Introduction

This document provides a methodology for loading the configuration file into a Zilker Labs Digital-DC™ device via the SMBus for commercial device programmers capable of supporting the I^2C bus protocol.

Configuring the Zilker Labs Controller via the SMBus

The configuration parameters can be stored in a Zilker Labs Digital-DC device using PMBus protocol via the SDA and SCL pins. SDA and SCL are a standard I^2C interface.

Table 1. SMBus Signals

Pin Symbol	I/O	Description
SCL	I/O	SMBus clock pin
SDA	I/O	SMBus data pin

The procedure for configuring a device is described on the following page.

Zilker Labs provides a parser program called ZLHLD, which takes the device configuration file as an input. The output file is in ZLHLD format (Zilker Labs Hex Line Delimited). Figure 3 shows an example of an output file in ZLHLD format.

The pin assignments and configuration circuits for the Zilker Labs Digital-DC devices are shown in Appendix III. All other pins must be connected to ground through a 22 kΩ resistor. For any device, wait 20 ms prior to the start of device programming.

When configuring a device, all standard SMBus protocols must be followed, including clock stretching. Additionally, a 750 µs delay between every SMBus transmission (between every stop & start condition) must occur. Refer to the *System Management Bus (SMBus) Specification*, from SBS Implementers Forum for SMBus electrical and timing requirements.

Configuring Zilker Labs devices

Below is a sequence of steps following from design to production:

1. A configuration file is generated by the design engineer (example – Figure 2).
2. A parser (ZLHLD Generator) supplied by Zilker Labs will convert this configuration file into a Zilker Labs Hex Line Delimited. This parser will calculate and append a Packet Error Checking byte for all write transactions. A checksum of the ZLHLD file is also generated in a separate file for verification by the programmer.
3. A Zilker Labs supplied low volume programmer or an industry standard high volume memory programmer, such as from BP Microsystems, may be used for configuring the devices.
4. The programmers will do the following:
    a. Verify the Device ID:
    All Zilker Labs devices have a unique identifier, which is loaded into the non-volatile memory at final test. The device ID should always be checked prior to start of configuring by the following method:
        i. Read DEVICE_ID (0xE4), a 16-byte identifier. The bytes written before reading are "40E441", with a repeated start performed before byte 0x41 is issued.
        ii. Compare the first 7 bytes of the response to the Device ID under test. These first 7 bytes should be something such as "ZL2005-", "ZL2105-", "ZL2006-", etc.

    b. Insert 6 byte serial number into configuration file. The placement of this command is device dependent, as shown in Appendix I. Upon generating the serial number, calculate the PEC(1) and append it after the serial number. See Appendix II for example code on calculating the PEC. Note that for BPMicro systems, the Zilker Labs external serialization program (ZL_ESP) is used to write the serial number in ASCII at the correct locations.

    c. Load ZLHLD File(2) – Example in Figure 3 via I^2C Bus, using SMBus protocol.
        i. Verification that the data was written to the device is done by checking the CML bit (bit [1]) of the STATUS_WORD register as well as checking the contents of STATUS_CML register to be equal to 0x00.
        ii. After the entire config file is loaded, read STATUS_WORD (0x79), bit [1].
            1. If this bit is set (1), the data transfer failed and the device needs to be rejected.
            2. If this bit is clear (0), we will do a second test to verify the data communication & flash write, as shown in step 3.
            3. Read the STATUS_CML (0x7E) register.
            4. If the single-byte data of STATUS_CML does NOT equal 0x00, the data transfer failed and the device needs to be rejected
            5. If the single-byte data of STATUS_CML equals 0x00, the data transfer has succeeded.

NOTES:

1) When loading the configuration file, always insert a 50 ms wait prior to or after the 000440F10087 string.

2) On DDC-enabled devices such as the ZL2004, ZL2006, & ZL2106, an EVENT command is inserted at the beginning of the file to put the device into a manufacturing mode. This addition is performed automatically by the ZLHLD generator when the appropriate device is selected.

