소스코드는 다음과 같다.

```
/*****************************************************
This program was produced by the
CodeWizardAVR V2.03.4 Standard
Automatic Program Generator
?Copyright 1998-2008 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project :
Version :
Date    : 2010-10-06
Author  :
Company :
Comments:


Chip type           : ATmega2560
Program type        : Application
Clock frequency     : 16.000000 MHz
Memory model        : Small
External RAM size   : 0
Data Stack size     : 2048
*****************************************************/

#include <mega2560.h>
#include <stdio.h>
#include <stdlib.h>
#include <delay.h>
#include <string.h>
#include <math.h>


unsigned char cmd[6];
unsigned char C328_ACK_data[256];
unsigned char PIC_data[256];

unsigned int g_nRxHead_0=0;

unsigned char GS0 =  '0' ;

#define RXB8 1
#define TXB8 0
#define UPE 2
#define OVR 3
#define FE 4
#define UDRE 5
#define RXC 7

#define FRAMING_ERROR (1<<FE)
#define PARITY_ERROR (1<<UPE)
#define DATA_OVERRUN (1<<OVR)
#define DATA_REGISTER_EMPTY (1<<UDRE)
#define RX_COMPLETE (1<<RXC)

#define DQ_HIGH (PORTD |= 0x01)
#define DQ_LOW (PORTD &= ~0x01)

void Boot_OBC(void);
void Boot_SD(void);

//uart I/O function
void put_string_0(unsigned char str[]);
void put_flashstring_0(flash unsigned char *str);
void put_string_3(flash unsigned char *str);
void uart0_send_int(int x);
void uart3_send_int(int x);

//wait for SD card.
void wait_message(void);

//functions for CAM
```

```
void C328_send_command(unsigned char *cmd);
void sync(void);
void ACK_sync(void);
void init_C328(void);
void set_package_size(void);
void snapshot(void);
void get_picture(void);
void package(char n);
void lastACK(void);
void connect(void);
void Boot_CAM(void);

void takePIC(int);  //Main Operation

void gpsparsing(void);   // GPS

//functions for Thermometer
unsigned char DS18S20_reset(void);
unsigned char read_data(void);
void write_data(unsigned char data);
void disp_data(unsigned int sign_byte, unsigned int temp_byte);
void read_temp(void);

void nominal_telemetry(void);

// USART0 Receiver buffer
#define RX_BUFFER_SIZE0 8
char rx_buffer0[RX_BUFFER_SIZE0];

#if RX_BUFFER_SIZE0<256
unsigned char rx_wr_index0,rx_rd_index0,rx_counter0;
#else
unsigned int rx_wr_index0,rx_rd_index0,rx_counter0;
#endif

// This flag is set on USART0 Receiver buffer overflow
bit rx_buffer_overflow0;

// USART0 Receiver interrupt service routine
interrupt [USART0_RXC] void usart0_rx_isr(void)
{
char status,data;
status=UCSR0A;
data=UDR0;

if (data=='1')   // Pic download via Radio Frequency
{
GS0='1';
}

if (data=='2')   // Call GPS coordiante at anytime

{
nominal_telemetry();
data='0';
}


if (data=='3')  // Initiate Microcontroller at anytime
{
Boot_OBC();
data='0';
}


if (data=='4')  // command for line-cut. release current on hot-wire.
{
PIND.4=1;
put_flashstring_0("₩r₩n CutCutCut ₩r₩n");

}
```

```
if (data==’5’)
{
// Port D initialization
// Func7=In Func6=In Func5=In Func4=Out Func3=In Func2=In Func1=In Func0=Out
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTD=0b00000000;
DDRD=0b00010001;

put_flashstring_0( “₩r₩n Cut END ₩r₩n” );        // command for line-cut, shut current on hot-wire.
