Generator funkcyjny z karty dźwiękowej

Generator funkcji jest podstawowym narzędziem elektronika konstruktora. Jednak fabryczne generatory są drogie i nie każdy może sobie pozwolić na taki luksus. Alternatywnym rozwiązaniem jest generator software'owy, czyli program komputerowy sterujący kartą dźwiękową. Koszt tego rozwiązania żaden, a program może być w wielu przypadkach użyteczny.

Podstawowe informacje:
- Zakres częstotliwości wyjściowej: 1 Hz...20 kHz (sinus).
- Rozdzielczość ustawiania częstotliwości: 0,1 Hz.
- Maksymalna amplituda napięcia wyjściowego: zależna od karty dźwiękowej.
- Generowanie przebiegów o kształtach: sinusoidalnym, trójkątnym, prostokątnym.

W opisanym generatorze wykorzystuje się przetworniki C/A na karcie dźwiękowej z częstotliwością próbkowania 96 kHz lub 44,2 kHz na kartach starszego typu. Sygnał jest pobierany z wyjść karty dźwiękowej, które są najczęściej typu jack. Oczywiście, jak widać z samej zasady działania, generator taki ma gorsze parametry niż generator sprzętowy. Generatory pozwalają uzyskać częstotliwości sygnału wyjściowego powyżej 1 MHz, a ten jedynie kilkadziesiąt kiloherców i to tylko dla sinusoidy. Do badania urządzeń akustycznych to jednak w zupełności wystarczy, a z powodu niskiej ceny nasz generator może być konkurencyjny w stosunku do firmowych. Inną zaletą jest to, że przy niskich częstotliwościach możemy uzyskać bardzo małe zniekształcenia. Sygnał odwzorowany jest z rozdzielczością 16 bitów, czyli 65536 poziomów napięć. Dodatkowym mankamentem kart dźwiękowych jest to, że na sygnał wyjściowy nakładają się zakłócenia elektromagnetyczne wytwarzane przez płytę główną komputera oraz to, że wyjście jest separowane kondensatorem sprzęgającym o niewielkiej pojemności, co ogranicza pasmo od dołu.

Zasada działania

Do bufora o odpowiedniej długości wpisywane są kolejne wartości chwilowe amplitudy sygnału generowanego. Następnie bufor jest cyklicznie odczytywany przez program obsługi urządzeń typu „Wave", a zapisane w nim wartości są kolejno podawane na wyjście przetwornika C/A karty dźwiękowej. Szybkość odczytu z bufora jest stała i równa maksymalnej częstotliwości próbkowania dla danej karty. Z tego powodu o częstotliwości wytwarzanego przebiegu decyduje rozłożenie danych w buforze.

Należy również zauważyć, że dane powinny być tak rozłożone, aby mieściły na całej długości bufora całkowitą liczbą okresów. Wynika to z faktu, że gdy licznik procedury odczytującej dojdzie do końca bufora, zostaje wyzerowany i odczytywanie zaczyna się od początku. Gdybyśmy nie zachowali ciągłości sygnału, wystąpiłby silny impuls zakłócający przy przejściu licznika do początku bufora.

Należy zauważyć pewną prawidłowość. Częstotliwość sygnału wyjściowego jest proporcjonalna do liczby okresów zawartych w buforze oraz zależy od czasu odczytu całego bufora. Można to wyrazić wzorem:

$$f_g = \frac{N}{T_{buf}}$$

gdzie:
- N: liczba okresów w całym buforze,
- T_{buf}: czas odczytu całego bufora.

