Application Report
SLUA563B–August 2010–Revised June 2011

bq76PL536/536A/536AQ1 Communications Basics

Bob Shoemaker
BMS HCE Automotive

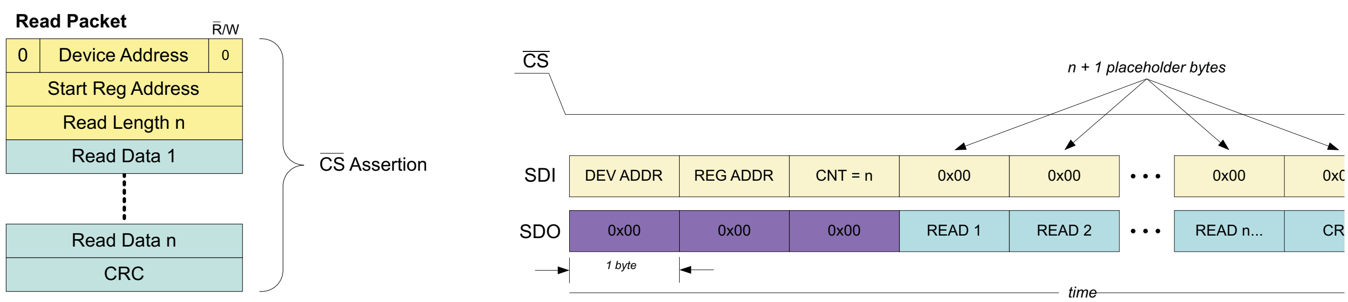
ABSTRACT

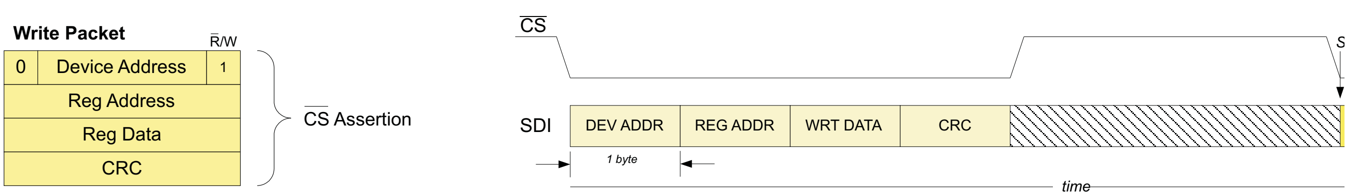
The bq76PL536 can be stacked vertically to monitor up to 192 cells without additional isolation components between the integrated circuits. A high-speed serial peripheral interface (SPI) bus operates between each bq76PL536 to provide reliable communication through a high-voltage battery cell stack. This application report discusses the communication basics of addressing, reading, and writing for the Texas Instruments bq76PL536, a stackable 3-to-6 series battery pack protector and analog front end.

Although only the bq76PL536 is referred to in this document, all information herein applies equally to the bq76PL536A and the bq76PL536AQ1.

1 bq76PL536 Packet Formats

The following illustrations show the Read and Write packet formats of the bq76PL536.





2 bq76PL536 Addressing, Reading, and Writing

2.1 Addressing

When powering up the integrated circuit (IC), the default address is 0. To change it to address 1, write: 01 3B 01 02. Other addresses require that the last two bytes change.

- Write to address 0 = 01
- Write to address control registers = 3B
- Write address to 1 = 01
- Write the CRC = 02

Now the device answers to address 1.

Note the following two things.

1. The length byte is implied because you can only write one byte at a time. No length byte is sent with Write packets.
2. The chip is manufactured with CRC enabled, so you must transmit the CRC. CRCs protect against communications errors.

Subsequent devices must receive unique addresses, usually sequential from 0x01 to 0xnn, where nn < 0x3F.

2.2 Reading Data

The following sequence occurs in reading the cell 1 ADC (the cell_1 registers are 0x03 MSB and 0x04 LSB) from the device at 0x01.

Both can be read in one 2-byte read like this: 02 03 02 00 00 00

- Read (logical) address 1 = 02
- Start read at register 3 = 03
- Read 2 bytes (registers 3 and 4) = 02
- Send 0 to clock in byte 1 (register 3 data) = 00*
- Send 0 to clock in byte 2 (register 4 data) = 00*
- Send 0 to clock in CRC = 00*

  *Padding bytes sent by the host CPU to clock out data bytes from the bq76PL536; the value is unimportant

The LSB in the address field is the R/W bit. The 6-bit logical address field is shifted one position left. So, to address device 1 in this example, a "2" is sent for the read address, and a "3" for the write address.