Configuring Zilker Labs devices Flowchart

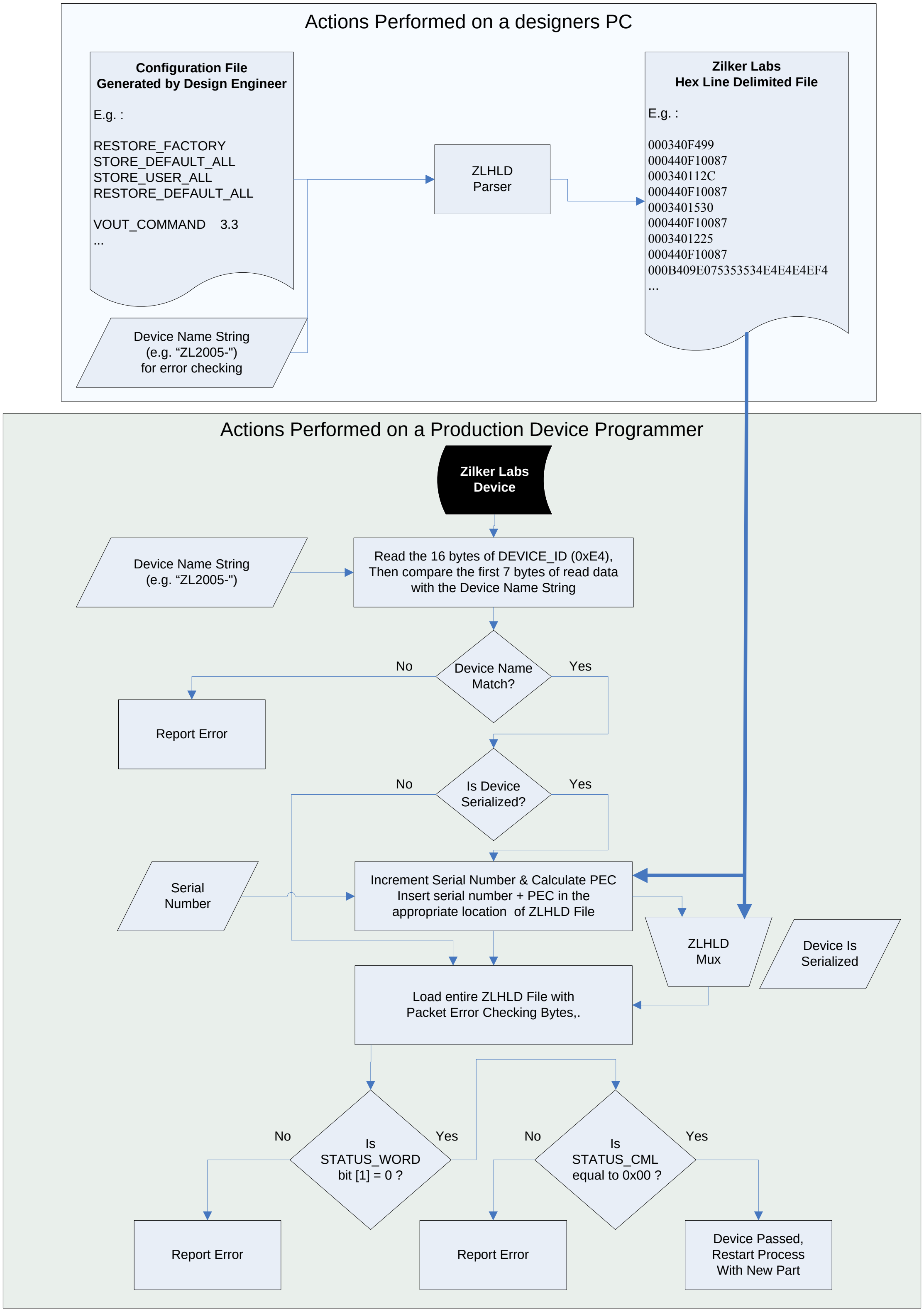

Figure 1. Production Programming Flowchart

Example of ZL2005 Configuration File in Text Format

```
#-----------------------------------------------------------------
# Configuration file for ZL2005EV4   10A design
# The next three lines clear the Default Store & User Store
RESTORE_FACTORY
STORE_DEFAULT_ALL
STORE_USER_ALL
RESTORE_DEFAULT_ALL
MFR_SERIAL                    SSSNNN
MFR_ID                        ZilkerLabs
MFR_MODEL                     ZL2005EV4_Config
MFR_REVISION                  Rev1.3
MFR_LOCATION                  Austin_TX
VOUT_COMMAND                  1.2   #V
FREQUENCY_SWITCH              600   #kHz
POWER_GOOD_DELAY              1   #ms
TON_DELAY                     15   #ms
TON_RISE                      5   #ms
TOFF_DELAY                    15   #ms
TOFF_FALL                     5   #ms
SEQUENCE                      0x0000
MFR_CONFIG                    0xABC1
USER_CONFIG                   0x0000
PID_TAPS                      A=1993.25, B=-3688.31, C=1795.06
IOUT_OC_FAULT_LIMIT           20.   #Amps
IOUT_AVG_OC_FAULT_LIMIT       15.   #Amps
IOUT_UC_FAULT_LIMIT           -10.   #Amps
IOUT_AVG_UC_FAULT_LIMIT       -8.   #Amps
#low FET not enabled for output OV, output OV and UV count to 2
OVUV_CONFIG                   0x01
IOUT_SCALE                    3.5   #mOhms
IOUT_CAL_OFFSET               0
TEMPCO_CONFIG                 0xAC
NLR_CONFIG                    0xA250
# Set Null PRIVATE_PASSWORD (to get correct security level even on used-but-erased part)
PRIVATE_PASSWORD              0x000000000000000000
# Set PRIVATE_PASSWORD
PRIVATE_PASSWORD              ExamplePW
#replace "ExamplePW" nine-character password
# Set UNPROTECT bits
UNPROTECT                     0xFFFFFDFFFDFFF7FCBFF7FFFFCCFFFFFFFFFFFFA1FFFFFFFFFFFF4CEE7EFEEFFF
