I2C PASM driver, I2C SPIN driver, I2C slave driver (PASM), lots of demos.

By: Chris Gadd, created: 2013-07-27 | updated: 2015-03-28

Total votes: 3
Request group membership

Includes Spin-based and PASM-based open-drain and push-pull drivers, with methods for reading and writing any number of bytes, words, and longs in big-endian or little-endian format. All drivers tested with the 24LC256 EEPROM, DS1307 real time clock, MMA7455L 3-axis accelerometer, L3G4200D gyroscope module, MS5607 altimeter module, and BMP085 pressure sensor. Includes a demo programs for each.

Also includes a polling routine that displays the device address and name of everything on the I2C bus.

Also includes a slave object that runs in a PASM cog with a Spin handler, and provides 32 byte-sized registers for a master running on another device to write to and read from. Tested okay up to 1Mbps (max speed of my PASM push-pull object). Includes demo for the slave object.

Newly added multi-master object that can share the bus with other masters, provided of course that the other masters also allow sharing.

Items per page
1
Apply

Original File Upload

Attachment	Size
I2C routines - Archive.zip	102.52 KB

Log in or register to post comments

Comments

AKH replied on Fri, 2014-03-28 17:27 PERMALINK

POSSIBLE BUG

I may have found a bug in the I2C slave driver.

In the `detect_st_or_sp` subroutine in "I2C slave v1.0.spin" there are two occurances of the following:

```
         if_z          jmp       detect_st_or_sp_ret
```

I suspect those should actually be:

```
         if_z          jmp       #detect_st_or_sp_ret
```

I may be mistaken but my compiler (BST) did emit warnings on those two lines and the latter version also seems more correct to me.

BTW, I'm really impressed with the work you've done on the drivers. It opens up a bunch of new possibilities for what one can do with the Prop. Thanks!

Log in or register to post comments

ChrisGadd replied on Tue, 2014-07-01 17:00 PERMALINK

RE: POSSIBLE BUG

Ordinarily you'd be right, but with **ret** instructions, the contents are the address so it does the same thing. Only difference is that it saves one clock cycle, at the possible expense of making the code more difficult to read. I learned that one from Phil Pilgrim's Propeller Tricks and Traps. http://forums.parallax.com/attachment.php?attachmentid=49618

Log in or register to post comments

AKH replied on Fri, 2014-07-18 13:32 PERMALINK

INTERESTING

Interesting, I'll take another look at the code and Tricks and Traps.

Log in or register to post comments

dstarkey replied on Tue, 2014-07-15 11:55 PERMALINK

READING WITHOUT ADDRESSING A REGISTER

Chris,

How would I read 2-bytes from an I2C device that doesn't require registers to be addressed.

ie. the TI ADS1000 A/D chip just wants to be read and it sequentially sends the 2-byte converter value and optionally a 3rd byte configuration byte.

When I use the I2C.read(addr,0), it doesn't work because the PASM code is sending the register byte while the ADS1000 is expecting the low byte to be sent.

Am I using your code wrong?

Thanks,

Don

Log in or register to post comments

ChrisGadd replied on Wed, 2014-08-13 09:46 PERMALINK

MANUFACTURERS NEVER CEASE TO

Manufacturers never cease to amaze me with their inventive variations on I2C.

Having examined the datasheet, any byte written to the device will be interpreted as a configuration setting. I would've thought that sending register address 0 would cause it to not start a conversion, remain in continuous conversion mode, with a gain of 1. It should then have read bits 15 through 8 of the output register.

I2C.read_next(addr) only sends the address byte with read/write set, then reads a single byte, so that should give you the top byte. Sending the same command a second time might give you the low byte, though I don't see any mention in the datasheet if this thing supports repeated reads.

If it doesn't support repeated reads, then the only way to get both bytes would be with a page_read. You'll need to make a modification to the PASM section, under

:read page, simply comment out the [b]call #send_start[/b] line. The Send_Start subroutine is responsible for sending the device ID with write bit, then the register address, so this device should work without it. It will require a two or three byte array to read the received bytes into, as read_page doesn't return an immediate value.

I'll have to consider making a non-standard method that writes and reads an arbitrary number of bytes for all of these edge cases.

Log in or register to post comments

dstarkey replied on Fri, 2014-08-15 08:52 PERMALINK

READING WITHOUT ADDRESSING A REGISTER

Chris,

I hacked your PASM code to do what I needed.

I added "I2C.read_noreg(I2C#AD2,2)" ' Read 2 bytes from device, with no register addressing

Seems to work ok.

I put a link to the modified code below. Let me know if you want me to remove it.

Thanks again for great programming.

Don

https://dl.dropboxusercontent.com/u/7183470/I2C%20PASM%20driver%20v1.3a.spin

Log in or register to post comments

geo_leeman replied on Sat, 2014-10-18 13:59 PERMALINK

HTU21D

Chris,

Thank you for this great work! I'm now implementing an HTU21D humidity sensor and can't get it to return (other than abort) on writes. The datasheet is here. Any ideas or thoughts would help a lot!

Thank you,

John

Log in or register to post comments

geo_leeman replied on Tue, 2014-10-21 06:27 PERMALINK

SCRATCH THAT

Chris,

Scratch that... Crummy datasheet, but all is well now. Appreciate your code!

- John

Log in or register to post comments