Environmental Sample Processor
I^{2}C Bus Messages

3/3/10 Brent Roman   brent@mbari.org

Inter-Integrated Circuit Bus Basics

- Invented by Philips (now called NXP) in the early 1980's
- Two-wire serial bus: Clock, Data, and common Ground (OK, three wires :-)
    - Clock and Data float high via a resistor (or active pullup) for logical '1'
    - Driven low (shorted to GND) to represent a logical '0'
    - Data rate we use is 100kBits/s – quite slow by today's standard's
  - Supports very elegant multi-mastering with transparent collision handling
      - If two nodes “talk” simultaneously:
        - One of the two transmits successfully, the other retries later
        - Better than ethernet, where both nodes must back off and retry.
  - Supports up to 127 unique addresses (or nodes)
    - Extended to support > 1000, but we use the original standard
    - Address 0 is reserved for broadcasting to all attached nodes
    - Addresses are preassigned – there is nothing like DHCP
      - But, our dwarves do verify that they each are assigned unique addresses
  - Intended to be used on large circuit boards or board sets
    - Not intended to be transmitted long distances over cables
    - But, we've tested our cables to 12 feet w/o incident
  - Positive acknowledgement of each byte after it is sent
    - Receivers can “extend the clock” to throttle data rate
  - Basic Standard has no Cyclic Redundancy Code (error) checking
    - We bend the rules by appending an 8-bit CRC to each message
    - And, a non-standard 1-bit CRC Message valid ACK (or Nack)

ESP Dwarf I^{2}C messages

- Messages are strings of 8-bit bytes
  - Raw bytes are typically displayed in hexadecimal (e.g. 0x20 hex == 32 decimal)
- Three basic message types:
  - ***Commands***
    - Unacknowledged beyond confirmation of receipt
      - Typically used only for configuration messages
  - ***Requests***
    - Allow the receiver to “talk back” to the requester
      - To indicate status and/or tell when a long operation completes
    - Requesters do not wait for replies
      - Many unrelated messages are interleaved between a given request and its replies
      - Each request includes a 7-bit (1..127) “reply tag” to facilitate this
  - ***Replies***
    - Always refer to a “reply tag” from a previously sent request
    - A reply tag is recycled a few seconds after the last reply for it is received
      - Otherwise one would run out of unique tags after 126 requests
    - There may be more than one reply message to a given request

  - Any ESP dwarf may send *Commands*, *Requests* and *Replies* to any other
    - But we currently only sent requests to dwarves from the host gateway
    - Dwarves only send replies back to the host gateway

ESP I^{2}C Gateways

- Bridges between the I^{2}C bus a single (fast) RS-232 serial port
  - All operations pass through
  - Gateway has no knowledge of what the bytes “mean”
- Data flows through the gateway without buffering
  - Reduces latency and gateway's memory requirements
- RS-232 serial port normally configured for 115.2 kbaud, 8 data bits + 1 stop bit
  - CTS/RTS “hardware” flow control required to help ensure against lost data
  - Data is binary, not ASCII
    - RS-232 “break” condition signals attention
      - “I^{2}C Gateway not responding to BREAK” error message
        - Means that the host cannot communicate with that gateway
- Gateways are configured from there RS-232 side (by the host)
  - Address of the gateway on the I^{2}C bus (typically 0x20)
  - Number of retries after errors (typically 3)
  - Delay between each retry
  - And many other parameters

- A host may communicate with multiple I^{2}C buses
  - Each via its own gateway on a dedicated RS-232 serial port
  - The ESP-DWSM functions over such an additional, dedicated gateway
    - The main I^{2}C bus is the “core” bus.

Some other ESP I^{2}C Bus Exceptions/Error Messages

- *Unexpected ACK | String | etc.*
  - Low level protocol error, probably due to electrical noise or firmware error
- *Address 0x?? already in use*
  - Two dwarves' dip switches are configured to select the same address
- *I^{2}C Bus Error*
- *Timeout waiting for bus startup*
- *Invalid bus start*
- *Slave NACK …*
- *Master NACK …*
- *Rejected message's CRC*
- *CRC was invalid*
  - Probably an electrical problem on the bus or a misbehaving dwarf
- *NodeOffline*
  - Dwarf is powered off or has been removed from the bus
- *Unknown Command, Query, or Reply*
  - Could be old dwarf firmware being used with new Host Ruby code
- *I^{2}C::Request::Timeout* errors **do not** generally indicate an I^{2}C bus problem
  - More likely, a valve or motor is stuck
  - When a request receives no reply within the expected maximum time
    - The request is said to have “timed-out”
- *Unexpected Reply*
  - Results if the ESP software restarted while dwarves had replies pending
  - ESP host receives the reply, but doesn't “remember” original request

ESP I^{2}C Bus Error Message Format

- @16:02:00.06 I2C::Request::Timeout in simfast --
  - No Response to I2C::Servo::AbsMove3Request[09:->25] during Processing Clamp move

- *I2C::Request::Timeout* => Timeout error on an I^{2}C Request message

- *Simfast* => mode or thread in which error occurred

- *No Response to* … => details about this particular timeout error
  - *I2C::Servo::AbsMove3Request* => Exact message type
  - [*tag*:*source*->*destination*] => addressing information
    - *Tag* => Request tag number
    - *Source* => Sending node (may be omitted for the host)
    - *Destination* => Receiving node (usually a dwarf address)
    - <u>All the fields within [square backets] are in hexadecimal !!!</u>

- *During* … => what operation was being performed when the error occurred

Dwarf DIP Switches

- Each dwarf is functionally identical to every other
- The only difference between them are there 6 DIP switch settings
- The low-order 4 switches determine the dwarf's I^2C network address
  - The address selected is 0x20 + the 4-bit code selected
  - Code 0 (all off) is for debugging only
    - Recall the gateway uses address 0x20 !!
  - 0x25 => processing dwarf
  - 0x26 => manipulator dwarf
  - 0x27 => collection dwarf
  - 0x28 => puck storage dwarf
  - 0x29 => sampler dwarf
  - 0x2A => microfluidic block dwarf
  - 0x2B => 4km DWSM deep sampler dwarf
  - 0x2C => 4km DWSM deep resampler dwarf

- The top two switches select debugging on the dwarf's serial port
  - Both off => no debugging input or output on serial port
    - (required when an instrument is attached to dwarf's serial port)
  - Most significant on => input one-letter debug commands
  - 2nd Most significant on => output debug messages on serial port

- Leave top two switches off unless you are debugging

Dwarf One letter debug commands

- ? => output a very short help message
- i => stop displaying I2C message data
- I => start displaying I2C message data (as binary hexadecimal)
- o => stop displaying detailed servo status for channel 0
- 0 => toggle display of detailed servo status for channel 0
- O => start displaying detailed servo status for channel 0
- l => stop displaying detailed servo status for channel 1
- 1 => toggle display of detailed servo status for channel 1
- L => start displaying detailed servo status for channel 1

- Each letter is acted upon as soon as typed
  - There is no “Enter” key
- Note that o is lower-case O and 0 is the number zero.
- Note that l is lower-case L and 1 is the number one.

- Servo performance may suffer if detailed display of status is enabled for both channels simultaneously

- Debugging data is always input and output at:
  - 115.2kBaud, 8 data bits, 1 stop bit