Understanding the Pratham OBC Flight Code

by

Electrical Subsystem

Student Satellite Lab
Aerospace Department
Indian Institute of Technology, Bombay
Mumbai 400 076

Contents

Chapter 1

OBC Master Flight Code

1.1 Details of Available Hardware

1.2 Constants

- F_CPU - This is the frequency at which the microprocessor is operating. We have used clocks of 8 MHz for all the microprocessors on-board. Uniformity is maintained as a mismatch in operating frequency can cause errors in the communication between two microprocessors.
-

1.3 Header and C Files

1.3.1 comm

Define

Defining centre and radius for communication check. The transmission Regions. Latitude and Longitude of India, France and GS at IIT Bombay, Mumbai, India. The angles defined are -

- IN_LAT 22.5833
- IN_LON 82.7666
- IN_RAD 15
- FR_LAT 48.861101
- FR_LON 2.352104
- FR_RAD 6
- GS_LAT 19.133167
- GS_LON 72.915144
- PHI_MIN_ANGLE 6
- IN 0xAA
- GS 0xBB
- FR 0xCC

Variables

- extern Vector v_sat
- float latitude - calculated by converting the value of v_sat[0] from radians to degrees
- float longitude - calculated by converting the value of v_sat[1] from radians to degrees
- float altitude - calculated by converting the value of v_sat[2] from meters to kilometers
- float dx - check difference in latitude from centre of circle
- float dy - check difference in longitude from centre of circle
- float dis_sq - square of radial distance of the satellite location from the centre of circle
- float thres - threshold for comparing the square of radial distance and selecting whether it is inside the circle or not

- static uint_8 Transmission initialized 0. It takes values "FR", "IN" to be used as flags for transmission over France and India. in every cycle of function comm(), Transmission is assigned the value of variable sat_pos
- static uint_8 downlink_time initialized 0
- uint_8 sat_pos. Takes value returned by function check_satellite_position()

Functions

- uint8_t **check_satellite_position(void)** -Checks whether the position of Satellite is over India/France/Ground-Station and return values accordingly.
  - Return IN 0xAA when Satellite over India in a circle of
  - Return GS 0xBB when Satellite over IIT Bombay, Mumbai, India in a circle of
  - Return FR 0xCC when Satellite over Paris, France in a circle of
  - Return 0 when Satellite is not on India and Paris.
- void **comm(void)** - Performs the various tasks to be performed for communication to ground station including configuring of Transceiver CC1020 through SPI interface and ordering OBC Slave micro-controller to start transfer of appropriate data when over GS, India, or France as per the lat/long values. It uses the check_satellite_position() function.This function has a "watch_dog(T_COMM)" to set the watchdog timer for the task.
  - Return 0 when GPS_done is less than 0 and we are not in Preflight mode.
  - After every 90 seconds send data to OBC Slave HM_DATA using slave_send() function.
  - Once every 10 seconds between two slave_send() function which communicate HM_DATA (that is in 90 second interval, during 8 times), slave_send() sends REAL_TIME when over France or India.
  - It check time and then opens up a window for Uplink to be received by using slave_send() to send END_TX, checking the acknowledgment and then switching off the CC1101
  - It reopens the communication channel by using slave_send() to send BEGIN_TX_GS when transmission is happening over GS and by using slave_send() to send BEGIN_TX_COMM
  -

1.3.2 common

Contains the various constant/macro/struct definitions, clock frequency and general functions.Modes of operation of satellite are listed in this section.

Variables

- F_CPU 8000000 Operating frequency of clock for all onboard microprocessors
- NULL - 0
- Boolean Values Assigned
  - TRUE - 1
  - FALSE - 0
- Modes of Satellite Operation
  - Preflight 0
  - Nominal 1
  - Safe 2
  - Emergency 3
  - Detumbling 4
- Preflight Mode
  - DDR_PF DDRA Port A assigned
  - PORT_PF PINA
  - PIN_PF PA0 Pin to check for preflight check mode
- FRAME_TIME 2 Time frame of 2 seconds for the main loop

Functions

- sbi(x,y) ??
- cbi(x,y) ??
- tbi(x,y) ??

