



VOICE PROGRAMMING

by :

**PRIMA KRISTALINA
(DIGITAL COMMUNICATION LAB.)**

Presented On :

**INHOUSE TRAINING ON
TELECOMMUNICATION DEPT.**

16 DECEMBER 2005



Pemrograman Voice meliputi 3 hal :

- Voice Driver
- Voice Library
- Fungsi Voice.

Voice Driver, digunakan untuk berkomunikasi dan mengontrol voice hardware

Voice Library menyajikan interface dengan Voice Driver

Fungsi Voice untuk menjalankan aplikasi-aplikasi pada Dialogic board

Model Pemrograman Single-threaded Asynchronous

- Memungkinkan sebuah program tunggal dapat mengontrol berbagai kanal suara dalam satu kendali.
- Model ini dapat digunakan untuk pengembangan aplikasi kompleks dimana beberapa tugas dapat dikoordinir secara simultan.
- Dapat mensupport manajemen *polled* maupun *callback*.

Model Pemrograman Multi-threaded Synchronous

- Menggunakan fungsi-fungsi yang mem-blok eksekusi aplikasi sampai seluruh fungsi selesai.
- Aplikasi mengontrol masing-masing kanal dari kendali yang berbeda.
- Model ini memungkinkan untuk mengatur aplikasi berbeda pada kanal yang berbeda secara dinamis dan realtime.



Voice Library yang harus dicantumkan untuk aplikasi Pemrograman Dialogic adalah :

- libdxxmt.lib → Voice Library utama
- libsrImt.lib → Standard Run-time Library

Fungsi Voice yang disediakan Dialogic

1. Fungsi Manajemen Device
2. Fungsi Konfigurasi
3. Fungsi I/O
4. Fungsi Play & Record
5. Fungsi Deteksi Global Tone
6. Fungsi Pembangkit Global Tone
7. Fungsi Pengatur Kecepatan dan Volume
8. Fungsi Analisa Panggilan Perfect Call
9. Fungsi Caller ID
10. Fungsi Manipulasi File

1. Fungsi Manajemen Device

Digunakan untuk membuka dan menutup device (board dan kanal).

Meliputi :

`dx_open()` → berfungsi membuka kanal

`dx_close()` → berfungsi menutup kanal

2. Fungsi Konfigurasi

digunakan untuk mengubah, menguji dan mengontrol konfigurasi fisik sebuah device yang terbuka

Meliputi :

- `dx_clrldigbuf()` → membersihkan buffer digit firmware
- `dx_getparm()` → mendapatkan parameter device kanal / board
- `dx_setdigtyp()` → set tipe kumpulan digit
- `dx_sethook()` → set status switch hook
- `dx_setparm()` → set device parameters
- `dx_wtring()` → wait for number of ring

3. Fungsi I/O

digunakan untuk mentransfer data ke dan dari kanal idle yang terbuka

Meliputi :

`dx_playiottdata()` → play data suara dari berbagai sumber

`dx_rec()` → merekam data suara ke satu atau lebih tujuan

`dx_reciottdata()` → merekam data suara ke berbagai tujuan

`dx_RxlottData()` → menerima data pada kanal tertentu

`dx_setdigbuf()` → set mode buffering digit

`dx_stopch()` → stop I/O yang sedang berlangsung

`dx_TxlottData()` → Transmit data pada kanal tertentu

`dx_TxRxlottData()` → mulai penerimaan inisiasi transmit data

`dx_wink()` → mematikan kanal sebentar

4. Fungsi Play dan Record

digunakan untuk memainkan atau merekam data suara, baik dari sebuah kanal atau lebih

Meliputi :

`dx_play()` → play data suara yang sudah direkam

`dx_playf()` → play data suara yang sudah direkam dari sebuah file tunggal

`dx_playiottdata()` → play data suara dari berbagai sumber

`dx_playvox()` → plays file vox tunggal

`dx_playwav()` → plays file wave tunggal

`dx_mreciottdata()` → rekam data suara dari dua kanal menjadi sebuah file, device atau memory tunggal.