data=’0’;
}


if ((status & (FRAMING_ERROR | PARITY_ERROR | DATA_OVERRUN))==0)
   {
   rx_buffer0[rx_wr_index0]=data;
   if (++rx_wr_index0 == RX_BUFFER_SIZE0) rx_wr_index0=0;
   if (++rx_counter0 == RX_BUFFER_SIZE0)
      {
      rx_counter0=0;
      rx_buffer_overflow0=1;
      };
   };
}

#ifndef _DEBUG_TERMINAL_IO_
// Get a character from the USART0 Receiver buffer
#define _ALTERNATE_GETCHAR_
#pragma used+
char getchar(void)
{
char data;
while (rx_counter0==0);
data=rx_buffer0[rx_rd_index0];
if (++rx_rd_index0 == RX_BUFFER_SIZE0) rx_rd_index0=0;
#asm( “cli” )
--rx_counter0;
#asm( “sei” )
return data;
}
#pragma used-
#endif


// Write a character to the USART0 Transmitter
#pragma used+
void putchar0(unsigned char c)
{
while ((UCSR0A & DATA_REGISTER_EMPTY)==0);
UDR0=c;
}
#pragma used-

// Get a character from the USART1 Receiver
#pragma used+

unsigned char usart1_rx_isr(void)
{
unsigned char status,data;
while (1)
      {
      while (((status=UCSR1A) & RX_COMPLETE)==0);
      data=UDR1;
      if ((status & (FRAMING_ERROR | PARITY_ERROR | DATA_OVERRUN))==0)
         return data;
      };
}
#pragma used-

// USART2 Receiver buffer
#define RX_BUFFER_SIZE2 8
unsigned char rx_buffer2[RX_BUFFER_SIZE2];
```

```
#if RX_BUFFER_SIZE2<256
unsigned char rx_wr_index2,rx_rd_index2,rx_counter2;
#else
unsigned int rx_wr_index2,rx_rd_index2,rx_counter2;
#endif

// This flag is set on USART2 Receiver buffer overflow
bit rx_buffer_overflow2;

// USART2 Receiver interrupt service routine
interrupt [USART2_RXC] void usart2_rx_isr(void)
{
unsigned char status,data;
status=UCSR2A;
data=UDR2;
#asm( "cli" )
C328_ACK_data[g_nRxHead_0++]=data;
#asm( "sei" )

if ((status & (FRAMING_ERROR | PARITY_ERROR | DATA_OVERRUN))==0)
   {
   rx_buffer2[rx_wr_index2]=data;
   if (++rx_wr_index2 == RX_BUFFER_SIZE2) rx_wr_index2=0;
   if (++rx_counter2 == RX_BUFFER_SIZE2)
      {
      rx_counter2=0;
      rx_buffer_overflow2=1;
      };
   };
}

// Get a character from the USART2 Receiver buffer
#pragma used+
unsigned char getchar2(void)
{
unsigned char data;
while (rx_counter2==0);
data=rx_buffer2[rx_rd_index2];
if (++rx_rd_index2 == RX_BUFFER_SIZE2) rx_rd_index2=0;
#asm( "cli" )
--rx_counter2;
#asm( "sei" )
return data;
}
#pragma used-

// Write a character to the USART2 Transmitter
#pragma used+
void putchar2(unsigned char c)
{
while ((UCSR2A & DATA_REGISTER_EMPTY)==0);
UDR2=c;
}
#pragma used-

// USART3 Receiver buffer
#define RX_BUFFER_SIZE3 8
unsigned char rx_buffer3[RX_BUFFER_SIZE3];
unsigned char fRX=0;

#if RX_BUFFER_SIZE3<256
unsigned char rx_wr_index3,rx_rd_index3,rx_counter3;
#else
unsigned int rx_wr_index3,rx_rd_index3,rx_counter3;
#endif

// This flag is set on USART3 Receiver buffer overflow
bit rx_buffer_overflow3;

// USART3 Receiver interrupt service routine
interrupt [USART3_RXC] void usart3_rx_isr(void)
```

```
{
unsigned char status,data;
status=UCSR3A;
data=UDR3;

if (data==’O’) // OK 응답: 영문대문자 O (0x4F)  // for SD card
{
fRX=1;
}

if ((status & (FRAMING_ERROR | PARITY_ERROR | DATA_OVERRUN))==0)
   {
   rx_buffer3[rx_wr_index3]=data;
   if (++rx_wr_index3 == RX_BUFFER_SIZE3) rx_wr_index3=0;
   {
   if (++rx_counter3 == RX_BUFFER_SIZE3)
      {
      rx_counter3=0;
      rx_buffer_overflow3=1;
      };
   }


   };


}

// Get a character from the USART3 Receiver buffer
#pragma used+
unsigned char getchar3(void)
{
unsigned char data;
while (rx_counter3==0);
data=rx_buffer3[rx_rd_index3];
if (++rx_rd_index3 == RX_BUFFER_SIZE3) rx_rd_index3=0;
#asm(“cli”)
--rx_counter3;
#asm(“sei”)

return data;
}
#pragma used-

// Write a character to the USART3 Transmitter
#pragma used+
void putchar3(unsigned char c)
{

while ((UCSR3A & DATA_REGISTER_EMPTY)==0);
{
UDR3=c;
}

}
#pragma used-


void main(void)
{

Boot_OBC();
Boot_SD();
nominal_telemetry();
Boot_CAM();


takePIC(1000);       //1000 is the number of pic.
