Contents

1.0 Release Information ................................................................. 5
  1.1 Release Summary .................................................................... 5
    1.1.1 Intended Audience .......................................................... 5
    1.1.2 Customer Support ............................................................ 5
    1.1.3 Intel® Quark™ Microcontroller Website ............................ 5
    1.1.4 Intel® Quark™ Microcontroller Open-Source Repository .......... 5
  1.2 Open-Source Software Declaration ........................................ 6
  1.3 Terminology ........................................................................ 6
  1.4 Critical Release Documentation ............................................. 7
  1.5 External Dependencies .......................................................... 10
  1.6 Hardware and Software Compatibility ..................................... 10
  1.7 New Features ...................................................................... 11

2.0 Build/Installation Information .................................................. 12
  2.1 Related Documentation .......................................................... 12
  2.2 Intel® QMSI Build Instructions .............................................. 12
    2.2.1 Build Modes .................................................................. 12
    2.2.2 SoC Selection ............................................................... 12
    2.2.3 SoC Targets ................................................................ 13
    2.2.4 Targets ....................................................................... 13
    2.2.5 Advanced Build Options .............................................. 13
  2.3 Bootloader Build Instructions ................................................ 14
    2.3.1 Target SoC Selection ...................................................... 14
    2.3.2 Firmware Management .................................................. 14
    2.3.3 Return from Sleep .......................................................... 15
    2.3.4 Flash Write Protection .................................................... 15

3.0 Issue Updates ......................................................................... 16
  3.1 Fixed Issues from Previous Release ......................................... 16
  3.2 Known Limitations and Workarounds ...................................... 17

4.0 Change Log ........................................................................... 20
  4.1 Supported Features .............................................................. 20
  4.2 Unsupported or Unimplemented Features ................................. 21
## Revision History

<table>
<thead>
<tr>
<th>Date</th>
<th>Revision</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>April 2017</td>
<td>003</td>
<td>Corresponds to Release 1.4.0 for Intel® Quark™ SE Microcontroller C1000 (ARC, Lakemont) and Intel® Quark™ Microcontroller D2000</td>
</tr>
<tr>
<td>August 2016</td>
<td>002</td>
<td>Corresponds to Release 1.1.0; Gold release for Intel® Quark™ SE Microcontroller C1000 (ARC, Lakemont) and Intel® Quark™ Microcontroller D2000</td>
</tr>
<tr>
<td>February 2016</td>
<td>001</td>
<td>Corresponds to Release 1.0.1; Gold release for Intel® Quark™ SE Microcontroller C1000 (LMT) and Intel® Quark™ Microcontroller D2000</td>
</tr>
<tr>
<td>December 2015</td>
<td>001</td>
<td>Corresponds to Release 1.0.0; Gold release for Intel® Quark™ Microcontroller D2000</td>
</tr>
</tbody>
</table>


1.0 Release Information

1.1 Release Summary

This Intel® release software comprises an Intel® Quark™ Microcontroller Software Interface (Intel® QMSI) and Bootloader Board Support Package (BSP) for two Intel® Quark™ microcontroller Customer Reference Boards (CRBs). The Intel® Quark™ Microcontroller D2000 and Intel® Quark™ SE Microcontroller C1000 CRBs include MCU-Class microprocessors from Intel. The Intel® QMSI BSP comprises drivers for the microcontroller units, internal peripherals, and sample applications.

This document provides important information regarding this release.

To learn more about this product, see the following:

- Features listed in Section 4.1, “Supported Features”.
- Online documentation at the Intel® Quark™ microcontroller website.
- Reference documentation listed in Section 1.4, “Critical Release Documentation”.

1.1.1 Intended Audience

This document is intended for system and application microcontroller developers who develop for Intel® Quark™ microcontroller-based systems and devices.

1.1.2 Customer Support

For technical support, contact us through our product website or your Intel® representative.