STORE_DEFAULT_ALL
# User Store Data
# Set PRIVATE_PASSWORD, to change SECURITY_LEVEL to 3, in order to do a RESTORE_FACTORY
PRIVATE_PASSWORD              ExamplePW
RESTORE_FACTORY
# Set PUBLIC_PASSWORD
PUBLIC_PASSWORD               MyPW
#replace "MyPW" with your own four-character password
# Set PRIVATE_PASSWORD to NULL (USER_STORE Private password),
PRIVATE_PASSWORD              0x000000000000000000
# Set PRIVATE_PASSWORD
PRIVATE_PASSWORD              ExampleP2
#replace "ExampleP2" with your own nine-character password
UNPROTECT                     0xFFFFD9FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEFFF
STORE_USER_ALL
-----------------------------------------------------------------
```

Figure 2. Example ZL2005 Configuration Text File

Example of ZL2005 Configuration File in ZLHLD (Zilker Labs Hex Line Delimited)

```
000340F499
000440F10087
000340112C
000440F10087
0003401530
000440F10087
0003401225
000440F10087
000B409E075353534E4E4E4EF4
000E40990A5A696C6B65724C6162732D
0014409A105A4C323030354556345F436F6E66696729
000A409B06526576312E3325
000D409C0941757374696E5F5458B7
000540216626CA
0005403358026D
000540D400BA30
00054060C0D384
0005406180CAFB
00054064C0D32F
0005406580CA50
000540E0000055
000540D0C1AB14
000540D10000DF
000D40D50928F97B85E6FC62E07BE2
0005404680DADE
000540E7C0D399
0005404B80D562
000540E800CC6E
000440D80193
0005403880C3DA
00054039000040
000440DCAC8D
000540D750A2C9
000D40FB09000000000000000000BB
000D40FB094578616D706C6550571A
002440FD20FFFFFDFFFDFFF7FCBFF7FFFFCCFFFFFFFFFFFFA1FFFFFFFFFFF4CEE7EFEEFFFF7B
000340112C
000440F10087
000D40FB094578616D706C6550571A
000340F499
000440F10087
000840FC044D79505788
000D40FB09000000000000000000BB
000D40FB094578616D706C65503226
002440FD20FFFFD9FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEFFFF73
0003401530
000440F10087
```