put_string_3(“fcreate END.jpg₩r₩n”); //make a file
printf(“Function END₩r₩n”);
wait_message();
```

```
}


void Boot_OBC(void)
{
#asm( “cli” )

// Declare your local variables here

// Crystal Oscillator division factor: 1
#pragma optsize-
CLKPR=0x80;
CLKPR=0x00;
#ifdef _OPTIMIZE_SIZE_
#pragma optsize+
#endif

// Input/Output Ports initialization
// Port A initialization
// Func7=out Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTA=0b00000000;
DDRA=0b00000000;

// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTB=0x00;
DDRB=0x00;

// Port C initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTC=0x00;
DDRC=0x00;

// Port D initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTD=0b00000000;
DDRD=0b00010001;

// Port E initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTE=0x00;
DDRE=0x00;

// Port F initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTF=0x00;
DDRF=0x00;

// Port G initialization
// Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State5=T State4=T State3=T State2=T State1=T State0=T
PORTG=0x00;
DDRG=0x00;

// Port H initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTH=0x00;
DDRH=0x00;

// Port J initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTJ=0x00;
DDRJ=0x00;
```

```
// Port K initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTK=0x00;
DDRK=0x00;

// Port L initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTL=0x00;
DDRL=0x00;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=FFh
// OC0A output: Disconnected
// OC0B output: Disconnected
TCCR0A=0x00;
TCCR0B=0x00;
TCNT0=0x00;
OCR0A=0x00;
OCR0B=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// OC1C output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
// Compare C Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;
OCR1CH=0x00;
OCR1CL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer 2 Stopped
// Mode: Normal top=FFh
// OC2A output: Disconnected
// OC2B output: Disconnected
ASSR=0x00;
TCCR2A=0x00;
TCCR2B=0x00;
TCNT2=0x00;
OCR2A=0x00;
OCR2B=0x00;

// Timer/Counter 3 initialization
// Clock source: System Clock
// Clock value: Timer 3 Stopped
// Mode: Normal top=FFFFh
// Noise Canceler: Off
// Input Capture on Falling Edge
// OC3A output: Discon.
```

```
// OC3B output: Discon.
// OC3C output: Discon.
// Timer 3 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
// Compare C Match Interrupt: Off
TCCR3A=0x00;
TCCR3B=0x00;
TCNT3H=0x00;
TCNT3L=0x00;
ICR3H=0x00;
ICR3L=0x00;
OCR3AH=0x00;
OCR3AL=0x00;
OCR3BH=0x00;
OCR3BL=0x00;
OCR3CH=0x00;
OCR3CL=0x00;

// Timer/Counter 4 initialization
// Clock source: System Clock
// Clock value: Timer 4 Stopped
// Mode: Normal top=FFFFh
// OC4A output: Discon.
// OC4B output: Discon.