1.1.3 Intel® Quark™ Microcontroller Website

Information and support regarding Intel® Quark™ microcontrollers can be found in the following website:

http://intel.com/quark

1.1.4 Intel® Quark™ Microcontroller Open-Source Repository

https://github.com/quark-mcu
1.2 Open-Source Software Declaration

Copyright (c) 2016, Intel Corporation

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted if the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

* Neither the name of Intel Corporation nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

1.3 Terminology

The following acronyms and terms are used in this document (arranged in alphabetic order):

<table>
<thead>
<tr>
<th>Acronym/Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC</td>
<td>Analog-to-Digital Converter</td>
</tr>
<tr>
<td>AON</td>
<td>Always-on</td>
</tr>
<tr>
<td>ARC</td>
<td>Argonaut RISC Core</td>
</tr>
<tr>
<td>CRB</td>
<td>Customer Reference Board</td>
</tr>
<tr>
<td>DMA</td>
<td>Direct Memory Access</td>
</tr>
<tr>
<td>FPR</td>
<td>Flash Protection Region</td>
</tr>
<tr>
<td>GDB</td>
<td>GNU Project Debugger</td>
</tr>
<tr>
<td>Acronym/Term</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>GPIO</td>
<td>General-purpose input/output</td>
</tr>
<tr>
<td>I²C</td>
<td>Inter-INtegrated Circuit</td>
</tr>
<tr>
<td>LMT</td>
<td>Lakemont</td>
</tr>
<tr>
<td>MCU</td>
<td>Microcontroller Unit</td>
</tr>
<tr>
<td>MPR</td>
<td>Memory Protection Regions</td>
</tr>
<tr>
<td>MVIC</td>
<td>Microcontroller Vector</td>
</tr>
<tr>
<td></td>
<td>Interrupt Controller</td>
</tr>
<tr>
<td>OCD</td>
<td>On-chip debugger</td>
</tr>
<tr>
<td>PIC</td>
<td>Programmable Interrupt</td>
</tr>
<tr>
<td></td>
<td>Controller</td>
</tr>
<tr>
<td>PWM</td>
<td>Pulse Width Modulation</td>
</tr>
<tr>
<td>QMSI</td>
<td>Intel® Quark™ Microcontroller</td>
</tr>
<tr>
<td></td>
<td>Software Interface</td>
</tr>
<tr>
<td>RISC</td>
<td>Reduced Instruction Set</td>
</tr>
<tr>
<td></td>
<td>Computing</td>
</tr>
<tr>
<td>ROM</td>
<td>Read Only Memory</td>
</tr>
<tr>
<td>RTC</td>
<td>Real Time Clock</td>
</tr>
<tr>
<td>SoC</td>
<td>System-on-Chip</td>
</tr>
<tr>
<td>SPI</td>
<td>Serial Peripheral Interface</td>
</tr>
<tr>
<td>SS</td>
<td>Sensor Subsystem</td>
</tr>
<tr>
<td>UART</td>
<td>Universal asynchronous</td>
</tr>
<tr>
<td></td>
<td>receiver/transmitter</td>
</tr>
<tr>
<td>WDT</td>
<td>Watchdog Timer</td>
</tr>
</tbody>
</table>

