Embedded RTOS Implementation for Twin Nano-Satellite STUDSAT-2

Kamal Lamichhane, Kiran M.
UG Student, Dept. of ECE
Center for Small Satellite Research
Nitte Meenakshi Institute of Technology, Bangalore-560064.
kalmichhane18@gmail.com

Kannan T, Divyanshu Sahay, Ranjith H.G
Center for Small Satellite Research
Nitte Meenakshi Institute of Technology, Bangalore-560064.
kannan.thotumugath@gmail.com

Dr S. Sandya
Project Director, STUDSAT-2
Professor & HOD, Dept. of ECE
Nitte Meenakshi Institute of Technology, Bangalore-560064.
sandya9prasad@gmail.com

***Abstract*—This paper describes the task management and scheduling algorithm for Twin Nano- satellite, "STUDSAT-2". STUDSAT (STUDent SATellite) project is undertaken by a group of students pursuing undergraduate course from INDIA. The mission is to demonstrate inter-satellite communication between two satellites STUDSAT-2A and STUDSAT-2B.Nano-satellites are designed for the dimension 300*300*150 mm weighing 6 kg and 5.5 kg respectively. Developing a reliable and robust Real Time Operating System (RTOS) functioning under the space constraints is of the highest importance. The basic prototype is expanded to operate in sophisticated conditions when system will be in space. The RTOS is developed on ARM CORTEX –M4 platform. Kernel is used from FreeRTOS Inc.**

This paper illustrates the development of RTOS which includes task management, priority assignment and scheduling their functionality. Hybrid configuration system which is combination of interrupt driven and preemptive priority systems is used for task management and scheduling. Starvation in preemptive scheduling is eliminated by the 'Aging' technique i.e. temporarily boosting the priority of a task which is starving for a long time. In this RTOS, direct to task notifications, queues, binary semaphores, counting semaphores, recursive semaphores and mutexes for communication and synchronisation between tasks, or between real time tasks and interrupts is implemented. The thread tick method is used to switch tasks depending on priority and a round-robin scheduling scheme. Task Synchronization technique, Mutual Exclusion - Sleep & Wakeup Events, is implemented in the RTOS which uses notification mechanism for synchronization. Power saving is achieved by the method of tickless mode which is limited by the necessity to periodically exit and then re-enter the low power state to process tick interrupts. Three types of task are defined for the purposed architecture which are Periodic Task, Periodic Update Task and Aperiodic Task. Periodic Task takes action on a regular basis; Periodic Update Task collects data and places it in a global memory area for use by other task on regular basis. Aperiodic Task runs as a result of some external command response from ground station. The real time operating system (RTOS) is designed basically a task-scheduling program that responds to events such as interrupts in real time, i.e. a few milliseconds at most. Flight software architecture is developed using the types of task proposed above which is explained in detail in this paper.

Keywords—STUDSAT; RTOS; Scheduling; Task management; FreeRTOS.

I. INTRODUCTION

The successful launch of STUSAT-1 on 2010 developed in India by undergraduate students of seven engineering colleges across south India inspired Team STUDSAT to come up with a new project STUDSAT-2. Project STUDSAT-2 is an unique twin satellite technology endeavor undertaken by Under-graduate students at Nitte Meenakshi Institute of Technology, Bangalore, India. The main mission objective of Project Studsat-2 is to prove inter- satellite link (ISL) between the two satellites (StudSat-2A and StudSat-2B). STUDSAT-2 hosts three payloads. One is CMOS image sensor for earth imaging. Second is the Inter-Satellite Link (ISL) to demonstrate Inter-Satellite Communication between the two Nano Satellites in space and the third is Drag sail mechanism to De-orbit satellite after mission life. Along with these payloads, each satellite hosts an On-Board GPS Receiver, efficient power system with deployable solar panels, beacon in Morse code, 3-axes attitude stabilization system- reaction–wheels, magnetic torquers, and antenna deployment and various other systems that are required for a successful space mission. This paper describes the embedded RTOS implementation of twin satellite (StudSat-2A and StudSat-2B) . The operational flow provides the functional integration and performance of small satellite that simplifies software development, increase code modularity, and results in maintainable and expandable high-performance that will reduce the effort required to develop real time operating system for STUDSAT-2A/2B Mission.

