

HALCON Extension Package を用いたユーザ定義関数の組み込み

本文書では、HALCON Extension Package を使用してユーザ定義関数を HALCON ライブラリに組み込む方法について説明します。

□用意するもの

- ・ユーザ定義関数（C 言語などで書かれた独自アルゴリズムなど）

※ここでは、サンプルプログラムのビーブ音を鳴らす関数を HALCON ライブラリに組み込む方法を説明します。

- ・HALCON 10以上
- ・Microsoft Visual Studio 2005以上

※ここではHALCON 11.0とMicrosoft Visual Studio 2005 を扱う例を説明します。

□試してみよう

LinX Express vol.159 のサンプルプログラムをビルドしてみましょう。

1. Visual Studio 環境変数の設定

今回の一連の作業は、コマンドプロンプト上にて行います。「スタート」メニューから「コマンドプロンプト」を起動してください。

※ Windows VistaやWindows 7をお使いの場合は、管理者モードで起動してください。

次に Visual Studio 2005の環境設定を行います（すでに環境設定を行っている場合には必要ありません）。コマンドプロンプト上で下記のバッチファイルを実行してください。

32bit 版 HALCON をお使いの場合

```
"C:¥Program Files (x86)¥Microsoft Visual Studio 8¥VC¥bin¥vcvars32.bat"
```

64bit 版 HALCON をお使いの場合

```
"C:¥Program Files (x86)¥Microsoft Visual Studio 8¥VC¥bin¥amd64¥vcvarsamd64.bat"
```

※ バッチファイルの格納先は、Visual Studio をインストールしたディレクトリにより異なります。上記はVisual Studio 2005 をデフォルト状態でインストールした時の設定です。Visual Studio 2010 の場合には、 "C:¥Program Files (x86)¥Microsoft Visual Studio 10.0¥VC¥bin¥vcvars32.bat" となります。

2. サンプルプログラムのビルド

コマンドプロンプトの作業ディレクトリを、サンプルプログラムを解凍してできたbeepディレクトリに移動してください。

例として、C:¥Program Files¥MVTec¥HALCON-11.0¥extension_package以下に解凍した場合は下記のコマンドを入力します。

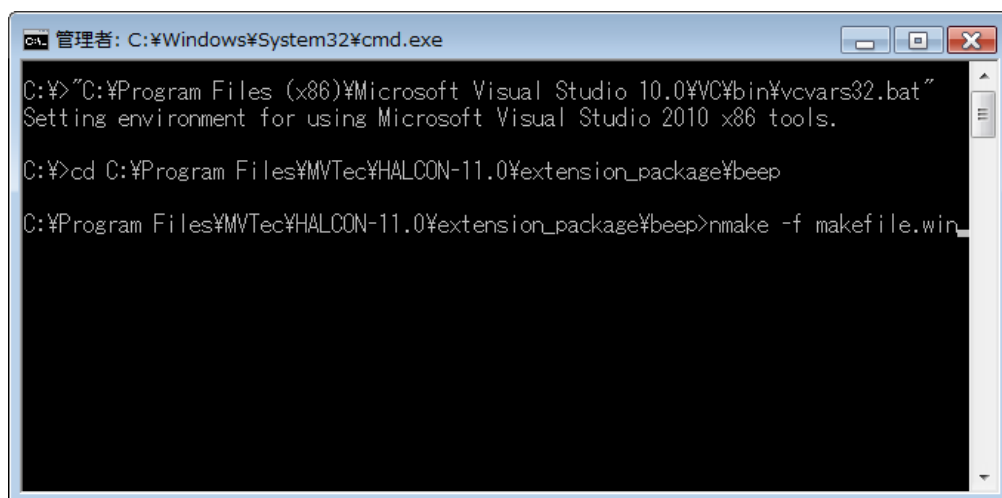
```
cd C:¥Program Files¥MVTec¥HALCON-11.0¥extension_package¥beep
```



