How a Lightweight RTOS can Drive CubeSat Flight Software

Andrew Kalman
Pumpkin, Inc.
Flight Software Requirements

• Reliable - Duh
• Modular – so that multiple coders can work on it simultaneously
• (Re-)configurable – e.g. for testing, or to optimize performance
• Efficient – use a minimum of RAM, Flash, power
• Fast – satisfy responsiveness & throughput requirements
• Capable – permit the use of (all) of the MCU’s hardware / peripherals, without getting in the way
• Clean – enable a consistent programming methodology
What is an RTOS?

- A chunk of software that:
  - Has a well-defined API, clear documentation, etc.
  - Provides a variety of services to build an application on top of it:
    - Scheduling
    - Multitasking
    - Time-based services
    - Inter-process communications
  - Has 'soft' or "hard" real-time performance
  - Is configurable, extensible, etc.
  - Has a user base of > 1

- Pumpkin’s Salvo™ RTOS is a lightweight RTOS designed for embedded MCUs (MSP430, PIC, C8051, etc.)
Non-RTOS vs RTOS Coding

• Non-RTOS applications:
  • Are typically very linear in their coding and execution
  • Typically don’t have scheduling, priorities, etc.
  • Their run-time performance is typically strongly affected by additions / deletions to the code
  • Must often utilize interrupts heavily to achieve a modicum of responsiveness
  • Are initially smaller ... but eventually become larger than the RTOS equivalent

• RTOS applications
  • Are very loosely-coupled
  • Leverage multitasking, priorities and scheduling to maximize responsiveness, minimize load and reduce power consumption
  • Concentrate functionality via a few modules, ultimately reducing code size
  • Have a very consistent look and feel to the code
Who uses (Embedded) RTOSes?

• Some Salvo RTOS applications:
  • Automated shrimp feeders in Patagonia
  • Industrial process controls
  • Health / fitness monitors
  • SDL’s DICE mission
  • Sports watches
  • Bowling lanes
  • Electronic toys
  • Geotagging devices
  • Earth science sensors
  • SSDL’s LMRST-Sat mission
  • All of Pumpkin’s sub-Linux-size embedded controllers

Don’t leave Earth without it
Multitasking

OSInit();

OSCreateTask(task_cmd_do, TASK_CMD_P, 2);
OSCreateTask(task_scpi, TASK_SCPI_P, 1);
OSCreateTask(task_status, TASK_STATUS_P, 3);
OSCreateTask(task_led, TASK_LED_P, 15);
OSCreateTask(task_self_test, TASK_SELF_TEST_P, 5);
OSCreateTask(task_vinti7, TASK_VINTI7_P, 8);

while (1) {
    OSSched();
}

Multitasking
void task_TAP_EPS_temperature(void) {

    TAP_set_name(TAP_ID_EPS_TEMPERATURE, "EPS temperature");
    TAP_set_interval(TAP_ID_EPS_TEMPERATURE, TAP_ID_EPS_TEMPERATURE_INTERVAL_DEFAULT);
    TAP_set_size(TAP_ID_EPS_TEMPERATURE, SIZEOF_TAP_ID_EPS_TELEM_TEMPERATURE);
    TAP_set_action(TAP_ID_EPS_TEMPERATURE, SEND_TAP_SDCARD);
    TAP_set_carton_fn(TAP_ID_EPS_TEMPERATURE, carton_EPS_temperature_fill_TAP):

    while(1) {
        OS_DelayTS(TAP_get_interval(TAP_ID_EPS_TEMPERATURE));
        TAP_push_TAP(TAP_ID_EPS_TEMPERATURE);
        WDT_inc_counter(TAP_ID_EPS_TEMPERATURE);
    }
}
void task_GPS_timeout(void) {

    while(1) {
        OS_WaitBinSem(BINSEM_GPS_TIMEOUT_S_P,OSNO_TIMEOUT);
        OSTryBinSem(BINSEM_GPS_TIMEOUT_E_P);
        OS_WaitBinSem(BINSEM_GPS_TIMEOUT_E_P,GPS_TIMEOUT_TIME);
        if(OSTimedOut()) {
            gps_power(0);
        }
    }
}
char * time_elapsed_DDHHMMSSTT(void) {
    OStypeTick sec, tt;
    int dd, hh, mm, ss;
    static char str[15];

    tt = OSGetTicks();
    sec = (tt / TICKS_PER_SEC);
    dd = (sec / SEC_PER_DAY);
    hh = (sec / SEC_PER_HOUR)
        - (dd * HOUR_PER_DAY);
    mm = (sec / SEC_PER_MIN)
        - (hh * MIN_PER_HOUR) - (dd * MIN_PER_DAY);
    ss = sec - (mm * SEC_PER_MIN) - (hh * SEC_PER_HOUR) - (dd * SEC_PER_DAY);
    tt = tt%TICKS_PER_SEC;
    dd %= 100;
    sprintf(str, "%02d:%02d:%02d:%02d.%02d", dd, hh, mm, ss, tt);

