GPS

경희대학교 컴퓨터공학과

조 진 성

GPS 정의

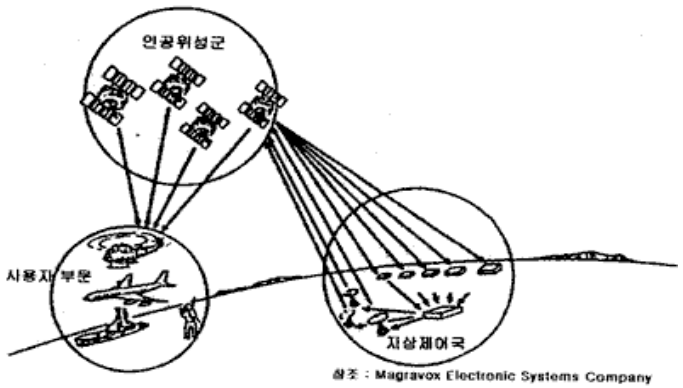
- GPS는 Global Positioning System의 약자로 범지구적 위치결정 체계로 미국 국방부에서 군사용으로 개발한 새로운 위성항법시스템.
- 지상, 해상, 공중 등 지구상의 어느 곳에서나 시간제약 없이 인공위성에서 발신하는 정보를 수신하여 정지 또는 이동하는 물체의 위치를 측정가능
- 우주부분, 제어부분, 사용자부분 등의 3부문으로 구성
- 현재 24개의 GPS위성을 이용한 항법서비스를 전세계적으로 무료로 제공 중.
- 기존 항법시스템에 비해 정확성이 높고 사용이 간편하며 시간과 장소 그리고 기상상황에 관계없이 사용할 수 있다.

GPS 특징

- GPS는 이동하는 사용자의 3차원 위치, 속도, 자세, 시간에 대한 10가지 정보를 동시 제공.
- 무제한 수의 사용자 이용 가능(One-Way Service)
- 3개 이상의 위성 정확한 시간과 거리를 측정하여 3개의 각각 다른 거리를 삼각방법에 의하여 현 위치를 정확히 계산.
- 위도·경도·고도의 위치뿐만 아니라 3차원의 속도정보와 함께 정확한 시간까지 얻을 수 있다.
- 민간용은 수평·수직 오차가 10~15m 정도이며 속도측정 정확도는 초당 3cm이다.
- 인공위성에는 3개의 원자시계가 탑재되어 있어 3만 6000년에 1초만의 오차를 갖는 시간정보를 제공

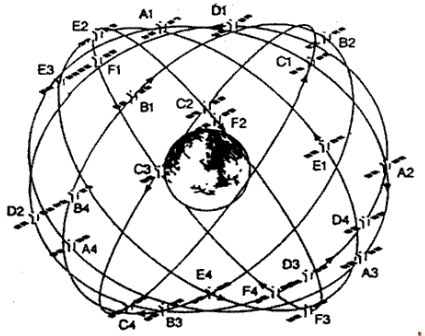
GPS의 구성요소

- GPS 는 우주 부문(Space Segment) , 관제 부문(Control Segment) , 사용자 부문(User Segment) 3가지 영역이 있다.



GPS의 구성요소

- 우주부분
  - ✓ GPS 우주 부문은 모두 24개의 위성으로 구성 (21개 항법 3개 예비용)
  - ✓ 고도 20,200 km 상공에서 12시간을 주기로 지구 주위를 돌고 있으며 궤도면은 지구의 적도면과 55의 각도를 이루고 있다.
  - ✓ 모두 6개의 궤도는 60도씩 떨어져 있고 한 궤도면에는 4개의 위성이 위치한다.
  - ✓ 지구상 어느 지점에서나 동시에 5개에서 최대 8개까지 위성을 볼 수 있다.