この beep ディレクトリには、ユーザ定義関数として5つの新しい関数を追加するための定義ファイルおよびユーザ定義関数のソースプログラムが用意されています。ファイルの具体的な説明をする前に、まずはこれらユーザ定義関数をメイクしてみましょう。

コマンドプロンプト上で次のコマンドを入力し、メイクをします。

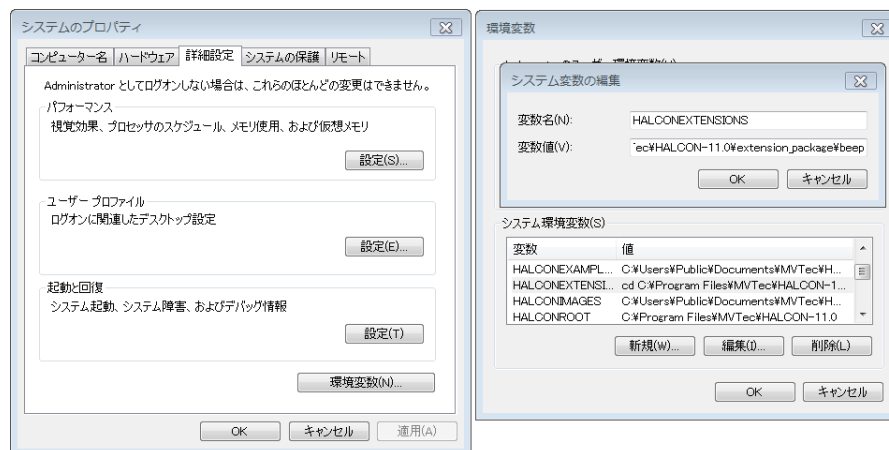
```
nmake -f makefile.win
```



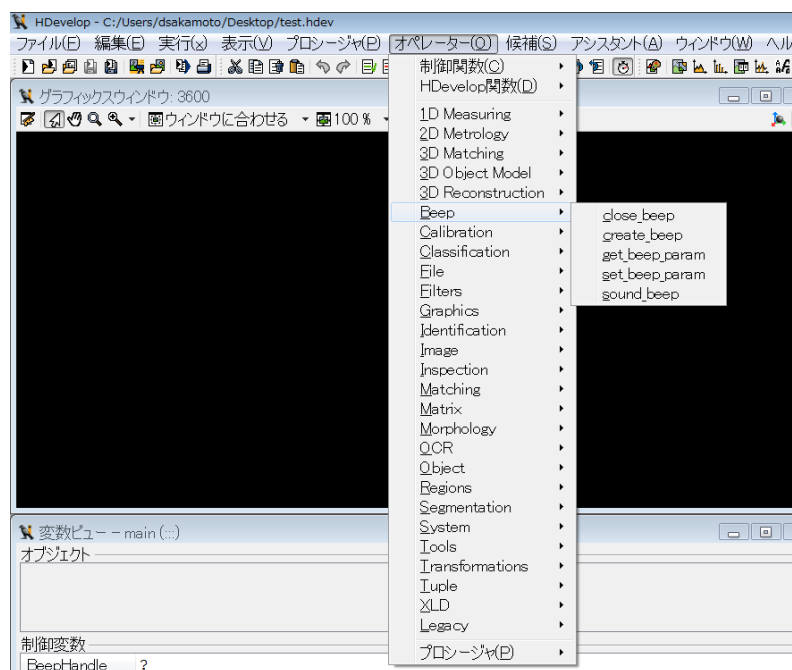
3. HDevelop の起動

Windows のシステムの環境変数の設定にて HALCONEXTENSIONS の設定を行います。これによりHALCON上にてユーザ定義関数を呼び出せるようになります。

