ATMEL AVR DESIGN CONTEST 2006

PROJECT : A BUTTERFLY GPS

AVR MICROCONTROLLER : ATMEGA169

PROJECT SUMMARY :

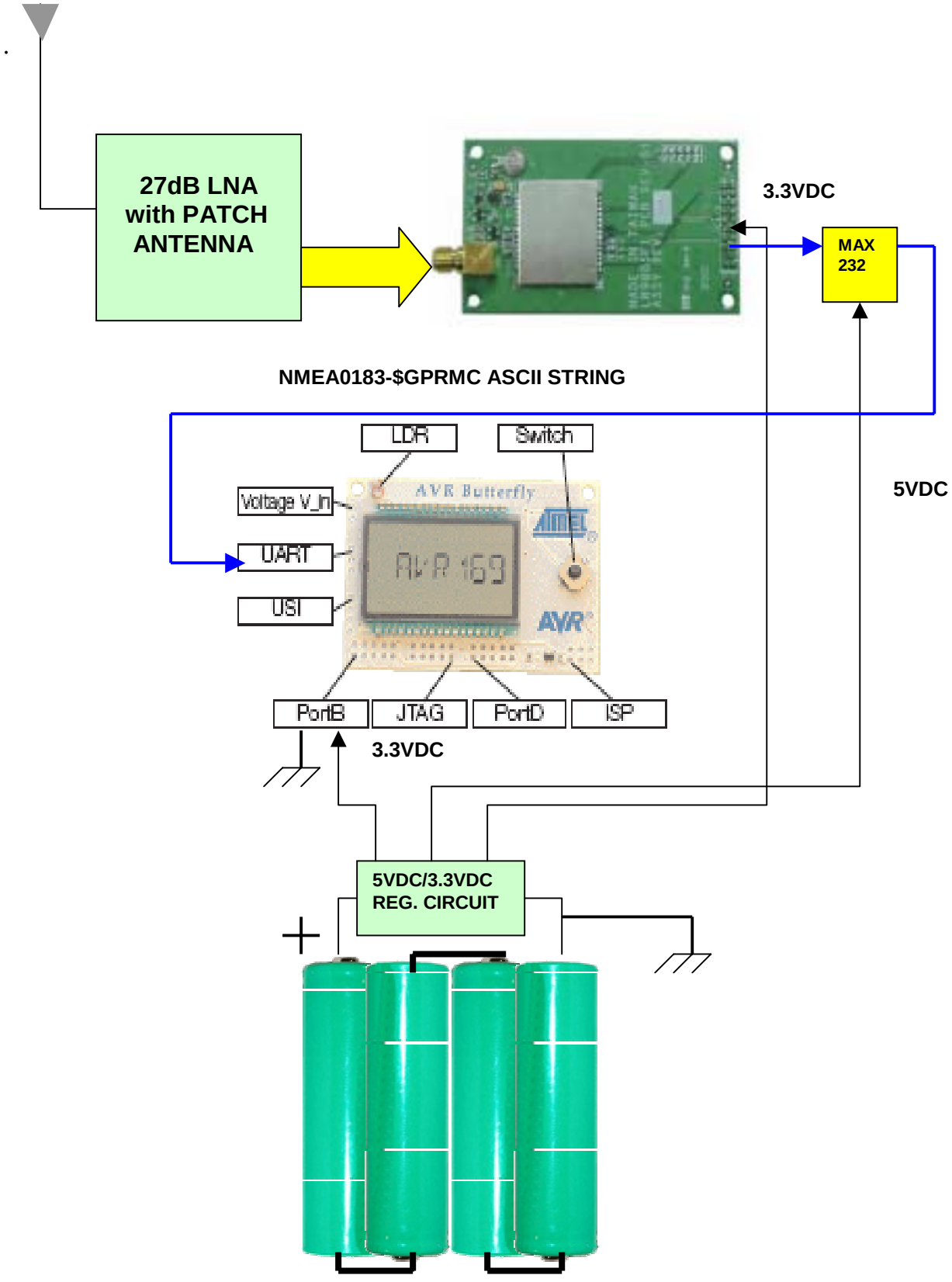
The Butterfly GPS is a handy gadget for GPS position locating using the ATMEL Butterfly kit. It's a trade off between an expensive PDA navigator and a full fledged GPS receiver with mapped display. The objective of this design is to create a simple cost effective GPS device that would give position fixes of Latitude and Longitude along with date and time in UTC. I always thought that we could have something that wouldn't cost like a PDA navigator or a PDA phone with LBS built-in but a handy device that would tell us where we stand. With the Google Earth revolutionising the world, its now pretty easy to navigate yourself in any city on this planet. With this idea in mind, I created the Butteryfly GPS based on the ATMEL ATMEGA169 microcontroller and a highly sensitive GPS receiver engine board based on the SiRF chipset. The idea is to generate a Vanilla GPS that that can always be flavoured.

