AN-2093
APPLICATION NOTE

One Technology Way • P.O. Box 9106 • Norwood, MA 02062-9106, U.S.A. • Tel: 781.329.4700 • Fax: 781.461.3113 • www.analog.com

ADBMS1818 Slave Module Solution

by Amina Bahri

INTRODUCTION

An accurate report of a battery state of charge (SOC) during the battery lifetime prevents a system damage and explosion (for example, because of battery overcharging) and maximizes the longevity of the battery and, therefore, the system host. Monitoring the battery SOC is achieved by connecting the individual cells of the battery back to a battery management system (BMS) comprised of one or multiple slave units that report SOC data to a master unit that manages the algorithm (see Figure 3 for a BMS system example).

The main function of a BMS is to keep the different cells of the battery pack inside their safe operating area by closely computing their SOC and state of health (SOH).

The ADBMS1818 solution described in this application note was developed in collaboration with the German Institute Fraunhofer IISB and features the ADBMS1818, an Analog Devices, Inc., cell monitoring IC.

This application note introduces the ABMS1818 multicell battery stack monitor and the evaluation platform, which consists of the foxBMS master unit (see Figure 1) and the ADBMS1818 slave unit (see Figure 2). This application note also describes the system setup from a hardware and software perspective as well as the SOC test results and different implementation scenarios.

Refer to the ADBMS1818 data sheet and the master unit documentation referenced throughout this application note while using this application note.

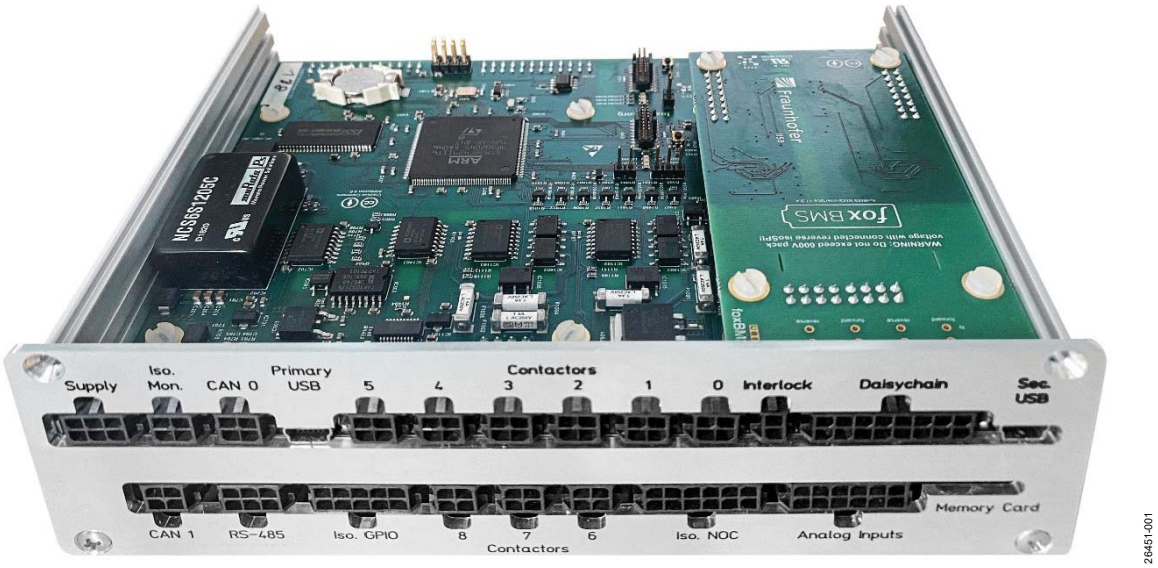

Figure 1. foxBMS Master Unit

[1] Figure 1 foxBMS Master Unit. © 2021 Analog Devices, Inc. All rights reserved. The foxBMS unit displayed above is proprietary to Fraunhofer-Gesellschaft zur Förderung der angewandten Forschung e.V. foxBMS and the foxBMS® are the exclusive trademarks and/or registered trademarks of Fraunhofer.

TABLE OF CONTENTS

REVISION HISTORY

8/2021—Revision 0: Initial Version

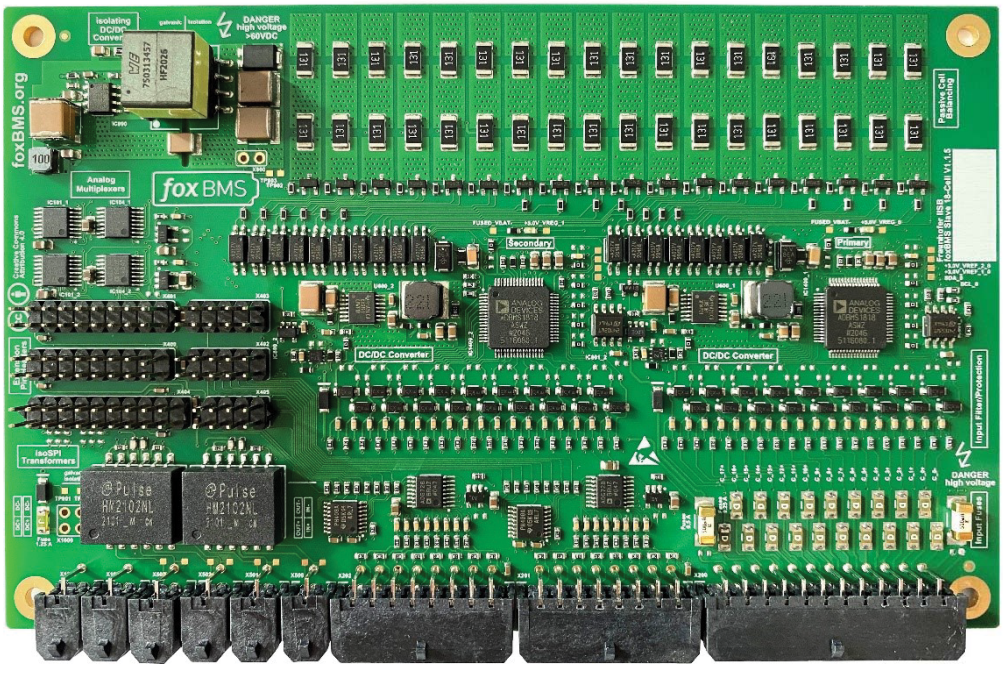

Figure 2. ADBMS1818 Slave Unit Reference Design

BMS SYSTEM SETUP

This section describes the evaluation of the ADBMS1818 platform and lists the required steps for both hardware and software setup.

Figure 3 shows the main elements and interconnections of the BMS system. The implementation (the number of slave units, the size of the battery, and additional safety elements such as fuses or ground fault detectors) depends on the user application and specifications.

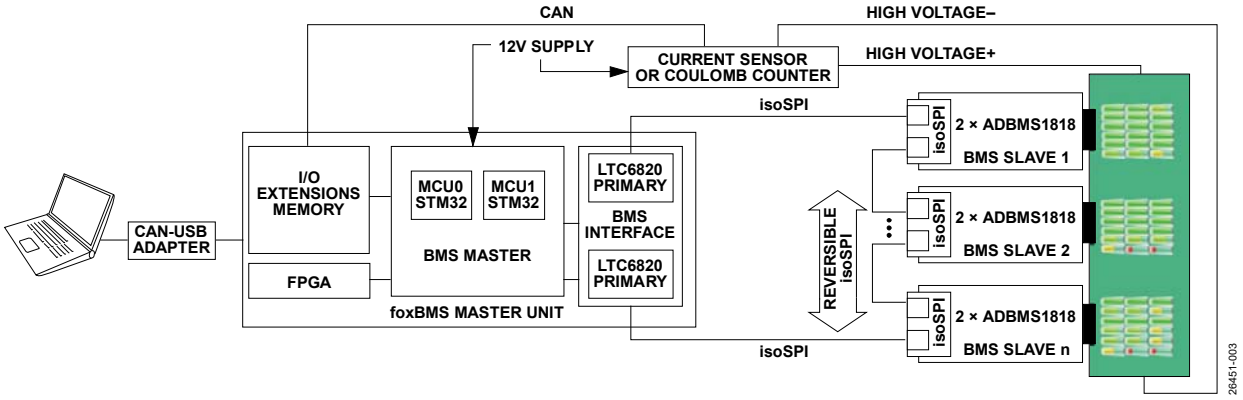

Figure 3. BMS System Block Diagram

SOFTWARE SETUP

The software, documentation, and workspace required for software setup and use are all open source and free to download.