W naszym przypadku chcemy, aby częstotliwość ustawiana była z rozdzielczością 0,1 Hz, więc długość bufora ustalamy tak, aby całkowity czas odczytu wynosił 10 s. Można to wyrazić za pomocą następującego wzoru:

$$L_{buf} = f_p \cdot T_{buf} \cdot Kanały \cdot Bajty$$

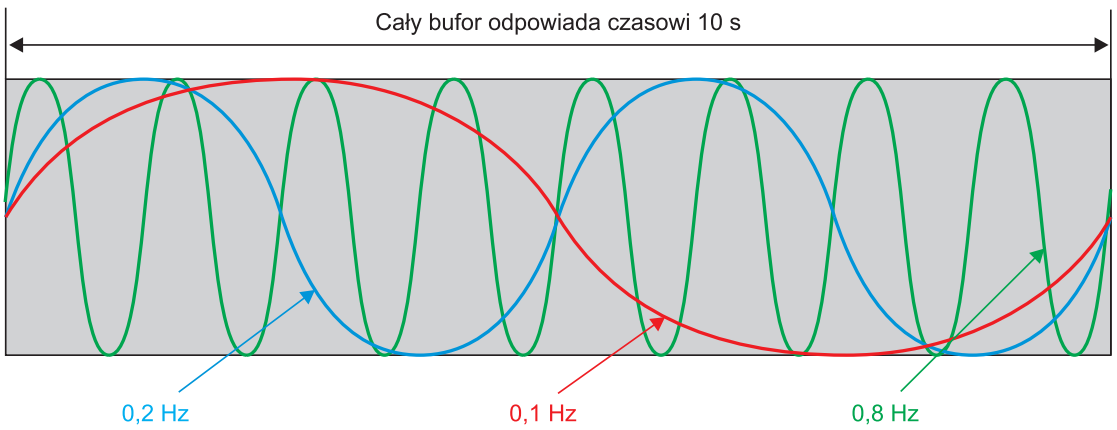
gdzie:
- f_p: częstotliwość próbkowania.
- T_{buf}: czas odczytu całego bufora.
- kanały: liczba kanałów.
- bajty: liczba bajtów na próbkę.

W naszym przypadku f_p=96 kHz, T_{buf}=10 s, kanały=2 (stereo), bajty=2 (16 bitów). Dane są zapisywane w postaci liczby całkowitej ze znakiem w notacji U2, czyli odpowiadają typowi *short* języka C++.

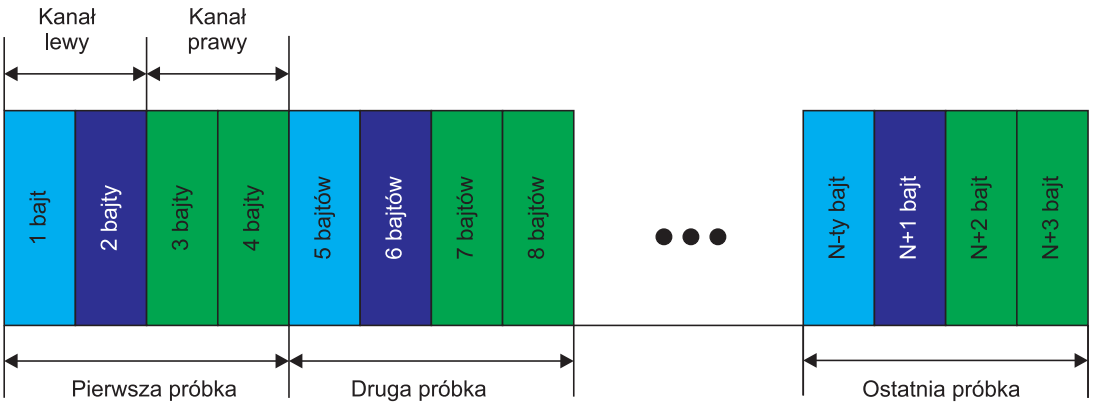
Opisane wyżej zależności pokazano graficznie na **rysunku 1** i **rysunku 2**.

