Enabling IoT OSs for Intel Quark MCU Platforms: the fast way

OpenIoT Summit Europe
Andre Guedes
Agenda

• Intel Quark MCU Platforms
• Intel Quark Microcontroller Software Interface (QMSI)
• Zephyr/QMSI Integration
• Contiki/QMSI Integration
• Final Considerations
Intel Quark Microcontrollers
Intel Quark MCU D2000

Processor core
• 32 MHz clock frequency
• Pentium 586 ISA compatible without x87 Float Point Unit

Memory system
• 32 KB Flash
• 8 KB OTP Code
• 4 KB OTP Data
• 8 KB SRAM
Intel Quark MCU D2000

I/O Peripheral
- UART, I2C, SPI, GPIO, PWM, ADC, Analog Comparator, and DMA
- Timer
- RTC, Watchdog, Always-On Counters
Intel Quark MCU Developer Kit D2000

• Small form-factor
• Flash storage
• 6-axis compass/accelerometer
• Temperature sensor
• Arduino-Uno compatible shield interface
Intel Quark SE MCU C1000

Processor core
- 32 MHz clock frequency
- Pentium 586 ISA compatible without x87 Float Point Unit

Memory system
- 384 KB Flash
- 8 KB OTP
- 80 KB SRAM
Intel Quark SE MCU C1000

I/O Peripheral
• UART, I2C, SPI, GPIO, PWM, Analog Comparator, DMA, USB, I2S, Mailbox
Timer
• RTC, Watchdog, Always-on Counters

Sensor subsystem
• ARC EM4 DSP with floating point extension
• I2C, SPI, GPIO, ADC, Timers
Intel Curie Module

- Intel Quark SE MCU C1000
- BLE radio
- Pattern Matching Engine
- 6-axis accelerometer/gyroscope
- Used in Arduino/Genuino 101
Intel Quark Microcontroller Software Interface (QMSI)
QMSI

• Hardware Abstraction Layer for Intel Quark MCUs
• Specifies APIs for all peripherals present in Quark MCUs
• Provides a common API for different MCUs
• Reduces learning curve for new MCUs
• Currently supports Quark D2000 and Quark SE C1000
• Current version: 1.2
QMSI BSP

- Open source implementation of QMSI specification
- Bootloader
- Device drivers for almost all peripherals
  - Missing I2S, SPI and I2C slave mode
  - Static library libqmsi
- Linker scripts and crt0
- Newlib system calls
  - fstat(), write(), and sbrk()
  - pico_printf()
- Bare-metal example applications
QMSI BSP

- BSD 3-clause License
- Drivers available on github.com/quark-mcu/qmsi
- Bootloader available on github.com/quark-mcu/qm-bootloader
QMSI Structure Overview

$ tree -d -L 1
```
|-- board  <- Board level drivers
|-- doc    <- Project documentation and guidelines
|-- drivers <- QMSI drivers
|-- examples <- Example applications using QMSI APIs
|-- include <- Top level headers
|-- soc     <- SoC-specific support
|-- sys     <- crt0 and newlib syscalls
|-- tools   <- Helper tools
`-- usb     <- USB device stack
```
Zephyr/QMSI Integration
Zephyr Overview

• Open source small-footprint RTOS
• Multiple architecture support
  • x86, ARC, ARM, and Nios2
• Networking support
  • Bluetooth Low Energy, IEEE 802.15.4, Ethernet
• IoT communication protocols
  • 6LoWPAN, RPL, CoAP, MQTT
• Supports Quark D2000 and Quark SE C1000
• Apache License version 2.0
• Available in zephyrproject.org
Zephyr/QMSI Integration

Reused components
- Bootloader
- Device drivers*

Not reused components
- Newlib system calls
- Linker script and crt0
- Device drivers
  - Interrupt controller
  - Local APIC Timer
  - Mailbox
Zephyr Device Driver Model Overview

- Unified peripheral API
- Applications access peripherals devices via Zephyr peripheral APIs
  - Top-level include/ directory
- Device drivers implement Zephyr peripheral APIs
  - Top-level drivers/ directory
- Device driver is selected during kernel configuration
Driver Model Architecture

User Application

Zephyr GPIO API

gpio_dw  gpio_k64  gpio_stm32  ...