C-code files must be compiled to generate the firmware (the binary file) with which the microcontrollers, MCU0 and MCU1, are flashed.

Table 1 summarizes the different software components necessary to run the BMS platform.

Table 1. foxBMS Master Unit System Software Components

foxBMS Embedded Software	Name	Function
Software Toolchain Python-Based Environment	foxconda3	This component compiles the embedded software and generates the HTML documentation.
Integrated Development Environment	Eclipse CDT	This component works with foxBMS sources to write, compile, debug, and run the foxBMS C source code.
Python Integrated Development Environment (IDE) for Eclipse	PyDev	This plugin enables Eclipse to be used as a Python IDE.

To compile the C program, take the following steps:

1. Install the foxconda3 installer including proxy configuration. No further libraries and programs need to be installed because these programs are already installed with the foxconda3 installer.
2. Build the software and generate the HTML documentation for both microcontrollers, MCU0 and MCU1. The foxconda3 installer provides an Anaconda prompt, which is the interface used to write Conda commands. Therefore, all commands needed to build the binaries must be entered on the Anaconda prompt.
3. Set the Eclipse workspace to work with the foxBMS sources.

This process requires installing and configuring the Eclipse C/C++ development tooling (CDT) and downloading the foxBMS source files into the CDT. The PyDev plugin must be installed from the Eclipse marketplace to enable Eclipse to be used as a Python IDE. The PyDev plugin is available for download from the PyDev official website.

This software configuration step ensures that the workspace is properly set up and that the software files are properly imported into the IDE (Python interpreter and paths configuration).

For more details on this setup process, visit the foxBMS website and navigate to the documentation related to Gen1.

To flash and run the C program on the microcontroller unit, take the following steps:

1. Flash the microcontrollers.
2. Supply the master unit through the **Supply** header to flash MCU0 and MCU1. The order of the wires must be respected when inserting and crimping the wires into the proper Molex receptable. The supply voltage range for the master unit is rated from 12 V to 24 V. At 16 V, the unit draws 150 mA.
3. Connect a USB cable from the PC to the master unit.

The master unit must be connected to the PC via the mini USB jack labeled **prim. USB** and **sec. USB** to program MCU0 and MCU1, respectively.

When connecting the master unit to a PC for the first time, an internet connection must be available because the required drivers install automatically and the operating system looks for the drivers on the internet.

The generated binary files, the header and main code (refer to the previous steps), must be flashed.

When the master unit is supplied and the MCUs are properly programmed, the user can proceed to connect the slave unit(s) and measure the SOC of the battery cells connected to the salve unit(s).

Running the Graphical User Interface (GUI)

To run the GUI for the first time, take the following steps:

1. Write the C:\foxconda3\Scripts\activate command in the command prompt window.
2. Write the conda install matplotlib commands to install the matplotlib Python library, which downloads the required packages.
3. Download the GUI from the foxBMS website under the master tools section.

To display the GUI, take the following steps:

1. Write the C:\foxconda3\Scripts\activate command in cmd.
2. Write the GUI path in cmd to access the GUI folder.
3. Write the python foxbms_interface.py script to visualize the data from the foxBMS master unit. The interface shown in Figure 4 appears.

The bit rate default setting is 50 kbit/sec.

The user must specify the number of modules, which translates to the number of slave units that are daisy-chained as well as the number of cells per module.

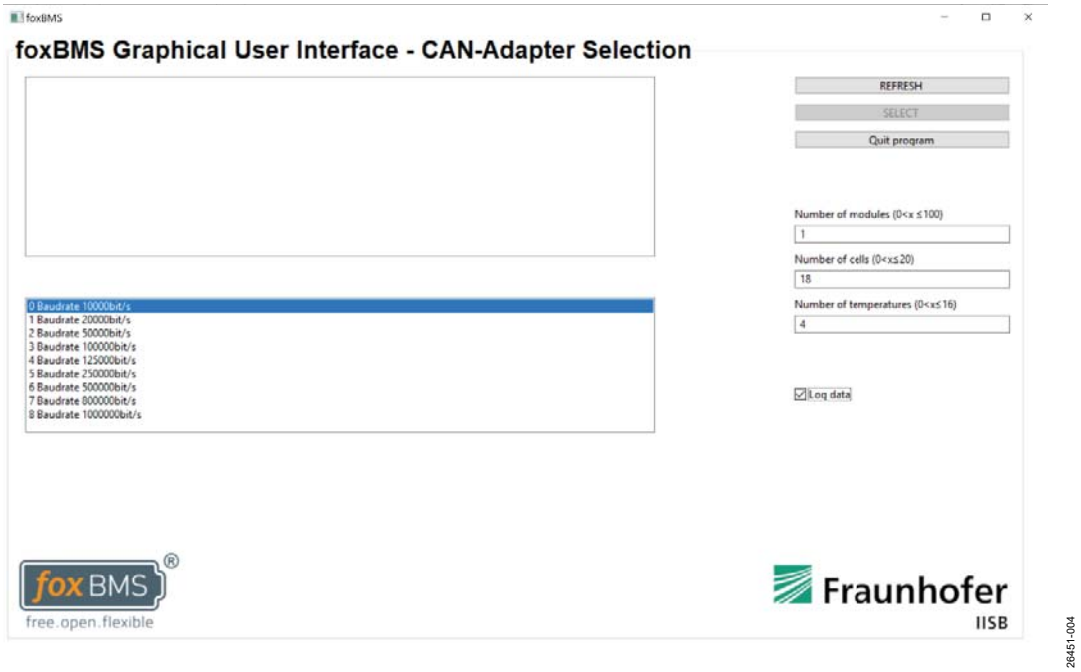

Figure 4. foxBMS GUI

HARDWARE

This section focuses on the main elements that intercommunicate in the system. See Table 2 for the required hardware.

Table 2. Main Hardware Components Required

Equipment	Manufacturer	Part Number
foxBMS Master Unit	Fraunhofer IISB	foxBMS version 1
ADBMS1818 Slave Units	Analog Devices	Slave 18-cell V1.1.5
CAN-Bus to PC Adapter	Peak System	PCAN-USB FD
Battery Simulator Board	Analog Devices	DC2472A
Current Sensor	Isabellenhütte	IVT-S-100-U3-I-CAN2-12/24
Debugger	Segger	J-Link LITE

FOXBMS MASTER UNIT

The foxBMS master unit is comprised of the following main components:

- BMS master. This board has two STM32F429 microcontroller units to ensure redundant safety.
- BMS interface. This board is located on top of the BMS master unit and connects to the first BMS slave in the daisy chain through isoSPI. This block features the LTC6820 SPI isolator (see the LTC6820 data sheet for further information).
- BMS extension board. This board consists of an isolated RS485 interface (which can be an alternative to the controller area network (CAN) or universal asynchronous receiver/transmitter (UART) interface when communicating with the foxBMS master unit), analog inputs, and isolated general-purpose input/outputs (GPIOs).

The master unit also comes with a separate SPI interface to reduce the data traffic going through the isoSPI interface.

ADBMS1818 18-CELL BMS SLAVE UNIT

On-Board Solutions

The devices used on the slave unit (see Figure 5) include the ADMBS1818, LTC6280, and LT8302. For more information on these devices, see the ADMBS1818, LTC6280, and LT8302 data sheets.

Using the DC2472A 18-Cell Simulator Demo Board

The DC2472A USB powered board provides a simple battery stack simulator and can be used to mimic up to 18 cells connected in series. The number of output rails (cells) can be reduced and adjusted using the jumpers. This board simulates 1V5 and 4V2 per cell, which is suitable for several battery chemistries.

Cabling the BMS System

All connectors used in this system are type Molex Micro-Fit 3.0.

Test Setup Equipment

Table 3 lists the test equipment used in this system. The oscilloscope and the digital multimeter (DMM) are required for debugging and verification purposes.