I always believe that the most complex matters can always be solved in a very simple way. If I had to navigate my way in a city, town, a desert or some place where I need to track my route, all that I would need is a hardcopy of the map of the terrain with Lat/Lon grids and a small handy GPS to get instantaneous updates of my position. Expensive navigators are no doubt very useful devices where you can easily track your route. I opted for something which would cost me less than a $100 or a sub $100 gadget. This prompted me to think of something very simple but I just didn't find the right platform till I found the Butterfly Kit. You don't have to think of designing your own printed circuit board for having a uC onboard, then putting in all that hardware to interface an LCD display and get the board running. Infact this kit is what would be the future in embedded design where uC modules with LCD, on board memory etc. would be readily available for application development.

The design idealogy is simple. I didn't want to create just another PDA navigator with a TFT display and some city maps in digital format pushed into the PDA memory. It had to be a simple machine that would display the postion data and date/time over an LCD display. A GPS engine interfaced to the Butterfly Kit is all that is needed to create the basic system architecture. The GPS module selected is a highly sensitive GPS engine board based on the SiRFIII GPS chipset and offers a sensitivity of –159dBm with a position accuracy of better than 10 metres under average conditions of satellite availability and signal strength from the GPS satellites. You simply can't find a better GPS engine in the market that gives a 1Hz update over a NMEA0183 format and cosumes a max. current of 80mA on trickle power mode. The Butterfy Kit is just what I needed to create this device. The 1Hz Lat/Lon position updates coming out of the GPS engine UART are received over the Butterfly UART and stored in the EEPROM on board the Butterfly Kit. The NMEA format is a $GPRMC string that contains information on the Lat/Lon, Date, Time, Course Heading & Speed. The ASCII string stored is displayed over the LCD display and the memory location is refreshed with another update. The display is a smooth scrolling display of the Lat, Lon, Date and UTC time.