1.4 Critical Release Documentation

<table>
<thead>
<tr>
<th>Component</th>
<th>Location</th>
<th>Revision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firmware Manager</td>
<td>bootloader/fw-manager</td>
<td>1.4</td>
</tr>
<tr>
<td>ROM</td>
<td>bootloader/bootstrap/</td>
<td>1.4</td>
</tr>
<tr>
<td></td>
<td>soc/quark_d2000/rom</td>
<td></td>
</tr>
<tr>
<td></td>
<td>soc/quark_se/rom</td>
<td></td>
</tr>
<tr>
<td>APIC Interrupt Handler</td>
<td>drivers/interrupt/apic.h</td>
<td>1.4</td>
</tr>
<tr>
<td>MVIC Interrupt Handler</td>
<td>drivers/interrupt/mvic.h</td>
<td>1.4</td>
</tr>
<tr>
<td>Newlib SysCalls Code</td>
<td>sys/newlib-syscalls.c</td>
<td>1.4</td>
</tr>
<tr>
<td>ADC</td>
<td>drivers/adc/qm_adc.c</td>
<td>1.4</td>
</tr>
<tr>
<td></td>
<td>drivers/include/qm_adc.h</td>
<td></td>
</tr>
<tr>
<td></td>
<td>examples/quark_d2000/adc</td>
<td></td>
</tr>
<tr>
<td></td>
<td>examples/quark_d2000/adc_continuous</td>
<td></td>
</tr>
<tr>
<td>Component</td>
<td>Location</td>
<td>Revision</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>--------------------------------------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>SS ADC</td>
<td>drivers/adc/qm_ss_adc.c drivers/include/qm_ss_adc.h examples/sensor/adc examples/sensor/adc_continuous</td>
<td>1.4</td>
</tr>
<tr>
<td>AON Counters</td>
<td>drivers/qm_aon_counters.c drivers/include/qm_aon_counters.h examples/aon_counters</td>
<td>1.4</td>
</tr>
<tr>
<td>Comparator</td>
<td>drivers/comparator/qm_comparator.c drivers/include/qm_comparator.h examples/comparators</td>
<td>1.4</td>
</tr>
<tr>
<td>DMA</td>
<td>drivers/dma/qm_dma.c drivers/dma/qm_dma.h drivers/include/qm_dma.h examples/dma</td>
<td>1.4</td>
</tr>
<tr>
<td>Flash</td>
<td>drivers/flash/qm_flash.c drivers/include/qm_flash.h examples/flash_access</td>
<td>1.4</td>
</tr>
<tr>
<td>Flash Protection Regions</td>
<td>drivers/fpr/qm_fpr.c drivers/include/qm_fpr.h examples/fpr</td>
<td>1.4</td>
</tr>
<tr>
<td>GPIO</td>
<td>drivers/gpio/qm_gpio.c drivers/include/qm_gpio.h examples/gpio</td>
<td>1.4</td>
</tr>
<tr>
<td>SS GPIO</td>
<td>drivers/gpio/qm_ss_gpio.c drivers/include/qm_ss_gpio.h examples/sensor/gpio</td>
<td>1.4</td>
</tr>
<tr>
<td>I2C</td>
<td>drivers/i2c/qm_i2c.c drivers/include/qm_i2c.h examples/i2c examples/i2c_slave</td>
<td>1.4</td>
</tr>
<tr>
<td>SS I2C</td>
<td>drivers/i2c/qm_ss_i2c.c drivers/include/qm_ss_i2c.h examples/sensor/i2c</td>
<td>1.4</td>
</tr>
<tr>
<td>Identification</td>
<td>drivers/qm_identification.c drivers/include/qm_identification.h examples/identification</td>
<td>1.4</td>
</tr>
<tr>
<td>Led Blink</td>
<td>examples/blinky</td>
<td>1.4</td>
</tr>
<tr>
<td>D2000 Power States</td>
<td>soc/quark_d2000/drivers/power_states.c soc/quark_d2000/include/power_states.h examples/quark_d2000/power</td>
<td>1.4</td>
</tr>
<tr>
<td>Component</td>
<td>Location</td>
<td>Revision</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------------------------------------------------------------------</td>
<td>----------</td>
</tr>
</tbody>
</table>
| SE Power States | soc/quark_se/drivers/power_states.c  
soc/quark_se/include/power_states.h  
soc/quark_se/drivers/ss_power_states.c  
soc/quark_se/include/ss_power_states.h  
examples/quark_se/power_soc  
examples/quark_se/power_core  
examples/quark_se/power_lpss  
examples/sensor/power_core  
examples/sensor/power_lpss | 1.4      |
| Mailbox     | drivers/mailbox/qm_mailbox_se.c  
drivers/mailbox/qm_mailbox_se_2.c  
drivers/include/qm_mailbox.h  
examples/quark_se/mailbox | 1.4      |
| PIC Timer   | drivers/timer/qm_pic_timer.c  
drivers/include/qm_pic_timer.h  
examples/pic_timer | 1.4      |
| Pin Muxing  | drivers/pinmux/qm_pinmux.c  
drivers/include/qm_pinmux.h | 1.4      |
| PWM / Timers | drivers/pwm/qm_pwm.c  
drivers/include/qm_pwm.c  
examples/pwm  
examples/timers | 1.4      |
| SS Timer    | drivers/timer/qm_ss_timer.c  
drivers/include/qm_ss_timer.h  
examples/sensor/timer | 1.4      |
| RTC         | drivers/rtc/qm_rtc.c  
drivers/include/qm_rtc.h  
examples/rtc | 1.4      |
| Clock Control | drivers/ss_clk.c  
drivers/include/ss_clk.h  
examples/clk_div | 1.4      |
| SPI         | drivers/spi/qm_spi.c  
drivers/include/qm_spi.h  
examples/spi  
examples/spi_slave | 1.4      |
| SS SPI      | drivers/sensor/include/qm_ss_spi.h  
examples/sensor/spi | 1.4      |
| UART        | drivers/uart/qm_uart.c  
drivers/include/qm_uart.h  
examples/uart | 1.4      |
<table>
<thead>
<tr>
<th>Component</th>
<th>Location</th>
<th>Revision</th>
</tr>
</thead>
<tbody>
<tr>
<td>USB</td>
<td>drivers/uart/qm_usb.c</td>
<td>1.4</td>
</tr>
<tr>
<td></td>
<td>drivers/include/qm_usb.h</td>
<td></td>
</tr>
<tr>
<td></td>
<td>examples/quark_se/usb_cdc_acm</td>
<td></td>
</tr>
<tr>
<td></td>
<td>examples/quark_se/usb_dfu</td>
<td></td>
</tr>
<tr>
<td>WDT</td>
<td>drivers/wdt/qm_wdt.c</td>
<td>1.4</td>
</tr>
<tr>
<td></td>
<td>drivers/include/qm_wdt.h</td>
<td></td>
</tr>
<tr>
<td></td>
<td>examples/wdt</td>
<td></td>
</tr>
<tr>
<td>MPR</td>
<td>drivers/mpr/qm_mpr.c</td>
<td>1.4</td>
</tr>
<tr>
<td></td>
<td>drivers/include/qm_mpr.h</td>
<td></td>
</tr>
<tr>
<td></td>
<td>examples/mpr</td>
<td></td>
</tr>
<tr>
<td>BMC sensor driver</td>
<td>board/drivers/bmx1xx/bmc1xx.c</td>
<td>1.4</td>
</tr>
<tr>
<td>Sensor examples</td>
<td>board/drivers/bmx1xx/bmc1xx.h</td>
<td></td>
</tr>
<tr>
<td></td>
<td>examples/accel/main.c</td>
<td></td>
</tr>
<tr>
<td></td>
<td>examples/quark_d2000/magneto/main.c</td>
<td></td>
</tr>
<tr>
<td></td>
<td>examples/sensor/gyro/main.c</td>
<td></td>
</tr>
<tr>
<td>Documentation</td>
<td>doc</td>
<td>1.4</td>
</tr>
</tbody>
</table>