GPS의 구성요소

- 관제부분 (Control Segment)
  - ✓ GPS의 관제는 하나의 주관제국 (MCS: Master Control Station)과 무인으로 운영되는 다섯 개의 부관제국(Monitor Station)으로 구성.
  - ✓ 위성에서 송신되는 신호의 품질 점검, 위성 궤도의 추적, 위성에 탑재된 각종 기기의 동작상태 점검 및 그 밖의 각종 제어작업 등을 시행.
  - ✓ 부관제국 : 주어진 시간에 관측할 수 있는 모든.GPS 위성의 신호를 추적, 신호를 저장한 다음 주관제국으로 전송하며, 이 통신 시설을 DSCS (Defense Satellite Communication System)이라고 부른다
  - ✓ 주관제국:
    - 위성의 궤도를 수정
    - 사용불능 위성을 예비위성으로 교체하는 업무를 담당

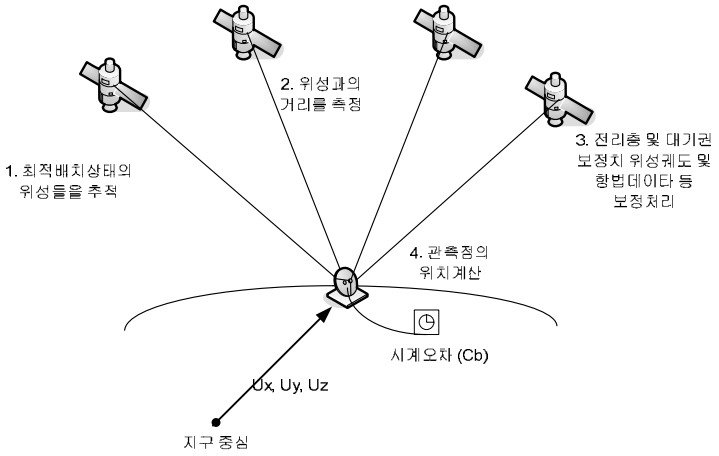
GPS의 구성요소

- 사용자 부문 (User Segment)
  - ✓ GPS 수신기와 사용자 단체.
  - ✓ GPS 수신기
    - 위성으로 부터 수신 받은 신호를 처리하여 수신기의 위치와 속도, 시간을 계산한다.
    - 4개 이상 위성의 동시관측을 필요. (3차원 좌표와 시간이 합쳐져 4개의 미지수를 결정)
    - GPS 수신기는 현재 항해와, 위치 측량, 시간보정 등 다양한 분야에 이용되고 있다.

GPS 원리

- GPS의 기본 원리
  - ✓ GPS는 삼각측량의 원리를 사용하며, 알고 싶은 점을 사이에 두고 있는 두 변의 길이를 측정함으로 미지의 점의 위치를 결정한다.
  - ✓ 인공위성으로부터 수신기까지의 거리는 각 위성에서 발생시키는 부호 신호의 발생 시점과 수신 시점의 시간 차이를 측정한 다음 여기에 빛의 속도를 곱하여 계산한다.
  - ✓ 거리 = 빛의 속도 * 경과시간
  - ✓ 위성의 정확한 위치를 계산하는 데는 GPS 위성으로부터 전송되는 궤도력을 사용한다.

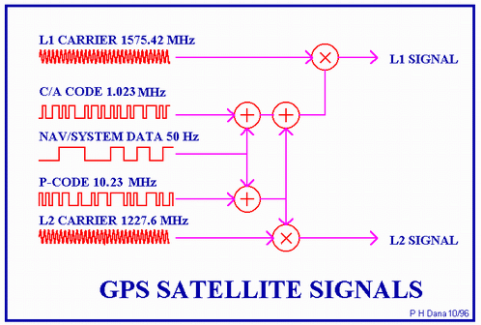
GPS 원리



GPS 위치 확인 과정

GPS 신호

- GPS 신호
  - ✓ L1, L2 두 Microwave 반송파에 신호를 실어 보내게 되는데, 어느 반송파에 실리느냐에 따라 PPS, SPS의 특성이 결정되게 된다.
  - ✓ L1(1575.42㎒) 반송파는 Navigation Message 와 SA를 적용하는 C/A Code Signal을 싣고, L2(1227.60㎒)의 반송파는 전리층에서 생기는 Delay를 측정하는데 쓰이게 된다.