Table 3. Test Equipment

Equipment	Manufacturer	Part Number
Oscilloscope	Rohde&Schwarz	RTE 1104 1GHz
DC Power Supply	Voltcraft	LSP-1403
DMM	FLUKE	Fluke 287

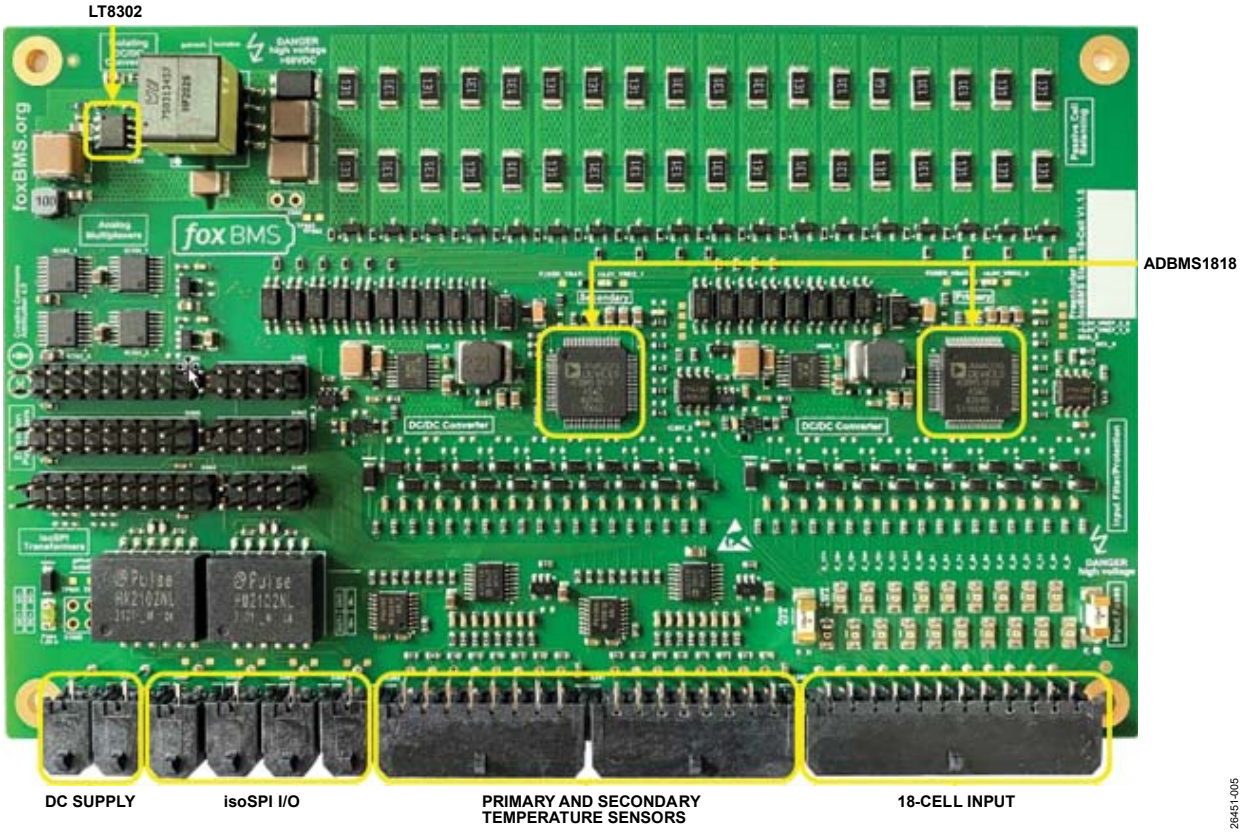

Figure 5. ADBMS1818 Slave Unit

ESTABLISHING MASTER TO SLAVE UNIT(S) COMMUNICATION

CAN-Bus Communication

A CAN-bus to USB adapter is required to establish the communication between the master unit and the PC.

Debugging CAN Communication

Apart from using an oscilloscope to see the CAN_H and CAN_L signals, it is possible to view, transmit, and record CAN data traffic using a CAN monitor software called PCAN-View.

The CAN-USB adapter must be connected to the master unit through the CAN0 header. This connection requires some cabling work. One possible method is to use a female DB9 connector to take out three wires (CAN_H, CAN_L, and GND) and connect the wires to the 4-pin Molex header.

To ensure that the adapter is recognized by the PC, it is important to download the proper PCAN USB driver.

When all required connections are made and the CAN-bus to USB driver is installed, the CAN-bus device is detected.

Note the following when setting up the CAN communication:

- It is not possible to use the PCAN-View software and the GUI at the same time. The hardware is already in use by another software and the bit rate cannot be changed.
- When the adapter is connected to the PC, the LED is on (illuminated red).
- When selecting the available device, the LED on the adapter blinks, which means the adapter is ready to receive and transmit data.

Measuring the Battery Stack Current

A current sensor or a coulomb counter can be connected between the most positive and most negative battery voltage (BAT+ and BAT−, respectively) to measure the current going through the battery stack.

If no current measuring device is included in the BMS system, a small firmware modification is required.

In the **embedded-software\mcu-primary\src\general\config\batterysystem_cfg.h** header file, the current measurement is set by default to the TRUE command, as follows:

```
#define CURRENT_SENSOR_PRESENT          TRUE
```

This switch must be set to the FALSE command to allow the system to start without the current sensor.

EVALUATION RESULTS

The following test setup is for two ADBMS1818 slave units in a daisy chain (see Figure 6).

Note that when connecting more than one slave unit in the system, the number of defined modules in the firmware must be changed accordingly.

Use the **/primary/mcu-primary/src/general/config/batterysystem_cfg.h** path to go inside the **batterysystem_cfg.h** header file.

Change the defined number of modules in the following line to the desired number of slave units:

```
#define BS_NR_OF_MODULES                    3
```

For simplicity, only one cell is attached to each battery simulator board. Figure 7 and Figure 8 show the calculated cell voltage and temperature measurement displayed on the GUI where the measurement error is less than 2 mV.

The graphs of the minimum and maximum cell voltage over time can also be seen on the GUI (see Figure 9).

