I2C Foo

Maintainer:
tux

Status
Solved - kind of

Problem

RaspberryPi 2B und Attiny verstehen sich nicht per I2C.

Effekt: Der RPi erhält die Antwort auf die vorherige Anfrage. Es handelt sich nicht um ein Timing-Problem, das Signal sieht gut aus und die Übertragung ist auf dem PHY sauber. Nur die Inhalte stimmen nicht.

Lösung

- https://github.com/netz39/space_notification/pull/12
- Repeated Starts im I2C Frame führen dazu, dass die Bibliothek die Schwelle zwischen Schreiben zum Slave und Lesen vom Slave nicht mehr erkennt und den Callback zu spät ausführt, sodass die Puffer-Daten aus dem jeweils letzten Request zurückgelesen werden.
- Der aktuelle Workaround ist, nach dem Empfang des ersten Bytes mit der Verarbeitung zu beginnen.
- Der Blick auf das Direction Bit könnte ein generischer Marker sein. Diese Bibliothek nutzt aber in ihrer Implementierung die Zeit, die der Master mit dem Senden der Adresse für ein Read auf dem Bus verbringt, für die Berechnung. Diese Zeit fehlt dann und wahrscheinlich ist der µC damit nicht mehr schnell genug. Clock-Stretching wird nicht unterstützt.
- Optimal wäre, die Repeated Starts abzuschalten, dazu habe ich aber keinen Switch gefunden.
- PRs
  - https://github.com/netz39/rollladensteuerung/pull/20
  - https://github.com/netz39/space_notification/pull/12

Logbuch des Wahnsinns

- RaspberryPi kann kein Clock Stretching
- Problem tritt auf RPi 2B, aber nicht auf RPi A auf
- Abstand zwischen den Abfragen ist nicht relevant, es können mehrere Minuten dazwischen liegen
- Start und Stop Condition werden vom Oszi nicht dekodiert (mit anderen Transfers gegenchecken)

2021-11-06

- Offenbar sendet der RPi2 Repeated-Start-Conditions, die den AVR dazu veranlassen, sofort den Puffer zurückzugeben, ohne die Interrupt-Routine aufzurufen.
- Workaround: i2cget und i2cset trennen
- Besser wäre es, wenn der Kernel das schon auseinander nehmen könnte
- https://github.com/torvalds/linux/blob/master/drivers/i2c/algos/i2c-algo-bit.c#L552 → Unterscheidung Start/Stop oder Repeated Start
- https://github.com/thenaran/linux-rpi/blob/master/drivers/i2c/busses/i2c-bcm2708.c#L77
    - Es gibt den Parameter "combined", über den festgelegt wird, ob Nachrichten über Repeated-Start zusammengefasst werden
- Für den relevanten Bus ist bcm-2835 zuständig … oder auch nicht. Es ist bcm2708
- Offenbar kommt das repeated-start aus dem Userspace
- Benutzt die USITWI-Library für den AVR die Stop-Condition, um das callback aufzurufen?
    - Der Callback wird nicht in der ISR aufgerufen (was eig. sinnvoll ist)
    - Ablauf wäre dann so:
        - Master sendet Adresse und Parameter im ersten Block
        - Parameter landet im Puffer
        - Stop-Condition führt dazu, dass Callback (mit Parameter aus Puffer) ausgeführt wird
        - Während der Master das Adress-Byte für den Read-Request schickt (USI ist Hardware-basiert, unterbricht also den Programmablauf nicht), führt der Prozessor den Callback auf dem Parameter aus dem Eingangspuffer aus und schreibt das Ergebnis in den Ausgangspuffer.
        - Wenn der Master die das Adress-Byte geschrieben hat, liegt das Ergebnis zum Senden im Ausgangspuffer
    - Beim Repeated Start fehlt die Stop-Condition
        - Callback wird nicht nach Empfang des Eingangs-Parameters (erster Frame) aufgerufen
        - d.g. beim Antworten liegen noch die Daten des vorherigen Requests im Ausgangspuffer
        - Callback wird _nach_ dem gesamten Request/Response aufgerufen
        - d.h. bei der nächsten Anfrage liegen dann die Daten der letzten Anfrage im Puffer und werden zurückgegeben

Lösung könnte sein:

- 1)
    - Einen anderen Trigger verwenden (es gibt eine interne State Machine)
- 2)
    - I2C-Zugriff zentralisieren, wie das schon mal geplant war.
    - Bei der Gelegenheit könnte auch gleich der ganze I3C-Teil umgesetzt werden.
    - Read/Write trennen, sodass der AVR auch definitiv genug Zeit für die Verarbeitung hat

Linksammlung

- https://community.atmel.com/forum/e70-twi-master-read-communication-problem
- https://elinux.org/BCM2835_datasheet_errata#p35_I2C_clock_stretching
- http://ww1.microchip.com/downloads/en/AppNotes/doc8380.pdf
- https://hackaday.com/2016/07/19/what-could-go-wrong-i2c-edition/
- Attiny Datasheet
    - ab Seite 117 bzw. Seite 121
- Software emulated I2C for Raspberry Pi
    - nur 8 bit built-in, den Rest muss man emulieren
    - kann Clock Stretching