### 1.5 External Dependencies

- i586-intel-elfiamcu toolchain is required to build the packages for the LMT core.
- arc-elf32 toolchain is required to build the packages for the ARC.
- OpenOCD is required to flash applications and ROM files onto the SoC.
- GDB is optional, it is used as a supplement to OpenOCD.
- Intel® System Studio for Microcontrollers is optional but does include all the previous dependencies.
- GNU Make is required to build packages.

### 1.6 Hardware and Software Compatibility

This release has been validated with the following hardware:
- Intel® Quark™ SE Microcontroller C1000.
- Intel® Quark™ SE Microcontroller C1000 Development Platform

This release has been validated on the following operating systems:
- Ubuntu* Linux 14.04 LTS (64-bit)
• Microsoft Windows* 7
• Microsoft Windows* 8.1
• Microsoft Windows* 10

1.7 New Features

The following items are the new features included with this release:

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bootloader</strong></td>
<td>DFU Secure firmware update Secure Firmware Update via UART and USB</td>
</tr>
<tr>
<td><strong>QMSI</strong></td>
<td>SPI Slave SPI Slave Driver</td>
</tr>
<tr>
<td></td>
<td>SoCWatch Instrumentation for SocWatch Energy Analysis for Microcontrollers</td>
</tr>
</tbody>
</table>
2.0 Build/Installation Information

2.1 Related Documentation

- The Intel® Quark™ Microcontroller Software Interface 1.4 document is included as part of this release.