II. SPECIFICATION OF TWIN SATELLITES

A. Category:	Nano-Satellites
B. Mass:	5 kg each
C. Dimension:	30 x 30 x 15 cm^3
D. Payload:	CMOS Camera (on 2B), ISL, Drag Sail

E. Orbit:

Altitude:	LEO (680 Kms)
Inclination:	98.3^o
Type:	Polar Sun-Synchronous

F. Resolution: 40 m

G. Swath: 40 x 30 km^2

H. Inter-Satellite Link

Frequency:	S band (2.4GHz @ 256Kbps),
Antenna type:	Patch

I. Communication:

Downlink :	UHF (434MHz – 437MHz),
Antenna type:	Monopole
Uplink:	VHF (144MHz-147MHz),
Antenna type:	Monopole
Beacon:	UHF (434MHz – 437MHz),
Antenna type:	Monopole

J. ADCS:

3-Axes Stabilized

Nadir pointing < 1 degree

Sensors:	GPS, Sun sensor, Gyro, Magnetometer
Actuators:	Magnetic Torquer Coils and Reaction Wheel (on 2B)

K. Controller: ARM Cortex M-4

L. Operating System: RTOS

M. Battery: Li-Ion Battery(5.2Ah)

N. EPS Efficiency: Above 85%

O. Solar Panels:

Body mounted:	1(on each 2A &2B)
Deployable:	2(on each 2A &2B)
Solar cells efficiency:	18%

P. Mission lifetime: 1 year

III. SUBSYSTEMS

A. Attitude Determination & Control System

The Attitude Determination and Control System (ADCS) is responsible for maintaining the desired orientation of the satellite in the orbit. The satellite employs a combination of sensors and actuators to achieve the same. The system use sun sensors and magnetometer for data acquisition and magnetic torquer coils and reaction wheel for the active actuation. With the present configuration it is expected to attain a pointing accuracy of ± 1^0 for STUDSAT 2B and ± 6^0 for STUDSAT-2A.

B. ISl Subsystem

Inter-Satellite link is two-way communication path between satellites. The objective is to demonstrate Inter-Satellite Communication between twin Nano Satellites by transmitting payload's image data to ground station and the slave satellite (STUDSAT-2B). Microhard's n2420 OEM radio is used for the communication between the satellites.

C. Payload Subsystem

STUDSAT-2 incorporates some unique application payloads that define the mission and govern the basic flow of the overall working of the satellite. ISL being the prime payload is considered as a separate subsystem because of its vastness. As mentioned above it aims to establish a link between the master and slave satellite in space.

Taking the problem of space debris because of defunct satellites into consideration, the system employs an indigenously developed drag sail, made using Kaplan sheets. With that being used the satellite de-orbits and burns off in space. As mentioned before image payload data has to be transmitted to the ground station (NASTRAC) and the slave satellite. STUDSAT-2B uses a CMOS camera for capturing image.

D. Command & Data Handling Subsystem

Being called as the brain of satellite, this subsystem is responsible for on board control and processing. The Read/Write Non-Volatile Radiation Resistant Memory is used as code memory. A separate Secure Digital card is used for payload and telemetry data. The controller will be supervised

by a watchdog timer to catch runaway programs by resetting the controller.
An indigenously developed Real Time Operating system having kernel memory footprint of 10 Kb is used. The preemptive and cooperative priority based scheduling algorithm is used in the same. The controller provides serial peripheral interface (SPI), I2C, USART, ADC etc to communicate with sub-modules of other sub-systems.

E. On Board Communication

The objective of this system is to establish communication link with ground station. The payload data along with telemetry is transmitted to the ground station using half duplex mode of communication .A customized AX.25 protocol is used for data packeting. The communication system is designed and developed to be well within the constraints of very low power of 2Watt, small size and mass.