// OC4C output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 4 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
// Compare C Match Interrupt: Off
TCCR4A=0x00;
TCCR4B=0x00;
TCNT4H=0x00;
TCNT4L=0x00;
ICR4H=0x00;
ICR4L=0x00;
OCR4AH=0x00;
OCR4AL=0x00;
OCR4BH=0x00;
OCR4BL=0x00;
OCR4CH=0x00;
OCR4CL=0x00;

// Timer/Counter 5 initialization
// Clock source: System Clock
// Clock value: Timer 5 Stopped
// Mode: Normal top=FFFFh
// OC5A output: Discon.
// OC5B output: Discon.
// OC5C output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 5 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
// Compare C Match Interrupt: Off
TCCR5A=0x00;
TCCR5B=0x00;
TCNT5H=0x00;
TCNT5L=0x00;
ICR5H=0x00;
ICR5L=0x00;
OCR5AH=0x00;
OCR5AL=0x00;
OCR5BH=0x00;
OCR5BL=0x00;
OCR5CH=0x00;
```

```
OCR5CL=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
// INT3: Off
// INT4: Off
// INT5: Off
// INT6: Off
// INT7: Off
EICRA=0x00;
EICRB=0x00;
EIMSK=0x00;
// PCINT0 interrupt: Off
// PCINT1 interrupt: Off
// PCINT2 interrupt: Off
// PCINT3 interrupt: Off
// PCINT4 interrupt: Off
// PCINT5 interrupt: Off
// PCINT6 interrupt: Off
// PCINT7 interrupt: Off
// PCINT8 interrupt: Off
// PCINT9 interrupt: Off
// PCINT10 interrupt: Off
// PCINT11 interrupt: Off
// PCINT12 interrupt: Off
// PCINT13 interrupt: Off
// PCINT14 interrupt: Off
// PCINT15 interrupt: Off
// PCINT16 interrupt: Off
// PCINT17 interrupt: Off
// PCINT18 interrupt: Off
// PCINT19 interrupt: Off
// PCINT20 interrupt: Off
// PCINT21 interrupt: Off
// PCINT22 interrupt: Off
// PCINT23 interrupt: Off
PCMSK0=0x00;
PCMSK1=0x00;
PCMSK2=0x00;
PCICR=0x00;

// Timer/Counter 0 Interrupt(s) initialization
TIMSK0=0x00;
// Timer/Counter 1 Interrupt(s) initialization
TIMSK1=0x00;
// Timer/Counter 2 Interrupt(s) initialization
TIMSK2=0x00;
// Timer/Counter 3 Interrupt(s) initialization
TIMSK3=0x00;
// Timer/Counter 4 Interrupt(s) initialization
TIMSK4=0x00;
// Timer/Counter 5 Interrupt(s) initialization
TIMSK5=0x00;

// USART0 initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART0 Receiver: On
// USART0 Transmitter: On
// USART0 Mode: Asynchronous
// USART0 Baud Rate: 9600
UCSR0A=0x00;
UCSR0B=0x98;
UCSR0C=0x06;
UBRR0H=0x00;
UBRR0L=0x67;

// USART1 initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART1 Receiver: On
// USART1 Transmitter: Off
```

```
// USART1 Mode: Asynchronous
// USART1 Baud Rate: 9600
UCSR1A=0x00;
UCSR1B=0x10;
UCSR1C=0x06;
UBRR1H=0x00;
UBRR1L=0x67;

// USART2 initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART2 Receiver: On
// USART2 Transmitter: On
// USART2 Mode: Asynchronous
// USART2 Baud Rate: 38400
UCSR2A=0x00;
UCSR2B=0x98;
UCSR2C=0x06;
UBRR2H=0x00;
UBRR2L=0x19;

// USART3 initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART3 Receiver: On
// USART3 Transmitter: On
// USART3 Mode: Asynchronous
// USART3 Baud Rate: 38400
UCSR3A=0x00;
UCSR3B=0x98;
UCSR3C=0x06;
UBRR3H=0x00;
UBRR3L=0x19;




// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
ADCSRB=0x00;


#asm("sei")


delay_ms(1000);
put_flashstring_0("₩r₩nOBC INITIATED₩r₩n");
delay_ms(1000);
}

void Boot_SD(void)
{
put_flashstring_0("SD healthcheck start₩r₩n");
    delay_ms(200); //wait for SDcard
    put_string_3("mode /m₩r₩n");
    delay_ms(20);
    put_string_3("fcreate SDIN.jpg₩r₩n"); //make a file
    wait_message();
    put_string_3("fcreate INTERUPT.jpg₩r₩n"); //make a file to make sure the condition of SD card.
    wait_message();
delay_ms(1000);
put_flashstring_0("SD INITIATED₩r₩n");
delay_ms(1000);
}

void put_string_0(unsigned char str[])     // Output function for limited string array via uart0.