Fungsi `dx_mreciottdata()` ini men-support fitur Transaction Record.

`dx_rec()` → rekam data suara ke satu atau lebih tujuan

`dx_recf()` → rekam data suara ke file tunggal

`dx_reciottdata()` → rekam data suara ke berbagai tujuan

`dx_recvox()` → rekam data suara ke file vox tunggal

`dx_recwav()` → rekam data suara ke file wave tunggal

5. Fungsi Deteksi Global Tone

digunakan untuk mendefinisikan dan mendeteksi nada frekuensi tunggal dan frekuensi dual yang dikeluarkan oleh Voice Driver.

Nada-nada ini termasuk nada DTMF dengan range 0-9, * dan #.

Meliputi :

- dx_addtone() . tambahkan tone / nada yang didefinisikan user
- dx_blddt() . buat deksripsi dual frequency tone
- dx_blddtcad() . buat deskripsi irama dual frequency tone
- dx_bldst() . buat deskripsi single frequency tone
- dx_bldstcad() . buat deskripsi irama single frequency tone
- dx_deltone() . hapus tone yang didefinisikan user

6. Fungsi Pembangkit Global Tone

digunakan untuk mendefinisikan dan memainkan nada tunggal dan dual.

Meliputi :

`dx_bldtngen()` · membuat sebuah template pembangkit tone yang didefinisikan user

`dx_playtone()` · play tone yang didefinisikan user

7. Fungsi Pengatur Kecepatan dan Volume

digunakan untuk mengatur volume dan kecepatan play

Meliputi :

`dx_adjsv()` · atur kecepatan atau volume

`dx_clrsvcond()` · bersihkan kondisi pengaturan digit
kecepatan atau volume

`dx_setsvcond()` · set kondisi pengaturan kecepatan
atau volume

`dx_getcursv()` · dapatkan setting kecepatan dan volume
saat ini

`dx_getsvmt()` · dapatkan table Modifikasi Kecepatan /Volume

`dx_setsvmt()` · set Tabel Modifikasi Kecepatan /Volume

8. Fungsi Analisa Panggilan Perfect Call

digunakan untuk memonitor sebuah panggilan setelah proses dialing ke PSTN (Public Switch Telephone Network).

PerfectCall Call Analysis → digunakan untuk memperbaiki metode identifikasi sinyal dan dapat mendeteksi mesin fax maupun mesin penjawab telepon

Basic Call Analysis → digunakan untuk analisa sebelum PerfectCall Analysis dipakai.

Call Analysis diinisialisasikan dengan fungsi `dx_dial()`
→ menggunakan input dari struktur data Call Analysis Parameter (DX-CAP).

Meliputi :

- `dx_chgdur()` → mengubah durasi sinyal Analisa Panggilan PerfectCall
- `dx_chgfreq()` → mengubah frekuensi sinyal Analisa Panggilan PerfectCall
- `dx_chgrepcnt()` → mengubah hitungan pengulangan Analisa Panggilan PerfectCall
- `dx_initcallp()` → inisialisasi Analisa Panggilan PerfectCall pada sebuah kanal

9. Fungsi Caller ID

digunakan untuk memonitor nomor pemanggil

Meliputi :

`dx_gtcallid()` → Mendapatkan Directory Number jalur pemanggil

`dx_gtextcallid()` → Mendapatkan pesan Caller ID yang diminta

`dx_wtcallid()` → Menunggu Ring dan laporan Caller ID

10. Fungsi Manipulasi File

memetakan fungsi-fungsi run-time C, dan hanya bisa digunakan jika file dibuka dengan fungsi `dx_fileopen()`.