Figure 3. Example ZL2005 Hex Format Configuration File

Appendix I. Line Locations of Serial Number, by Device Type

Product	Config File Line Location	ZLHLD File Serial Data Byte Location	Comments
ZL2005	5	0x31	
ZL2005-02		0x31	
ZL2005P		0x31	
ZL2105		0x31	
ZL2004		0x37	ZLHLD Byte Location accounts for EVENT command insertion
ZL2004-01		0x37	ZLHLD Byte Location accounts for EVENT command insertion
ZL2006		0x37	ZLHLD Byte Location accounts for EVENT command insertion
ZL2106		0x37	ZLHLD Byte Location accounts for EVENT command insertion

NOTE: 1) The above ZLHLD Serial Data Byte Locations are calculated with the inclusion of device address, 2-byte length preamble, CRC-command delay after STORE/RESTORE commands, and PEC bytes.
2) Config File Line Location does not account for the EVENT command, as it is automatically inserted by the ZLHLD Generator. Accounting for this will increase the Configuration File Line Location by one line.

Appendix II. Example Code For Calculating PEC

```c
/* PEC.c
 * Implements a CRC-8 checksum using the direct method.
 */

/* public global values */
extern unsigned char PEC_CurrentCRC;  //CRC calculation of all bytes called //through
CRC_Process_Byte since last call to CRC_Reset

#define CRCPoly 0x8380 //polynomial of x^8 + x^2 + x^1 + 1 in most significant 9 bits
#define CRCDone 0x0083 //CRC is done after polynomial shifts one byte

/* function prototypes */
void PEC_ProcessByte(UBYTE crcInput);
void PEC_ResetCRC(void);

unsigned char PEC_CurrentCRC;

// CRC_Process_Byte performs a direct-mode calculation of
// one byte along with a previous CRC calculation
void PEC_ProcessByte(unsigned char crcInput)
{
  unsigned short crcTemp;
  unsigned short polyTemp = CRCPoly; //polynomial shifts as opposed to CRC
  unsigned short testMask = 0x8000;  //used to evaluate whether we should XOR

  //XOR previous CRC and current input for multi-byte CRC calculations
  //temporary is shifted left one byte to perform direct mode calculation
  crcTemp = (  ( (unsigned short)(PEC_CurrentCRC ^ crcInput) ) << 8  );

  do {
    if(crcTemp & testMask)
      crcTemp = crcTemp ^ polyTemp;

    testMask = testMask >> 1;
    polyTemp = polyTemp >> 1;
  }
  while (polyTemp != CRCDone);

  PEC_CurrentCRC = (unsigned char) crcTemp;
}

//CRC_Reset will reset PEC_CurrentCRC to 0
//this should be called before a new multi-byte calculation needs to be done
void PEC_ResetCRC(void)
{
  PEC_CurrentCRC = 0;
}
```

Appendix III. Pin Assignments and Configuration Circuits

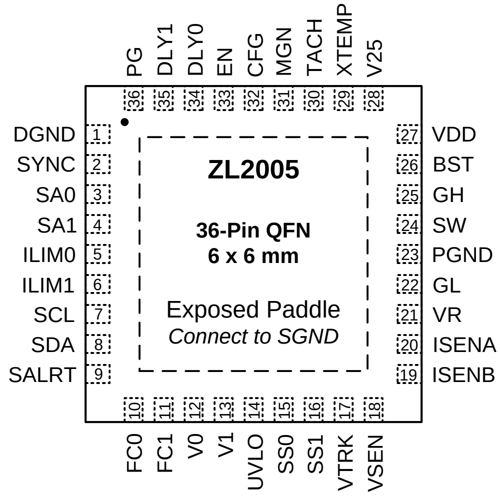

Figure 4. ZL2005 Pin Assignments (top view)