GPS 신호

- C/A Code
  - ✓ C/A Code는 L1 반송파에 담겨지는 Data 이다.
  - ✓ 이 Code는 대역폭이 1㎒인 Pseudo Random Noise (PRN)를 반복하게 되는데, PRN은 Noise같이 보이지만, 실제로는 일정한 규칙성을 나타내는 사람이 만들어낸 Signal이다.
  - ✓ 이 PRN은 각 위성마다 달라서 각각의 위성의 고유한 Code Number로서 위성을 식별할 수 있는 지표가 된다. C/A Code는 L1 반송파에 변조되어 일반 SPS에게 제공된다
- P-Code.
  - ✓ P-Code (Precise)는 L1 과 L2에 모두 변조되는 주기가 매우 긴(7일) 10㎒ PRN Code이다.
  - ✓ 이 Code는 특정한 사람에게만 쓰일 수 있게 Anti-Spoofing (AS) Mode로 동작하기 위해서 Y-Code로 Encode되어 보내진다.
  - ✓ Encode 된 Y-Code는 사용자의 Receiver Channel에서 AS Module을 분류하여 암호해독이 된다. 따라서 이 Code는 PPS에서 사용되게 된다.
- Navigation Massage.
  - ✓ Navigation Massage는 C/A Code와 함께 L1에 변조된다. 이 Message는 50㎐의 신호로 GPS 위성의 궤도, 시간 그리고 다른 System Parameter들을 포함하는 Data Bit이다

GPS 정확도

기법	내용	정밀도
단독측위	GPS 수신기 1 대로 위치측정	100 m
DGPS	측량용과 항법용 수신기를 결합하여 이동체의 후처리 및 실시각 정밀위치 측정	1 m - 5 m
후처리 상대측위	2 대 이상의 측량용 GPS 수신기를 이용하여 고정밀 상대위치 측정하나 실시각 불가능	수 mm
실시각 이동측위	2 대 이상의 측량용 수신기를 이용하여 실시각 고정밀 위치 측정	1 cm - 2 cm

NMEA

- NMEA는 National Marine Electronics Association의 약자로, GPS Module 제어 / 데이터 송수신을 위해 사용되는 프로토콜이다.
- 모든 NMEA 메시지는 ASCII 문자로 구성된다.
  - ✓ 20 – 127개의 10진 데이터 또는 14 – 7E개의 헥사 코드로 구성된다.

Sentence	Description
$GPGGA	Global positioning system fixed data
$GPGLL	Geographic position - latitude / longitude
$GPGSA	GNSS DOP and active satellites
$GPGSV	GNSS satellites in view
$GPRMC	Recommended minimum specific GNSS data
$GPVTG	Course over ground and ground speed
$GPGGA	Sentence (Fix data)

NMEA

- GGA – Global Positioning System Fix Data
  - ✓ 시간, 위치 등의 정보 데이터 표현
  - ✓ $GPGGA,hhmmss.dd,xxmm.dddd,<N|S>,yyymm.dddd,<E|W>,v,ss,d.d,h.h,M,g.g,M,a.a,xxxx *hh<CR><LF>

    Ex) $GPGGA,092204.999,4250.5589,S,14718.5084,E,1,04,24.4,19.7,M,,,,0000*1F

Field	Example	Comments
Sentence ID	$GPGGA	
UTC Time	092204.999	hhmmss.dd
Latitude	4250.5589	xxmm.dddd
N/S Indicator	S	N = North, S = South
Longitude	14718.5084	yyymm.ddd
E/W Indicator	E	E = East, W = West
Position Fix	1	v : 0 = Invalid, 1 = Valid SPS, 2 = Valid DGPS, 3 = Valid PPS
Satellites Used	04	ss : Satellites being used (0-12)
HDOP	24.4	d.d : Horizontal dilution of precision
Altitude	19.7	h.h :Altitude in meters according to WGS-84 ellipsoid
Altitude Units	M	M = Meters
Geoid Seperation		g.g : Geoid seperation in meters according to WGS-84 ellipsoid
Seperation Units		M = Meters
DGPS Age		a.a :Age of DGPS data in seconds
DGPS Station ID	0000	xxxx
Checksum	*1F	*hh
Terminator	CR/LF	

NMEA

- GLL – Geographic Position – Latitude/Longitude
  - ✓ 위도 경도 시간 등의 , ,UTC 정보 데이터 표현
  - ✓ $GPGLL,xxmm.dddd,<N|S>,yyymm.dddd,<E|W>,hhmmss.dd,S,M*hh<CR><LF>

    Ex) $GPGLL,4250.5589,S,14718.5084,E,092204.999,A*2D

Field	Example	Comments
Sentence ID	$GPGLL	
Latitude	4250.5589	xxmm.dddd
N/S Indicator	S	N = North, S = South
Longitude	14718.5084	yyymm.ddd
E/W Indicator	E	E = East, W = West
UTC Time	092204.999	hhmmss.dd
status	A	A = Valid, V = Invalid
Checksum	*2D	*hh
Terminator	CR/LF	