Hardware
Device Driver Integration

- QMSI shim drivers
  - Shim layer between Zephyr and QMSI
  - Implements Zephyr driver APIs using QMSI APIs
  - Eventually translates Zephyr APIs parameters into QMSI APIs parameters
  - Located in drivers/ directory
- Zephyr has a copy from the latest QMSI release
  - ext/hal/qmsi/ directory
- QMSI drivers are built by Zephyr build system when the corresponding shim driver is selected during kernel configuration
- Optional: link against libqmsi (see CONFIG_QMSI_LIBRARY option)
static int rtc_qmsi_set_config(struct device *dev, struct rtc_config *cfg) {
    qm_rtc_config_t qm_cfg;

    qm_cfg.init_val = cfg->init_val;
    qm_cfg.alarm_en = cfg->alarm_enable;
    qm_cfg.alarm_val = cfg->alarm_val;
    qm_cfg.callback = (void *) cfg->cb_fn;
    qm_cfg.callback_data = dev;

    return qm_rtc_set_config(QM_RTC_0, &qm_cfg);
}
QMSI Shim Driver Architecture

User Application

Zephyr GPIO API

gpio_qmsi
qm_gpio
gpio_k64
gpio_stm32

Hardware
Integration Highlights

- QMSI shim drivers are used by all Quark MCU based boards
  - D2000 CRB, SE C1000 devboard, and Arduino 101
- Shim drivers approach enabled us to rapidly support Intel Quark MCUs peripherals
- Minimum overhead
Contiki/QMSI Integration
Contiki Overview

- Open source OS for the Internet of Things
- Multiple architecture support
  - x86, ARM, 8051, AVR, MSP430
- IoT communication protocols support
- BSD 3-clause License
- Available in contiki-os.org
- Quark D2000 and Quark SE C1000 ports available in github.com/otcshare/contiki-x86
- Pull-request for Contiki upstream (Quark D2000 devkit)
Contiki/QMSI Integration

- Helper script to download and build libqmsi and bootloader
- All BSP components are reused out-of-the-box
  - Bootloader
  - Linker scripts and crt0
  - Newlib syscalls implementation
  - Device drivers
- Contiki links against libqmsi
Contiki Port

Timers Support

- Clock module support
  - Enables Timer, Stimer, Etimer, and Ctimer libraries
  - Located in platform/qmsi-common/core/sys/clock.c
  - Implemented using qm_pic_timer driver
- Rtimer library support
  - Located in platform/qmsi-common/core/sys/rtimer-arch.c
  - Implemented using qm_RTC driver
- Watchdog support
  - Located in platform/qmsi-common/core/dev/watchdog.c
  - Implemented using qm_wdt driver
Contiki Port

I/O Support

- No unified API
  - I/O APIs are platform dependent
- Serial
  - C library API for output
  - `core/dev/serial-line.h` for input
Contiki Port Architecture

- User Application
- Contiki APIs
- Contiki Core
- QMSI Drivers APIs
- Hardware
Integration Highlights

• Initial Quark D2000 port done in 4 small patches
• Initial Quark SE C1000 port done in 2 patches
• Quark SE C1000 port was very straightforward once another QMSI-based port was already in place
  • Clock, rtimer, watchdog, stdout support is shared between both ports
• Initial ports are pretty decent
  • Support for main Contiki subsystems
  • Support for all I/O devices via QMSI APIs
Final Considerations
Final Considerations

• QMSI is the way to go with Intel Quark MCU platforms
  • Abstracts away peripheral device complexities
  • No need to jump into datasheet at first
  • APIs are very simple
  • Good documentation
  • Example applications are very useful
• Reusing QMSI BSP components can accelerate your porting work
• Zephyr and Contiki showcase two different approaches on how to integrate with QMSI
• They can provide guidelines on porting other IoT OSs to Intel Quark MCU platforms
Enabling IoT OSs for Intel Quark MCU Platforms: the fast way

OpenIoT Summit Europe
Andre Guedes