{
  int k;
  for(k=0;*(str+k)!=0;k++)
  {

    putchar0(*(str+k));
```

```
  }


}

void put_flashstring_0(flash unsigned char *str) // Output function for unlimited string array marked with  “  “ (quotes)
via uart0.
{
  int j;


  for(j=0;*(str+j)!=0;j++)
  {
    putchar0(*(str+j));

  }

}



void put_string_3(flash unsigned char *str)  // Output function for unlimited string array marked with  “  “ (quotes)via
uart3.
{
  int i;
  for(i=0;*(str+i)!=0;i++)
  {
    putchar3(*(str+i));

    }
}


void uart0_send_int(int x)  // Send integer variables via uart0
{
int divide = 10000;
char i,c;
for( i = 0 ; i < 4; i ++)
{
c = x / divide;
x = x % divide;
putchar(c +  '0' );
divide = divide / 10;
}
putchar(x +  '0' );

}

void uart3_send_int(int x)  // Send integer variables via usart3
{
int divide = 10000;
char i,c;
for( i = 0 ; i < 4; i ++)
{
c = x / divide;
x = x % divide;
putchar3(c +  '0' );
divide = divide / 10;
}
putchar3(x +  '0' );

}

void wait_message(void)       // OBC waits for SD card until  "O"  from uart3(SDcard).
{
unsigned char dump;
                // SD-COM으로 부터 응답 기다림
 while(fRX==0)
 {
 dump = getchar3();
 if(dump=='O')
```

```
 {
 fRX=1;
 }

 }

fRX=0;
}

// Camera C328 fuctions start//
void C328_send_command(unsigned char *cmd)
{
    putchar2(cmd[0]);
    putchar2(cmd[1]);
    putchar2(cmd[2]);
    putchar2(cmd[3]);
    putchar2(cmd[4]);
    putchar2(cmd[5]);
}

void sync(void)
{
    cmd[0]=0xAA;
    cmd[1]=0x0D;
    cmd[2]=0x00;
    cmd[3]=0x00;
    cmd[4]=0x00;
    cmd[5]=0x00;
    C328_send_command(cmd);
}

void ACK_sync(void)
{
    cmd[0]=0xAA;
    cmd[1]=0x0E;
    cmd[2]=0x0D;
    cmd[3]=0x00;
    cmd[4]=0x00;
    cmd[5]=0x00;
    C328_send_command(cmd);
}

void init_C328(void)
{
    cmd[0]=0xAA;
    cmd[1]=0x01;
    cmd[2]=0x00;
    cmd[3]=0x07;
    cmd[4]=0x00;
    cmd[5]=0x05;    //320x240=05, 640x480=07
    C328_send_command(cmd);
}

void set_package_size(void)
{
    cmd[0]=0xAA;
    cmd[1]=0x06;
    cmd[2]=0x08;
    cmd[3]=0x00;
    cmd[4]=0x01;
    cmd[5]=0x00;
    C328_send_command(cmd);
}

void snapshot(void)
{
    cmd[0]=0xAA;
    cmd[1]=0x05;
    cmd[2]=0x00;
    cmd[3]=0x00;
    cmd[4]=0x00;
    cmd[5]=0x00;
```

```
    C328_send_command(cmd);
}

void get_picture(void)
{
    cmd[0]=0xAA;
    cmd[1]=0x04;
    cmd[2]=0x01;
    cmd[3]=0x00;
    cmd[4]=0x00;
    cmd[5]=0x00;
    C328_send_command(cmd);
}

void package(char n)
{
    cmd[0]=0xAA;
    cmd[1]=0x0E;
    cmd[2]=0x0A;
    cmd[3]=0x00;
    cmd[4]=0x00+n;
    cmd[5]=0x00;
    C328_send_command(cmd);
}

void lastACK(void)
{
    cmd[0]=0xAA;
    cmd[1]=0x0E;
    cmd[2]=0x0A;
    cmd[3]=0x00;
    cmd[4]=0xF0;
    cmd[5]=0xF0;
    C328_send_command(cmd);
}

void connect(void)
{
    unsigned int i=0, success=0;
    while(success==0)