Meliputi :

- `dx_fileclose()` · menutup file yang berhubungan dengan handle
- `dx_fileerrno()` · mendapatkan nilai error sistim
- `dx_fileopen()` · membuka file yang dispesifikasikan dengan filep
- `dx_fileread()` · membaca data dari file yang berhubungan dengan handle
- `dx_fileseek()` · memindahkan file pointer yang berhubungan dengan handle
- `dx_filewrite()` · menulis data dari buffer ke file yang berhubungan dengan handle

2. Fungsi dx_sethook()

```
#include <srllib.h>
#include <dxxplib.h>
#include <windows.h>
main()
{
    /* open a channel with chdev as descriptor */
    .
    .
    do
    /* put the channel on-hook */
    if (dx_sethook(chdev,DX_ONHOOK,EV_SYNC) == -1) {
        /* error setting hook state */
        MessageBox("Error on hook");
        exit(1);
    }
    SetDlgItemText(IDC_HOOK,"On hook success");

    /* take the channel off-hook */
    if (dx_sethook(chdev,DX_OFFHOOK,EV_SYNC) == -1) {
        /* error setting hook state */
        MessageBox("Error off hook");
        exit(1);
    }
    SetDlgItemText(IDC_HOOK,"Off hook success");
}
```

3. Fungsi dx_getdig()

```
#include <stdio.h>
#include <srllib.h>
#include <dxxlib.h>
#include <windows.h>
main()
{
    DV_TPT tpt[3];
    DV_DIGIT digp;
    int chdev, numdigs, cnt;
    /* open the channel with dx_open()
    /* Set up the DV_TPT and get the digits */
    dx_clrtppt(tpt, 3);
    tpt[0].tp_type = IO_CONT;
    tpt[0].tp_termno = DX_MAXDTMF; /* Maximum number of digits */
    tpt[0].tp_length = 4; /* terminate on 4 digits */
    tpt[0].tp_flags = TF_MAXDTMF; /* terminate if already in buf
    */
    tpt[1].tp_type = IO_CONT;
    tpt[1].tp_termno = DX_LCOFF; /* LC off termination */
```

```

tpt[1].tp_length = 3; /* Use 30 ms
                        * (10 ms resolution timer) */
tpt[1].tp_flags = TF_LCOFF|TF_10MS; /* level triggered, clear
                        * history, 10 ms resolution*/
tpt[2].tp_type = IO_EOT;
tpt[2].tp_termno = DX_MAXTIME; /* Function Time */
tpt[2].tp_length = 100; /* 10 seconds (100 ms
                        * resolution timer) */
tpt[2].tp_flags = TF_MAXTIME; /* Edge-triggered */
/* clear previously entered digits */
if (dx_clrdigbuf(chdev) == -1) {
/* process error */
}
if ((numdigs = dx_getdig(chdev,tpt, &digp, EV_SYNC)) == -1) {
/* process error */
}
for (cnt=0; cnt < numdigs; cnt++) {
printf("\nDigit received = %c, digit type = %d",
        digp.dg_value[cnt], digp.dg_type[cnt]);
}
/* go to next state */
.
.
}

```

4. Fungsi dx_playwav()

```
#include "srllib.h"
#include "dxxxlib.h"

main()
{
int chdev;          /* channel descriptor */
DV_TPT tpt;        /* termination parameter table */
:
:
:
/* Open channel */
if ((chdev = dx_open("dxxxB1C1",0)) == -1) {
    printf("Cannot open channel\n");
/* Perform system error processing */
    exit(1); } /* Set to terminate play on 1 digit */
tpt.tp_type = IO_EOT;
tpt.tp_termno = DX_MAXDTMF;
tpt.tp_length = 1;
tpt.tp_flags = TF_MAXDTMF;
/* Wait forever for phone to ring and go offhook */
if (dx_wtring(chdev,1,DX_OFFHOOK,-1) == -1) {
    printf("Error waiting for ring - %s\n",
ATDV_LASTERR(chdev));
    exit(3); }
/* Start playback */
if (dx_playwav(chdev,"HELLO.WAV",&tpt,EV_SYNC) == -1) {
    printf("Error playing file - %s\n",
ATDV_ERRMSGP(chdev));
    exit(4);
}
}
```