2.2 Intel® QMSI Build Instructions

The build system for the Intel® Quark™ Microcontroller Software Interface is based on the make tool. This section describes the command line interface for the BSP package.

2.2.1 Build Modes

Two different build modes are available when building packages in this release: debug and release. The BUILD variable controls these modes; the default value is release.

To build in debug mode:

```bash
make BUILD=debug
```

To build in release mode:

```bash
make BUILD=release
```

2.2.2 SoC Selection

Two SoCs are available when building packages in this release: quark_se and quark_d2000. The SOC variable controls these modes; the default SoC depends on the application Makefile.

To build for the Intel® Quark™ SE Microcontroller C1000 SoC:

```bash
make SOC=quark_se
```

To build for the Intel® Quark™ Microcontroller D2000 SoC:

```bash
make SOC=quark_d2000
```
2.2.3 **SoC Targets**

When compiling applications for the Intel® Quark™ SE Microcontroller C1000 SoC, there are two separate cores available for compilation: x86 and sensor. The `TARGET` variable controls compilation for these targets; the default Core depends on the application Makefile.

To build for the Lakemont (x86) core:

```
make SOC=quark_se TARGET=x86
```

To build for the ARC:

```
make SOC=quark_se TARGET=sensor
```

2.2.4 **Targets**

The top level Makefile contains one main make target: `libqmsi`. The output directory is build.

Libqmsi is a library archive of all the Intel® QMSI drivers for the SoC.

To build the `libqmsi` target, run the following command from the top level directory:

```
make libqmsi
```

To build any of the provided example apps run `make` inside the corresponding directory or use the `–C` make option from the top level directory.

For example, to build the `hello_world` example app (by default it will be built in release mode):

```
make –C examples/hello_world
```

**Note:** Specific examples have sensible defaults for SOC and TARGETS. In the `hello_world` example, the default is SOC=quark_d2000.

2.2.5 **Advanced Build Options**

Some operating systems may use their own interrupt system instead of the one provided by Intel® QMSI. To properly integrate with those OSs, the ISRs defined in Intel® QMSI drivers should be compiled as regular functions (for example, no interrupt-related prologue and epilogue, no end-of-interrupt handling). So when interrupts are handled externally, you should set 'ENABLE_EXTERNAL_ISR_HANDLING=1' when building libqmsi.
For instance, the following command builds libqmsi for D2000 with external interrupt handling support:

```
make libqmsi SOC=quark_d2000 ENABLE_EXTERNAL_ISR_HANDLING=1
```

## 2.3 Bootloader Build Instructions

The bootloader Makefile supports the following build parameters:

- SOC
- ENABLE_FIRMWARE_MANAGER
- ENABLE_FIRMWARE_MANAGER_AUTH
- ENABLE_RESTORE_CONTEXT
- ENABLE_FLASH_WRITE_PROTECTION

### 2.3.1 Target SoC Selection

Both D2000 and SE C1000 are supported. You can select them by setting the SOC variable.

To build for D2000:

```
make SOC=quark_d2000
```

To build for SE C1000:

```
make SOC=quark_se
```

On the Intel® Quark™ SE Microcontroller C1000, the ARC is not started as part of the ROM flow. The user application / RTOS must start the ARC. In a bare metal environment, to start the ARC, the function `sensor_activation()` must be called. The function is provided by Intel® QMSI in the `ss_init.h` header file.

### 2.3.2 Firmware Management

ENABLE_FIRMWARE_MANAGER is used to enable firmware management inside the bootloader.