Interfejs użytkownika

Interfejs użytkownika napisano w postaci okna dialogowego o stałej wielkości. Można w nim wydzielić dwie podstawowe sekcje: ustawiania częstotliwości oraz ustawiania amplitudy. Parametry te można zmieniać za pomocą suwaków lub wpisując odpowiednie wartości w pola tekstowe. Do zmiany zakresów służą przyciski typu *„Radio button"* i odpowiednio dla częstotliwości są to:



Rysunek 1. Związek między rozmieszczeniem danych w buforze a częstotliwością



Rysunek 2. Rozmieszczenie danych w buforze

– 100 Hz: umożliwia uzyskanie częstotliwości do 100 Hz,
– 1 kHz: umożliwia uzyskanie częstotliwości do 1 kHz,
– 20 kHz: umożliwia uzyskanie częstotliwości do 20 kHz.

Dla amplitudy są to:

– 1:1: maksymalna amplituda równa maksymalnemu napięciu wyjściowemu karty dźwiękowej,
– 1:10: maksymalna amplituda równa 10% maksymalnego napięcia wyjściowego karty dźwiękowej,
– 1:100 maksymalna amplituda równa 1% maksymalnego napięcia wyjściowego karty dźwiękowej.

Przy wpisaniu do pola tekstowego odpowiedniej wartości położenia suwaków oraz przyciski zakresów są aktualizowane automatycznie.

Ponadto, przyciskami w sekcji ustawiania kształtu możemy zmieniać kształt generowanego sygnału. Do wyboru są opcje podobne jak w typowym generatorze funkcji, czyli: sinus, trójkąt, prostokąt.

W przypadku gdy w komputerze mamy więcej niż jedno urządzenie typu „Wave Audio", możemy je wybrać za pomocą kontrolki typu „Combo box".

Po przyciśnięciu przycisku „Parametry" wyświetlane są parametry bieżącej (wybranej) karty dźwiękowej, natomiast wybranie przycisku „Informacja" powoduje wyświetlenie podstawowych informacji na temat programu.

Procedura generująca dźwięk

Działanie tej procedury w zasadzie polega na odpowiednim wypełnieniu bufora danych oraz uruchomieniu odpowiedniej procedury odtwarzającej. Ważniejsze fragmenty programu realizującego generator funkcyjny umieszczono na **listingu 1**, którego dokładne przestudiowanie pozwoli zrozumieć sposób działania programu.

Dokładniejszy opis funkcji „Wave Audio" można znaleźć na stronie: *http://msdn.microsoft.com/en-us/library* w sekcji: *„Win32 and COM Development/Audio and Video/Windows Multimedia/Multimedia Audio/Waveform Audio*.

Program napisano jako zwykłą aplikację Win32, niewykorzystującą żadnych rozszerzeń ani dodatkowych bibliotek. Projekt został skompilowany z użyciem *Microsoft-Visual C++ ver.6.0*, ale można go zaimportować do nowszych wersji.