1.3.3 controller

Contains the function for the Control Law.

Variables

- MAG_B 2.5e-5
- K_DETMUBLING 4e4
- TOLW_D2N 4e-3
- TOLW_n2D 8e-3
- D_TIME 2000
- N2D_TIME 10000
- M_MAX 0.95
- N_TURNS 60
- AREA 0.0442
- PWM_RES 0XFFFF
- I_MAX 0.273
- static vector v_B
- static vector v_w = 0.0, 0.0, 0.0
- static vector v_ieu = 0,0,0
- quaternion q_o;
- static uint8_t first_B = 1
- static uint8_t light
- static uint8_t w_ctrl = 0
- int flag = 1
- int tol_time = 30
- int switch_n2d_tol_time = 0
- double mod_w
- int flag_N = 0

- int flag_D = 0
- uint64_t t_now = 0
- int8_t star_flag=0
- int16_t start_time=0
- double norm_m_d=0
- double md_start=0
- double avg_md=0
- double avg_md_new=0
- uint16_t num=2
- uint16_t flag_on=0
- double t_orbit = 5676.599
- double eclipse_time = 2270.6
- double gps_start = 2270.6
- double light_dur = 3406
- double gps_max = 34060
- int gps_power= 0
- int16_t time_e = 0
- int16_t time_l = 0
- vector v_m_temp=0,0,0
- static matrix m_Kp
- static matrix m_Ki
- static matrix m_Kd

Functions

- void control(void)
- detumbling(vector v_m_D)
- quaternion q_o
- nominal(vector v_m_M)
- apply_torque(vector v_m)
- control(void)

1.3.4 frame

Implements Frame Transformations

Variables

- STPERUT - 1.00273790935
- W_EARTH_ROT - 7.294070543852040e-5
- A - 6378137.0
- B - 6356752.314245
- F - 0.003352811
- E2 - 0.006694381
- EP2 - 0.006739498
- extern uint64_t seconds_since_equinox
- extern uint64_t seconds_since_pivot

Functions

- uint64_t days_in_months(uint8_t month)
- uint64_t get_seconds_since_pivot(uint16_t year, uint8_t month, uint8_t date, uint8_t hours, uint8_t minutes, uint8_t seconds)
- void get_seconds_since_equinox(void)
- void ecef2eci(vector v_ecef, vector v_eci)
- void eci2ecef(vector v_eci, vector v_ecef)
- void eci2orbit(vector v_r, vector v_v, vector v_eci, vector v_orbit)
- void ecef2lla(vector v_ecef, vector v_lla)
- void ned2ecef(vector v_ned, vector v_lla, vector v_ecef)

1.3.5 gps

Functions to read the GPS device

Variables

- volatile static uint32_t buffer = 0 - 4-byte buffer for the GPS reading
- Position variables for the data in GPS structure
  - volatile static uint8_t pos = 0xFF
  - volatile static uint8_t vel = 0xFF
  - volatile static uint8_t dop = 0xFF
  - volatile static uint8_t geo = 0xFF
  - volatile static uint8_t time = 0xFF
- Variables to check whether the message has ended
  - volatile static uint8_t last_byte
  - volatile static uint8_t message_end
- volatile struct GPS_reading gps - Temporary GPS reading
- char arrayx[40]
- char arrayy[40]
- char arrayz[40]

Functions

- void init_UART_GPS(void) UART initialised for communication with GPS. Double Baud Rate setting switched on. Frame Format- 8 bit Data Byte, 2 stop bits. Deafult values are assigned to variables in gps struct in current_ state struct.
- ISR interrupt routine for USART to receive data from GPS This interrupt routine is responsible for receieving the data from GPS, sorting the data and storing it in appropriate variables for further processing.

1.3.6 hm

Contains functions that are required for health monitoring data acquisition.