HALCONEXTENSIONS = C:\Program Files\MVTec\HALCON-11.0\extension_package\beep



環境変数 HALCONEXTENSIONS を設定することにより、HDevelop のオペレーターメニューに新しいメニュー Beep が追加されたのが確認できます。



動作を確認するために、簡単なプログラムを組んでみましょう。
※このプログラムはビーブ音が鳴ります。ご注意ください。

```
create_beep (300, 1000, BeepHandle)

for i := 0 to 20 by 1
    sound_beep (BeepHandle)
    get_beep_param (BeepHandle, 'frequency', Value1)
    get_beep_param (BeepHandle, 'time', Value2)

    set_beep_param (BeepHandle, 'frequency', Value1+50)
    set_beep_param (BeepHandle, 'time', Value2+((i%2)*2-1)*500)
endfor

close_beep(BeepHandle)
```

ここで、以下の5つが今回用意したユーザ定義関数です。

create_beep ... ビーブ音を鳴らすためのハンドルを作成します
sound_beep ... ビーブ音を鳴らします
get_beep_param ... ビーブ音のパラメータを取得します
set_beep_param ... ビーブ音のパラメータを設定します
close_beep ... ビーブ音のハンドルをクローズします

□ユーザ定義関数の組み込み

ここでは、今回サンプルで用意したユーザ定義関数を例にいかに簡単にHALCONに組み込むことができるかについて具体的に見ていきます。

1. HALCON コンパイラ用定義ファイルの準備

HALCON にはヘルプコンパイラと呼ばれるツールが用意されています。あらかじめ定まった書式に基づいて定義ファイル(.def)を用意することで、オンラインヘルプ、テンプレートプログラムなどを自動的に生成します。下記に例題ファイルと同様のファイルを自動生成するための定義ファイルbeep.defを示します。

<beep.def >

```
1: /***** create_beep *****/
2:
```

関数名の定義。[]内は引数の定義（入力画像オブジェクト、出力画像オブジェクト、入力Tuple、出力Tuple）。この関数は、入出力画像なし、入力がビーブ音の周波数と時間、出力が作成したハンドルであることを意味しています。

```
3: create_beep <- CIPCreateBeep[:,Frequency,Time:BeepHandle]
4:
```

オペレータの説明を記述します。

```
5: short.german
6:   create beep handle;
7:
8: short.english
9:   create beep handle;
10:
```

オペレータの機能の定義を行います。ヘルプファイルの中でこの設定に基づきオペレータのカテゴリーが判断されます。

```
11: module
12:   foundation;
13:
14: chapter.german
15:   Beep;
16:
17: chapter.english
18:   Beep;
19:
```

並列化に関する定義を記述します。

```
20: parallelization
```

```

21: process_exclusively:      false;
22: process_locally:          false;
23: process_mutual:            false;
24: method:                    none;
25:

```

入出力に関する各パラメータを定義します。

```

26: parameter
27:   Frequency:                input_control;
28:   sem_type:                  integer;
29:   default_type:              integer;
30:   multivalue:                false;
31:
32: parameter
33:   Time:                      input_control;
34:   sem_type:                  integer;
35:   default_type:              integer;
36:   multivalue:                false;
37:
38: parameter
39:   BeepHandle:                output_control;
40:   sem_type:                  integer;
41:   default_type:              integer;
42:   multivalue:                false;

```

以上のような関数の入出力を定義する `def` ファイルを用意すれば、ユーザ定義関数を組み込む準備はほぼ終了したことになります。

2. C プログラム

今回例題として用意したユーザ定義関数のオリジナルソース `cipbeep.c` に含まれる `create_beep` 関数を例に説明します。重要と思われる箇所に適宜コメントを入れていますのでご参照ください。

```

1: #ifndef __APPLE__
2: #   include "Halcon.h"

