GPS Signal Description

1. The baseband signal **transmited** by the satellite is given as

$$S(t) = S_{PPS}(t) + jS_{SPS}(t) \tag{1}$$

- $S_{SPS}(t) = \sum_{i=-\infty}^{\infty} c_{sps}(|i|_{L\_sps}).d([i]_{CD\_sps}).rect_{T_{c,sps}}(t - iT_{c,sps})$ — Standard Positioning Service
- $S_{PPS}(t) = \sum_{i=-\infty}^{\infty} c_{pps}(|i|_{L\_pps}).d([i]_{CD\_pps}).rect_{T_{c,pps}}(t - iT_{c,pps})$ —- Precision Positioning Service

2. Let $x_{in}[n]$ be the incoming signal at the **receiver** end and is given as

$$x_{in}[n] = A(t)s_T(t - \tau(t))e^{j(2\pi f_D(t)t + \phi(t))}|_{t=nT_s} + n(t)|_{t=nT_s} \tag{2}$$

where

- $A(t)$ is Amplitude
- $s_T(t)$ is Complex baseband signal
- $\tau(t)$ is code delay(time varying)
- $f_D(t)$ is Doppler shift(time varying)
- $\phi(t)$ is carrier phase shift(time varying)
- $n(t)$ is Random noise with zero mean
- T_s is Sampling period
- f_s is Sampling frequency

Pseudo code for GPS Signal Acquisition

1.1 Functions for computing the PRN codes of GPS satellite

1. (a) g1_lfsr()

```
int16 state = 0x3FF
int8 out[1023]
int8 new_bit
for i=0 to i=1022
    out[i] = (state ≫ 9) & 0x1
    new_bit = ((state ≫ 9) ⊕ (state ≫ 2)) & 0x1
    state = ((state ≪ 1) | new_bit) & 0x3FF
end for
return out
```

   (b) g2_lfsr(**int8** tap0,**int8** tap1)

```
int16 state = 0x3FF
int8 out[1023]
tap0 = tap0-1
tap1 = tap1-1
for i=0 to i=1022
    out[i] = ((state ≫ tap0) ⊕ (state ≫ tap1)) & 0x1
    new_bit = ((state ≫ 9) ⊕ (state ≫ 8) ⊕ (state ≫ 7) ⊕ (state ≫ 5) ⊕ (state ≫ 2) ⊕ (state ≫ 1)) & 0x1
    state = ((state ≪ 1) | new_bit) & 0x3FF
end for
return out
```

   (c) combine_g1_g2(**int16** *g1,**int16** *g2)

```
int8 out[1023]
for i=0 to i=1022
    out[i] = g1[i] ⊕ g2[i]
end for
return out
```

1.2 Main function

int32 f_c = 1023000 /* PRN code frequency */

int32 f_s = 2048000 /* Sampling frequency */

int16 N = 2048 /* Samples for 1ms is 2048 */

cint16 incoming_signal [4096] /* Incoming 2ms Samples */

cint16 *incoming_signalpower

/*Calculate the power of signal*/

CEVA_DSP_LIB_MAT_CX_MUL_TRANS_Q15(**x**,**1**,**2048**,**incoming_signalpower**)

The power of incoming signal should be **incoming_signalpower**> **threshold** . If true, **go to** step **8**. else, **stop** the process

int16 max_power[5] = {0}

int8 visible_satellites_withMaxPower[5] = {0}

int16 codePhase[5] = {0}

int16 visible_PRN_codes[5][2048] = {{0}} /* Matrix to store the PRN codes of visible satellites */

int32 log2_FFT_size = 11; /* FFT size = 2^{11} */

int16 temp[4096]

int16 input_signal_FFT[4096]

/********** lookup tables for computing fft of size 2048 **********/

int16 temp_buff[4096]

int8 ScalVal[13] = 0

int32 br = 1

/***************Copying the incoming signal (real and imaginary parts as required for FFT function) to temp for computing fft***************/

for i = 0 to i = 2047

temp[2***i**] = incoming_signal[**i**].re

temp[2***i** + 1] = incoming_signal[**i**].im

end for

/************ Function for computing the FFT of input signal ************/

CEVA_FFT_LIB_CX16_FFT(log2_FFT_size, temp, input_signal_FFT , CEVA_FFT_LIB_cos_sin_fft_16 , (int16*)bitrev16bit_16_to_2048, temp_buff , ScaleVal, br)

int8 SVs[32][2] = { {2, 6}, {3, 7}, {4, 8}, {5, 9}, {1, 9}, {2, 10}, {1, 8}, {2, 9}, {3, 10}, {2, 3}, {3, 4}, {5, 6}, {6, 7}, {7, 8}, {8, 9}, {9, 10}, {1, 4}, {2, 5}, {3, 6}, {4, 7}, {5, 8}, {6, 9}, {1, 3}, {4, 6}, {5, 7}, {6, 8}, {7, 9}, {8, 10}, {1, 6}, {2, 7}, {3, 8}, {4, 9} };

int8 g1[1023] /* Array for g1 LFSR */

int8 g2[1023] /* Array for g2 LFSR */

g1 = g1_lfsr() /* Fucntion call */

int8 tap0,tap1

int16 gold_code[2048]

int16 PRN_code_FFT[4096]

int16 M[4096]

cint16 INABS[2048]

uint16 OUTABS[2048]

for sv=0 to **sv**=31:

/******** PRN code generation ********/

int8 index=0 /* index for iterating the SVs arraay */

tap0 = SVs[**sv**][index]

tap1 = SVs[**sv**][index++]

g2 = g2_lfsr(tap0,tap1) /* Fucntion call */

gold_code = combine_g1_g2(g1,g2) /* Fucntion call */

for i = 0 to **i** = 1022

if gold_code[**i**] > 0

gold_code[**i**] = -1

else

gold_code[**i**] = 1

/* Upsampling the PRN code */

for i = 0 to **i** = 1022

c[**2i**] = gold_code[**i**]

c[**2i + 1**] = gold_code[**i**]

end for

c[2046] = 0

c[2047] = 0

/***************Copying the PRN code (real and imaginary parts as required for FFT function) to temp for computing FFT***************/

for i = 0 to **i** = 2047

temp[2**i**] = c[**i**]

temp[2**i** + 1] = 0

end for

/********* Computing the FFT for PRN code ***************/

CEVA_FFT_LIB_CX16_FFT(log2_FFT_size, temp, PRN_code_FFT, CEVA_FFT_LIB_cos_sin_fft_16 , (int16*)bitrev16bit_16_to_2048,temp_buff , ScaleVal, br)

/********** Multiplying the FFT of input signal and conjugate of FFT of PRN codes **********/

for i = 0 to **i** = 2047

M[2**i**] = input_signal_FFT [2**i**] * PRN_code_FFT[2**i**] + input_signal_FFT [2**i**+1] * PRN_code_FFT[2**i**+1]

M[2**i**+1] = PRN_code_FFT[2**i**] * input_signal_FFT [2**i**+1] - input_signal_FFT [2**i**] * PRN_code_FFT[2**i**+1]

end for

/******* Computing IFFT for the resultant signal **********/

CEVA_FFT_LIB_CX16_IFFT (log2_FFT_size, M, temp , CEVA_FFT_LIB_cos_sin_fft_16 , (int16*)bitrev16bit , ScaleVal, br)

/********** Copy the IFFT output to the complex array for finding power ********/

for i = 0 to i = 2047

INABS[**i**].re = temp[**2i**]

INABS[**i**].im = temp[**2i+1**]

end for

/******** function for computing the absolute value in array **********/

CEVA_DSP_LIB_VEC_CX_ABS_Q15 (INABS,OUTABS,2048,1)

int16 max = 0

for n=0 to **n** = N-1

if OUTABS[**n**] > max:

max = OUTABS[**n**]

max_index = **n**

end if

end for

/* Update max_power, visible_satellites_withMaxPower and codePhase arrays and the corresponding PRN codes are stored */

for i = 0 to **i** = 4

if max > max_power[**i**]

for j = 4 to **j** = **i**-1

max_power[**j**] = max_power[**j**-1]

visible_satellites_withMaxPower[**j**] = visible_satellites_withMaxPower[**j**-1]

codePhase[**j**] = codePhase[**j**-1]

j = **j**-1

end for

max_power[**i**] = max

visible_satellites_withMaxPower[**i**] = **sv**

codePhase[**i**] = max_index

for b = 0 to **b** = 2048

visible_PRN_codes[**i**][**b**] = c[**b**]

end for

break the loop

end if

end for

end for

/* Finding the Doppler shift for 5 satellites */

int16 p_inp[2048]

int16 cos_sin_out[4096]

cint32 x_sh[2048]

cint16 code[2048]

cint16 *signal_power

int16 fd[5]

for sv = 0 to **sv** = 4

Code phase $\hat{\tau}$ = codePhase[**sv**]

Initialize max_of_max=0

Initialize max_fd = 0

for $f_D = f_{min}$ to $f_D = f_{max}$ in f_{step} steps:

/********* Computing the x[n].$e^{-j2\pi F_D t}$, for n = 0 to N-1 *********/

for i = 0 to **i** = 2047

p_inp[**i**] = (2 * pi * f_D * **i** * 10430)/2048000

end for

CEVA_DSP_LIB_COSSIN_Q15(p_inp,cos_sin_out,2048)

for i = 0 to **i** = 2047

x_sh[**i**].re = x[**i** + $\hat{\tau}$].re * cos_sin_out[2i] + x[**i** + $\hat{\tau}$].im * cos_sin_out[2i + 1]

x_sh[**i**].im = x[**i** + $\hat{\tau}$].im * cos_sin_out[2i] - x[**i** + $\hat{\tau}$].re* cos_sin_out[2i+1]

code[**i**].re = visible_PRN_codes[**sv**][**i**]

code[**i**].im = 0

end for

/******Multiply the PRN code with Shifted signal and Compute the power of signal *****/

CEVA_DSP_LIB_MAT_CX_MUL_Q15(x_sh,code,signal_power,1,2048,1)

z = *signal_power.re

if (z > max_of_max)

max_of_max = z

max_fd = f_D

end if

end for

Doppler Frequency offset fd[**sv**] = max_fd

end for