Variables

- struct HM_data - Captures load bytes as obtained from power for health monitoring.
  - uint8_t LoadStatus;
  - uint8_t BatteryCurrent;
  - uint8_t BatteryVoltage;
  - uint8_t Panel;
  - uint8_t SixVolt;
  - uint8_t ThreeVolt;
  - uint8_t BatteryStatus;
- ADDR - 0x22 - I2C slave address of the power MuC
- Load Status Bits
  - BEACONPOWER - 7
  - PCONTROL - 6
  - PGPS - 5
  - PCC - 4
  - POBC - 3
  - PMAG - 2

Functions

- void get_HM_data(void) Get the health monitoring data from the Power Microcontroller using the I2C interface.
- void send_loads(void) Send the CommandByte to power uC after suitable modifications

1.3.7 igrf

Contains the various constant/macro/struct definitions and functions for Magnetic Field Estimation

Variables

- RE - 6371.2
- A2 - 40680631.59
- B2 - 40408299.98
- IGRF_YEAR - 2015
- static float p[50]
- static float q[50]
- static float cl[15]
- static float sl[15]
- const static float agh[196]? progmem ?
- const static float dgh[196]? progmem ?
- vector v_lla
- float years
- uint8_t order
- vector v_B_ned
- double lat=v_lla[0]
- double lon=v_lla[1]
- double alt=v_lla[2]/1000
- double x
- double y
- double z
- double one
- double two

- double three
- double four
- double slat=sin(lat)
- double clat=cos(lat)
- double cd
- double sd
- double ratio
- double r
- double rr
- double t=years-IGRF_YEAR
- double agh_p
- double dgh_p
- uint8_t l
- uint8_t m
- uint8_t n
- uint8_t max
- uint8_t k
- uint8_t fn
- uint8_t fm

Functions

- void igrf(vector v_lla, float years, uint8_t order, vector v_B_ned)

1.3.8 mag

Read the magnetic field vector from the Magnetometer

Variables

- uint8_t mag_count = 0
- uint8_t mag_data[7]
- volatile stati int16_t x
- volatile stati int16_t y
- volatile stati int16_t z

Functions

- void init_UART_MM(void) - Initializes UART at 9600 bps
- void send_MM_cmd(char *data) - Sends magnetometer command
- void poll_MM(void) - Poll the Magnetometer for B readings
- void poll_MM1(void)
- uint8_t receive_MM(void) - Receive byte through UART
- Interrupt routine for UART1 for Magnetometer

1.3.9 mathutil

Contains the various constant/macro/struct definitions and functions for matrix/vector manipulation

Variables

- typedef double vector[3]
- typedef double quaternion[4]
- typedef double matrix[3][3]

Functions

- void copy_vector(vector v_src, vector v_dest)
- void copy_quaternion(quaternion q_src, quaternion q_dest)
- double vector_norm(vector v)
- double quatrenion_norm(quaternion q)
- double vector_dot_product(vector v_a, vector v_b)
- void add_vectors(vector v_a, vector v_b, vector v_res);
- void vector_into_matrix(vector v, matrix m, vector v_res)
- void vector_cross_product(vector v_a, vector v_b, vector v_res)
- void scalar_into_vector(vector v, double s)
- void scalar_into_quaternion(quaternion q, double s)
- void convert_unit_vector(vector v)
- void convert_unit_quaternion(quaternion v)

1.3.10 peripherals

Variables

- struct GPS_reading
  - int32_t x
  - int32_t y
  - int32_t z
  - int32_t v_x
  - int32_t v_y
  - int32_t v_z
  - int32_t lat
  - int32_t lon
  - int32_t alt
  - uint8_t hours
  - uint8_t minutes
  - uint8_t seconds
  - uint8_t date
  - uint8_t month
  - uint16_t year
  - uint16_t pdop
  - uint8_t gps_OC
  - uint8_t gps_power_main
  - uint16_t time_since_reading
- struct SS_reading - Capture Sun Sensor reading
  - uint16_t reading[6]
  - double read[6]
- struct MM_reading - Captures Magnetometer readings
  - float B_x
  - float B_y
  - float B_z
- struct PWM_values