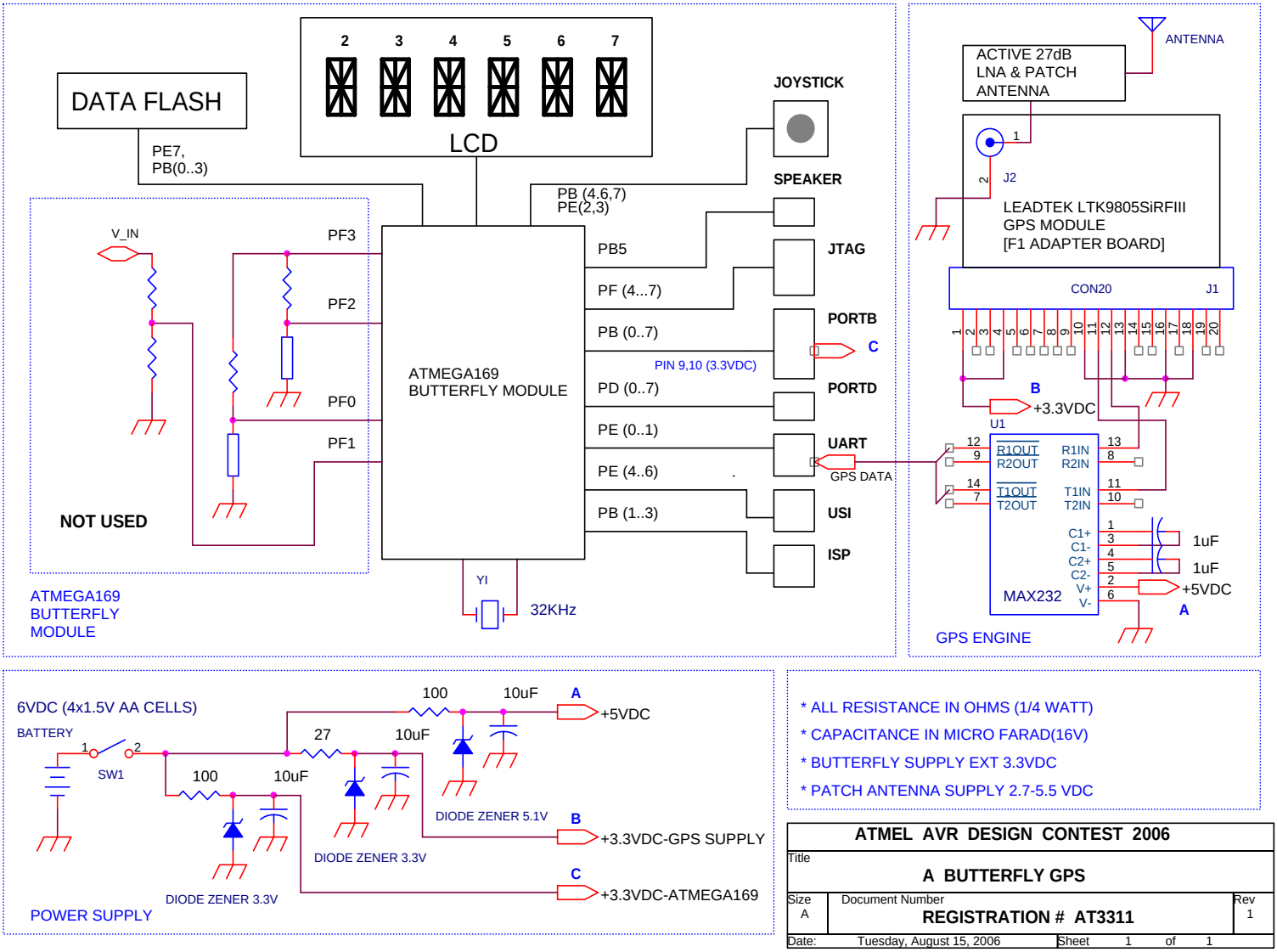
The application runs of 4x1.5V AA cells with a total current consumption of around 90mA when taking position fixes. The ATMEGA169 Butteryfly draws less than 10mA or so and added to the GPS current of 80mA makes up the total cuurent for this unit. One big switch to turn on power to the device and wait for around 40 secs. before you get the first fix. The power switch supplies DC to the GPS and the ATMEGA169 board. However, the Butterfly board is activated after sometime by pushing the joystick UP and the message starts scrolling on the display. The Butterfly board is switched ON after a delay since the GPS engine takes around 40 secs. or less to get the first fix. This is called the TTFF parameter or the Time To First Fix. The TTFF may vary depending on satellite availability at the point of fix. The gadget is simple to operate and needs no special care except that you need to find a reasonably clear spot looking towards the sky to get a good fix. Once you have the Lat/Lon fix, use a map with grid locators to find your position. Date and UTC is displayed on the LCD.

SYSTEM BLOCK DIAGRAM :



AT3311

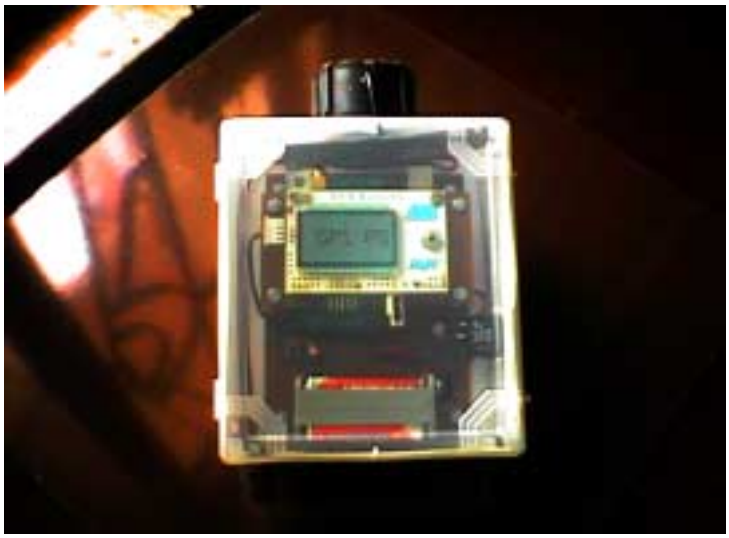
SYSTEM SCHEMATIC DIAGRAM :



* ALL RESISTANCE IN OHMS (1/4 WATT)
* CAPACITANCE IN MICRO FARAD(16V)
* BUTTERFLY SUPPLY EXT 3.3VDC
* PATCH ANTENNA SUPPLY 2.7-5.5 VDC

ATMEL AVR DESIGN CONTEST 2006		
Title: A BUTTERFLY GPS		
Size: A	Document Number: REGISTRATION # AT3311	Rev: 1
Date: Tuesday, August 15, 2006	Sheet 1 of 1	