To disable firmware manager:

```
make ENABLE_FIRMWARE_MANAGER=none
```

To enable firmware manager over UART:

```
make ENABLE_FIRMWARE_MANAGER=uart
```
To enable firmware manager over USB:

```
make ENABLE_FIRMWARE_MANAGER=2nd-stage
```

To use the firmware manager over USB a 2nd-stage bootloader must be flashed. By default, firmware management mode is enabled over UART.

### 2.3.3 Return from Sleep

The Intel® Quark™ SE has support for sleep states that power off the CPU. When a wake event happens, the CPU starts over from the reset vector as in a normal power on. To do so, build both the bootloader and libqmsi with `ENABLE_RESTORE_CONTEXT=1`:

```
make SOC=quark_se ENABLE_RESTORE_CONTEXT=1
```

The hardware restores the context of the Intel® Quark™ Microcontroller D2000. For that reason, the `ENABLE_RESTORE_CONTEXT` option has no effect on Intel® Quark™ D2000 SoC.

By default, context save and restore management is enabled on Intel® Quark™ SE.

### 2.3.4 Flash Write Protection

By default the bootloader write-protects all the SoC flash memory to avoid any possible modification of the firmware.

It's possible to deactivate this feature by compiling the bootloader with the following command:

```
ENABLE_FLASH_WRITE_PROTECTION=0
```

However, this command renders the SoC vulnerable to malware that could access to the firmware and overwrite it.

```
make SOC=quark_se ENABLE_FLASH_WRITE_PROTECTION=0
```
## 3.0 Issue Updates

### 3.1 Fixed Issues from Previous Release

<table>
<thead>
<tr>
<th>Reference No.</th>
<th>Description</th>
<th>Implication</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>1804657428</td>
<td>SE C1000 SPI / Sensor SPI example may return incorrect CHIP ID.</td>
<td>There is a chance after flashing the example application that the returned value of CHIP ID is incorrect causing the application to fail.</td>
<td>A dummy read is added to the example to force the device into SPI communication mode.</td>
</tr>
<tr>
<td>1804660713</td>
<td>After restoring context on I²C slave, the I²C master reports tx_abort events, but slave receives data.</td>
<td>I²C master reports no ACK bit from slave after restoring context of I²C on slave, when running on SE C1000.</td>
<td>Do not enter low-power modes on SE C1000 when performing I²C slave transfers and using save/restore context functionality.</td>
</tr>
<tr>
<td>1804662808</td>
<td>SPI master transfer speeds are suboptimal when using save/restore builds on SE C1000.</td>
<td>SPI master performance is degraded when using save/restore functionality on SE C1000.</td>
<td>Either reduce transfer speeds on SPI master or disable save/restore context.</td>
</tr>
<tr>
<td>1804660206</td>
<td>Mixing debug and release binaries for Mailbox example causes issues.</td>
<td>When executing Mailbox on SE C1000, if there is a mixture between releases and debug binaries used, unexpected results can occur.</td>
<td>Use either only release binaries or debug binaries.</td>
</tr>
<tr>
<td>1804665427</td>
<td>I²C master terminate TX IRQ fails in release builds.</td>
<td>For D2000, attempting to terminate an IRQ-based TX transfer when operating at FAST_PLUS mode results in failure. For SE C1000 sensor, attempting to abort a multi-master IRQ-based TX transfer operating at STANDARD mode results in failure.</td>
<td>Operate I²C at different speeds or run in debug builds.</td>
</tr>
<tr>
<td>1804344257</td>
<td>Repeated runs of AON counters for SE C1000 may result in failures.</td>
<td>AON periodic timer may fail to have ready bit set if example is run multiple times on SE C1000.</td>
<td>Perform cold reset in-between runs.</td>
</tr>
<tr>
<td>1804663645</td>
<td>SE C1000 DMA freezes if performed after a save/restore cycle.</td>
<td>In debug mode, if DMA is configured after performing a sleep/restore cycle, the system floods with block interrupts and stalls. This renders it unusable.</td>
<td>Disable save/restore context or run in release mode.</td>
</tr>
</tbody>
</table>
## Known Limitations and Workarounds

The current list of known BSP open issues and workarounds includes the following.