Tomasz Krogulski
tomaszkro@op.pl

Listing 1. Ważniejsze fragmenty programu generatora funkcyjnego

```
                                                //zmienne globalne:
char FU_Buffer[96000*4*10];        bufor odtwarzanych danych;
HWAVEOUT FU_Hw = NULL;     handler urządzenia „Wave Audio";
WAVEFORMATEX      FU_Wf;   struktura parametrów przetwarzania;
WAVEHDR  FU_Hdr;  nagłówek bufora;

/*parametry:
        freq - częstotliwość*10;
        amp - amplituda (0-100000);
        shape - kształt: 0 - sinus, 1 - trójkąt, 2 - prostokąt;
        idx - indeks urządzenia ;
        chn - liczba kanałów;
        fsample - częstotliwość próbkowania; */

void StartWave(int freq, int amp, int shape, int idx, int chn, int fsample)
{
        if(FU_Hw)                                   // Zamknięcie urządzenia „Wave Audio", jeśli jest otwarte
        {
                waveOutReset(FU_Hw);
                waveOutUnprepareHeader(FU_Hw, &FU_Hdr,  sizeof(FU_Hdr));
                waveOutClose(FU_Hw);
        }

        FU_Wf.nChannels = chn;                      // wypełnienie struktury
        FU_Wf.nSamplesPerSec = fsample;             // parametrów przetwarzania
        FU_Wf.wBitsPerSample = 16;                  // która jest następnie przekazana
        FU_Wf.nAvgBytesPerSec = fsample*4;          // do funkcji "waveOutOpen"
        FU_Wf.wFormatTag = WAVE_FORMAT_PCM;
        FU_Wf.nBlockAlign = 4;
        FU_Wf.cbSize = 0;
                                                    // otwarcie urządzenia „Wave Audio"
        MMRESULT re1 = waveOutOpen(&FU_Hw, idx, &FU_Wf, NULL, NULL, CALLBACK_NULL);
                                                    // wypełninie struktury nagłówka w której zawarty jest sposób
                                                    // odtwarzania dźwięku (sygnału)
        FU_Hdr.dwBufferLength = fsample*chn*2*10;   // długość bufora
        FU_Hdr.lpData = FU_Buffer;                  // adres bufora
```

Listing 1. cd.

```
        FU_Hdr.dwLoops = 0xFFFFFFFF;                // liczba powtórzeń odczytu (całego
                                                    // bufora) wpisałem maksymalną wartość, co w praktyce oznacza
                                                    // nieskończoną liczbę powtórzeń
        FU_Hdr.dwFlags = WHDR_BEGINLOOP|WHDR_ENDLOOP;
                                                    // Nagłówek jest pierwszym i zarazem ostatnim w pętli odtwarzania.
        union {
                char    *cpt;
                short   *wpt;
                short   (*lpt)[2];
        } pt;
        int     i;

        pt.cpt = FU_Buffer;
        for(i=0; i<fsample*10; ++i)                 // wypełnienie bufora odtwarzania
        {
                short val;
                if(shape == 0)                      // sinusoida
                {
                                                    // wyliczanie wartości amplitudy
                        val = (short)(32767.0*sin((double)2*M_PI*i*freq/(fsample*10.0)));
                }
                else if(shape == 1)                 // trójkąt
                {
                        double freal = freq/10;
                                                    // wyliczanie wartości amplitudy
                        double tf = (double)i/(double)fsample-((double)(int)((double)i*freal/(double)fsample))/freal;
                        if(tf<=(0.5/freal))
                        {
                                val = 32767.0*(4.0*tf*freal-1.0);
                        } else
                        {
                                val = 32767.0*(3.0-4.0*tf*freal);
                        }
                }
                else                                // prostokąt
                {
                        double freal = freq/10;
                                                    // wyliczanie wartości amplitudy
                        double tf = (double)i/(double)fsample-((double)(int)((double)i*freal/(double)fsample))/freal;
                        val = tf>(0.5/freal) ? -32768 : 32767;
                }
                if(chn==1)                          // przepisanie wartości do bufora
                        pt.wpt[i] = val;
                else {
                        pt.lpt[i][0] = val;
                        pt.lpt[i][1] = val;
                }
        }

                                                    // aby odtworzyć zawartość bufora, należy najpierw utworzyć nagłówek
                                                    // za pomocą
                                                    // funkcji        „waveOutPrepareHeader”
        waveOutPrepareHeader(FU_Hw, &FU_Hdr,  sizeof(FU_Hdr));
                                                    // ustawianie amplitudy za pomocą funkcji “waveOutSetVolume”
        waveOutSetVolume(FU_Hw,(unsigned short)((double)amp/100000.0*65535.0));
                                                    // rozpoczęcie odczytu - czyli start generatora
        waveOutWrite(FU_Hw, &FU_Hdr,  sizeof(FU_Hdr));
}

                                                    // Opis funkcji interfejsu API Windowsów obsługi urządzeń “Wave
                                                    // AUDIO” wykorzystywanych w naszej procedurze:
                                                    // Otwarcie danego urządzenia “Wave AUDIO”
MMRESULT waveOutOpen(
        LPHWAVEOUT phwo,                            // zwracany handler dla nowo otwartego urządzenia „Wave AUDIO”
        UINT_PTR uDeviceID,                         // numer urządzenia
        LPWAVEFORMATEX pwfx,                        // struktura z parametrami odtwarzania
        DWORD_PTR dwCallback,                       // wskaźnik do procedury obsługi odtwarzania
        DWORD_PTR dwCallbackInstance,               // dana przesyłana do procedury obsługi
        DWORD fdwOpen                               // flagi - w naszym przypadku 0
);

                                                    // Zamknięcie urządzenia „Wave AUDIO”
waveOutClose(
        LPHWAVEOUT phwo                             // handler urządzenia
);

                                                    // Wyłączenie urządzenia (zatrzymanie odtwarzania)
waveOutReset(
        LPHWAVEOUT phwo                             // handler urządzenia
);
```

