BCC_S32K144_Monitoring

Example project for Battery Cell Controller SW Driver
(Lite version)

Date: *12 June 2019*

Revision: *1.1*

Overview

The purpose of this example project is to demonstrate how to use the Lite version of Battery Cell Controller (BCC) SW Driver to get measurements and fault status information from the BCC using S32K144. Raw values of status registers and interpretation of measured values is printed to the console.

The example is determined for all BCC devices and all communication types: MC33771B SPI, MC33771B TPL, MC33772B SPI and MC33772B TPL.

Hardware Requirements

Following is required:

- S32K144EVB-Q100
- FRDM33771BSPIEVB or FRDM33772BSPIEVB or FRDM33771BTPLEVB or FRDM33772BTPLEVB board
- Configurable AAA battery pack (BATT-14AAAPACK) with batteries or Battery pack emulator (BATT-14EMULATOR for MC33771B or BATT-6EMULATOR for MC33772B)
- USB Micro B cable

Moreover, **SPI boards** require four board-to-board connectors (CON 2x8, CON 2x8, CON 2x10 and CON 2x6) shipped with the boards.

TPL boards require FRDM33664BEVB, wires for interconnection of the boards, and (optionally) a Translator board (X-DEVKIT-TRANSLT; provided upon request by NXP).

For detailed information about interconnection of S32K144 evaluation board and BCC evaluation boards, refer to UM11143 (User manual for BCC SW driver).

Setting up Hardware

1. Make sure that you **disconnect** SBC (System Basis Chip) and power your S32K144EVB-Q100 from OpenSDA USB instead. SBC initialization and setting is not in scope of this example.

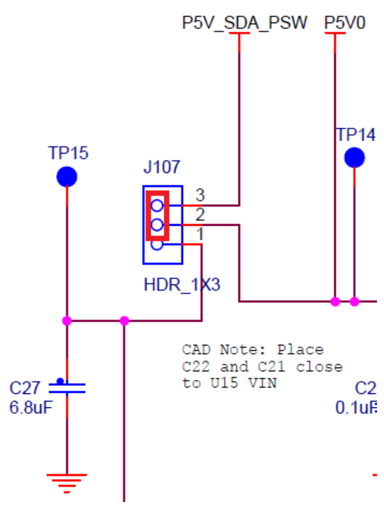


Figure 1. S32K144EVB-Q100 power source selection

2. Configure and connect selected BCC board (FRDM33771BSPIEVB, FRDM33772BSPIEVB, FRDM33771BTPLEVB or FRDM33772BTPLEVB) to the S32K144 evaluation board according to UM11143 (User manual for Battery cell controller software driver).
3. Setup the battery pack (or battery pack emulator) according to its user manual. If you use BATT-14AAAPACK battery pack, use 14 batteries for MC33771-based evaluation boards and 6 batteries for MC33772-based evaluation boards.
4. Connect battery pack (or battery pack emulator) using parallel cable to the utilized BCC board.
5. Connect USB cable to USB interface on the MCU board and to your PC.

Figure 2 shows the complete HW setup for MC33771B SPI evaluation board.