F. Electrical Power System

This is the major subsystem which supplies regulated power for proper functioning of all the subsystems. The solar energy is harnessed by the solar panels and converted to electrical energy which is stored in Li-ion batteries (2S 2P configuration). These batteries supply power to other systems during the eclipse. This system consists of three major components, solar arrays, power storage and distribution system and a control unit. The energy is collected from the solar energy available in the orbit through solar cells, and is stored in secondary power sources, which in this case are batteries.

IV. Implementation of RTOS

We chose uVision Keil for implementing the RTOS for our project. Firstly, we listed out the tasks required for the satellite operation. Table 1 illustrates the list of tasks devised for the system. There are three types of task basis.

• Periodic task: This task must take action on a regular basis.

• Regularly update task: This task will collect data and place for regular use by the global storage area for other tasks.

• Aperiodic tasks: This task will be as a result of some external command response, such as running from the results of the ground. The tasks are assigned a priority based on criticality of the task and execution time.

Each task will have a message queue assigned to it and will block on the message queue, with a timeout equal to its period. If the task is aperiodic, it will have an infinite timeout. Using this mechanism, there are two ways to activate a sleeping task: send a message to its message queue, which will be processed immediately (as long as another, higher priority task is not already active) or wait for the timeout period to expire (assuming the task is periodic). The task manager is responsible for context switching between active tasks, based on their priority.

Camera (tCamera): This task is responsible for capturing image .Following threads run in the Camera Task (tCamera):

```
Cam_GetStatus();
Cam_set_mode();
 Cam_TakePicture();
 Cam_Jpeg_Compress();
 Cam_Transmit_Image_to_Buffer();
 Cam_Rec_Image_from_Buffer();
 Cam_LowPower();
 Cam_PowerOK();
 Cam_telemetry.();
 Cam_PictureData();
```

• Intersatellite Link (tISL): This task is responsible for communication with the other satellite through Inter-Satellite Link (2.4Ghz) and establish communication between two satellites. The communication would be master salve configurations. OBC will determine when the data needs to be gathered for the telemetry buffer and when to communicate with the slave satellite. Following Threads run in Intersatellite Link (tISL):

```
ISL_GetStatus();
 ISL_Get_TelemBuffer();
 ISL_Get_IMIBuffer();
 ISL_Get_ImageData();
 ISL_Send_Cross Send();
 ISL_GetResponse();
 ISL_Get_Cross_GPS();
 ISL_End_Link();
```

Similarly task and threads are listed and execute in round robin fashion. The clock frequency is set at 120 MHZ,. Depending upon the criticality of the task and worst case execution time, time scheduling is carried out.

V. Algorithms Used

When STUDSAT-1 was developed the idea of utilizing a real-time operating system and various algorithms did not strike us. However, after its launch we did realize how utilization of a RTOS could have significantly reduced the complexity of the code and saved a lot of time. Algorithms used for the development purpose are Prioritized Preemptive Scheduling and Co-operative Scheduling.

A. Prioritized Preemptive Scheduling

This Prioritized Preemptive Scheduling techniques has following features
• Each task is assigned a priority.
• Each task can exist in one of several states.
• Only one task can exist in the Running state at any one time.
• The scheduler will always select the highest priority Ready state task to enter the Running state. This type of scheme is called ‘Fixed Priority Preemptive Scheduling’. ‘Fixed Priority’ because each task is assigned a priority that is not altered by

the kernel itself (only tasks can change priorities). 'Preemptive' because a task entering the Ready state or having its priority altered will always pre-empt the Running state task if the Running state task has a lower priority.

B. Co-operative Scheduling

A hybrid scheme is utilized where interrupt service routines are used to explicitly cause a context switch. This allows synchronization events to cause pre-emption, but not temporal events. The result is a preemptive system without time slicing.

VI. IMPLEMENTATION OF TASK AND PRIORITY

Each task is assigned a priority from 0 to (configMAX_PRIORITIES - 1). configMAX_PRIORITIES is defined within FreeRTOSConfig.h and can be set on an application by application basis. The higher the value given to configMAX_PRIORITIES the more RAM the FreeRTOS kernel will consume. Low priority numbers denote low priority tasks, with the default idle priority defined by tskIDLE_PRIORITY as being zero. The scheduler will ensure that a task in the ready or running state will always be given processor time in preference to tasks of a lower priority that are also in the ready state. In other words, the task given processing time will always be the highest priority task that is able to run.