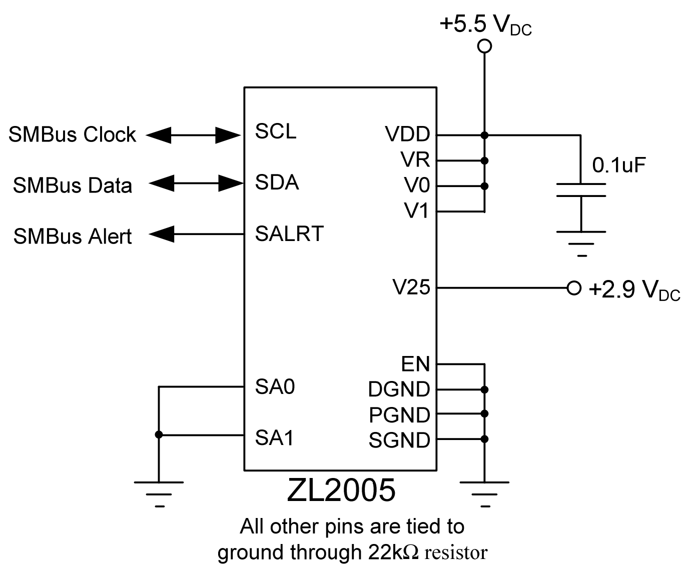

Figure 5. ZL2005 Configuration Circuit

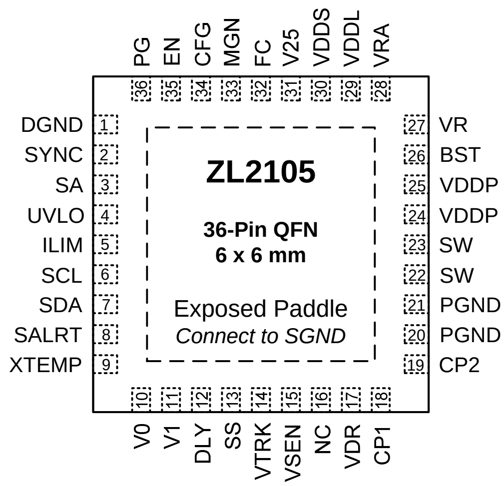

Figure 6. ZL2105 Pin Assignments (top view)

