LabJack Measurement & Automation Simplified

5.2.19 - I²C

Requires U3 hardware version 1.21+. Sends and receives serial data using I²C (I2C) synchronous communication.

Command:		
Byte		
0	Checksum8	
1	0xF8	
2	4 + NumI2CWordsSend	
3	0x3B	
4	Checksum16 (LSB)	
5	Checksum16 (MSB)	
6	I2COptions	
		Bits 7-4: Reserved
		Bit 3: Enable Clock Stretching
		Bit 2: No Stop when restarting
		Bit 1: ResetAtStart
		Bit 0: Reserved
7	SpeedAdjust	
8	SDAPinNum	
9	SCLPinNum	
10	AddressByte	
11	Reserved	
12	NumI2CBytesToSend	
13	NumI2CBytesToReceive	
14	I2CByte0	
...	...	
Response:		
Byte		
0	Checksum8	
1	0xF8	
2	3 + NumI2CWordsReceive	
3	0x3B	
4	Checksum16 (LSB)	
5	Checksum16 (MSB)	
6	Errorcode	
7	Reserved	
8	AckArray0	
9	AckArray1	
10	AckArray2	
11	AckArray3	
12	I2CByte0	
...	...	

- **NumI2CWordsSend:** This is the number of I²C bytes to send divided by 2. If the number of bytes is odd, round up and add an extra zero to the packet. This parameter is actually just to specify the size of this packet, as the NumI2CbytesToSend parameter below actually specifies how many bytes will be sent.
- **I2COptions:** If ResetAtStart is true, an I²C bus reset will be done before communicating.
- **SpeedAdjust:** Allows the communication frequency to be reduced. 0 is the maximum speed of about 150 kHz. 20 is a speed of about 70 kHz. 255 is the minimum speed of about 10 kHz.
- **SDAP/SCLP -PinNum:** Assigns which digital I/O line is used for each I²C line. Value passed is 0-19 corresponding to the normal digital I/O numbers as specified in Section 2.8 (/support/u6/users-guide/2.8) . Note that the screw terminals labeled “SDA” and “SCL” on hardware revision 1.20 or 1.21 are not used for I²C. Note that the I²C bus generally requires pull-up resistors of perhaps 4.7 kΩ from SDA to Vs and SCL to Vs.
- **AddressByte:** This is the first byte of data sent on the I²C bus. The upper 7 bits are the address of the slave chip and bit 0 is the read/write bit. Note that the read/write bit is controlled automatically by the LabJack, and thus bit 0 is ignored.
- **NumI2CBytesToSend:** Specifies how many I²C bytes will be sent (0-50).
- **NumI2CBytesToReceive:** Specifies how many I²C bytes will be read (0-52).
- **I2Cbyte#**: In the command, these are the bytes to send. In the response, these are the bytes read.
- **NumI2CWordsReceive:** This is the number of I²C bytes to receive divided by 2. If the number of bytes is odd, the value is rounded up and an extra zero is added to the packet. This parameter is actually just to specify the size of this packet, as the NumI2CbytesToReceive

parameter above actually specifies how many bytes to read.

- **AckArray#:** Represents a 32-bit value where bits are set if the corresponding I²C write byte was ack'ed. Useful for debugging up to the first 32 write bytes of communication. Bit 0 corresponds to the last data byte, bit 1 corresponds to the second to last data byte, and so on up to the address byte. So if n is the number of data bytes, the ACKs value should be (2^(n+1))-1.

Comments

No comments yet. Speak up. We're listening.