Intro to Microcontrollers

I2C on AVR

November 11, 2008

Outline

Outline

Resources

- http://en.wikipedia.org/wiki/I2c
- http://www.atmel.ru/Disks/AVR
- "I2C" copyright Phillips.
  It's called TWI (two-wire interface) in AVR docs
- Discussion in the Mega48P datasheet starts on p. 214

What Is I2C?

- Synchronous serial protocol: clock and data lines
- Bus-master: one "master" sets the clock at a time
- All other devices are slaved to that clock,
  slaves only speak when spoken to
- When current master is done, next master can set clock
- Packets have protocol: address packets and data packets
- Can have 128 devices on the same two wires
- Reasonably fast: 100kHz and 400kHz specs.

Electrical implementation

- Two lines: clock (SCL) and data (SDA)
- Pullup resistors on the two lines: normally-high.
- Devices either tri-state (allowing line high)
  or pull the line low
- Data is read off during the high parts of the clock
- Data must be stable during the clock high, because...

Bus Protocol Stuff

Start and Stop

- Start is signalled by a falling voltage change *during* a clock
- Stop is rising voltage *during* a clock
- (Remember: falling is pulled low, rising is tri-state)

Packets

- Address packet: Seven address bits, data direction bit (read or write), acknowledge bit
- Data packets: Eight data bits, one acknowledge bit
- Ack bit: Pulled low for ACK, left floating high for NACK.

Modes

Four modes: Master-Slave & Transmit-Receive

- *Master Transmitter*: write to a memory device
  master sends start, transmitter sends address + Write
- *Master Receiver*: read back from the memory, poll sensors
  master sends start, receiver sends address + Read
- *Slave Transmitter*: AVR acting as the sensor, relaying info
  wait for address + Read, then transmit
- *Slave Receiver*: take commands from another chip? I2C-SD bridge?
  wait for address + Write, then listen and process
- Devices can/do change modes frequently

Multi-Mastering and Arbitration

Avoiding data collisions

- Since anyone can act as the master, need arbitration
- Line can be pulled down by anyone: it's like an AND operation
- Masters listen for someone else pulling the data line down when they're transmitting. If they hear anything, they stop and let the other master talk through.
- (Suggests a (completely trivial) hardware DoS attack on I2C busses?)
- AVR implements arbitration with a collision register flag.
- I've never used more than one master, so it's all new to me.

Outline

Hardware Resources

- On Mega48/88/168, lots of stuff done in hardware
- Clock, bit-rate synchronizer (TWBR register)
- Arbitration detection (TWWC flag)
- Start/stop detection (can wake chip up, throw interrupts)
- Address matching (TWAR register)
- Shifting the bits into a nice 8-bit data register (TWDR)
- What's left for you to do? All of the protocol stuff.

The Registers

- Pullups: Can cheat and use the internal pullups on the SDA/SCL pins
- Bit rate: When master, set how fast want to talk in TWBR along with pre-scaler in TWSR (status register)
- TWCR (Control register): TWIE (interrupt enable), TWEN (enable), TWWC (write collision), TWSTO (stop), TWSTA (start), TWEA (enable acknowledge), TWINT (interrupt)
- TWSR (Status register): prescaler bits live here, but mostly it's status bits. Check these periodically during communication.
- TWDR (Data register): data sent/received goes here
- TWAR (Address register): Set your address here if you're a slave.

Mastering

Transmit

- Set TWEN, TWSTA, TWINT
- When AVR clears TWINT, write address + W to TWDR (data register) and clear TWSTA, set TWINT again
- On next TWINT, put data in TWDR, set TWINT
- Continue until done, when set TWSTO
- Check status registers along the way for arbitration

Mastering

Receive

- As before: Set TWEN, TWSTA, TWINT, and wait for TWINT
- Then transmit address + R
- Read data out of TWDR every time TWINT cleared
- After last byte, instead of regular ACK, leave line high
- Set TWSTO to signal stop
- Status registers for arbitration

Slaving

Transmit

- Set address in TWAR, TWEN, TWEA (enable ack).
- TWSTA, TWSTO cleared
- When AVR hears address + R, it'll enter transmit mode
- Put bytes into TWDR, set TWINT and TWEA to acknowledge

Slaving

Receive

- As above, up to address + W
- Read out of TWDR when TWINT
- Read TWSR to see what to do
- Read data out of TWDR
- Set TWINT and TWEA to acknowledge

Outline

AVR-to-AVR Bus

The full Monte

- One AVR set up as Master, does both a transmit and a receive
- Other AVR set up as Slave, receive and transmit
- Master sends x, Slave receives.
- Master asks for a receive, Slave transmits $x + 1$

The End

◂ Outline