Task is created in the following fashion:

```
void emptyTask(void *pvParameters)
{
    char *pcTaskName = "Satellite is booting up\n";
    //To display which task is running
    for(;;)  //Infinite loop
    {
    USART_puts(USART1, pcTaskName);
    //For printing on terminal software through USART1
    Delay ();
    }
}
```

Multitasking in satellite is achieved by using the following sample code utilizing the resources from FreeRTOS.

```
void tUplink (void *pvParameters)
{
        // Variable declarations and other initializations
        const char *pcTaskName = \r\ntUlink task running\r\n;
        uint8_t i=0;
                for( ;; ) {
         // Print out the name of this task.
         USART_puts(USART1, pcTaskName);
        //tUplink code
        }
}
```

```
void tDownlink (void *pvParameters)
{
        // Variable declarations and other initializations
        const char *pcTaskName = \r\ntDownlink task
running\r\n;
        uint8_t i=0;
        for( ;; ) {
         // Print out the name of this task.
         USART_puts(USART1, pcTaskName);
        //tDownlinklink code
         }
    }

int main( void )
{
         init_USART1(9600);
         // initialize USART1 @ 9600 baud
         USART_puts(USART1, "USART Initialization
         complete !\r\n");
         //Task Creation
         xTaskCreate( tUplink, "tUplink",
         configMINIMAL_STACK_SIZE, NULL, 1,
         &hUplink );
         //Note that the priority for both the tasks are set the
         same (which is 1)
         xTaskCreate( tDownlink, "tDownlink",
         configMINIMAL_STACK_SIZE, NULL, 1,
         &hDownlink );
         //This means that on execution, both tasks will be
         running alternately which makes it look like its
         running simultaneously
         //Similarly, any number of tasks can be set the same
         priority for multitasking
         //Execute the task
         vTaskStartScheduler();
         // This should never return.
         // Will only get here if there was insufficient memory
         to create
         // the idle task.
         return 0;
}
```

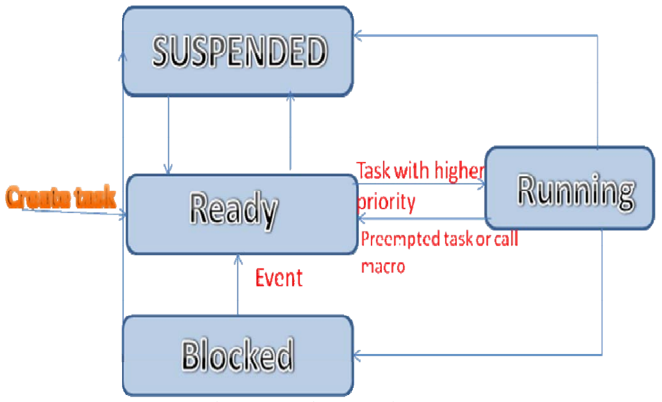


Figure 1 Task State Diagram

VII. RTOS ARCHITECTURE TASK MANAGEMENT PLAN

Following Task has been identified for the proper functionality of the satellite. As different processor is used for Attitude Determination and Control System, tADCS has a separate list of thread which will run in another processor which is exclusively used for attitude control and pointing accuracy of the satellite. tADCS contains Sunsensors, Magnetometer, Magnetic torquer coil, GPS etc. Identified task are listed in Table 1.