    {



        for(i=0;i<100;i++)
        {
            sync();

            delay_ms(100);
            if(i%2==0)
            {
            put_flashstring_0( "." );
            }

               if(C328_ACK_data[0]==0xAA && C328_ACK_data[1]==0x0E && C328_ACK_data[2]==0x0D && C328_ACK_
data[6]==0xAA && C328_ACK_data[7]==0x0D)
            {
                success=1;
                delay_ms(100);
                i=500;
            }
        }



    }
}


void Boot_CAM(void)        //  CAM initiates
{
```

```
    put_flashstring_0("CAM healthcheck start₩r₩n");
    delay_ms(2000);
    connect();              //C328과 연결 (sync보내기)


    delay_ms(100);
    ACK_sync();
    delay_ms(50);

    g_nRxHead_0=0;


    init_C328();          // C328 initial 설정
    delay_ms(50);
    if(C328_ACK_data[0]==0xAA && C328_ACK_data[1]==0x0E && C328_ACK_data[2]==0x01)
        {

            delay_ms(500);
        }
    delay_ms(50);

    g_nRxHead_0=0;
    set_package_size(); // package size 설정
    delay_ms(50);
    if(C328_ACK_data[0]==0xAA && C328_ACK_data[1]==0x0E && C328_ACK_data[2]==0x06)
        {

            delay_ms(500);
        }
    delay_ms(50);

    g_nRxHead_0=0;
    printf("₩r₩nCAM INITIATED ₩r₩n");
}

// Camera C328 fuctions end//


void takePIC(int PN)    // Take Pic. PN stands for the number of Pics.
{
    int imgsize, numpack;
    int m,n,cnt,picn,filesizen;
    int min,sec;

    unsigned char filesizes[11];

    unsigned int i=0;
    cnt=1;


    while(cnt!=PN)
    {

    snapshot();             // snapshot (사진찍기)
    picn=rand();  // prevent overwriting when OBC restarting.


    put_string_3("fcreate P"); //make a file

    uart3_send_int(picn);

    put_string_3(".jpg₩r₩n"); //make a file

    wait_message();

    put_string_3("fopen P"); //make a file
    uart3_send_int(picn);

    put_string_3(".jpg /w₩r₩n"); //make a file
    wait_message();

    cnt++;
```

```
if(C328_ACK_data[0]==0xAA && C328_ACK_data[1]==0x0E && C328_ACK_data[2]==0x05)
    {

        delay_ms(500);
    }
delay_ms(50);

g_nRxHead_0=0;
get_picture();        // get picture (사진 받기)
delay_ms(50);


imgsize = C328_ACK_data[10]*(16*16) + C328_ACK_data[9];
numpack = imgsize/250 + 1;
n=numpack;



m=0;

delay_ms(1000);

while(m<n)
{
    g_nRxHead_0=0;
    for(i=0;i<256;i++)        // C328_ACK_data를 모두 0으로 셋팅
    {
        C328_ACK_data[i]=0;
    }
    package(m);
    delay_ms(50);
    put_string_3("fwrite /250");
    put_string_3("₩r₩n");
    wait_message();
    for(i=4;i<254;i++)
    {
        putchar3(C328_ACK_data[i]);

    }

    put_string_3("₩r₩n");
    wait_message();

    m++;
}
lastACK();
put_string_3("fclose₩r₩n");
wait_message();


if(GS0=='1')  //Interrupt from Ground Station to send current pic via uart0(RF module).