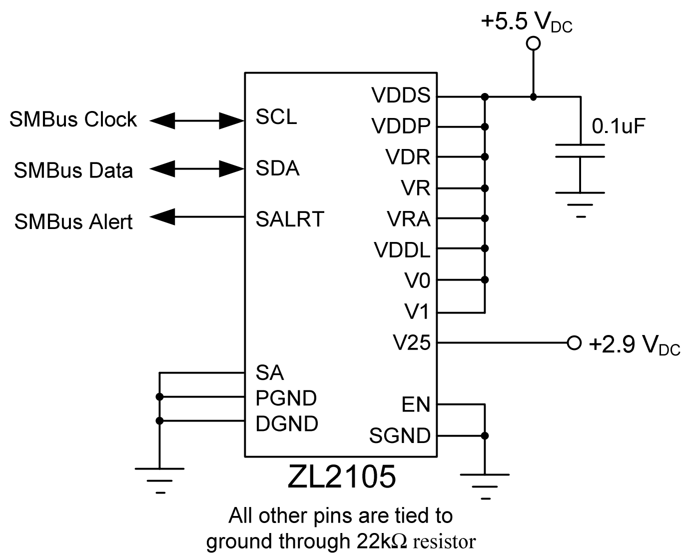

Figure 7. ZL2105 Configuration Circuit

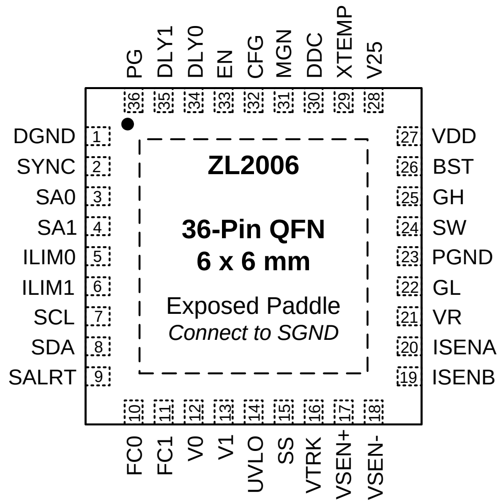

Figure 8. ZL2006 Pin Assignment (top view)

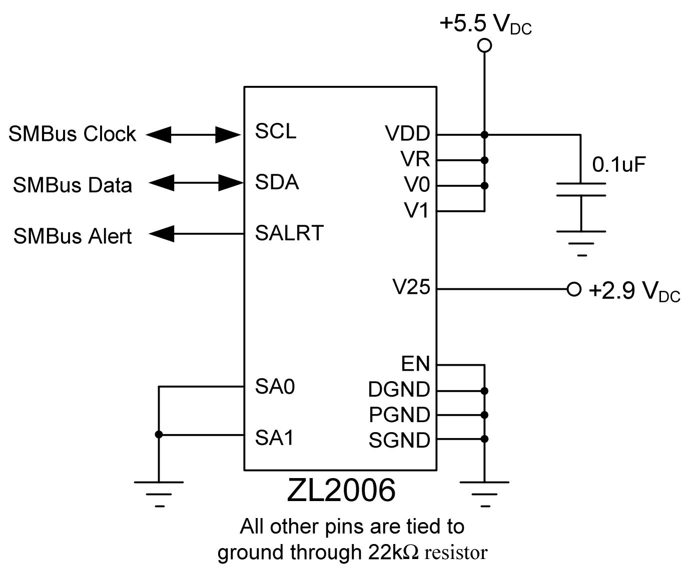

Figure 9. ZL2006 Configuration Circuit

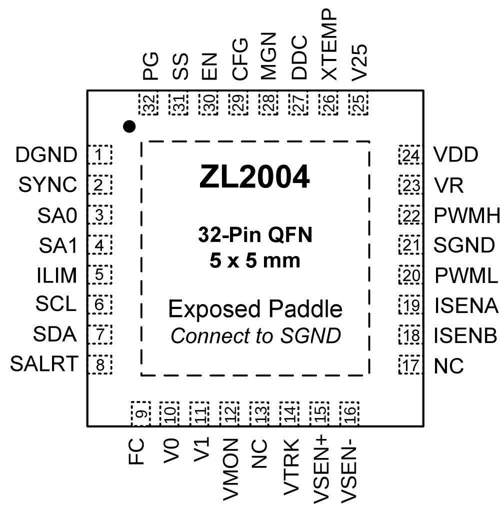

Figure 10. ZL2004 Pin Assignment (top view)