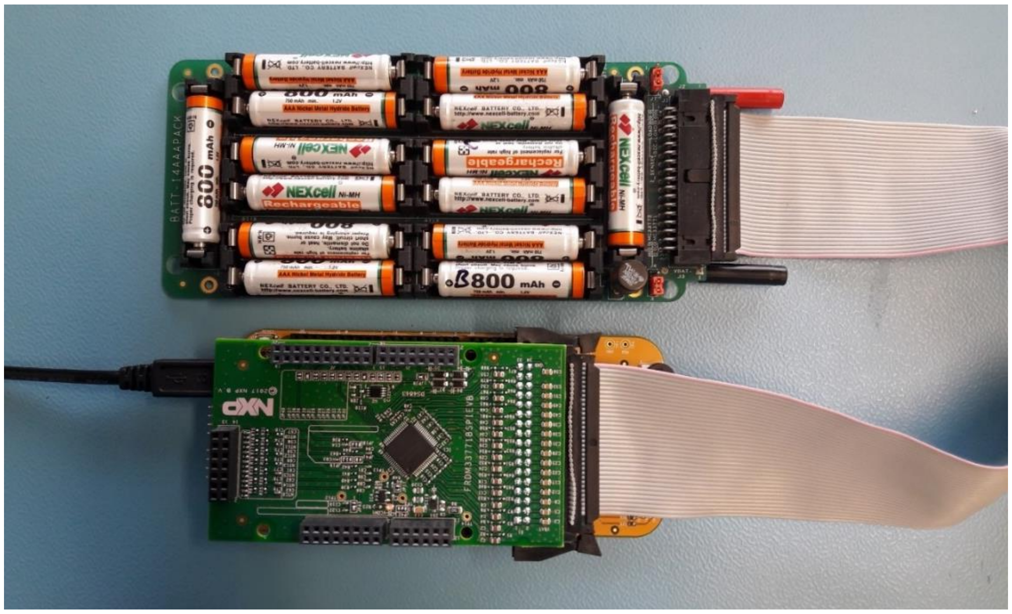


Figure 2. HW setup for FRDM33771BSPIEVB

Setting up Software

Make sure that you have installed S32 Design Studio for ARM Version 2018.R1.

The application uses debug interface with virtual serial port to print user messages. Check that your debug connection has been set up properly. Type of used debug connection depends on used MCU. S32K144EVB-Q100 uses OpenSDA, see Figure 3. Note that number of COM port may differ because of different system resource usage. Baud rate is 115200 Bd.

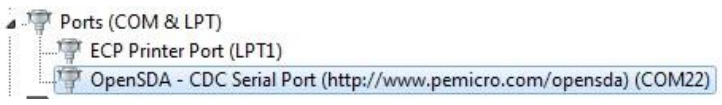

Ports (COM & LPT)
ECP Printer Port (LPT1)
OpenSDA - CDC Serial Port (http://www.pemicro.com/opensda) (COM22)

Figure 3. OpenSDA virtual port

Description

The purpose of this example project is to demonstrate how to use Lite version of the Battery Cell Controller SW Driver to get measurements and fault status information from the BCC device. Raw values of status registers and interpretation of measured values are printed to the console.

1. Initial configuration of BCC registers is printed.
2. Values of fuse mirror registers as well as BCC GUID is printed.
3. Following measured values (listed in ***bcc_measurements_t***) are printed.
    a. Coulomb counter registers.
    b. ISENSE current.
    c. Battery stack voltage.
    d. Battery cells voltage.
    e. Temperature on GPIOs configured as analog inputs.
    f. Internal circuit temperature.
    g. ADC1A and ADC1B voltage band gap reference.
4. Raw values of fault status registers (listed in ***bcc_fault_status_t***) is printed.

Note that application behavior is closely related to HW and SW setup. It is user obligation to adjust this settings (e.g. OV/UV thresholds, etc.) according to used equipment.

The project uses the following peripherals:

- LPSPI0 – SPI drive communication.
- LPSPI1 (TPL only) – SPI drive communication.
- GPIO – SW controlled SPI chip select, RESET pin (SPI only), INTB and EN pins (TPL only), FAULT pin and red onboard LED
- LPUART1 – serial communication

Pin selection for all mentioned peripherals is listed in Table 1 (SPI) and Table 2 (TPL).

Table 1. Pin selection for BCC SPI boards

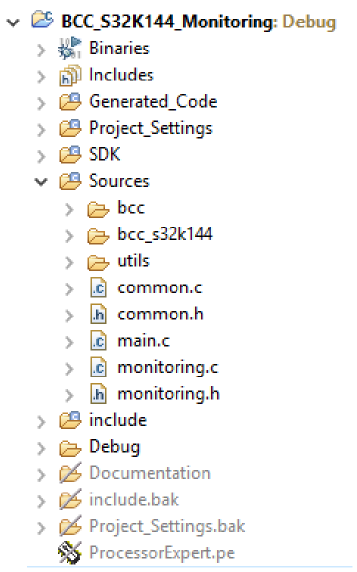
Pin Function	**S32K144** (macro **SPI**)
MISO_ALT0	PTB3/LPSPI0_SIN
MOSI_ALT0	PTB4/LPSPI0_SOUT
SCLK_ALT1	PTB2/LPSPI0_SCK
CSB_ALT1	PTB5
FAULT_ALT1	PTB9
RESET_ALT0	PTD4
Red on-board LED	PTD15
LPUART_RX	PTC6/LPUART1_RX
LPUART_TX	PTC7/LPUART1_TX