```

```

3: # include "hlib/HRLClip.h"
4: #else
5: #   ifndef HC_LARGE_IMAGES
6: #       include <HALCON/Halcon.h>
7: #       include <HALCON/HRLClip.h>
8: #   else
9: #       include <HALCONxl/Halcon.h>
10: #       include <HALCONxl/HRLClip.h>
11: #   endif
12: #endif
13:
14: #include <conio.h>
15: #include <stdio.h>
16: #include <windows.h>
17: #include <string.h>
18:

```

ビープ音の各種設定を保持する構造体を定義します。このサンプルでは、構造体の配列を用意し、作成したハンドルを保存していきます。

```

19: typedef struct _beep {
20:     int no;
21:     int frequency;
22:     int time;
23: }beep;
24:
25: beep g_extension_beep[100];
26:

```

実際に HDevelop から呼び出される関数。後述するヘルプコンパイラにて自動生成されたCプログラムHbeep.c が本関数を呼び出しています。

```

27: Herror CIPCreateBeep(Hproc_handle proc_handle)
28: {
29:     static int i = 0;
30:     Hcpar *beep_handle, frequency, time;

```

31:

戻り値として返すデータのメモリを確保します。

```
32:   HCkP(HAlloc(proc_handle,sizeof(*beep_handle),&beep_handle));
```

33:

HALCONのオペレータ実行時に引数として与えられた値を獲得し、構造体に保存します。

```
34:   HGetSPar(proc_handle,1,LONG_PAR,&frequency,1);
```

```
35:   HGetSPar(proc_handle,2,LONG_PAR,&time,1);
```

36:

```
37:   g_extension_beep[i].no = i;
```

```
38:   g_extension_beep[i].frequency = (INT)frequency.par.l;
```

```
39:   g_extension_beep[i].time = (INT)time.par.l;
```

40:

戻り値を設定します。

```
41:   beep_handle[0].par.l = i++;
```

```
42:   beep_handle[0].type = LONG_PAR;
```

43:

```
44:   HPutPPar(proc_handle,1,beep_handle,1);
```

45:

HALCON では特にエラーが発生せずに処理が完了した場合、H_MSG_OK(=2)を返します。エラー番号とその内容に関しては、HALCON のインクルードディレクトリHconst.h を参照ください。

```
46:   return H_MSG_TRUE;
```

```
47: }
```

3. Makefile

今回例題として用意したMakefileの中で、関数を自作する際に修正が必要となる個所を説明します。

31: # Name of HALCON user extension package that contains the new operators:

32: PACKAGE_NAME = beep

このExtension Packageの名前を定義します。今回はbeepという名前に設定します。

64: # Name(s) of new HALCON operator description file(s):

65: DEFS = \$(DEF)\$(SLASH)beep.def

先ほど作成したdefファイルを指定します。多数のdefファイルがある場合には、それらをすべて記述します。

67: # Name(s) of new HALCON operator object file(s):

68: OBJS = \$(OBJ)\$(SLASH)cipbeep.\$(O)

69: OBJS_XL = \$(OBJ_XL)\$(SLASH)cipbeep.\$(O)

先ほど作成した c プログラム名を指定します。複数のプログラムが存在する場合には、それらをすべて記述します。

4. ビルド

作業ディレクトリを作成し、下記のようなディレクトリ構成にしてください。

/work_directory

 /def

 /beep.def 定義ファイル

 /source

 /cipbeep.c ユーザ定義関数

 makefile.win メイクファイル

 make.x86sse2-win32 メイクファイル (32bit sse2用)

 make.x64-win64 メイクファイル (64bit用)

 make.dotnet C#用dll 作成用メイクファイル

作業ディレクトリに移動し、最初の例題と同様、nmake にてメイクしてください。自動的に下記のようなディレクトリが生成され、HDevelop が必要とする各種dll、ヘルプファイルなどが作成されます。

HDevelop を起動することで、今回作成したビープ音を鳴らす関数が追加されることが確認できます。

このように、簡単に HALCON にユーザ定義関数を組み入れることが可能です。