<table>
<thead>
<tr>
<th>Reference No.</th>
<th>Description</th>
<th>Implication</th>
<th>Workaround</th>
</tr>
</thead>
<tbody>
<tr>
<td>H/W Issue</td>
<td>UART 0 is reserved on SE C1000 development platform</td>
<td>On SE C1000 development platform, the BLE module reserves UART 0.</td>
<td>Use UART 1 instead</td>
</tr>
<tr>
<td>H/W Issue</td>
<td>I2C transfer speeds have not been validated</td>
<td>Maximum I2C transfer speeds (as advertised in datasheet) have not been validated</td>
<td></td>
</tr>
<tr>
<td>H/W Issue</td>
<td>Last frame of a receive SPI slave transfer appears to be stuck in the RX shift logic if /CS line is never de-asserted</td>
<td>If the SPI master does not de-assert the /CS line after completing the transfer, the SPI slave user application will wait forever for the last frame to come in.</td>
<td>Use two qm_spi_irq_transfer() calls to handle two separate SPI transactions initiated by the SPI master.</td>
</tr>
<tr>
<td>H/W Issue</td>
<td>SPI slave max speed as stated in the datasheet is not achievable in TX mode or TX/RX mode.</td>
<td>Random data is inserted into the transfer for fast transfer speeds.</td>
<td>SPI transfer speeds up to the following values ensure that this issue does not occur: 1.6 MHz for release build 0.8 MHz for debug build</td>
</tr>
<tr>
<td>H/W Issue</td>
<td>SPI Slave controller doesn't generate an interrupt if the /CS line is de-asserted.</td>
<td>The SPI Slave driver cannot detect an incomplete interrupt based SPI transaction (SPI Master terminates the transfer before sending/receiving all data frames). At the moment if the SPI Master prematurely ends the ongoing interrupt based transaction, there is no way for the SPI Slave user application to recover; it will spin and wait forever for the user callback to be called.</td>
<td>Short the /CS line with a GPIO pin. The GPIO can be setup in the application with an interrupt to alert the application that the /CS line has been de-asserted</td>
</tr>
<tr>
<td>1804658157</td>
<td>Bootloader occasionally fails to update over USB.</td>
<td>A small chance that, when using dfu-utils to update an image on SE C1000 SoC, the transfer hangs.</td>
<td>Reinitiate file transfer.</td>
</tr>
</tbody>
</table>
## Issue Updates