    return str;
}
All it takes is one API call ...

```c
void time_ISR_TimerA0(void) __interrupt[TIMERA0_VECTOR] {
    TACCR0 += SYSTEM_TICK_10ms;
    OSTimer();
    ...
}
```
ISR-to-task Communications

```c
void task_cmd_do(void) {
    unsigned char cmd;

    while (1) {

        OS_WaitSem(SEM_CMD_CHAR_P, OSNO_TIMEOUT);
        if ((cmd=uart1_getchar())) {
            switch (tolower(cmd)) {

                // Help.
                case 'h': case '?':
                    cmd_explain();
                    break;

                ...
            }
        }
    }

    void __attribute__ ((interrupt,no_auto_psv)) _U1RXInterrupt(void) {
        uart1_inchar(ReadUART1());
        OSSignalSem(SEM_CMD_CHAR_P);
    }
```
void OSIdlingHook(void) {
    asm(" PWRSAV #1 ");
}

Sleeping whenever Possible
High Runtime Performance

```c
int main() {

    init();
    i2c1_init();
    I2C1.Msgs_Received = 0;
    SCPI_Init(&scpi_context);
    scpi_cmds = 0;

    [SNIP]

    while (1) {
        if(I2C1.Msgs_PENDING) {
            //SCPI MESSAGE RECEIVED
            OSSignalBinSem(BINSEM_SCPI_RCVD_P);
        }
        if(I2C1STATbits.I2COV) {
            //I2C OVERFLOW -- CLEAR AND RESET I2C
            i2c1_init();
        }
        OSSched();
    }
}
```
Lightweight Footprint

Pumpkin GPSRM 1 v0.3.9 Flash Memory Utilization
(PIC24EP256MC206 w/262,144 bytes Flash)

- Pumpkin GPSRM utility functions [478]
- Microchip 16-bit self-test functions [1066]
- Pumpkin SupMCU utility functions [1250]
- Miscellaneous functions [1294]
- Pumpkin UART1 & UART2 library [1364]
- init(), main() & tasks [2318]
- Pumpkin Salvo RTOS [2360]
- SCPI Command Processing [9816]
- Vinti7 Orbit propagator [27734]
- C library functions [48353]
- Free [166111]
Conclusion

• Pumpkin’s lightweight Salvo RTOS has been used as the basis for flight software on multiple successful CubeSat missions

• A well-designed lightweight RTOS
  • Can have minimal impact on Flash and RAM
  • Can be exceptionally robust (see spaceflight heritage), in part because of its simplicity
  • Provides a wealth of useful features
  • Is conducive to team-based software development
  • Does not “get in the way” of real-time performance
Thank you for attending this Pumpkin presentation at the 2015 CubeSat Developers’ Summer Workshop!

Q&A Session
Notice

This presentation is available online at:

Appendix

• Speaker information
  ▪ Dr. Kalman is Pumpkin's president and chief technology architect. He entered the embedded programming world in the mid-1980's. After co-founding Euphonix, Inc – the pioneering Silicon Valley high-tech pro-audio company – he founded Pumpkin, Inc. to explore the feasibility of applying high-level programming paradigms to severely memory-constrained embedded architectures. He is the creator of the Salvo RTOS and the CubeSat Kit. He holds several United States patents. He is a consulting professor in the Department of Aeronautics & Astronautics at Stanford University and directs the department’s Space Systems Development Laboratory (SSDL). Contact Andrew at aek@pumpkininc.com.

• Acknowledgements
  ▪ Pumpkin’s Salvo, CubeSat Kit, MISC and SUPERNOVA customers, whose real-world experience with our products helps us continually improve and innovate.

• CubeSat Kit information
  ▪ More information on Pumpkin’s CubeSat Kit can be found at http://www.cubesatkit.com/. Patented and Patents pending.

• Copyright notice
  © 2000-2015 Pumpkin, Inc. All rights reserved. Pumpkin and the Pumpkin logo, Salvo and the Salvo logo, The RTOS that runs in tiny places, CubeSat Kit and the CubeSat Kit logo, CubeSat Kit Bus, nanoLab Kit and the nanoLab Kit logo, MISC and SUPERNOVA are all trademarks of Pumpkin, Inc. Don’t leave Earth without it is a service mark of Pumpkin, Inc. All other trademarks and logos are the property of their respective owners. No endorsements of or by third parties listed are implied. All specifications subject to change without notice. Unless stated otherwise, all photographs, images and illustrations are the property of Pumpkin, Inc. and may not be used without permission.

First presented at the Pre-Conference CubeSat Workshop in Logan, Utah on Sunday, August 8, 2015, prior to the 29th Annual AIAA/USU Conference on Small Satellites.