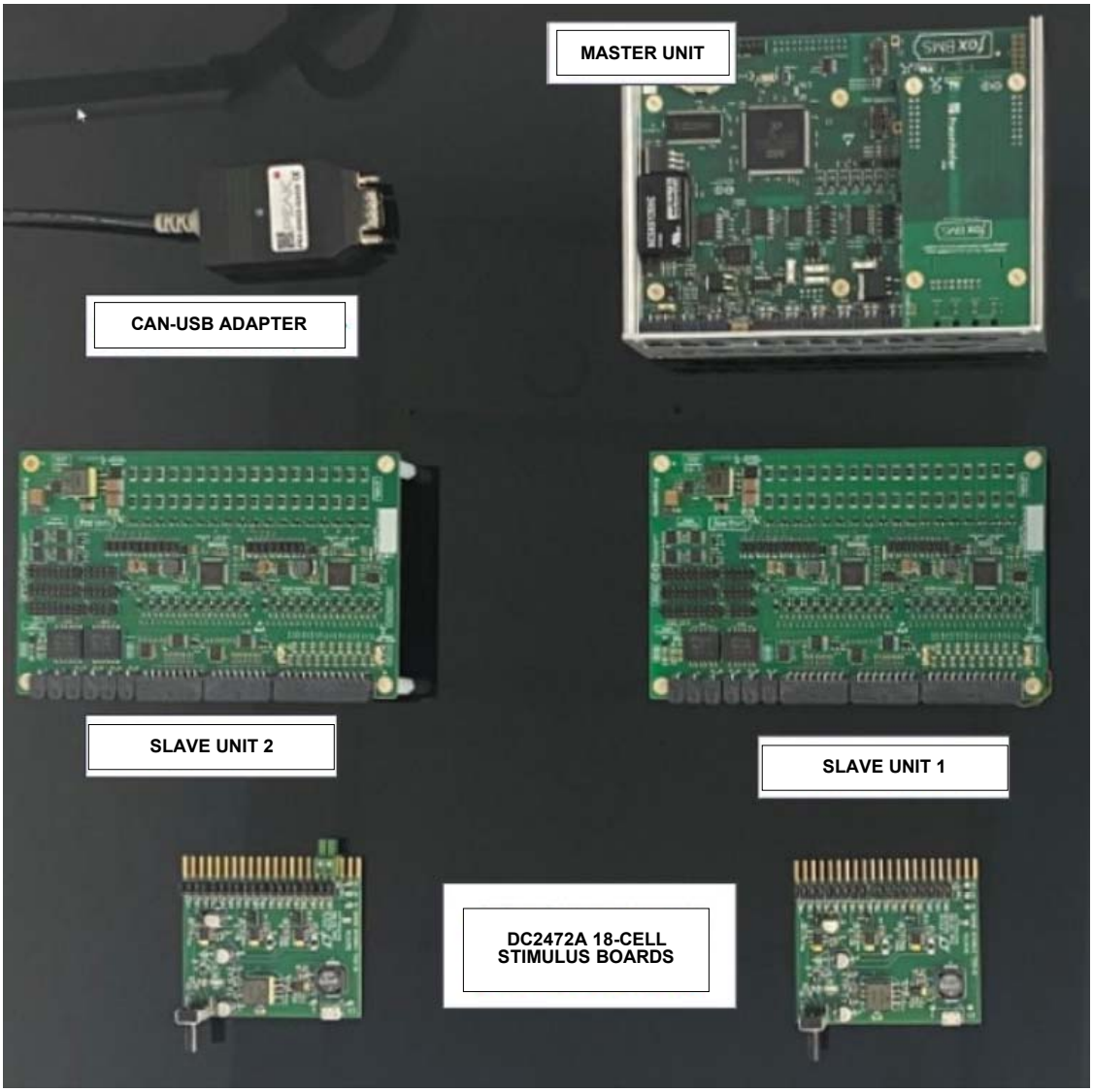

Figure 6. Test Setup

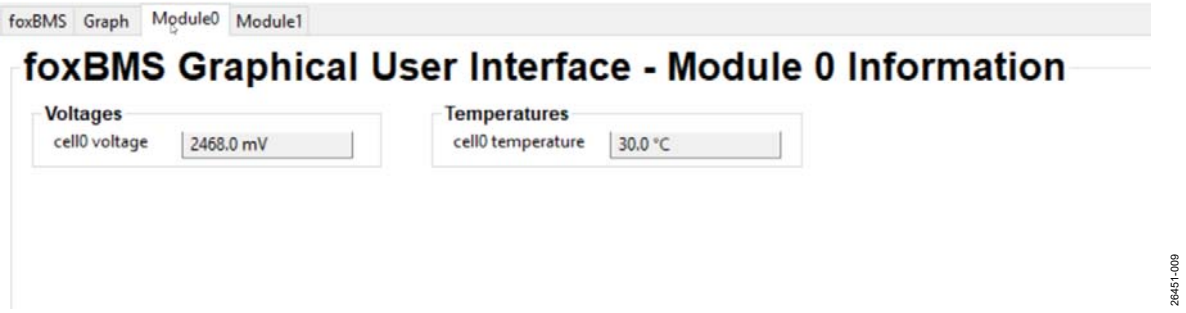

Figure 7. Cell Voltage Measurement on Slave 1

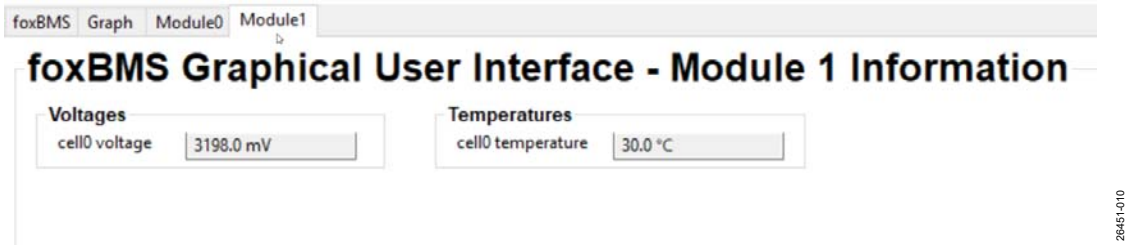

Figure 8. Cell Voltage Measurement on Slave 2

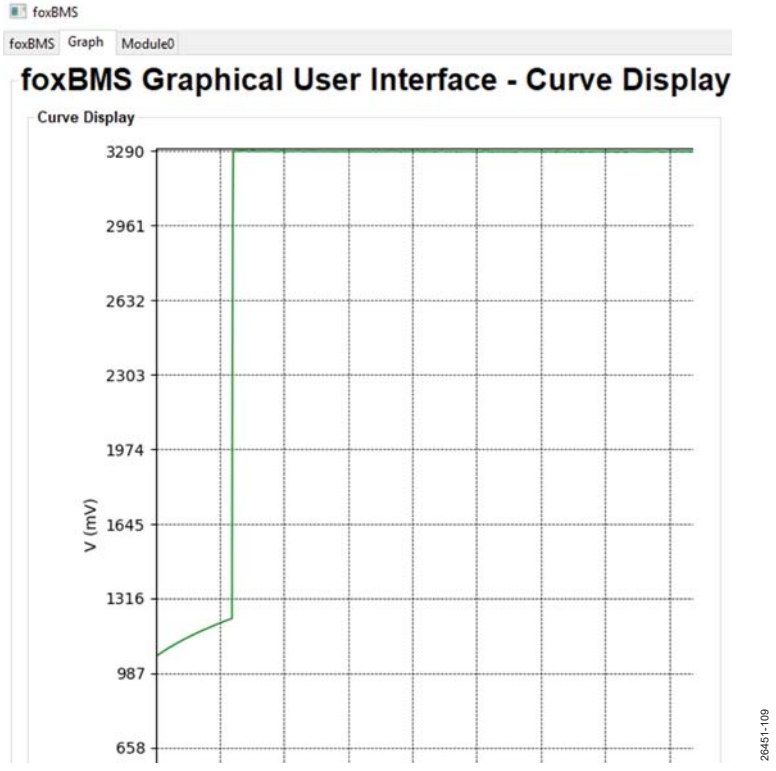

Figure 9. Voltage Across One Cell over Time

SCHEMATICS

Figure 10 to Figure 21 show the schematics for the ADBMS1818 slave unit. Only the primary ADBMS1818 is illustrated here for simplicity.

