der Struct braucht.
    ascii buffer[6];
    data_buffer = (smb380_data*) buffer;

    // Messbereich auf 4g-Interval (+/-2g) schalten.
    SMB380_SET (SMB380_C2, SMB380_C2_RANGE, 0x04);
    // Timer anmelden
    timer_n = adl_tmrSubscribe(
        TRUE,
        10, ADL_TMR_TYPE_100MS,
        timer_handler_n);

    // Interrupt anschalten und Handler registrieren
    irq_1_enable(TRUE);
    smb380_subscribe(accHandler);
}
```

Testlauf:

Beim Neigen hat das Modul (mit gewisser Ungenauigkeit) die LEDs geschaltet und bei etwas rütteln alle LEDs eingeschaltet. Es scheint also alles zu funktionieren.