Table 1 RTOS Task Interface

Task	Task Name	Timing
Task Manager		Continuous
Attitude Control	tAttControl	Periodic
Attitude Determination (De-tumble)	tAttDetDtum	Periodic
Attitude Determination (Slow)	tAttDetSlow	Periodic
Inter-Satellite Communication	tISL	Periodic
Fault	tFault	Periodic
STUDSAT Beacon	tBeacon	Periodic
Watchdog	tWatchdog	Periodic
Orbital	tOrbital	Periodic
Payload	tPayload	Periodic
Power	tPower	Periodic
Mode Manager	tModeMgr	Periodic
GPS	tGPS	Periodic Update
Camera	tCamera	Periodic Update
Uplink Communications	tUplink	Aperiodic
Downlink Communications	tDownlink	Aperiodic
Mission Timeline Manager	tMTM	Aperiodic
Reprogramming	tReprogram	Aperiodic

VIII. RTOS PERFOMANCE EVALUATION

The hardware platform used in this benchmark is a Cortex-M4 microcontroller, with 1 Mb of Flash, 192Kb of RAM and a 168 MHz clock. The compiler used was the Rowley Cross studio v.5. Time measurements were performed with an oscilloscope that monitors two I/O pins. The same code was used in all tests to set/reset the I/O pins.

Task Switching time: Measures the time to transfer control of one task to another at the same priority level after a call to Yield(), hence, the measured time includes the execution time of the Yield( ) plus the context switch time.

Table 2 Execution testing

Task 1	Task 2	
Pulse pin 1 Yield()	Pulse pin 2 Yield()	Measure time from pulse 1 to pulse 2

Similar measurement has been made for various parameters of RTOS task execution, which are listed as:

Table 3 RTOS Performance

Events/RTOS Metrics	Time in us
Task resumption	0.3
Task suspension	03
Obtaining a semaphore	0.5
Send message to queue	0.8
Set an event	0.5
Allocate memory	0.2
Allocate position	0.4

Memory foot prints:

1. Scheduler- 236 bytes [Reduced by using smaller data types]
2. Queue -76 bytes + queue storage
3. Task – 64 bytes [4character name] + Task stack size 256B
25 task; (64+256)*25= 8Kb, Code size (approx. 100 Kb)
4. Kernel – 10Kb
5. Total memory- 120 Kb

IX. RTOS FLOW & BLOCK DIAGRAM

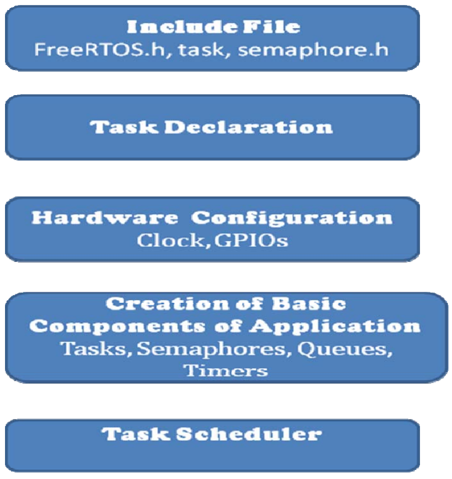
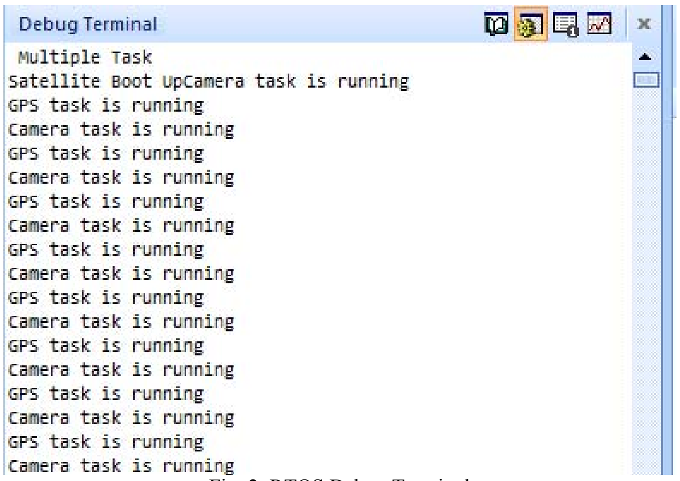

Include File
FreeRTOS.h, task, semaphore.h
Task Declaration
Hardware Configuration
Clock, GPIOs
Creation of Basic Components of Application
Tasks, Semaphores, Queues, Timers
Task Scheduler