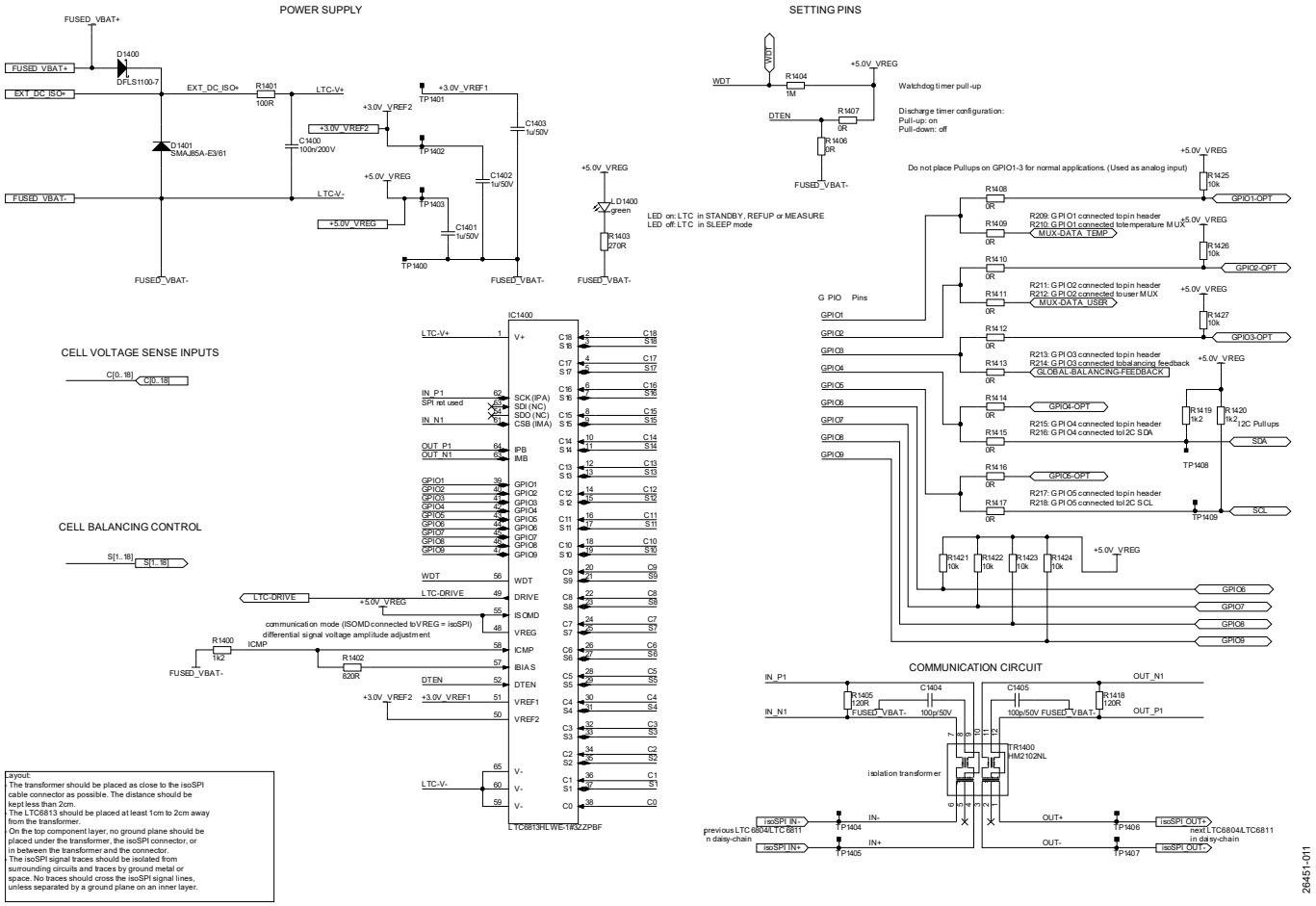

Figure 10. Primary ADBMS1818

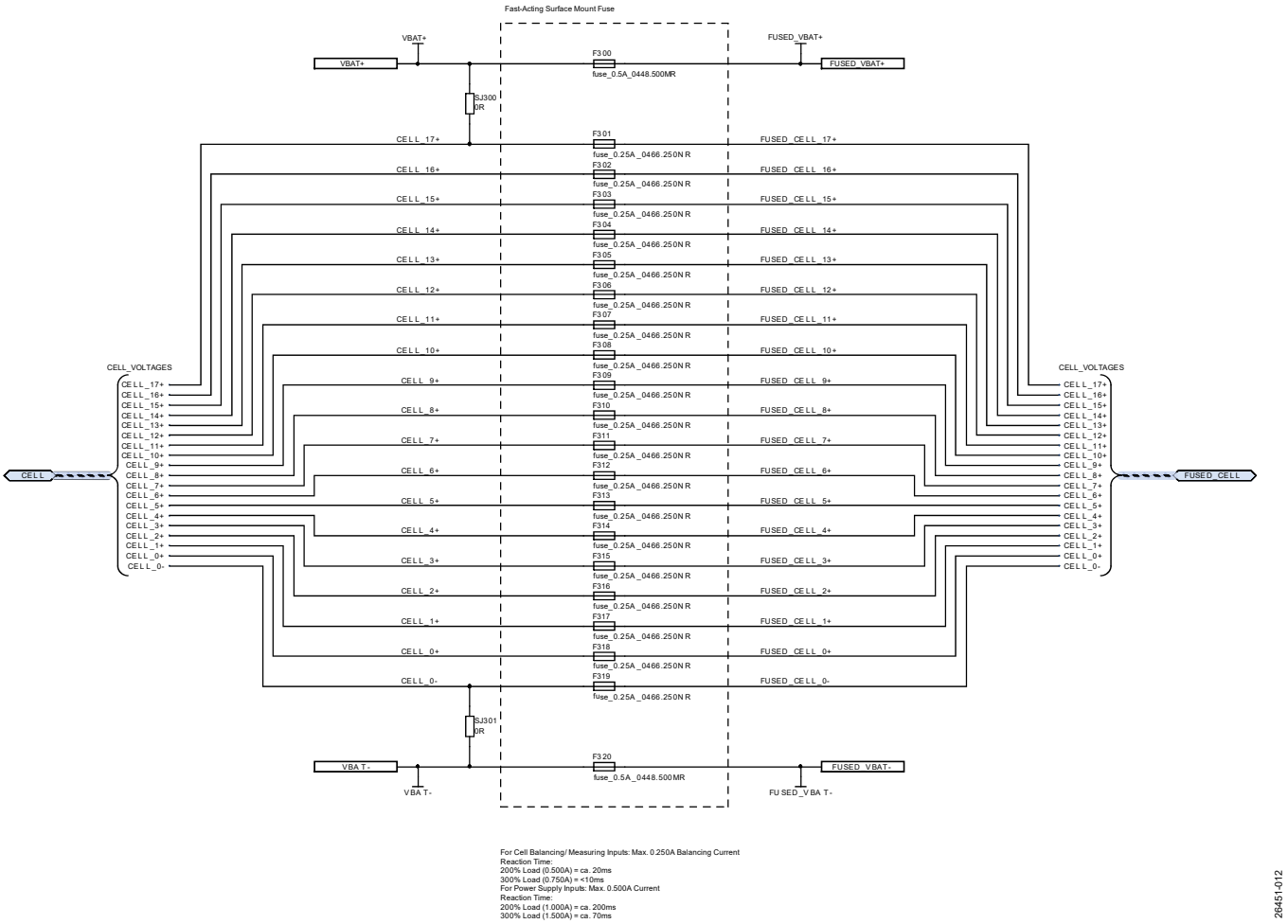

Figure 11. Cell Voltage Sense Fuses

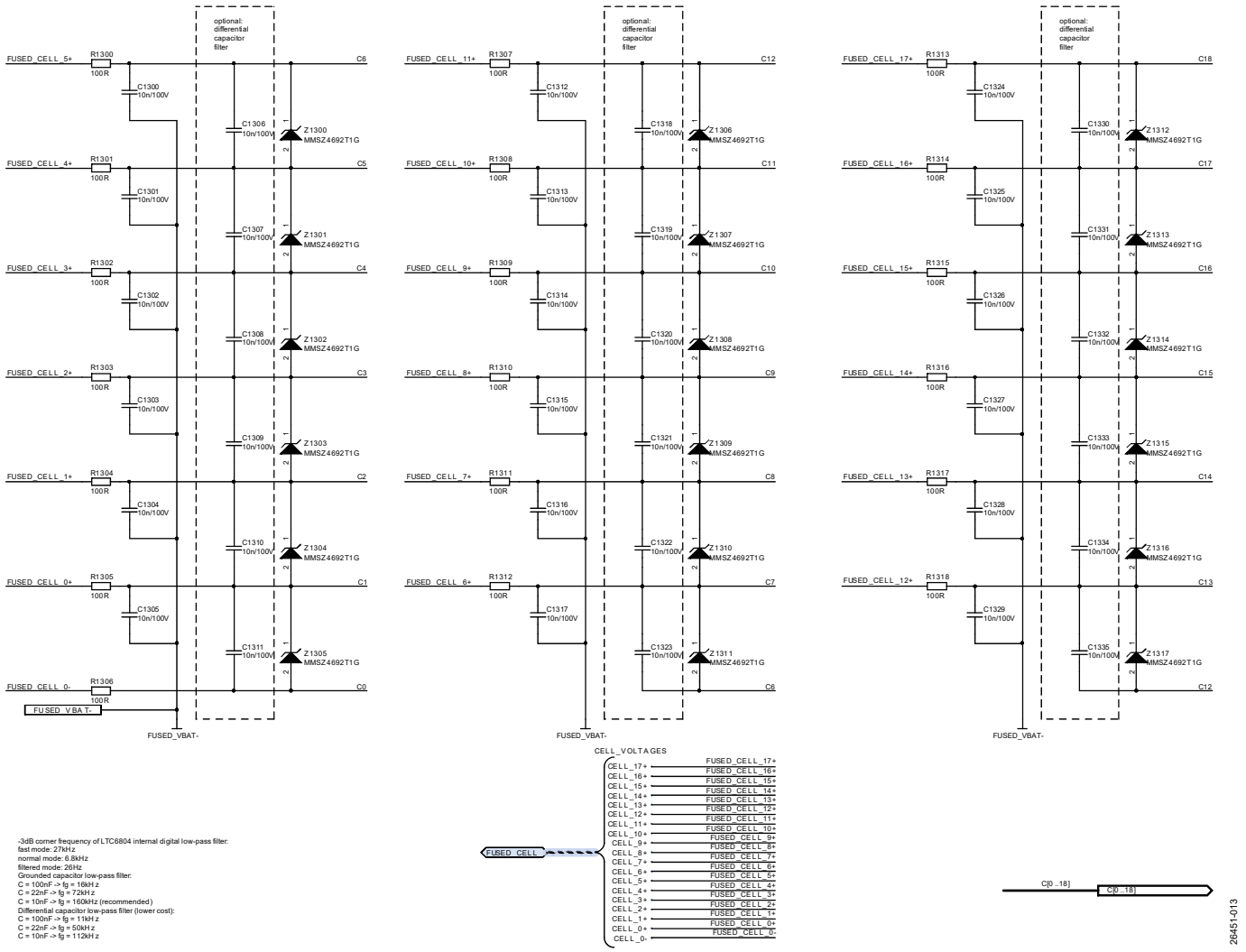

Figure 12. Primary Voltage Measurement

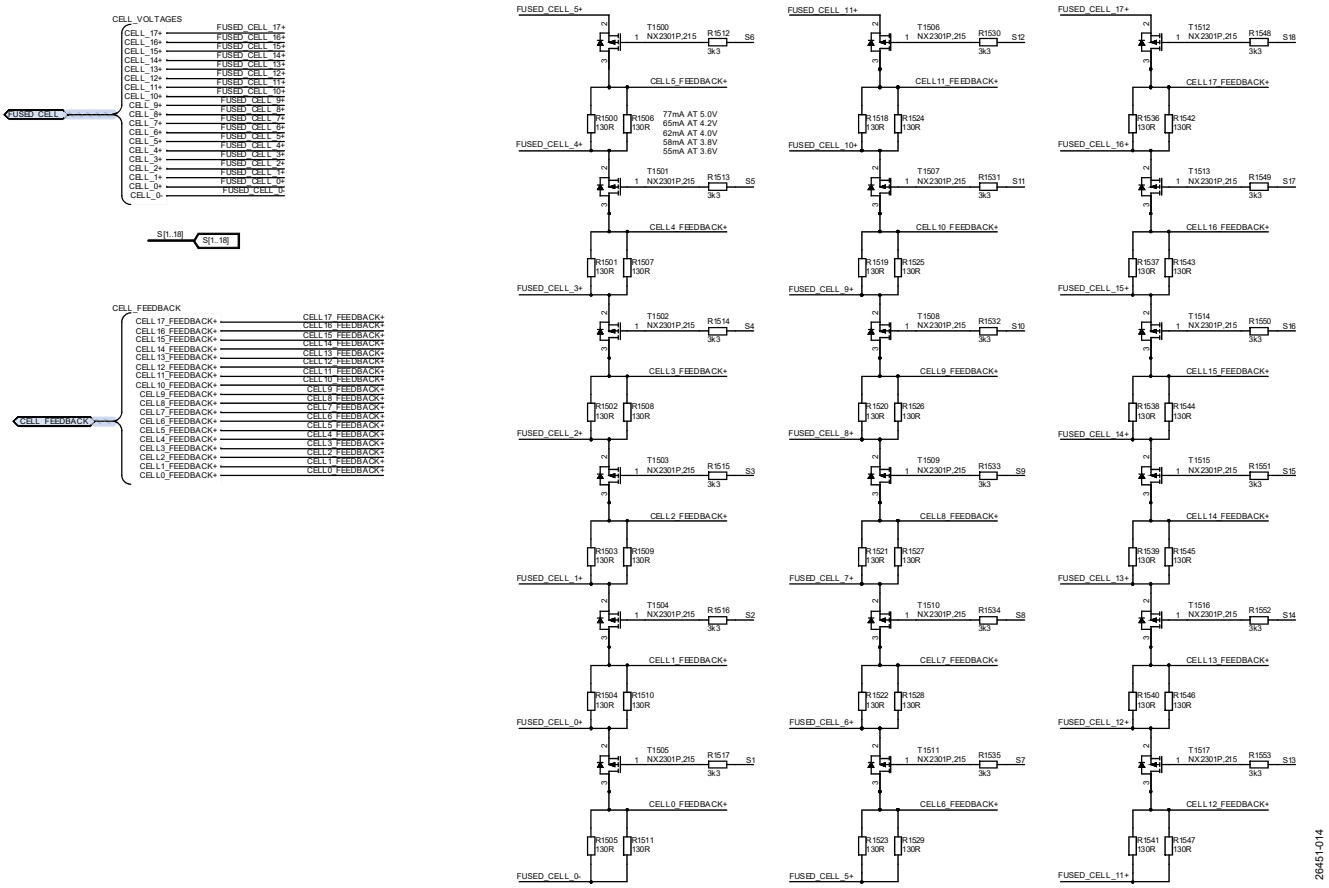

Figure 13. Passive Cell Balancing

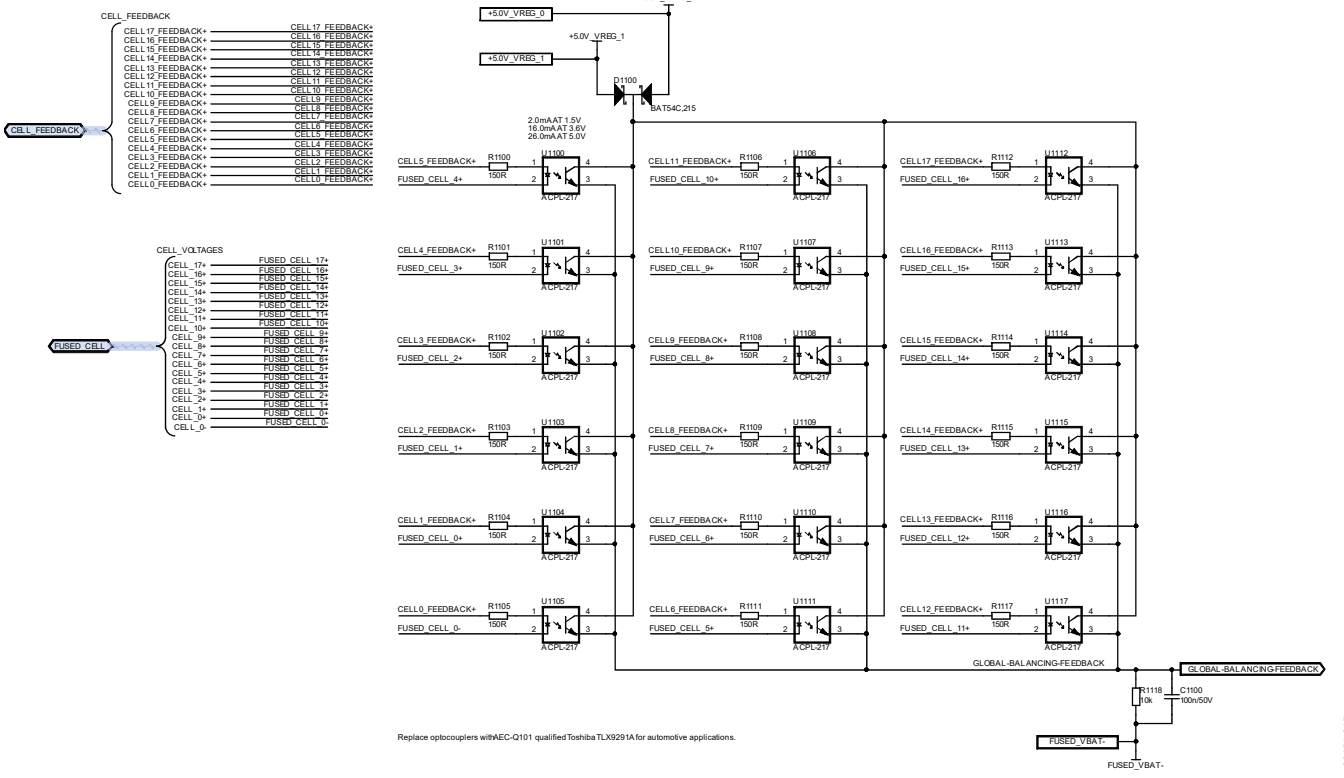

