Intel® Quark™ Microcontroller Software Interface (Intel® QMSI)

Malcolm Prinn,
October 2016
Intel® QMSI quick facts

- Hardware Abstraction Layer written in C
- ~100k LoC
- Supports most Intel® Quark™ MCU SoCs
- Small code size / stack usage
- Low power functionality
Intel® Quark™ MCU family

The new Intel® Quark™ microcontroller D1000, Intel® Quark™ microcontroller D2000, and Intel® Quark™ SE microcontroller C1000 for IoT extend Intel's product roadmap to the very edge of the Internet of Things (IoT), enabling a consistent architecture from things to the cloud, with a broad portfolio of Intel products spanning from Intel® Quark™ to Intel® Xeon® processors.

- **Low Power**
  Optimized for low power consumption, such as battery-powered applications

- **Integrated Security**
  With manageability and connectivity to help protect your data at every endpoint

- **Scalable Architecture**
  Maximize investment by reusing software to scale up and down to any Intel processor
### Intel® Quark™ Microcontroller D2000

- Ultra low power, Entry Level
- 32MHz, 32-bit x86 Microcontroller, 32kB Flash, 8kB SRAM
- Scalable Software Development Kit, with sample apps and libraries
- Pre-validated comms and sensor modules
- Full Intel® x86 instruction set architecture for compatibility and scalability

<table>
<thead>
<tr>
<th>Intel® Quark™ SoC</th>
<th>Peripheral Interfaces</th>
<th>Power Subsystem</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Lakemont- ULP @32MHz</td>
<td></td>
</tr>
<tr>
<td>8k SRAM</td>
<td>UART</td>
<td>Pwr. Supply &amp; Distr.</td>
</tr>
<tr>
<td>8k OTP</td>
<td>PWM / Timer</td>
<td>Battery Mgmt.</td>
</tr>
<tr>
<td>32k Intg. Flash</td>
<td>I2C</td>
<td>Power Mgmt.</td>
</tr>
<tr>
<td>2 ch DMA</td>
<td>ADC</td>
<td>JTAG &amp; Debug Support</td>
</tr>
<tr>
<td></td>
<td>Comparator</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPI (M/S)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GPIO</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RTC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WDT</td>
<td></td>
</tr>
</tbody>
</table>

**Optimization Notice**

Copyright © 2016, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.
- High-efficiency power consumption
- 32MHz, 32-bit x86 Microcontroller
- 384Kb Flash, 80kB SRAM
- Scalable Software Development Kit, with sample apps and libraries
- Pre-validated comms and sensor modules
- Full Intel® x86 instruction set architecture for compatibility and scalability
- Always sensing: always-listening Internal Sensor Hub
- Intelligent: Pattern Matching Engine

**Intel® Quark™ SE Microcontroller C1000**

**Intel® Quark™ SoC**
- Lakemont- ULP @32MHz
- ARC @32MHz

**Peripheral Interfaces**
- UART
- I2C (M/S)
- SPI (M/S)
- WDT
- USB 1.1
- ARC @32MHz
- Pattern Matching Engine
- GPIO
- PWM / Timer
- Mailbox
- RTC
- AON GPIO
- Comparator

**Memory Subsystem**
- 80k SRAM
- 8k OTP
- 2x 192kB Intg. Flash
- 8 ch DMA

**Power Subsystem**
- Pwr. Supply & Distr.
- Battery Mgmt.
- Power Mgmt.

**JTAG & Debug Support**
Intel® Quark™ Microcontroller Software Interface

- Intel® Quark™ Microcontrollers Hardware Abstraction Layer (HAL)
- APIs provide a standard interface to all functionality in Intel® Quark™ microcontrollers.
- APIs are consistent across the Intel® Quark™ microcontroller family of devices.
- Included with this API are a collection of sample applications to enable users to get started quickly.
- Embedded devices with a limited amount of memory,
  - Object code size & stack size minimization is a primary design objective.
  - Everything accessed as MMIO.
  - Nothing is dynamic, all the heavy work is done at compile time.
  - Garbage collection on unused functionality.
Bootloader / ROM

- Bootstrap

- TRIM code calculation

- Firmware update management
  - Host tools also provided (fork of DFU utils)

- Un-brick me
Boot flow

- Reset
  - Assembly Start-up
    - Primary peripherals
      - Un-brick me
        - Start ARC
          - Start Lakemont
            - Power down
  - Secondary peripherals
    - Flash controllers
      - Sanitize bootloader
        - Firmware mgmt.
Clocking