NMEA

- GSA – DOP and Active Satellites
  - ✓ GPS 수신 모드, 사용되는 위성 표시, 고도 등의 정보 데이터 표현
  - ✓ $GPGSA,a,b,xx,xx,xx,xx,xx,xx,xx,xx,xx,xx,xxx,xx,xx,p.p,h.h,v.v*hh<CR><LF>

    Ex) $GPGSA,A,3,01,20,19,13,,,,,,,,,40.4,24.4,32.2*0A

Field	Example	Comments
Sentence ID	$GPGSA	
a Mode	A	A = Auto 2D/3D, M = Forced 2D/3D
b Mode	3	1 = No fix, 2 = 2D, 3 = 3D
Satellite used 1~12	01,20,19,13	xx :Satellite used on channel 1~12
PDOP	40.4	p.p :Position dilution of precision
HDOP	24.4	h.h : Horizontal dilution of precision
VDOP	32.2	v.v :Vertical dilution of precision
Checksum	*0A	*hh
Terminator	CR/LF	

NMEA

- GSV – Satellites in view
  - ✓ 시야에 있는 위성의 개수, 위성 ID Number, SNR 값 등의 정보 데이터 표현
  - ✓ $GPGSV,n,m,ss,xx,ee,aaa,cn,xxxx,xx,ee,aaa,cn*hh<CR><LF>
    Ex)
    $GPGSV,3,1,10,20,78,331,45,01,59,235,47,22,41,069,,13,32,252,45*70

Field	Example	Comments
Sentence ID	$GPGSV	
Number of messages	3	n : Number of messages in complete message (1-9)
Sequence number	1	m : Sequence number of this entry (1-9)
Satellites in view	10	ss : total number of satellites in view
Satellite ID	20	xx : Satellite ID (PRN) number (1-32)
Elevation	78	ee : Elevation in degrees (0-90)
Azimuth	331	aaa : Azimuth in degrees (0-359)
SNR	45	cc : Signal to noise ration in dBHZ (0-99)
~	~	~
Checksum	*70	
Terminator	CR/LF	

NMEA

- RMC – Recommended Minimum Specific GNSS Data
  - ✓ 시간, 날짜, 위치, 경로, 스피드 등의 정보 데이터 표현
  - ✓ $GPRMC,hhmmss.dd,S,xxmm.dddd,<N|S>,yyymm.dddd,<E|W>,s.s,h.h,dd,mmyy,d.d,<E|W>,M*hh<CR><LF>
    Ex) $GPRMC,092204.999,A,4250.5589,S,14718.5084,E,0.00,89.68, 21,1200,,*25

Field	Example	Comments
Sentence ID	$GPRMC	
UTC Time	092204.999	hhmmss.dd
Status	A	S : A = Valid, V= Invalid
Latitude	4250.5589	xxmm.dddd
N/S Indicator	S	N = North, S = South
Longitude	14718.5084	yyymm.dddd
E/W Indicator	E	E = East, W = West
Speed over ground	0.00	s,s :Speed, Knots
Course over ground	0.00	h,h : Degrees
UTC Date	211200	DDMMYY date
Magnetic variation		d,d : Degrees
Magnetic variation		E = East, W = West
Mode indicator		M : A = autonomous N=data not valid
Checksum	*25	
Terminator	CR/LF	

NMEA

- VTG – Course Over Ground and Ground Speed
  - ✓ 경로 및 시간 등의 정보 데이터 표현
  - ✓ $GPVTG,h.h,T,m.m,M,s.s,N,s.s,K,M*hh<CR><LF>
    Ex) $GPVTG,89.68,T,,M,0.00,N,0.0,K*5F

Field	Example	Comments
Sentence ID	$GPVTG	
Course	89.68	h.h : Course in degrees
Reference	89.68	T = True heading
Course		m.m : Course in degrees
Reference	89.68	M = Magnetic heading
Speed	0.00	s.s : Horizontal speed
Units	N	N = Knots
Speed	0.00	s.s : Horizontal speed
Units	K	K = KM/h
Mode indicator		A = autonomous, N = data not valid
Checksum	*5F	
Terminator	CR/LF	