Listing 1. cd.

```
                                        // Przygotowanie nagłówka - trzeba wywołać przed wywołaniem
                                        // „waveOutWrite”
waveOutPrepareHeader(
        LPHWAVEOUT phwo                 // handler urządzenia
        LPWAVEHDR pwh,                  // przygotowywany nagłówek
        UINT cbwh                       // wielkość nagłówka
);

                                        // Zwolnienie nagłówka oraz związanego z nim bufora danych
                                        // trzeba ją wywołać przed zwolnieniem bufora danych
waveOutUnprepareHeader(
        HWAVEOUT hwo,                   // handler urządzenia
        LPWAVEHDR pwh,                  // przygotowywany nagłówek
        UINT cbwh                       // wielkość nagłówka
);

                                        // Ustawia poziom głośności
waveOutSetVolume(
        HWAVEOUT hwo,                   // handler urządzenia
        DWORD dwVolume                  // poziom głośności 0-65536
);

                                        // Procedura rozpoczynająca odtwarzanie dźwięku (zapis do kraty
                                        // dźwiękowej)
waveOutWrite(
        HWAVEOUT hwo,                   // handler urządzenia
        LPWAVEHDR pwh,                  // wskaźnik do nagłówka
        UINT cbwh                       // wielkość nagłówka
 );

                                        // Opis struktur wykorzystywanych przez powyższe funkcje
 WAVEFORMATEX                           // struktura z parametrami odtwarzania
    Poszczególne pola:
        WORD wFormatTag;                // typ format urządzenia
        WORD nChannels;                 // liczba kanałów
        DWORD nSamplesPerSec;           // częstotliwość przetwarzania
        DWORD nAvgBytesPerSec;          // średnia ilość bitów na sekundę = nBlockAlign*nSamplesPerSec
        WORD nBlockAlign;               // liczba bajtów dla jednej próbki - w naszym przypadku 4
        WORD wBitsPerSample;            // liczba bitów jednej próbki
        WORD cbSize;                    // rozmiar dodatkowych danych - w naszym przypadku 0

 WAVEHDR                                // nagłówek danych do odtwarzania
    Poszczególne pola:
        LPSTR lpData;                   // adres bufora
        DWORD dwBufferLength;           // długość bufora
        DWORD dwBytesRecorded;          // liczba odczytanych bajtów
        DWORD_PTR dwUser;               // dane użytkownika
        DWORD dwFlags;                  // najważniejsze flagi to: WHDR_BEGINLOOP - pierwszy nagłówek w
                                        // pętli, WHDR_ENDLOOP - ostatni nagłówek;
        DWORD dwLoops;                  // liczba powtórzeń pętli
        struct wavehdr_tag *lpNext;     // wskaźnik do następnego nagłówka
        DWORD_PTR reserved;             // zarezerwowane
```