<table>
<thead>
<tr>
<th>Reference No.</th>
<th>Description</th>
<th>Implication</th>
<th>Workaround</th>
</tr>
</thead>
<tbody>
<tr>
<td>1804662577</td>
<td>On SE C1000, if mass erase is performed on the ARC flash partition, garbled output can be observed on the UART.</td>
<td>If the ARC flash is mass erased when using save/restore functionality on SE C1000, the UART output displays garbled text.</td>
<td>Perform a cold reset after performing a mass erase of ARC flash.</td>
</tr>
<tr>
<td>1804662992</td>
<td>Cannot use comparator 0 after SE C1000 wakes up.</td>
<td>On SE C1000, after performing a sleep restore cycle, comparator 0 is behaves erratically regarding interrupts.</td>
<td>Use another comparator instead.</td>
</tr>
<tr>
<td>1804662634</td>
<td>Reset button issue with SE C1000 development platform fab E.</td>
<td>On Fab E of SE C1000, if the SoC has been halted and the reset button is pressed, the SoC does not resume properly.</td>
<td>Power down the board and repower it to restore correct operations. Or use the reset button while the SoC is in an active state.</td>
</tr>
<tr>
<td>1804662363</td>
<td>OpenOCD loses connection with ARC when it is in sleep mode.</td>
<td>OpenOCD resume command fails to affect ARC after it has been restored from sleep mode.</td>
<td>After restoring from sleep mode, re-establish OpenOCD connection ARC and then issue resume command.</td>
</tr>
<tr>
<td>1706636084</td>
<td>Setting watchdog on SoC Watch variable in bss causes system reset.</td>
<td>For SE C1000, attempting to set a watchdog on a SoC Watch variable in bss causes the system to reset when the system wakes after sleep.</td>
<td>In this situation, do not set a watchdog on SoC watch variables in bss.</td>
</tr>
<tr>
<td>1804671459</td>
<td>On rare occasions the Bootloader fails to enable USB.</td>
<td>On rare occasions, the USB fails to initialize when the device is powered on.</td>
<td>Reconnecting the device resolves the issue.</td>
</tr>
<tr>
<td>1804668166</td>
<td>jflash does not work properly for sensor application debugging.</td>
<td>Debugging of sensor applications using jflash fails.</td>
<td>Use gdb-arc directly when debugging sensor applications.</td>
</tr>
<tr>
<td>1804789056</td>
<td>Comparators fires a spurious interrupt during power up if being configured for negative polarity at the same time.</td>
<td>If a callback is registered the user will observe the spurious interrupt when setting the pin to be triggered at negative polarity while powering up the pin.</td>
<td>Don't set the polarity to negative immediately when powering up a comparator pin. The following will avoid the issue: 1. Power up the comparator pin and set cmp_en = 0. 2. Wait for 10us. 3. Change the comparator to negative polarity and set cmp_en=1.</td>
</tr>
<tr>
<td>Reference No.</td>
<td>Description</td>
<td>Implication</td>
<td>Workaround</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
<td>-------------</td>
<td>------------</td>
</tr>
</tbody>
</table>
| 1804860107    | The 'accel' example application fails to compile with the Intel(R) Integrated Performance Primitives (IPP) library enabled | When the IPP_LIBRARY_PATH environment variable is set the 'accel' application compilation fails as it cannot find the 'lippsq' library. | In the Makefile change this line:
LDLIBS += -L$(IPP_LIBRARY_PATH)/lib –lippsq
To this:
LDLIBS += -L$(IPP_LIBRARY_PATH)/lib/$(BUILD)/$(SOC)/$(TARGET) -lippsq |
4.0 Change Log

4.1 Supported Features

- ADC Continuous Conversion (Sample Application)
- Always-On (AON) Counters
- Always-On (AON) Periodic Timer
- Analog Comparators
- Analog-to-digital Converter (ADC)
- AON GPIO
- API/IO-APIC Support
- ARC ADC
- ARC General Purpose Input / Output (GPIO)
- ARC Interrupts
- ARC I2C
- ARC ROM
- ARC SPI
- ARC Timer
- ATP Clock Control
- ATP Power States
- BMI 160
- Bluetooth Low Energy
- Clock Control
- Device Manager
- Direct Memory Access (DMA)
- DMA I2C (Lakemont)
- DMA SPI (Lakemont)
- DMA UART (Lakemont)
- Continuous mode ADC conversions
- Firmware Update over UART / USB
- Flash library
• Flash Protection Regions (FPR)
• General-purpose input/output (GPIO)
• Inter-Integrated Circuit (I²C) master
• Inter-Integrated Circuit (I²C) slave
• Interrupt Controller Timer
• Interrupt Controller
• Mailbox (Intel® Quark™ SE Microcontroller C1000 only)
• Memory Protection Regions (MPR)
• Oscillator Trim Codes
• Pin Muxing
• Power states
• Power management for SE C1000
• Pulse Width Modulation (PWM)/Timers
• Real-Time Clock (RTC)
• Retention Alternating Regulator (RAR)
• Serial Peripheral Interface (SPI) master
• Sensor Subsystem (SS) Inter-Integrated Circuit (I²C) master
• Sensor Subsystem (SS) general-purpose input/output (GPIO)
• Sensor Subsystem (SS) Timer
• Sensor Subsystem (SS) Serial Peripheral Interface (SPI)
• System on Chip (SoC) Identification
• Tinycrypt Reference App
• Universal asynchronous receiver/transmitter (UART)
• Universal Serial Bus (USB) 1.1
• Update utilities
• Watchdog Timer (WDT)
• SoCWatch

4.2 Unsupported or Unimplemented Features
• Inter-IC Sound (I2S)