NMEA

- ZDA – Time and Date
  - ✓ 현재 UTC 시간, 날짜 등의 정보 데이터 표현
  - ✓ $GPZDA,<hhmmss.dd>,<ddmmyyyy>,<xx>,<yy>*hh
    Ex) $GPZDA,132358.14,04122002,00,00*6A

Field	Example	Comments
Sentence ID	$GPZDA	
UTC time	132358.14	Hours-minutes-seconds
UTC date	04122002	Day-month-year
Hour	00	Local zone hours
Minutes	00	Local zone minutes

PXA255-Pro 구성사항

- 하드웨어 구성
  - ✓ GPS 모듈에서 수신된 데이터는 RS232C Serial port 를 통하여 컴퓨터로 입력된다. (리눅스는 /dev/ttyS2에 standard UART로 연결됨)
- 소프트웨어 구성
  - ✓ 소프트웨어는 디바이스 드라이버 모듈과 응용프로그램으로 구성된다.
    - 디바이스 드라이버 모듈은 PXA255 내부 레지스터 설정을 하는 역할이다.
    - 응용프로그램에서는 Serial port로 들어오는 데이터를 구분하여 출력해 주는 역할을 한다.
  - ✓ 주요 함수
    - data_reset() : 사용하는 변수들을 초기화 하는 함수
    - gps_main() : GPS 모듈로부터 Serial로 값을 읽어 오는 함수
    - gps_parser() : Serial로 넘어온 데이터를 파싱하여 저장하는 함수
    - gps_print() : 파싱된 데이터를 화면에 출력하는 함수

GPS Device Driver

```
#ifndef __KERNEL__
#define __KERNEL__
#endif
#ifndef MODULE
#define MODULE
#endif

#include <linux/module.h>
#include <linux/kernel.h>                    // printk()
#include <linux/version.h>
#include <linux/fs.h>                        // everything...
#include <linux/errno.h>                     // error codes
#include <linux/types.h>                     // size_t
#include <linux/fcntl.h>              // O_ACCMODE
#include <linux/mm.h>
#include <linux/init.h>
#include <asm/io.h>
#include <asm/uaccess.h>
#include <asm/hardware.h>
```


GPS Device Driver

```
static int gps_init(void);
static void gps_exit(void);

static int gps_init(void)
{
        GAFR1_L &=~GPIO_28;// Set GPDR1, BIT GPIO_46 = 0
        GAFR1_L |=GPIO_29;// Set GPDR1, BIT GPIO_46 = 1
        GAFR1_L |=GPIO_30;// Set GPDR1, BIT GPIO_47 = 1
        GAFR1_L &=~GPIO_31;// Set GPDR1, BIT GPIO_47 = 0

        printk("<<<<<<<<<  init module  >>>>>>>>₩n");
        return 0;
}

static void gps_exit(void)
{
        printk("<<<<<<<<<  exit module  >>>>>>>>₩n");
}

EXPORT_NO_SYMBOLS;
module_init(gps_init);
module_exit(gps_exit);
```


GPS Test Program

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/ioctl.h>
#include <sys/signal.h>
#include <sys/types.h>
#include <string.h>
#include <termios.h>

static char receive_buff[256];
static char info_data[30][20];
static int handle;

// GPGGA Data
        short GGAHour;
        short GGAMinute;
        short GGASecond;
        double GGALatitude;                     // < 0 = South, > 0 = North
        double GGALongitude;                    // < 0 = West, > 0 = East
        short GGAGPSQuality;        // 0 = fix not available, 1 = GPS sps mode, 2 = Differential GPS, SPS mode, fix valid,
                                    // 3 = GPS PPS mode, fix valid
        short GGANumOfSatsInUse;
        double GGAHDOP;
        double GGAAltitude;                     // Altitude: mean-sea-level (geoid) meters
        unsigned short GGACount;
        int GGAOldVSpeedSeconds;
        double GGAOldVSpeedAlt;
        double GGAVertSpeed;
```


GPS Test Program

```
// GPSA
        short GSAMode;                  // M = manual, A = automatic 2D/3D
        short GSAFixMode;               // 1 = fix not available, 2 = 2D, 3 = 3D
        double GSAPDOP;
        double GSAHDOP;
        double GSAVDOP;
        unsigned short GSACount;