3 bq76PL536 Sample Data Exchange

The Address is 6 bit + LSB R/W bit:

0 for R
1 for W

Therefore, a read of board 1 Cell voltage is as follows:

SEND	START and LENGTH ADDRESS			+	STUFF BYTES TO CLOCK OUT DATA FROM IC		
SEND	IC Addr	Starting Reg	No. Bytes	+	Clk1 for data	Clk2 for data	Clk3 for CRC
SEND	0x02	0x03	0x02	+	0x0	00x00	0x00

The expected return is as follows: (On a read the device generates/returns the CRC)

RCV (clocked out by)	ADDR + START + LEN			+	(new data) Reg_0x03_result + Reg_0x04_result + IC_supplied_CRC			
RCV (clocked out by)	0x00	0x00	0x00	+	(new data)	0xMSB	0xLSB	0xCRC

You then can calculate the CRC based on the received message to confirm that the transmission was successful:

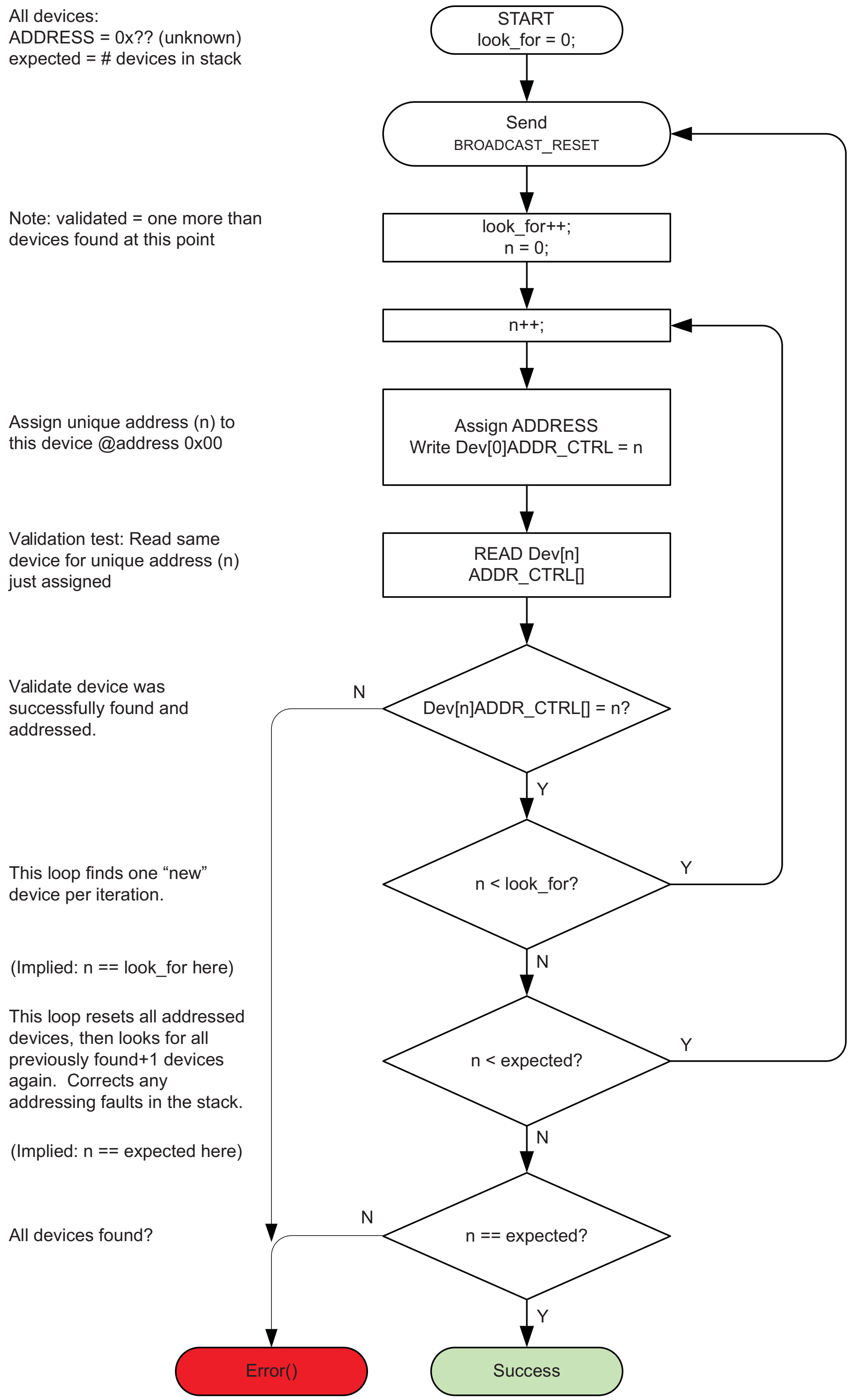
Transaction example:

Data sent to target	**02 03 02** 00 00 00
Data returned from target	00 00 00 **19 9A** 0D
Calculate the CRC	function_CRC (**02 03 02 19 9A**) = 0D

Note that the foregoing example assumes that this device first has been initialized to be address 1.

4 Simple Stack Addressing Algorithm

Each device in the stack must receive a unique address between 0x01 and 0x3E. Address 0x00 is reserved for device discovery and address 0x3F is reserved for BROADCAST messages to all devices. Typical addressing assigns address 1 to the base device, and incremental addresses go up (north) in the stack; 1-2-3-… n.