5. Fungsi dx_reciottdata()

```
#include "srllib.h"
#include "dxxxlib.h"
int chdev; /* channel descriptor */
int fd; /* file descriptor for file to be played */
DX_IOTT iott; /* I/O transfer table */
DV_TPT tpt; /* termination parameter table */
DX_XPB xpb; /* I/O transfer parameter block */
.
.
/* Open channel */
if ((chdev = dx_open("dxxxB1C1",0)) == -1) {
    printf("Cannot open channel\n");
    /* Perform system error processing */
    exit(1);
}
/* Set to terminate play on 1 digit */
tpt.tp_type = IO_EOT;
tpt.tp_termno = DX_MAXDTMF;
tpt.tp_length = 1;
tpt.tp_flags = TF_MAXDTMF;
/* Open file */
if ((fd = dx_fileopen("MESSAGE.VOX",O_RDWR|O_BINARY)) == -1) {
    printf("File open error\n");
    exit(2);
}
/* Set up DX_IOTT */
iott.io_fhandle = fd;
iott.io_bufp = 0;
iott.io_offset = 0;
iott.io_length = -1;
iott.io_type = IO_DEV | IO_EOT;
```

```

/*
 * Specify VOX file format for PCM at 8KHz.
 */
xpb.wFileFormat = FILE_FORMAT_VOX;
xpb.wDataFormat = DATA_FORMAT_PCM;
xpb.nSamplesPerSec = DRT_8KHZ;
xpb.wBitsPerSample = 8;
/* Wait forever for phone to ring and go offhook */
if (dx_wtring(chdev,1,DX_OFFHOOK,-1) == -1) {
    printf("Error waiting for ring - %s\n",
        ATDV_LASTERR(chdev));
    exit(3); }
/* Play intro message */
if (dx_playwav(chdev,"HELLO.WAV",&tpt,EV_SYNC) == -1)
{
    printf("Error playing file - %s\n",
        ATDV_ERRMSGP(chdev));
    exit(4); }
/* Start recording */
if
(dx_reciottdata(chdev,&iott,&tpt,&xpb,PM_TONE|EV_SYNC)
== -1) {
    printf("Error recording file - %s\n",
        ATDV_ERRMSGP(chdev));
    exit(4); }

```

6. Fungsi searching(int chdev)

```
Void CPelanggan::Searching(int chdev)
{
char x[1], y[10];
/* x[1] menyatakan buffer untuk masing-masing digit yang ditekan,
 * y[10] menyatakan panjang digit untuk setiap penekanan
 */
CString nomor;
x[0] = getDigit(chdev,y,1);
x[1] = getDigit(chdev,y,1);
x[2] = getDigit(chdev,y,1);
x[3] = getDigit(chdev,y,1);
x[4] = '\0' ;
/* penutup supaya tidak membaca digit berikutnya */
nomor = x;
/* masukkan digit yang sudah ditekan ke dalam variabel
 * string "nomor" */
if((nomor >= "1112") && (nomor <= "1114"))
{
query="select Nama from Pelanggan where No_Plg like
      '" + nomor + "'";
/* artinya : seleksi field "Nama" yang ada pada Tabel Pelanggan,
 * jika nomor yang ditekan sama dengan nomor yang ada di urutan
 * yang sama dengan Nama tersebut
 */
}
```

```
m_Adodc.SetCommandType(1);
m_Adodc.SetRecordSource(query);
m_Adodc.Refresh();
m_DataGrid.SetRefDataSource(m_Adodc.GetDSCCursor());
data=m_DataGrid.GetText();
char *data2;
data2=data.GetBuffer(255);

/* dari database, Nama tadi dimasukkan ke C++ berupa
* variable text selanjutnya dijadikan variabel Char
* Perhatikan, penamaan m_Adodc, m_DataGrid harus sesuai dengan
* yang sudah di-set di Properties ADO DC maupun ADO DataGrid
*/
}
```