Crystal Oscillator
32 MHz
Crystal determines
accuracy
~4mA

Silicon Oscillator
4/8/16/32 MHz
+/- 20000ppm accuracy
450uA@32MHz

RTC oscillator
32768 Hz

System clock source

RTC clock source

I2C Enable

Prescaler

I2C

Prescaler

SPI

Prescaler

RTC

Prescaler

AON Periodic
Timer

Prescaler

WDT
Power states - D2000

- Active
- Halt
- Deep Sleep
- Wake Event
- RST_N
- Sleep Request
Power states - C1000

- Host Processor C-State
  - C1: Interrupt, HLT
  - C0: P_LVL2 Read, Interrupt
  - C2/C2 LP

- Sensor Subsystem SS-State
  - SS1: Sleep Operand 000b, 001b or 010b
  - SS0: Sleep Operand 011b, 100b, 101b, 110b or 111b
  - SS2

- Low Power Sensing Standby
  - Wake Event

- Sensor Subsystem = SS2 & CCU_SS_LPS_EN
  - Host Processor Subsystem = C2/C2LP & CCU_SS_LPS_EN

Optimization Notice
Copyright © 2016, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.
Sample Real Time Clock driver usage

```c
qmem rtc config_t cfg;

/* Configure RTC and request the interrupt. */
cfg.init_val = 0;
cfg.alarm_en = true;
cfg.alarm_val = ALARM_INTERVAL;
cfg.callback = rtc_example_callback;
cfg.callback_data = NULL;
cfg.prescaler = CLK_RTC_DIV_1;

qmem irq_request(QM_IRQ_RTC_0, qmem rtc_isr_0);

/* Enable RTC. */
clk_periph_enable(CLK_PERIPH_RTC_REGISTER);

/* RTC actually starts here. */
qmem rtc_set_config(QM_RTC_0, &cfg);
```

- RTC configuration structure
- Register / request interrupt
- Enable clocking to RTC
- Set configuration and start RTC
Newlib syscalls

- Pico printf
  - Modular, support for formats can be disabled at compile time
  - Supports ‘d’, ‘u’, ‘x’, ‘X’ and ‘s’

- Puts

- Malloc / free

- Assert
Toolchain

Intel® Quark™ microcontroller-based systems

Bare metal / RTOS code

GCC

ARC GCC

OpenOCD based JTAG

GDB

https://software.intel.com/en-us/articles/issm-toolchain-only-download
TinyCrypt Library

- The TinyCrypt Library provides an implementation for constrained devices of a minimal set of standard cryptography primitives.
Intel® Performance Primitives for Microcontrollers

- **Supported data types**
  - Fixed point: q15, q31
  - Floating point: 32f (using floating point simulation)

- **Optimization criteria**
  - Size (~1 KB per function)
  - Performance
  - Accuracy
  - Power consumption

**Functional Groups**

- Basic Math
- Fast Math
- Complex Math
- Statistical Functions
- Filtering Functions
- Transform Functions
- Interpolation Functions
- Matrix Functions
- Controllers Functions
Floating Point Library

- Only applicable to Intel® Quark™ Lakemont processor cores

- Floating Point
  - The Floating Point Library emulates basic floating point operations with hardware integer instructions.
  - Compatible with the Intel® MCU Architecture; supports Intel® Pentium® processor instruction set minus instructions for x87 floating point unit.

- Fixed point data format
  - Intel® IPP for Microcontrollers functions operate on fixed-point data in Qn format.
  - Example: X in the Q15 format is X*2^-15, with the range of supported values for the Ipp16s data type equal to [-1, 1-2^-15].
Intel® System Studio for Microcontrollers

- Bundles components into a Software Development Kit (SDK)
- Integrated Development Environment (IDE)
  - Based on Eclipse
- Intel® System Studio for Microcontrollers plugins
- SoC debugger integration
- Windows USB Driver for:
  - Intel® Quark™ Microcontroller D2000 Development Board
  - Intel® Quark™ SE C1000 Development Board
Intel® system studio for Microcontrollers in action

Code

Disassembly

Registers
Customer Apps

Intel® system studio for Microcontrollers

Zephyr OS (comms. Stacks)

Libs

TinyCrypt

IPPM

Intel® QMSI

Intel® Quark™ MCUs

D2000

C1000
Build upon already existing example applications
Where to get Intel® QMSI™

Github.com/quark-mcu
Intel® Community

Intel® Quark™ Microcontroller Forum: https://communities.intel.com/community/tech/microcontrollers
Legal Disclaimer & Optimization Notice

INFORMATION IN THIS DOCUMENT IS PROVIDED “AS IS”. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

Copyright © 2016, Intel Corporation. All rights reserved. Intel, Pentium, Xeon, Xeon Phi, Core, VTune, Cilk, and the Intel logo are trademarks of Intel Corporation in the U.S. and other countries.

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804