PROJECT PHOTOGRAPH :



The Butterfly GPS assembled in a plastic enclosure as seen in this photograph…

The black dome on top is the Active LNA Patch type GPS antenna

APPLICATION SOURCECODE :

The application sourcecode is in ANSI C and compiled under the AVR Studio 4, SP2, GCC Complier and the project files are located in the folder named AT3311_Project Files_Firmware. Below is the C File for the GPS routine for collecting GPS data from the GPS engine through the UART and displaying the same on the LCD display.

```
#include "gps.h"
#include "usart.h"
#include "LCD_functions.h"

/*****************************************************************************
*Implemented
*Function Name: void GET_GPS_DATA(void);
*Purpose:       get Data From GPS
*
*****************************************************************************/

char GET_GPS_DATA(void)
{
short Counter=0;
char GET_DATA='\0',Valid_Data='N';


while(Counter<GPS_DATA_BUFF)
        GPS_BUFFER[Counter++]=GET_DATA;


Counter=0;
while(1)
                {

                 GET_DATA=Usart_Rx();
                 Usart_Tx(GET_DATA);

                        if((GET_DATA=='$')||(Valid_Data=='Y'))
                        {
                                Valid_Data='Y';
                                GPS_BUFFER[Counter++]=GET_DATA;

                                if((GET_DATA=='*')||(Counter==GPS_DATA_BUFF-1))
                                break;
                        }
                 }

/* if(GPS_BUFFER[18]=='A')
 return ('A');
 else
 return ('V');*/

 return 'a';
}


void MESSAGE_PERCING(void)
{

short Counter=0,i;
char TEMP_GPS_DATA[GPS_DATA_BUFF];

while((TEMP_GPS_DATA[Counter]=GPS_BUFFER[Counter])!='\0')
Counter++;

Counter=0;
while(Counter<GPS_DATA_BUFF)
        GPS_BUFFER[Counter++]='\0';


Counter=0;
GPS_BUFFER[Counter++]='G';
GPS_BUFFER[Counter++]='P';
GPS_BUFFER[Counter++]='S';
GPS_BUFFER[Counter++]=' ';
GPS_BUFFER[Counter++]='P';
GPS_BUFFER[Counter++]='O';
GPS_BUFFER[Counter++]='S';
```

```
GPS_BUFFER[Counter++]='-';

for(i=20;i<=43;i++)
GPS_BUFFER[Counter++]=TEMP_GPS_DATA[i];

i=GPS_DATA_BUFF-1;
while((TEMP_GPS_DATA[i--]=='*')||(i==12))
break;

i=51;

GPS_BUFFER[Counter++]=' ';
GPS_BUFFER[Counter++]='D';
GPS_BUFFER[Counter++]='A';
GPS_BUFFER[Counter++]='T';
GPS_BUFFER[Counter++]='E';
GPS_BUFFER[Counter++]='-';
GPS_BUFFER[Counter++]=TEMP_GPS_DATA[i++];
GPS_BUFFER[Counter++]=TEMP_GPS_DATA[i];
GPS_BUFFER[Counter++]='-';
GPS_BUFFER[Counter++]=TEMP_GPS_DATA[53];
GPS_BUFFER[Counter++]=TEMP_GPS_DATA[54];
GPS_BUFFER[Counter++]='-';
GPS_BUFFER[Counter++]=TEMP_GPS_DATA[55];
GPS_BUFFER[Counter++]=TEMP_GPS_DATA[56];



GPS_BUFFER[Counter++]=' ';
GPS_BUFFER[Counter++]='U';
GPS_BUFFER[Counter++]='T';
GPS_BUFFER[Counter++]='C';
GPS_BUFFER[Counter++]='-';
GPS_BUFFER[Counter++]=TEMP_GPS_DATA[7];
GPS_BUFFER[Counter++]=TEMP_GPS_DATA[8];
GPS_BUFFER[Counter++]='-';
GPS_BUFFER[Counter++]=TEMP_GPS_DATA[9];
GPS_BUFFER[Counter++]=TEMP_GPS_DATA[10];
GPS_BUFFER[Counter++]='-';
GPS_BUFFER[Counter++]=TEMP_GPS_DATA[11];
GPS_BUFFER[Counter++]=TEMP_GPS_DATA[12];


LCD_puts(GPS_BUFFER,1);
}

void DELAY_5_SEC(void)
{

int i,j;

 for(i=0;i<=1200;i++)
              for(j=0;j<=25000;j++)
                           {
                           }
}
```