FACULTAD DE INGENIERIA

Universidad de Buenos Aires

Presentación IMD

Maestría en Sistemas Embebidos

Device driver para Sensor de humedad y temperatura HTU21D

Autor: Esp. Ing. Marcelo Daniel Pistarelli

HTU21D

- Medición de humedad y temperatura
  - ✓ Interfaz I2C
  - ✓ Resolución humedad configurable 8 a 12 bits
  - ✓ Resolución temperatura configurable 11 a 14 bits
  - ✓ Salida de datos fija en 16 bits
  - ✓ Implementa detección de errores por CRC
  - ✓ Se implementó la verificación CRC en el driver

HTU21D

Control

Command	Code	Comment
Trigger Temperature Measurement	0xE3	Hold master
Trigger Humidity Measurement	0xE5	Hold master
Trigger Temperature Measurement	0xF3	No Hold master
Trigger Humidity Measurement	0xF5	No Hold master
Write user register	0xE6	
Read user register	0xE7	
Soft Reset	0xFE	

HTU21D

Configuración

<table>
<tr><th>Bit</th><th>#Bits</th><th>Description/Coding</th><th>Default</th></tr>
<tr><td>7,0</td><td>2</td><td>Measurement resolution
<table>
<tr><th>Bit 7</th><th>Bit 0</th><th>RH</th><th>Temp</th></tr>
<tr><td>0</td><td>0</td><td>12 bits</td><td>14 bits</td></tr>
<tr><td>0</td><td>1</td><td>8 bits</td><td>12 bits</td></tr>
<tr><td>1</td><td>0</td><td>10 bits</td><td>13 bits</td></tr>
<tr><td>1</td><td>1</td><td>11 bits</td><td>11 bits</td></tr>
</table></td><td>'00'</td></tr>
<tr><td>6</td><td>1</td><td>Status: End of Battery(1)
'0': VDD>2.25V
'1': VDD<2.25V</td><td>'0'</td></tr>
<tr><td>3, 4, 5</td><td>3</td><td>Reserved</td><td>'0'</td></tr>
<tr><td>2</td><td>1</td><td>Enable on-chip heater</td><td>'0'</td></tr>
<tr><td>1</td><td>1</td><td>Disable OTP reload</td><td>'1'</td></tr>
</table>

HTU21D

! Medición de humedad demora hasta 16ms

! Medición de temperatura demora hasta 50ms

Retardos dentro del device driver

ATOMIC CONTEXT:

ndelay(unsigned long nsecs)
udelay(unsigned long usecs)
mdelay(unsigned long msecs)

udelay is the generally preferred API; ndelay-level precision may not actually exist on many non-PC devices.

mdelay is macro wrapper around udelay, to account for possible overflow when passing large arguments to udelay. In general, use of mdelay is discouraged and code should be refactored to allow for the use of msleep.

Retardos dentro del device driver

NON-ATOMIC CONTEXT:

SLEEPING FOR "A FEW" USECS (< ~10us?):

* Use udelay

- Why not usleep?

  On slower systems, (embedded, OR perhaps a speed-stepped PC!) the overhead of setting up the hrtimers for usleep *may* not be worth it. Such an evaluation will obviously depend on your specific situation, but it is something to be aware of.

Retardos dentro del device driver

SLEEPING FOR ~USECS OR SMALL MSECS (10us - 20ms):

* Use usleep_range

- Why not msleep for (1ms - 20ms)?

  Explained originally here:

  http://lkml.org/lkml/2007/8/3/250

  msleep(1~20) may not do what the caller intends, and will often sleep longer (~20 ms actual sleep for any value given in the 1~20ms range). In many cases this is not the desired behavior.

Retardos dentro del device driver

SLEEPING FOR LARGER MSECS (10ms+)

* Use msleep or possibly msleep_interruptible

- What's the difference?

  msleep sets the current task to TASK_UNINTERRUPTIBLE whereas msleep_interruptible sets the current task to TASK_INTERRUPTIBLE before scheduling the sleep. In short, the difference is whether the sleep can be ended early by a signal. In general, just use msleep unless you know you have a need for the interruptible variant.

Prueba