5 CRC Calculations – SMBUS™ (ATM-8) Algorithm

```
namespace Communications
{
  class CRC8
   {
     private byte[] CrcTable = {
     0x00, 0x07, 0x0E, 0x09, 0x1C, 0x1B, 0x12, 0x15,
     0x38, 0x3F, 0x36, 0x31, 0x24, 0x23, 0x2A, 0x2D,
     0x70, 0x77, 0x7E, 0x79, 0x6C, 0x6B, 0x62, 0x65,
     0x48, 0x4F, 0x46, 0x41, 0x54, 0x53, 0x5A, 0x5D,
     0xE0, 0xE7, 0xEE, 0xE9, 0xFC, 0xFB, 0xF2, 0xF5,
     0xD8, 0xDF, 0xD6, 0xD1, 0xC4, 0xC3, 0xCA, 0xCD,
     0x90, 0x97, 0x9E, 0x99, 0x8C, 0x8B, 0x82, 0x85,
     0xA8, 0xAF, 0xA6, 0xA1, 0xB4, 0xB3, 0xBA, 0xBD,
     0xC7, 0xC0, 0xC9, 0xCE, 0xDB, 0xDC, 0xD5, 0xD2,
     0xFF, 0xF8, 0xF1, 0xF6, 0xE3, 0xE4, 0xED, 0xEA,
     0xB7, 0xB0, 0xB9, 0xBE, 0xAB, 0xAC, 0xA5, 0xA2,
     0x8F, 0x88, 0x81, 0x86, 0x93, 0x94, 0x9D, 0x9A,
     0x27, 0x20, 0x29, 0x2E, 0x3B, 0x3C, 0x35, 0x32,
     0x1F, 0x18, 0x11, 0x16, 0x03, 0x04, 0x0D, 0x0A,
     0x57, 0x50, 0x59, 0x5E, 0x4B, 0x4C, 0x45, 0x42,
     0x6F, 0x68, 0x61, 0x66, 0x73, 0x74, 0x7D, 0x7A,
     0x89, 0x8E, 0x87, 0x80, 0x95, 0x92, 0x9B, 0x9C,
     0xB1, 0xB6, 0xBF, 0xB8, 0xAD, 0xAA, 0xA3, 0xA4,
     0xF9, 0xFE, 0xF7, 0xF0, 0xE5, 0xE2, 0xEB, 0xEC,
     0xC1, 0xC6, 0xCF, 0xC8, 0xDD, 0xDA, 0xD3, 0xD4,
     0x69, 0x6E, 0x67, 0x60, 0x75, 0x72, 0x7B, 0x7C,
     0x51, 0x56, 0x5F, 0x58, 0x4D, 0x4A, 0x43, 0x44,
     0x19, 0x1E, 0x17, 0x10, 0x05, 0x02, 0x0B, 0x0C,
     0x21, 0x26, 0x2F, 0x28, 0x3D, 0x3A, 0x33, 0x34,
     0x4E, 0x49, 0x40, 0x47, 0x52, 0x55, 0x5C, 0x5B,
     0x76, 0x71, 0x78, 0x7F, 0x6A, 0x6D, 0x64, 0x63,
     0x3E, 0x39, 0x30, 0x37, 0x22, 0x25, 0x2C, 0x2B,
     0x06, 0x01, 0x08, 0x0F, 0x1A, 0x1D, 0x14, 0x13,
     0xAE, 0xA9, 0xA0, 0xA7, 0xB2, 0xB5, 0xBC, 0xBB,
     0x96, 0x91, 0x98, 0x9F, 0x8A, 0x8D, 0x84, 0x83,
     0xDE, 0xD9, 0xD0, 0xD7, 0xC2, 0xC5, 0xCC, 0xCB,
     0xE6, 0xE1, 0xE8, 0xEF, 0xFA, 0xFD, 0xF4, 0xF3};

/// <summary>
/// Calculates an 8 bit CRC
/// <summary>
/// <param name="buffer">An array of bytes</param>
/// <returns>A one byte CRC</returns>
public byte Pec(byte[] buffer)
{
  byte crc = 0;
  int temp = 0;
  for (int I = 0; < buffer.Length; I++)
  {
    temp = crc ^ buffer[i];
    crc = CrcTable[temp];
   }
   return crc;
  }
 }
}
// "namespace", "class", and associated braces may be removed for C code implementations

// Copyright © Texas Instruments. All rights reserved.
// THIS CODE AND INFORMATION IS PROVIDED "AS IS" WITHOUT WARRANTY
// OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT
// LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND
// FITNESS FOR A PARTICULAR PURPOSE.
```

CRC:

The CRC calculation implements the method used in the Smart Battery (SBData) over SMBUS – NBPC battery. It is also known as ATM-8. See the SMBUS Specification Sec 5.4.1, 5.4.1.3, at www.smbus.org/specs. For more specific implementation information, see http://smbus.org/faq/faq.htm