Table 2. Pin selection for BCC TPL boards

Pin Function	**S32K144** (macro **TPL** – FRDM33664BEVB connected by wires)	**S32K144** (macro **TPL_TRANSLT** – FRDM33664BEVB connected by translator board)
DATA_TX	PTB16/LPSPI1_SOUT	
SCLK_TX	PTB14/LPSPI1_SCK	
CSB_TX	PTD3/LPSPI1_PCS0	PTE9
DATA_RX	PTB3/LPSPI0_SIN	
SCLK_RX	PTB2/LPSPI0_SCK	
CSB_RX	PTB5/LPSPI0_PCS1	
EN	PTB0	
INTB	PTB1	PTA6
FAULT	PTE8	
Red on-board LED	PTD15	
LPUART_RX	PTC6/LPUART1_RX	
LPUART_TX	PTC7/LPUART1_TX	

Example project consists of several files/folders apart from main.c as you can see in Figure 4:

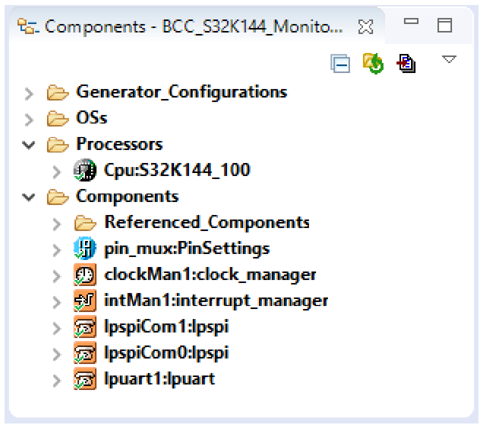
- **bcc** contains files related to BCC SW driver.
- **bcc_s32k144** contains external functions of BCC SW driver related to this example and S32K144 MCU.
- **utils** contains files used for serial communication.
- **main.c** contains functions for setup of used MCU peripherals and functions implementing functionality described above.
- **common.*** files contain common functions and definitions as well as initial BCC device register values.
- **monitoring.*** files contain functions for doing a measurement and printing results to the console.



```
BCC_S32K144_Monitoring: Debug
  Binaries
  Includes
  Generated_Code
  Project_Settings
  SDK
  Sources
    bcc
    bcc_s32k144
    utils
    common.c
    common.h
    main.c
    monitoring.c
    monitoring.h
  include
  Debug
  Documentation
  include.bak
  Project_Settings.bak
  ProcessorExpert.pe
```

Figure 4. Example Code Structure

Additionally configuration of several used MCU peripherals is done by Processor Expert components. Outputs of those are generated configuration structures (in *Generated_Code* folder) which are then passed to initialization functions in *main.c*.



```
Components - BCC_S32K144_Monito...
  Generator_Configurations
  OSs
  Processors
    Cpu:S32K144_100
  Components
    Referenced_Components
    pin_mux:PinSettings
    clockMan1:clock_manager
    intMan1:interrupt_manager
    lpspiCom1:lpspi
    lpspiCom0:lpspi
    lpuart1:lpuart
```

Figure 5. PEx components

Note that implementation of LPSPI driver (from S32K1xx SDK 3.0.0 RTM) was modified inside the provided example project in order to fully support "burst" read functionality of TPL devices. The specific change lies in setting of the TXMSK bit to LPSPIx_TCR register before a slave transfer (LPSPI_DRV_SlaveTransfer function in SDK/platform/drivers/src/lpspi/ lpspi_slave_driver.c file).

Import the Example Project

The following steps shows how to import an example project into S32 Design Studio for ARM Version 2018.R1.