{


put_flashstring_0("Ready 4 Pic! filename is P");
uart0_send_int(picn);
put_flashstring_0(".jpg_");
delay_ms(3000);
put_string_3("fsize P");
uart3_send_int(picn);
put_string_3(".jpg₩r₩n");



for(i=0;i<10;i++)
{
 filesizes[i]=0;
}
filesizes[10]=NULL;

i=0;
```

```
while(i<10)
{
filesizes[i]=getchar3();

if(filesizes[i]!=' O' )  // deleting  'O'
{
i++;
}

}


for(i=0;i<10;i++)
{
 putchar0(filesizes[i]);
}


filesizen=atoi(filesizes);

put_flashstring_0( "bytes_" );
delay_ms(5000);


numpack = filesizen/125;
n=numpack;

m=0;

min= n/6;
sec = (n-min*6)*10;

printf( "Downloading time %dmin %dsec₩r₩n" ,min,sec);
delay_ms(10000);    // ready 4 Ground Station (GS).

put_string_3( "fopen P" );
uart3_send_int(picn);
put_string_3( ".jpg /r₩r₩n" );  //open the file to read
wait_message();


while(m < n) //start file-transmit.
{
put_string_3( "fgetc /125₩r₩n" );

for(i=0;i<125;i++)
{
PIC_data[i]=getchar3();
}

for(i=0;i<125;i++)
{
putchar(PIC_data[i]);
}

delay_ms(10000);     // interval for RF module (to prevent overheat)
m++;

}




put_string_3( "fclose₩r₩n" ); // file-transmit end.
wait_message();
GS0=' 0' ;

delay_ms(10000);
Boot_OBC();  // initate OBC.
Boot_CAM();  //When error occurs after completing file-transmit, to revive CAM.
```

```
 }


    nominal_telemetry();

    }
}

void gpsparsing(void)  // read GPS coordiates
    {

    ////////////////////////////// IMU parsing 변수/////////////////////////////////////////

    //int i=0;
    int j=0;
    int k=0;
    int l=0;
    int n=0;
    int p=0;
    int LMW=0;




    unsigned char han[100];
    unsigned char dong[100];
    unsigned char lee[100];
    unsigned char time[30];
    unsigned char latitude[30];
    unsigned char longitude[30];
    unsigned char altitude[30];
    unsigned char nsatel[20];
    unsigned char speed[10];
    float latitude_float;
    float longitude_float;
    float latitude_deg_float;
    float longitude_deg_float;







    ////////////////////////////////// 체크섬  변수/////////////////////////////////////////////



    unsigned char *token;



    ////////////////////////////// 레지스터 설정/////////////////////////////////////////////////







      while(LMW!=1)
      {
          han[j]=usart1_rx_isr();
          if((han[j]=='A')&&(han[j-1]=='G'))
           {k=1;}          //GGA

          if((han[j]=='G')&&(han[j-1]=='T'))
           {k=2;}         //VTG
           j++;
```

```
//--G G A 시간, 위도, 경도, 위성수, 고도
 while(k==1)
{
dong[l++]=usart1_rx_isr();

      if(UDR1==13)
      {
       token = strtok(dong,  “,” );
       strncpy(time, token, strlen(token));        // 시간
       time[strlen(token)] = NULL;


       token = strtok(NULL,  “,” );
       strncpy(latitude, token, strlen(token));  // 위도
       latitude[strlen(token)] = NULL;

       latitude_float = atof(latitude);
       latitude_deg_float = (latitude_float – 3600)/60 + 36;        // latitude_degree_float


       token = strtok(NULL,  “,” );
       token = strtok(NULL,  “,” );
       strncpy(longitude, token, strlen(token));  // 경도
       longitude[strlen(token)] = NULL;

       longitude_float = atof(longitude);
       longitude_deg_float = (longitude_float – 12600)/60 + 126;  // longitude_degree_float


       token = strtok(NULL,  “,” );
       token = strtok(NULL,  “,” );
       token = strtok(NULL,  “,” );
       strncpy(nsatel, token, strlen(token));     // 위성 수
       nsatel[strlen(token)] = NULL;


       token = strtok(NULL,  “,” );