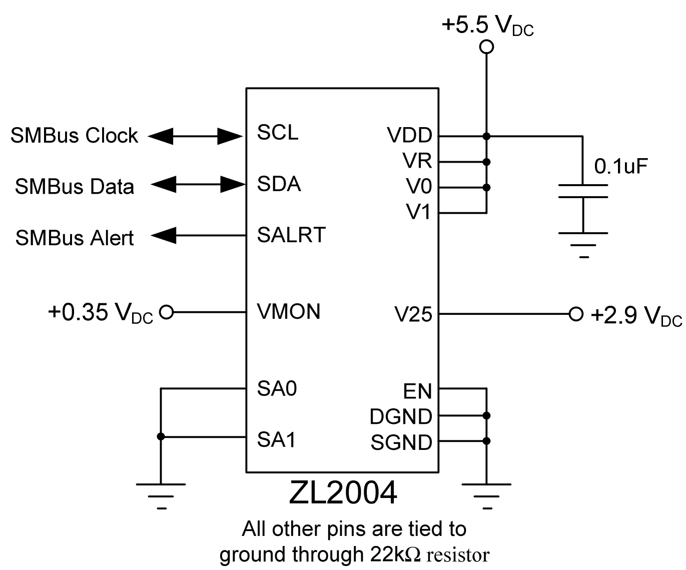

Figure 11. ZL2004 Configuration Circuit

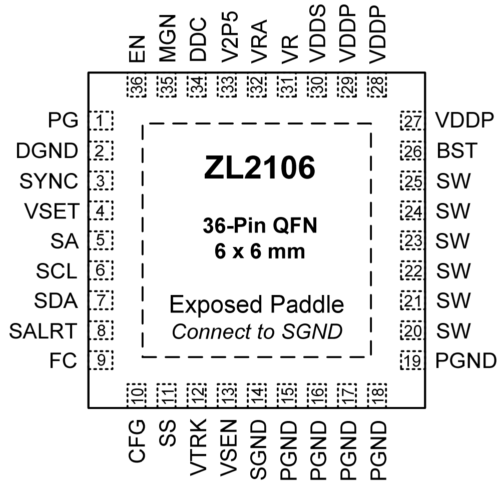

Figure 12. ZL2106 Pin Assignment (top view)

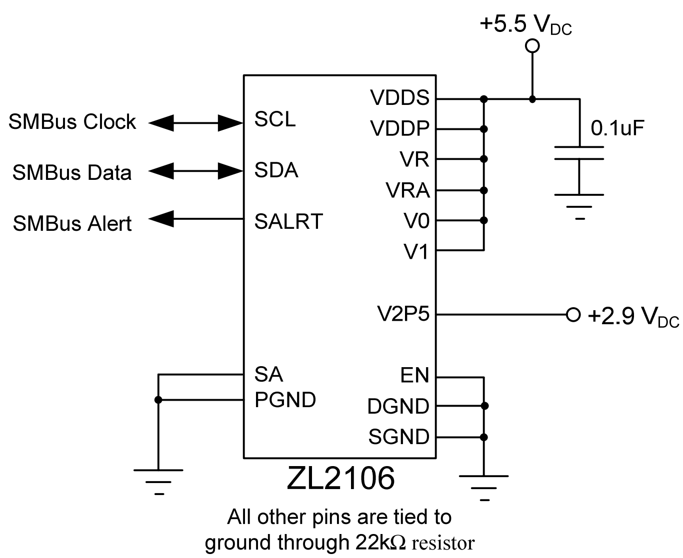

Figure 13. ZL2106 Configuration Circuit

Revision History

Date	Rev. #	Description
September 2007	0.5	Initial Release
10/28/07	0.6	Added ZL2005 and ZL2105 pin assignment diagrams Added ZL2105 configuration circuit
10/31/07	0.75	Added updated ZLHDC support for passwords and final Hex file format
11/27/07	0.8	Updated Fig 2 and Fig 4 pin configuration Added PEC to description on Page 3
12/6/07	0.85	Added configuration flowchart Added example code for calculating PEC
3/6/08	0.95	Modified Configuration Example Circuits to avoid factory VIN fault limits from causing SALRT to prematurely drop
4/1/08	0.96	Added ZL2006, ZL2004 schematics
4/30/08	0.97	Changed verification from SALRT to READ STATUS WORD
5/30/08	0.98	Updated to include additional STATUS_CML read, change to serial data byte locations
6/16/08	1.0	Added ZL2106 device
5/01/09	AN2030.0	Assigned file number AN2030 to app note as this will be the first release with an Intersil file number. Replaced header and footer with Intersil header and footer. Updated disclaimer information to read “Intersil and it’s subsidiaries including Zilker Labs, Inc.” No changes to application content.

Zilker Labs, Inc.

4301 Westbank Drive

Building A-100

Austin, TX 78746

Tel: 512-382-8300

Fax: 512-382-8329