// GPRMC
        short RMCHour;
        short RMCMinute;
        short RMCSecond;
        short RMCDataValid;             // A = Data valid, V = navigation rx warning
        double RMCLongitude;                    // current longitude
        double RMCGroundSpeed;          // speed over ground, knots
        double RMCCourse;               // course over ground, degrees true
        short RMCDay;
        short RMCMonth;
        unsigned short RMCYear;
        double RMCMagVar;               // magnitic variation, degrees East(+)/West(-)
        unsigned short RMCCount;
```


GPS Test Program

```
void data_reset(void)
{
// GPGGA Data
        GGAHour = 0;
        GGAMinute = 0;
        GGASecond = 0;
        GGALatitude = 0.0;                              // < 0 = South, > 0 = North
        GGALongitude = 0.0;                             // < 0 = West, > 0 = East
        GGAGPSQuality = 0;      // 0 = fix not available, 1 = GPS sps mode, 2 = Differential GPS, SPS
        mode, fix valid,
                                // 3 = GPS PPS mode, fix valid
        GGANumOfSatsInUse = 0;
        GGAHDOP = 0.0;
        GGAAltitude = 0.0;              // Altitude: mean-sea-level (geoid) meters
        GGACount = 0;
        GGAOldVSpeedSeconds = 0;
        GGAOldVSpeedAlt = 0.0;
        GGAVertSpeed = 0.0;

// GPGSA
        GSAMode = 'M';                          // M = manual, A = automatic 2D/3D
        GSAFixMode = 1;                         // 1 = fix not available, 2 = 2D, 3 = 3D
        GSAPDOP = 0.0;
        GSAHDOP = 0.0;
        GSAVDOP = 0.0;
        GSACount = 0;
```


GPS Test Program

```
// GPRMC
        RMCHour = 0;
        RMCMinute = 0;
        RMCSecond = 0;
        RMCDataValid = 'V';             // A = Data valid, V = navigation rx warning
        RMCLatitude = 0.0;              // current latitude
        RMCLongitude = 0.0;             // current longitude
        RMCGroundSpeed = 0.0;           // speed over ground, knots
        RMCCourse = 0.0;                // course over ground, degrees true
        RMCDay = 1;
        RMCMonth = 1;
        RMCYear = 2000;
        RMCMagVar = 0.0;                // magnitic variation, degrees East(+)/West(-)
        RMCCount = 0;
}

int gps_main(void)
{
        char Buff[256];
        int RxCount;
        int loop, input_count;
```


GPS Test Program

```
        while(1)
        {
                RxCount = read( handle, Buff, 1 );
                if( RxCount == 0 )
                {
                        printf( "Receive Time Over₩n" );
                        continue;
                }
                if( RxCount < 0 )
                {
                        printf( "Read Error₩n" );
                        break;
                }
                for( loop = 0; loop < RxCount; loop++ )
                {
                        receive_buff[input_count] = Buff[loop];
                        if (receive_buff[input_count] == '₩n')
                        {
                                receive_buff[input_count] = '₩0';
                                input_count=0;
                                printf("₩n%s₩n", receive_buff);
                                return 0;
                        }
                        else
                        {
                                input_count++;
                        }
                }
        }
        return 0;
}
```


GPS Test Program

```
int gps_parser(void)
{
        int i, j=0, index=0;

        for(i=0; receive_buff[i]; i++)
        {
         if (receive_buff[i] == ',') {
                        info_data[j][index] = '₩0';
                        j++;
         index = 0;
         }
         else
         {
           info_data[j][index] = receive_buff[i];
           index++;
         }
        }
        info_data[j][index] = '₩0';

//parser start  ================================================//

        char pBuff[10];

        if ((info_data[0][3] == 'G') && (info_data[0][4] == 'G'))    // If Data is $GPGGA
        {
//       printf("*********************************************GGA₩n");
        // Time
         // Hour
         pBuff[0] = info_data[1][0];
         pBuff[1] = info_data[1][1];
         pBuff[2] = '₩0';
         GGAHour = atoi(pBuff);
```


GPS Test Program

```
         // minute
         pBuff[0] = info_data[1][2];
         pBuff[1] = info_data[1][3];
         pBuff[2] = '₩0';
         GGAMinute = atoi(pBuff);


        // Second
         pBuff[0] = info_data[1][4];
         pBuff[1] = info_data[1][5];
         pBuff[2] = '₩0';
         GGASecond = atoi(pBuff);