Figure 14. Passive Balancing Feedback

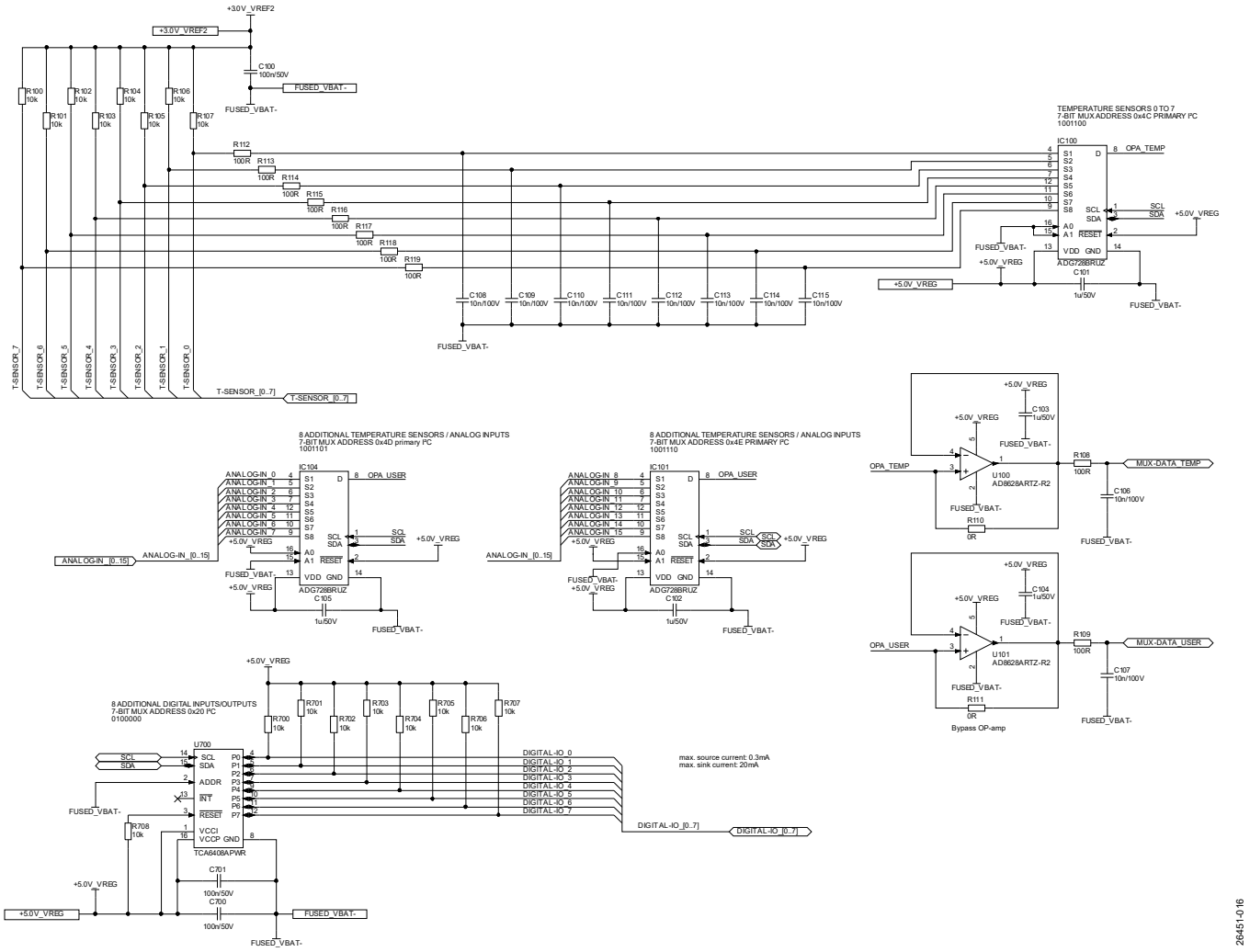

Figure 15. Primary Analog Inputs and Digital I/O

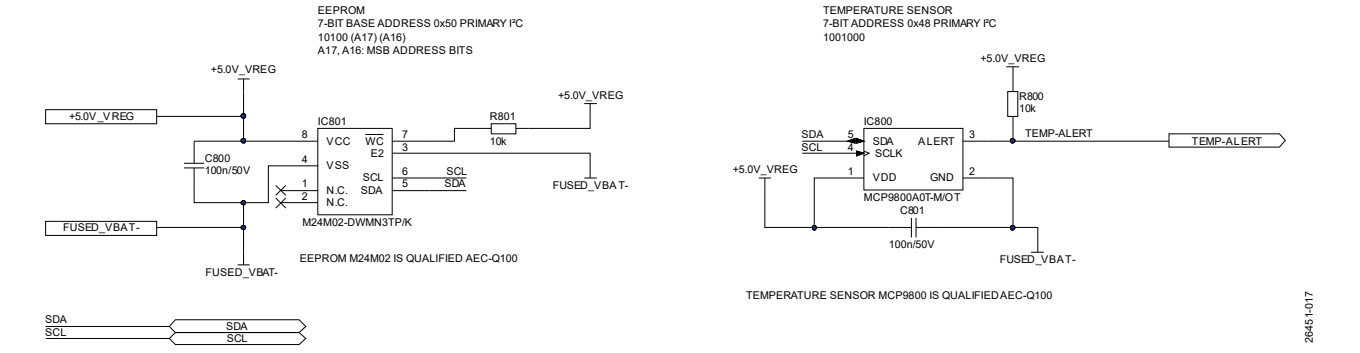

Figure 16. Primary EEPROM and Board Temperature Sensor

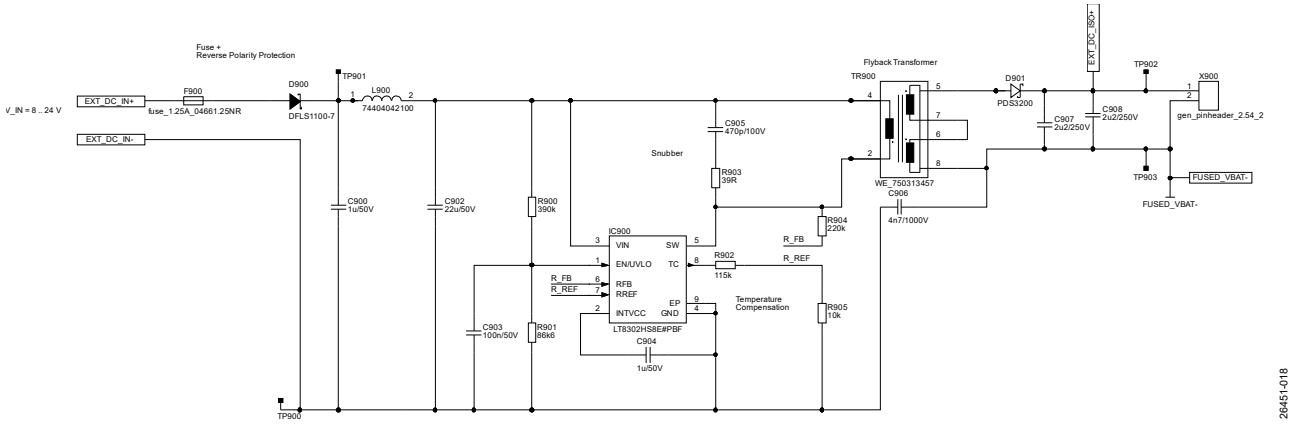

Figure 17. External Power Supply

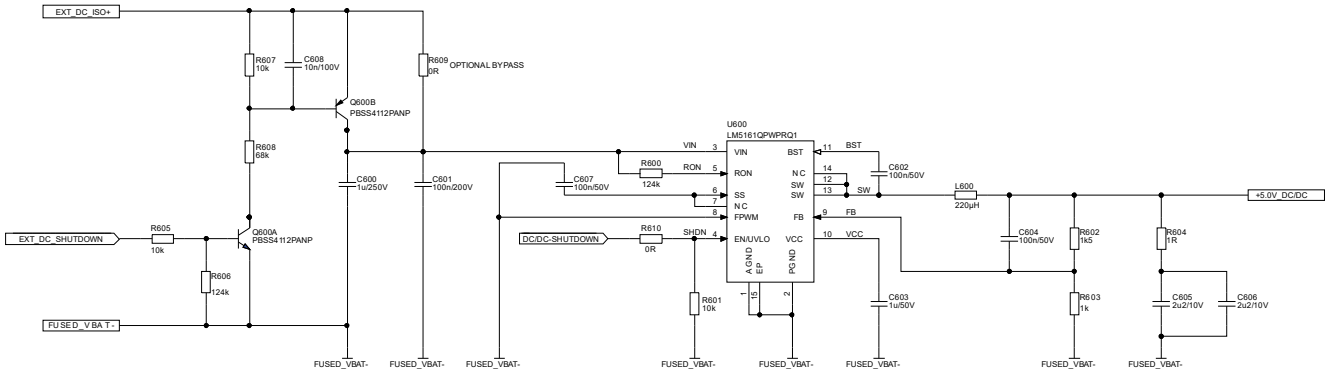

