

TN00129
Dec. 20, 1996

Generating Very Long FM Signal

In APOS, FM signal is usually generated using cumsum() function. When the signal is very long, result from the cumsum() may exceed floating point maximum and roll over.

As a result, a discontinuity will occur in the final signal.

One solution to this problem is: Do cumsum() on smaller buffers and concatenate them together. Make sure the angle is continuous in 360 degrees.

A routine has been written for this purpose, namely mycumsum(). There are two arguments in mycumsum(arg1, arg2). arg1 is the length of the small buffer, arg2 is a modular number used to prevent roll-over. For this purpose, arg2 is 2*PI. While calling this routine, the long buffer to be cumsumed should be on the top of the stack.

Attached below is the routine source code and test program. In the test program, effort has been made to make a very long FM signal possible:

```
/******  
FMTEST.C test program
```

This program generates a long FM signal.

Long FM is made in smaller buffers, each has 500000 points.

When cumsum is used for long buffer, data roll-over may happen (due to data type size). This routine (mycumsum) applies cumsum to smaller sections of a buffer and get rid of the problem.

```
*****/  
  
#include <stdlib.h>  
#include <math.h>  
#include <conio.h>  
#include <stdio.h>  
#include <string.h>  
#include <dos.h>  
  
#include "xbdrv.h"  
#include "apos.h"  
  
#define NPT1 500000L // small buffer length  
#define NPT2 5000L // cumsum section length  
#define WinLen 5.0 // buffer window length in ms  
  
void mycumsum(long size_block, float mod_value);  
  
void main()
```

```

{
int TONE1, TONE2, curbuf, PLAY_SPEC, CH1_SEQ;
float freqS, freqE, freqN, freq1, freq2;
float Srate, Dura;
long NPTS;
int sectN, i, first=1;

char tempF[10];

    if(!XB1init(USE_DOS))
    {
        printf("\n\n\nXBUS Error!!!\n\n\n");
        exit(0);
    }

    if(!apinit(APa))
    if(!apinit(APb))
    {
        printf("\n\n\nAP2 Error!!!\n\n\n");
        exit(0);
    }

clrscr();
gotoxy(5, 5);
printf("Enter start/end freq. (Hz), sampling rate (kHz) & duration (sec):\n");
scanf("%f %f %f %f", &freqS, &freqE, &Srate, &Dura);

strcpy(tempF, "junk.16");
NPTS = (long)(1000.0*Srate*Dura);
sectN = (int)((float)NPTS / (float)NPT1) + 0.51);
NPTS = (long)sectN * NPT1;
freqN = (freqS - freqE) / (float)sectN;
Srate = 1000.0 / Srate;

TONE1 = _allot16(NPT1);
TONE2 = _allot16(NPT1);
PLAY_SPEC = _allot16(20);
CH1_SEQ = _allot16(20);

/* making PLAY sequence ----- */
    dpush(10);
    make(0, CH1_SEQ);
    make(1,0);
    qpop16(PLAY_SPEC);

    dpush(10);
    make(0,TONE1);
    make(1,1);
    make(2,TONE2);
    make(3,1);
    make(4,0);
    qpop16(CH1_SEQ);

clrscr();

```

```

gotoxy(5, 5);
printf(" Generating Signal . . .\n\n");
for(i=0; i<sectN; i++)
{
    freq1 = freqN * (float)(sectN - i) + freqE;
    freq2 = freq1 - freqN;
    printf("Buffer: %d: %.0f - %.0f Hz\n", i, freq1, freq2);

    dpush(NPT1);
    fill(1.0, (freq2-freq1)/(float)NPT1/freq1);
    scale(freq1);
    scale(2.0);
    scale(M_PI);
    scale(Srate/1.0e6);

    mycumsum(NPT2, 2.0*M_PI); // first arg is the section length

    sine();
    scale(8000.0);
    qwind(WinLen, Srate);
    qpop16(TONE1);

    if(first)
        dama2disk16(TONE1, tempF, 0);
    else
        dama2disk16(TONE1, tempF, 1);
    first = 0;
}

printf("\n\n Presenting FM Signal . . .\n");

    DAClear(1);          /* First Clear DA3          */
    DASrate(1,Srate);    /* Spec the sample rate    */
    DANpts(1, NPTS);
    DAMode(1,DAC1);     /* Make DAC1 active        */

curbuf = TONE1;

disk2dama16(TONE1, tempF, 0);
seqplay(PLAY_SPEC);
DAarm(1);
DAgo(1);

disk2dama16(TONE2, tempF, NPT1);
i = 2;

/* play loop: ----- */
do
{
    do
    {
        if(kbhit()) /* if keyboard hit: */
        {
            DAstop(1);
        }
    }
}

```

```

                exit(0);
            }
        }while(playseg(1)==curbuf && DAstatus(1));

/* after curbuf is played... ----- */

        if(curbuf==TONE1)      /* switch curbuf      */
        {
            if(i<sectN)
                disk2dama16(TONE1, tempF, (long)i*NPT1);
            curbuf=TONE2;
        }
        else
        {
            if(i<sectN)
                disk2dama16(TONE2, tempF, (long)i*NPT1);
            curbuf=TONE1;
        }

        i++;

    }while(DAstatus(1));
    DAstop(1);
}

```

```

void mycumsum(long size_block, float mod_value)
{
    long int total_pts, dummy_int;
    int i, loops, frac_end=0;
    float end_value, module, dummy;

    total_pts = topline();
    loops = (int)(total_pts/size_block);
    if(total_pts%size_block > 0) frac_end = 1;

    module = 0.0;
    for(i=0; i<loops; i++)
    {
        block(i*size_block, i*size_block);
        shift(module);
        block(i*size_block, (i+1)*size_block-1);
        cumsum();
        noblock();
        end_value = whatis((i+1)*size_block-1);
        dummy = end_value/mod_value;
        dummy_int = (long int)dummy;
        module = mod_value*(dummy -(float)dummy_int);
    }

    if(frac_end)
    {
        block(loops*size_block, loops*size_block);
    }
}

```

```
shift(module);
block(loops*size_block, total_pts-1);
cumsum();
nblock();
}
}
```