        // Latitude
         GGALatitude = atof((char *)info_data[2]);

        // Nindicator
         if(info_data[3][0] == 'S')
         {
                        GGALatitude = -GGALatitude;
         }

        // Longitude
         GGALongitude = atof((char *)info_data[4]);

        // Eindicator
         if(info_data[5][0] == 'W')
         {
                        GGALongitude = -GGALongitude;
         }

        // GPS quality
         GGAGPSQuality = atoi(info_data[6]);
```


GPS Test Program

```
// Satellites in use
         pBuff[0] = info_data[7][0];
         pBuff[1] = info_data[7][1];
         pBuff[2] = '₩0';
         GGANumOfSatsInUse = atoi(pBuff);

        // HDOP
         GGAHDOP = atof((char *)info_data[8]);

        // Altitude
         GGAAltitude = atof((char *)info_data[9]);
         }


        else if ((info_data[0][3] == 'G') && (info_data[0][4] == 'S'))    // If Data is $GPGSA
        {
//       printf("*********************************************GSA₩n");

        // Mode
         GSAMode = info_data[1][0];

        // Fix Mode
         GSAFixMode = atoi(info_data[2]);

        // PDOP
         GSAPDOP = atof((char *)info_data[15]);

        // HDOP
         GSAHDOP = atof((char *)info_data[16]);

        // VDOP
         GSAVDOP = atof((char *)info_data[17]);
         }
```


GPS Test Program

```
else if ((info_data[0][3] == 'R') && (info_data[0][4] == 'M'))    // If Data is $GPRMC
        {
//       printf("*********************************************RMC₩n");

        // Time
         // Hour
         pBuff[0] = info_data[1][0];
         pBuff[1] = info_data[1][1];
         pBuff[2] = '₩0';
         RMCHour = atoi(pBuff);
         // minute
         pBuff[0] = info_data[1][2];
         pBuff[1] = info_data[1][3];
         pBuff[2] = '₩0';
         RMCMinute = atoi(pBuff);
         // Second
         pBuff[0] = info_data[1][4];
         pBuff[1] = info_data[1][5];
         pBuff[2] = '₩0';
         RMCSecond = atoi(pBuff);

        // Status incicator
         RMCDataValid = info_data[2][0];

        // latitude
         RMCLatitude = atof((char *)info_data[3]);

        if(info_data[4][0] == 'S')
        {
                        RMCLatitude = -RMCLatitude;
        }
```


GPS Test Program

```
// Longitude
        RMCLongitude = atof((char *)info_data[5]);

        if(info_data[6][0] == 'W')
        {
                RMCLongitude = -RMCLongitude;
        }

        // Ground speed
        RMCGroundSpeed = atof((char *)info_data[7]);

        // course over ground, degrees true
        RMCCourse = atof((char *)info_data[8]);

        // Date , Day
        pBuff[0] = info_data[9][0];
        pBuff[1] = info_data[9][1];
        pBuff[2] = '₩0';
        RMCDay = atoi(pBuff);

        // Month
        pBuff[0] = info_data[9][2];
        pBuff[1] = info_data[9][3];
        pBuff[2] = '₩0';
        RMCMonth = atoi(pBuff);

        // Year
        pBuff[0] = info_data[9][4];
        pBuff[1] = info_data[9][5];
        pBuff[2] = '₩0';
        RMCYear = 2000 + atoi(pBuff);
        }

        return 0;
}
```


GPS Test Program

```
int gps_print(void)
{
        printf("================================================₩n");
        printf("Current Time ₩t₩t: %d시 %d분 %d초₩n", GGAHour, GGAMinute, GGASecond);
        printf("GGA Latitude₩t₩t: %f₩n", GGALatitude);
        printf("GGA Longitude₩t₩t: %f₩n", GGALongitude);
        printf("GGA Altitude₩t₩t: %f₩n", GGAAltitude);
        printf("GGANum Of Sats In Use₩t: %d₩n", GGANumOfSatsInUse);
        printf("================================================₩n");
        printf("GSA A-Mode[A=automatic]₩t: %c₩n", GSAMode);
        printf("B-Mode[1=Not,2=2D,3=3D]₩t: %d₩n", GSAFixMode);
        printf("GSA PDOP₩t₩t: %f₩n", GSAPDOP);
        printf("GSA HDOP₩t₩t: %f₩n", GSAHDOP);
        printf("GSA VDOP₩t₩t: %f₩n", GSAVDOP);
        printf("================================================₩n");
        printf("Current Time ₩t₩t: %d시 %d분 %d초₩n", RMCHour, RMCMinute, RMCSecond);
        printf("RMC Latitude₩t₩t: %f₩n", RMCLatitude);
        printf("RMC Longitude₩t₩t: %f₩n", RMCLongitude);
        printf("RMC Ground Speed₩t: %f₩n", RMCGroundSpeed);
        printf("RMC Course₩t₩t: %f₩n", RMCCourse);
        printf("RMC data Valid₩t₩t: %c₩n", RMCDataValid);
        printf("Current Date ₩t₩t: %d년 %d월 %d일₩n", RMCYear, RMCMonth, RMCDay);
        printf("================================================₩n");