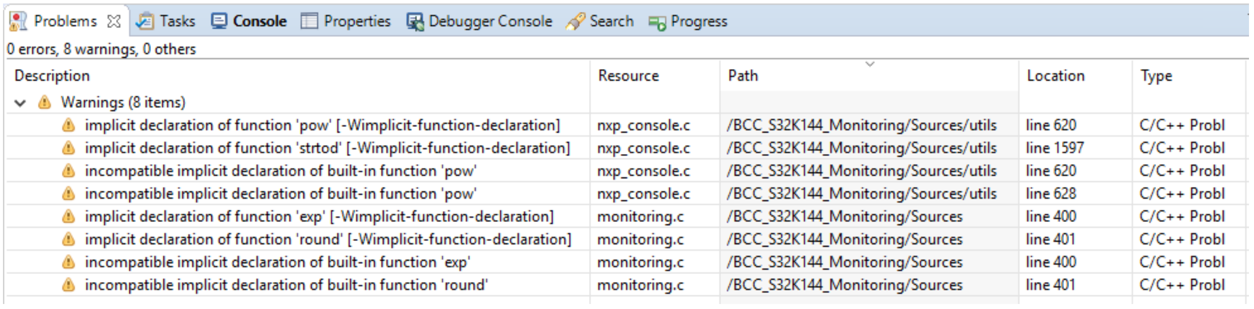
1. In S32 DS click on the ***File / Import***.
2. Choose ***General / Existing Projects into Workspace***.
3. Click ***Browse to select root directory*** with your downloaded example projects.
4. ***Select project*** named **BCC_S32K144_Monitoring** and click ***Finish*** to complete the process.
5. At the top of **Sources/common.h** file define either macro **MC33771** (if MC33771B device is used, defined by default) or **MC33772** directive.
6. At the top of **Sources/common.h** file define either macro **SPI** (in case of SPI EVB board usage, defined by default) **or TPL** (in case FRDM33664BEVB is connected to MCU board by wires) or **TPL_TRANSLT** (in case FRDM33664BEVB is connected to MCU board by translator board). See user manual of BCC SW driver (UM11143) for detailed information about HW configuration possibilities.
7. Now the example project should be in your workspace and ready to run.

Building and Running the Project

In order to build and run the project you need to ***build*** the project in a usual way. If the build is successful, ***debug and run*** the project. This can be accomplished in following steps:

1. Click on the **arrow** next to the **debug icon** and select **Debug Configurations**.
2. **Select** one of the existing configurations with **project name** under **PEMicro** group or **create** one by double clicking on this group.
3. Pick up proper **debug interface** and **USB port**.
4. Apply changes and click on **Debug**.

Note that a first build shows several warnings related to implicit declaration of math functions (Figure 6). These warnings have no impact on example functionality and will disappear after next build.



0 errors, 8 warnings, 0 others

Description	Resource	Path	Location	Type
Warnings (8 items)				
implicit declaration of function 'pow' [-Wimplicit-function-declaration]	nxp_console.c	/BCC_S32K144_Monitoring/Sources/utils	line 620	C/C++ Probl
implicit declaration of function 'strtod' [-Wimplicit-function-declaration]	nxp_console.c	/BCC_S32K144_Monitoring/Sources/utils	line 1597	C/C++ Probl
incompatible implicit declaration of built-in function 'pow'	nxp_console.c	/BCC_S32K144_Monitoring/Sources/utils	line 620	C/C++ Probl
incompatible implicit declaration of built-in function 'pow'	nxp_console.c	/BCC_S32K144_Monitoring/Sources/utils	line 628	C/C++ Probl
implicit declaration of function 'exp' [-Wimplicit-function-declaration]	monitoring.c	/BCC_S32K144_Monitoring/Sources	line 400	C/C++ Probl
implicit declaration of function 'round' [-Wimplicit-function-declaration]	monitoring.c	/BCC_S32K144_Monitoring/Sources	line 401	C/C++ Probl
incompatible implicit declaration of built-in function 'exp'	monitoring.c	/BCC_S32K144_Monitoring/Sources	line 400	C/C++ Probl
incompatible implicit declaration of built-in function 'round'	monitoring.c	/BCC_S32K144_Monitoring/Sources	line 401	C/C++ Probl

Figure 6. Warnings after the first build

If you have any questions related to how to work with debug configurations, see ***S32 Design Studio User's Guide***.