  - uint16_t x
  - uint16_t y
  - uint16_t z
  - uint8_t x_dir
  - uint8_t y_dir
  - uint8_t z_dir

- struct state
  - struct GPS_reading gps
  - struct SS_reading ss
  - struct MM_reading mm
  - struct HM_data hm
  - struct PWM_values pwm
- extern volatile struct state Current_state
- extern uint64_t Time
- extern uint8_t Mode
- extern uint8_t Mode_prev
- volatile uint8_t GPS_done
- uint8_t PWM_init=0

Functions

- void power_up_peripheral(uint8_t device)
- void power_down_peripheral(uint8_t device)
- void read_GPS(void)
- void read_SS(void)
- void read_MM(void)
- void configure_torquer(void)
- void set_PWM(void)
- void reset_PWM(void)

1.3.11 propagator

Functions and constants for SGP propagator

Variables

- R_E2 40680631590769.0
- J2 1.08263e-3
- GM 3.98658366e+14
- EPSILON 0.4102
- SECONDS_IN_YEAR 31536000.0
- static vector v_r
- static vector r_ecef_ash
- static float frtm=0
- vector v_sat
- extern volatile struct GPS_reading gps

Functions

- void copy_gps_reading(void)
- void sgp_get_acceleration(vector v_g)
- void sgp_orbit_propagator(void)
- void sun_vector_estimator(vector v_sun_o)
- void magnetic_field_estimator(vector v_B_o)

1.3.12 quest

Variables

- SS_GAIN 2.3
- SS_MAX_ANGLE 60
- MAG_WEIGHT 0.9
- N_SS 6
- TAU_W 60
- A_F 0.032258064516129031
- static quartenion q_B_old
- static vector v_w_old

Functions

- uint8_t quest(vector v_B_c, vector v_sun_c, quaternion q_triad, uint8_t * p_w_ctrl)
- void omega_estimation(quaternion q_B, vector v_w)
- uint8_t light_cal

1.3.13 slave_comm

Contains all information/functions used to communicate between master and slave muCs.

Variables

- HM_DATA 0x00
- REAL_TIME 0b11010101
- BEGIN_TX_GS 0b01010101
- BEGIN_TX_COMM 0b01011010
- END_TX 0b00110011
- END_SPI 0b10101010
- N_END_SPI 2
- ACK 0b10010010

Functions

- void slave_send (uint8_t command, char* buffer, uint8_t size)

1.3.14 spi

Contains the various pin/port definitions and procedures for spi.

Variables

- SPIDI PB3
- SPIDO PB2
- SPICLK PB1
- PORT_CS PORTB
- DDR_CS DDRB
- SLAVE PB0
- CC1020 PB4
- ADC_S PB5

Functions

- void init_SPI_slave(void)
- void init_SPI(void)
- void init_SPI_trans(void)
- void SPI_send(char* str, uint8_t size)
- uint8_t SPI_transfer(uint8_t transmit_byte)

1.3.15 sun

Functions to read the sunsensor

Variables

Functions

- void configure_SS(void)
- void poll_SS(void)
- void transmitSunSensorUART(int temp)
- uint16_t convert(uint8_t vall, uint8_t valh)
- void poll_SS1(void)

1.3.16 timer

All functions related to frame timer and watchdog.

Variables

- T_CONTROL WDT0_1S
- T_POWER WDTO_500MS
- T_COMM WDTO_1S
- TIMER_TWO_SEC 15624

Functions

- void timer_reset_two_sec(void)
- void watch_dog(int time)
- void timer_wait_reset(void)

1.3.17 uart

Contains the various pin/port definitions and procedures for uart.

Variables

-

Functions

- void init_UART0(void)
- void init_UART1(void)
- void transmit_UART1(char data)
- void transmit_UART0(uint8_t data)
- uint8_t receive_UART1(void)
- uint8_t receive_UART0(void)
- void transmit_string_UART0(char *buffer)
- void transmit_string_UART1(char *buffer)

1.3.18 xyz

This is the main function of the flight code of Master OBC. Not all the functions which had been listed till now as a part of other header files have been used in this code. Some of the codes are rewritten again.