        return 0;
}
```


GPS Test Program

```
int main()
{
        struct termios oldtio, newtio;

        handle = open( "/dev/ttyS2", O_RDWR | O_NOCTTY );     // GPS PORT OPEN
        if( handle < 0 )
            {
                printf( "Gps Port Open Fail [/dev/ttyS2]₩r₩n " );
                return -1;
            }
        tcgetattr( handle, &oldtio );
        memset( &newtio, 1, sizeof(newtio) );
        newtio.c_cflag = B9600 | CS8 | CLOCAL | CREAD ;
        newtio.c_iflag = IGNPAR | ICRNL;
        newtio.c_oflag = 0;
        newtio.c_lflag = 0;
        newtio.c_cc[VTIME] = 0;
        newtio.c_cc[VMIN]  = 0;

        tcflush( handle, TCIFLUSH );
        tcsetattr( handle, TCSANOW, &newtio );
        data_reset();

        int nCmd, nState = 1;

        printf("Coose what you want to do ₩n");
        printf("₩t1 : GPS Viewer ₩n");
        printf("₩t0 : QUIT ₩n");
        data_reset();
```


GPS Test Program

```
while(nState)
  {
        printf("Enter the command : ");
        scanf("%d", &nCmd);
        printf("₩n");

        switch(nCmd)
        {
                case 1:
                        printf("1. Current Time, latitude, longitude viewer₩n");
                        gps_main();
                        gps_parser();
                        gps_print();
                        break;
                case 0:
                        nState = 0;
                        printf("Program EXIT₩n");
                        exit(0);
                 default:
                        break;
        }
  }

  tcsetattr( handle, TCSANOW, &oldtio );
  close( handle );
  return 0;
}
```


GPS 실행

- GPS는 /dev/ttyS2를 사용하며, UART 통신을 한다.
  #ls -alsF /dev/ttyS2
  명령을 사용하여 ttyS2가 있는 지를 확인한다.

  없을 경우
  #la -alsF /dev/ttyS1
  #mknod /dev/ttyS2 c 4 66

- GPS가 UART 통신을 하기 위해서는 전체 Off 되어있는 Dip Swich 의 3번, 6번을 On 시켜야 한다.

GPS 실행

- Device Driver 및 Test Application를 컴파일 해준다.
  #arm-linux-gcc -c -D__KERNEL__ -DMODULE -Wall -O2 -o gps_driver.o gps_driver.c
  # arm-linux-gcc -o test_gps test_gps.c

- Device Driver 등록
  #insmod gps_driver.o

- 실행 Test
  #./test_gps

동작 화면

```
Enter the command : 1

1. Current Time, latitude, longitude viewer

$GPGGA,054846.00,3714.2757,N,12704.2540,E,1,03,11.3,-00012,M,019,M,,*44
=============================================
Current Time            : 5시 48분 46초
GGA Latitude            : 3714.275700
GGA Longitude           : 12704.254000
GGA Altitude            : -12.000000
GGANum Of Sats In Use   : 3
=============================================
GSA A-Mode[A=automatic] : M
B-Mode[1=Not,2=2D,3=3D] : 1
GSA PDOP                : 0.000000
GSA HDOP                : 0.000000
GSA VDOP                : 0.000000
=============================================
Current Time            : 5시 48분 45초
RMC Latitude            : 3714.275700
RMC Longitude           : 12704.254000
RMC Ground Speed        : 0.000000
RMC Course              : 0.000000
RMC data Valid          : A
Current Date            : 2005년 7월 27일
=============================================
Enter the command :
```