Figure 18. DC-to-DC Converter

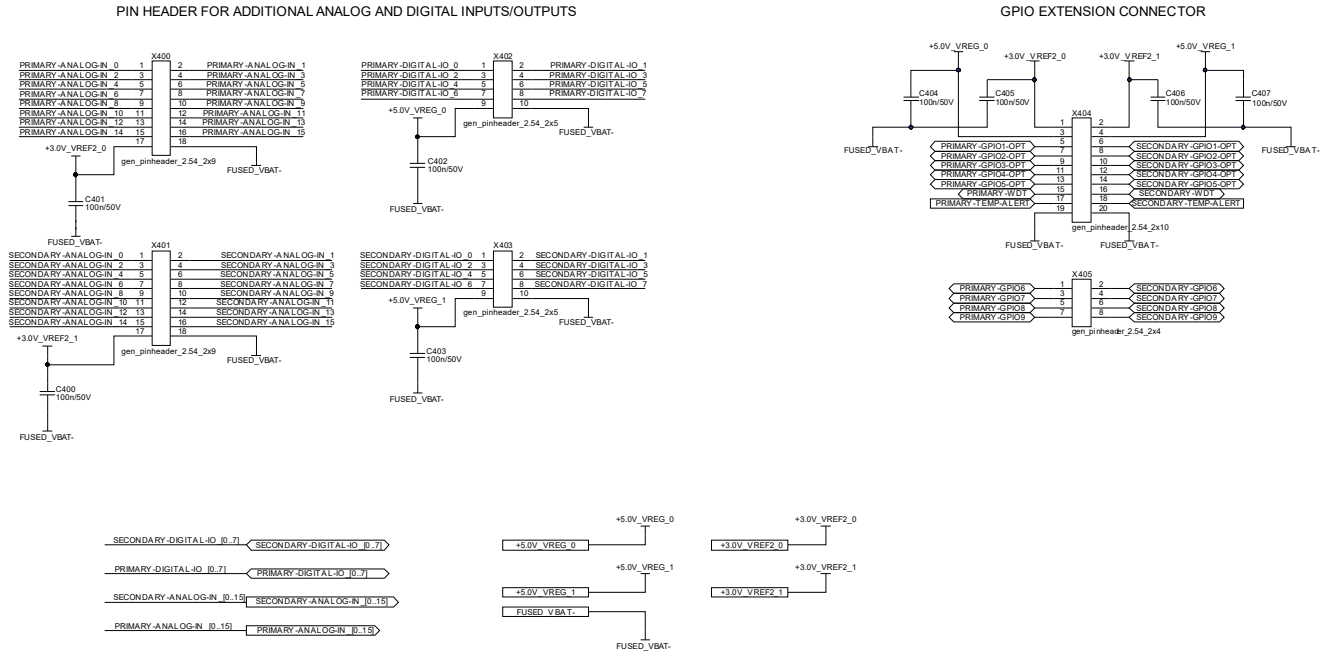

Figure 19. Connectors

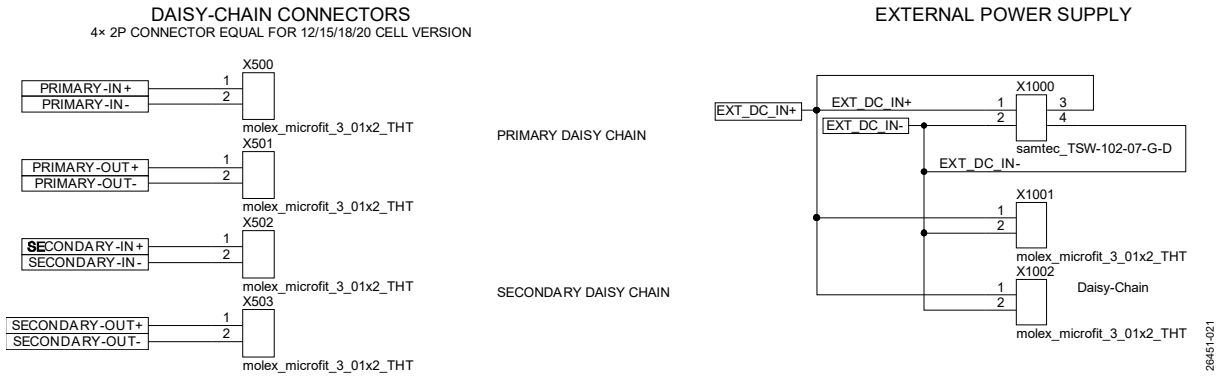

Figure 20. Daisy-Chain Connectors and External Power Supply Connectors

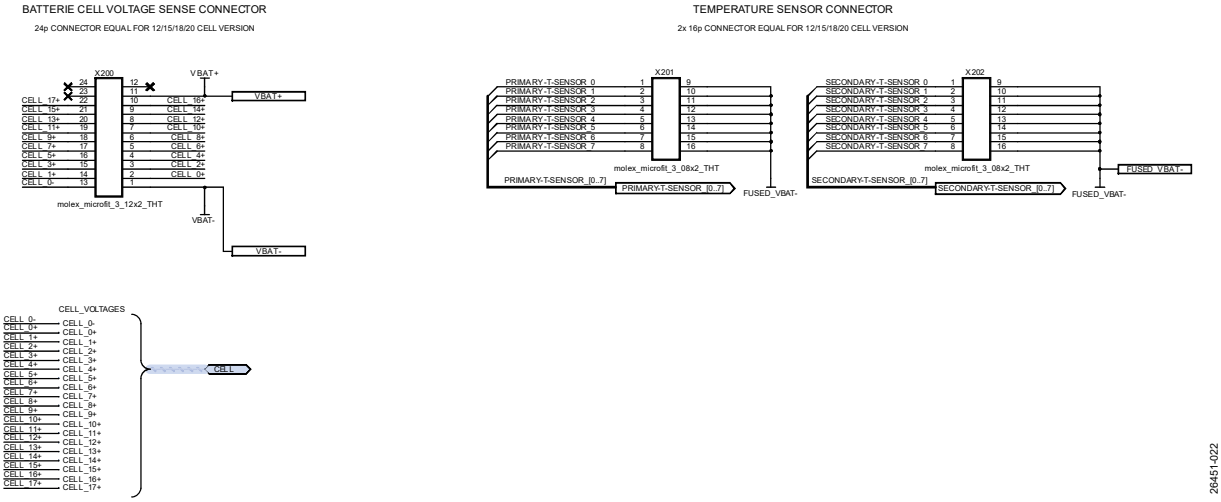

Figure 21. Battery Connectors

CONCLUSION

The ADBMS1818 slave unit is suitable for energy storage systems and industrial applications. For functional safety details, contact Analog Devices technical support.

This application note demonstrates the implementation of a BMS system using open-source software tools and firmware as well as slave unit schematic examples. This application note also proposes a test setup for battery cell SOC measurements.