Variables

- volatile uint8_t GPS_done = -1
- uint8_t Mode = DETUMBLING
- uint8_t Mode_prev = DETUMBLING
- uint64_t Time
- volatile struct state Current_state;
- unsigned char write_data=0x4C// OBC, Torquer, Magmeter
- unsigned int CyclesToCollectData = 1
- unsigned char recv_data
- uint8_t address=0x20, read=1, write=0
- unsigned int UniversalCycles = 1
- unsigned int counter1 = 0 //Beacon OverCurrent controller
- unsigned int counter2 = 0 //Control OverCurrent controller
- unsigned int counter3 = 0 //GPS OverCurrent controller
- unsigned int counter4 = 0 //Downlink OverCurrent controller
- char HM_Data[7] //Array for HM Data
- uint8_t FirstTimeOuter = 0
- uint8_t FirstTimeInner = 0
- uint8_t FirstTimeNormal = 0
- char GPS_Data[13] = "IamUndefined"
- uint8_t countd=0,countu=0
- uint8_t light_main = 1
- uint8_t flag_india = 0

- uint8_t flag_france = 0
- uint8_t flag_mumbai = 0

Functions

- Two Wire Interface (I2C)
  - void TWI_init_master(void) // Function to initialize master for I2C
  - void TWI_start(void) // //Function to send I2C start command
  - void TWI_repeated_start(void) // Function to send I2C repeated start command. Scarcely used
  - void TWI_write_address(unsigned char data)//Function for Master side to send slave address for I2C
  - void TWI_read_address(unsigned char data) //Function for slave side to read address sent by Master
  - void TWI_write_data(unsigned char data)//Function to write data on I2C data line
  - void TWI_read_data(void) //Function to read data from I2C data line
  - void TWI_stop(void)//Function to stop data transmission
- void SendHM(void) HM data sent to Slave OBC using SPI interface. 7 bytes of HM data is sent.
- int main(void) : Main function of the Master OBC loop
  - Initialization : SPI for Slave and ADC. SLave pins are selected. UART for GPS and Magnetometer TWI for Power Torquers are configured LED on port A are used as indication. Interrupts are enabled "Time" variable is initialized
  - While Loop
    - Reset timer for 2 second loop
    - Change write_data as per mode
    - Set slave select of ADC=1 and Slave OBC=0
    - "write_data" operations- this one byte flag has 8 single bit flags which are used by Power to switch loads on and off.
    - Communicate with Power using TWI , "write_data" , watchdog loop used.
    - Receive HM data from Power using TWI. It has 7 bytes.
    - 12 uint8_t bytes are processed. Struct is used for data manipulation. 2 bytes for latitude, 2 for longitude, 4 for time and 4 for quaternions. This data is flushed in to the GPS_Data array.

∗ Check for circles. Flags are used for checking the condition. Accordingly SLave OBC is instructed to start downlink. HM data and optionally based on location GPS data is sent. Later slave is instructed to go back to normal mode. In normal mode HM is sent to EEPROM every 20 seconds otherwise every 2 seconds.

∗ Cycle for Uplink 30 seconds and Downlink 120 seconds.

∗ Control function initiated in a watchdog loop.

∗ Transmit on UART0 the GPS_Data array - 12 bytes

∗ Transmit on UART0 the HM_Data array - 7 bytes

∗ Light data is updated. light_cal function is used to sore updates value in light_main flag

∗ Torquers are checked and reset in case of overcurrent.

∗ variable "Time" is incremented by "FRAME_TIME"

∗ Variables containing information about circles is updated. There are four cycles - India outer circle, France , India inner cicrle and Mumbai

∗ variable UniversalCycles incremented by 1

∗ timer_wait_reset restarts timer function timer_reset_two_sec.

–

1.4 Control flow in the code

Chapter 2

OBC Slave Flight Code

Chapter 3

Some Salient Features and Design Considerations

- A common header file for certain universal declarations has been used, named as **”common.h”**
- UART line from preflight and GPS is branched together and connected to UART of ATmega128 OBC Master.