Fig. 1. RTOS main Structure

X. RESULTS



```
Debug Terminal
 Multiple Task
Satellite Boot UpCamera task is running
GPS task is running
Camera task is running
GPS task is running
Camera task is running
GPS task is running
Camera task is running
GPS task is running
Camera task is running
GPS task is running
Camera task is running
GPS task is running
Camera task is running
GPS task is running
Camera task is running
GPS task is running
Camera task is running
```

Fig. 2. RTOS Debug Terminal

XI. CONCLUSION

The overall system architecture and requirements of the sub-systems is accomplished by the open-source real-time operating system (FreeRTOS) which supports various types of peripheral interfaces, telemetry storage and data processing. FreeRTOS was chosen for the project after performing several comparison and feasibility studies which have been discussed in the earlier sections. The only drawback of FreeRTOS is that tasks are designed to enter in infinite loops. Satellite function tasks should not be in infinite loop. The tasks should function in the specified conditions without going into infinite loops. To overcome this issue implementation of special techniques to avoid infinite loops is required. Currently, we are working on implementation of certain techniques to avoid infinite loops. This would help to modify the existing FreeRTOS code for STUDSAT-2.

ACKNOWLEDGMENT

The team of Project STUDSAT-2 would like to thank the Management of Nitte Meenakshi Institute of Technology, the lead college for Project STUDSAT, for their wholehearted support. The team would like to thank all the seven college in consortium for providing an opportunity to carry out the project smoothly. The authors sincerely thank all the scientists across various centers of ISRO and ISRO Satellite Centre (ISAC) for their motivation and having provided immense support and motivation towards the project.

REFERENCES

[1] FreeRTOS FAQ - Memory Usage, Boot Times & Context Switch Times, http://www.freertos.org/FAQMem.html

[2] Command and Data Handling Subsystem Design for the Ionospheric Observation Nanosatellite Formation (ION- F) John D. Jensen, Utah State University Dr. Charles M. Swenson (Advisor), Utah State University

[3] Saroj Kumar, Bheema Rajulu, V.V.Sasikiran, Abhijith Janardhan, Amrutha Varshini, ―Attitude Determination and Control System of STUDSAT-2A/2B for Inter-satellite link, UN/Japan workshop and the 4th Nano-Satellite Symposium, Nagoya, Japan, 2012.

[4] B. T. Blaylock, Magnetometers," Spacecraft Attitude Determination and Control, edited by J. R. Wertz, chap. 6.3, Kluwer Academic Publishers, The Netherlands, 1978

[5] Clarke, D. S., M. T. Hicks, A. M. Fitzgerald, J. J. Suchman, R. J. Twiggs, J. Ran-dolf and T. W. Kenny (1996). Picosat Free Flying Magnetometer Experiments. In proc.: 10th Anual AIAA Small Satellite Conference.

[6] C. Nichols, D. Schor, and J. P. Scowcroft, "A Command and Data Handling System for the WinCube Pico-Satellite Mission," April 2008, Undergraduate Capstone Design Project, Dept. of Electrical and Computer Eng., University of Manitoba.

[7] G. Smith (WA4SXM), "AMSAT O-51: Development, Operation and Specifications," Tech. Rep., 2004.

[8] Sethu Selvi, S.; Iyer, N.R.; Sandeep, G.S.P., "A facile approach to GPS navigation in unmanned ground vehicles," Advances in Technology and Engineering (ICATE), 2013 International Conference on , vol., no., pp.1,6, 23-25 Jan. 2013.

[9] DTUsat On Board Computer (OBC) System - DTUsat, http://etd.dtu.dk/thesis/200728/imm5238.pdf

[10] Open source Real time Operating Systems for the STM32 and Cortex m3 MCu's, https://sites.google.com/site/stm32discovery/stm32- resources-and-links/open-source-real-time-operating- systems-for-the-stm32-and-cortex-m3-mpus

[11] Why Use FreeRTOS, http://www.freertos.org/FAQWhat.html#WhyUseRTOS