       token = strtok(NULL,  “,” );
       strncpy(altitude, token, strlen(token));  // 고도
       altitude[strlen(token)] = NULL;

       l=0;
       k=k+3;
      }
}



  // V T G  속도
 while(k==2)
{
 lee[n++]=usart1_rx_isr();

      if(UDR1==13)
      {
       token = strtok(lee,  “,” );
       token = strtok(NULL,  “,” );
       token = strtok(NULL,  “,,” );
       token = strtok(NULL,  “,” );
       token = strtok(NULL,  “,” );
       token = strtok(NULL,  “,” );

       strncpy(speed, token, strlen(token));
       speed[strlen(token)] = NULL;                //속도

       n=0;
       k=k+2;
       p=1;
      }
}
```

```
            while(p==1)
            {

                //put_flashstring_0( " time =  ");
                put_string_0(time);
                put_flashstring_0( "  ");
                //put_flashstring_0( ", altitude =  ");
                put_string_0(altitude);
                put_flashstring_0( "  ");
                //put_flashstring_0( ", speed =  ");
                //put_string_0(speed);
                //put_flashstring_0( "  ");
                //put_flashstring_0( ", latitude =  ");
                put_string_0(latitude);
                put_flashstring_0( "  ");
                //put_flashstring_0( ", longitude =  ");
                put_string_0(longitude);
                //put_flashstring_0( "  ");
                delay_ms(3000);
                //put_flashstring_0( "₩r₩n" );
                p=0;
                LMW=1;




            }



      } // 전체 while end


   }



// Temperature fuctions
unsigned char DS18S20_reset(void)
{
    unsigned char presence;

    DQ_LOW;
    delay_us(480);
    DQ_HIGH;
    delay_us(70);
    do
    {
        presence = PIND.0;
    }while(presence);
    delay_us(424);
    return(presence);
}

unsigned char read_data(void)
{
    unsigned char i;
    unsigned char data = 0x00;

    for(i=0;i<8;i++)
    {
        DQ_LOW;
        DQ_HIGH;
        delay_us(15);

        if(PIND & 0x01)
            data |= 0x80;
```

```
        delay_us(120);

        if(i<7)
            data = data >> 1;
    }

    return data;
}

void write_data(unsigned char data)
{
    unsigned char i;

    for(i=0;i<8;i++)
    {
        DQ_LOW;
        delay_us(15);

        if(data & 0x01)
            DQ_HIGH;

        delay_us(70);
        DQ_HIGH;
        delay_us(5);

        data = data >> 1;
    }
}


void disp_data(unsigned int sign_byte, unsigned int temp_byte)
{
    unsigned char half, temp, sign, temper[6];

    if((temp_byte & 0x01) == 1)  // 소수점 판단
        half = 5;
    else
        half = 0;

    temp = temp_byte >> 1;    // 최하위 비트(소수점이하) 삭제

    if(sign_byte)
    {
        sign =  '-' ;
        temp = 128 - temp;    // 2의 보수 계산 (음수일때)
    }
    else
        sign =  '+' ;

    sprintf(temper,  " %c%d.%d₩r₩n" , sign, temp, half);

    put_string_0(temper);
    delay_ms(1000);

}

void read_temp(void)
{
    char get[10];
    unsigned char temp_lsb, temp_msb;
    unsigned char i;

    DS18S20_reset();
    delay_us(5);
    write_data(0xcc);
    delay_us(5);
    write_data(0x44);
    delay_us(104);

    DS18S20_reset();
    delay_us(5);
```

```
    write_data(0xcc);
    delay_us(5);
    write_data(0xbe);
    delay_us(104);

    for(i=0;i<9;i++)
        get[i] = read_data();

    temp_lsb = get[0];
    temp_msb = get[1];

    disp_data(temp_msb, temp_lsb);
    delay_ms(100);
    DS18S20_reset();
}

void nominal_telemetry(void)
{
gpsparsing();
read_temp();
}
```