

MICRO CADAM Helix 実践操作解説書

ACCESSプログラム開発ガイド 対話プログラム編 1

～ プログラムを新規に作成する ～

2022年8月

株式会社CAD SOLUTIONS



このチュートリアルは主にMICRO CADAM Helixの操作経験者で
C言語のプログラミング知識はあるが、
ACCESSのプログラミングは初めてという方を対象としています。

第1章 プログラム開発

1. 作成プログラムの概要
2. プログラム作成用フォルダの準備

第2章 リソースファイルの作成

1. メッセージ構造
2. 作成プログラムのメッセージ仕様
3. メニュー・メッセージ作成支援ツールによる作成

第3章 プログラムのコーディングと実行

1. 作成プログラムの全コード紹介
2. メインプログラム(初期化・終了処理)作成
3. コンパイル・リンク方法
4. メッセージの設定
5. グループ化要素の数取得とメッセージボックス表示
6. ポップアップ・メニューへの登録と実行
7. グループ化要素の取得とハイライト表示
8. ファイル選択画面の表示と要素の通常表示
9. CSVファイル出力
10. デバッグ方法

第1章 プログラム開発

1. 作成プログラムの概要

対話モードのプログラムを実際に作成してみましょう。

作成するプログラムは2022 R1からチュートリアルダウンロード用サンプル ①としてご提供している「グループ選択した点の座標をcsvに書き出す」です。

モジュール名：GETPOINT.DLL

機能概要：MC Helixであらかじめグループ化した点要素のXY座標をCSV形式のファイルに出力することができます。

グループ化された要素が存在しない場合などはメッセージ・ボックスでエラーメッセージが表示されます。

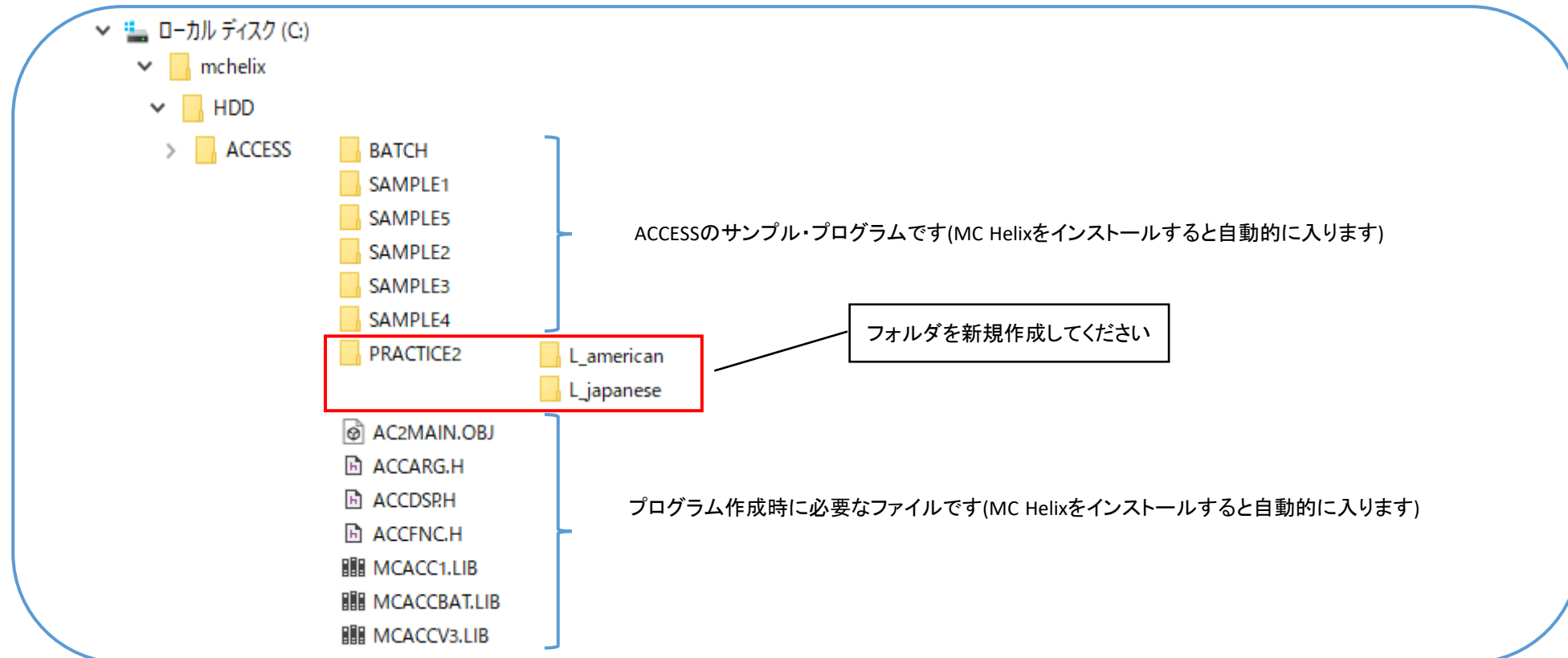
サンプルの動作をご確認いただくことで作成するプログラムの動作イメージがわくと思います。
一度ダウンロードしてお試してください。
『チュートリアル お役立ち情報 アクセス用サンプルプログラム』

2. プログラム作成用フォルダの準備

作業に必要なフォルダを作成します。

リソースファイル、ソースファイル、コンパイル・リンクに必要なバッチファイル、作成したDLLなどを格納するためのフォルダが必要です。

特に作成場所の規定はありませんが、今回はc:¥mhelix¥HDD¥ACCESSの下に¥PRACTICE2という名前で作成してください。さらに¥PRACTICE2の下に¥L_japaneseと¥L_americanというフォルダを作成してください。



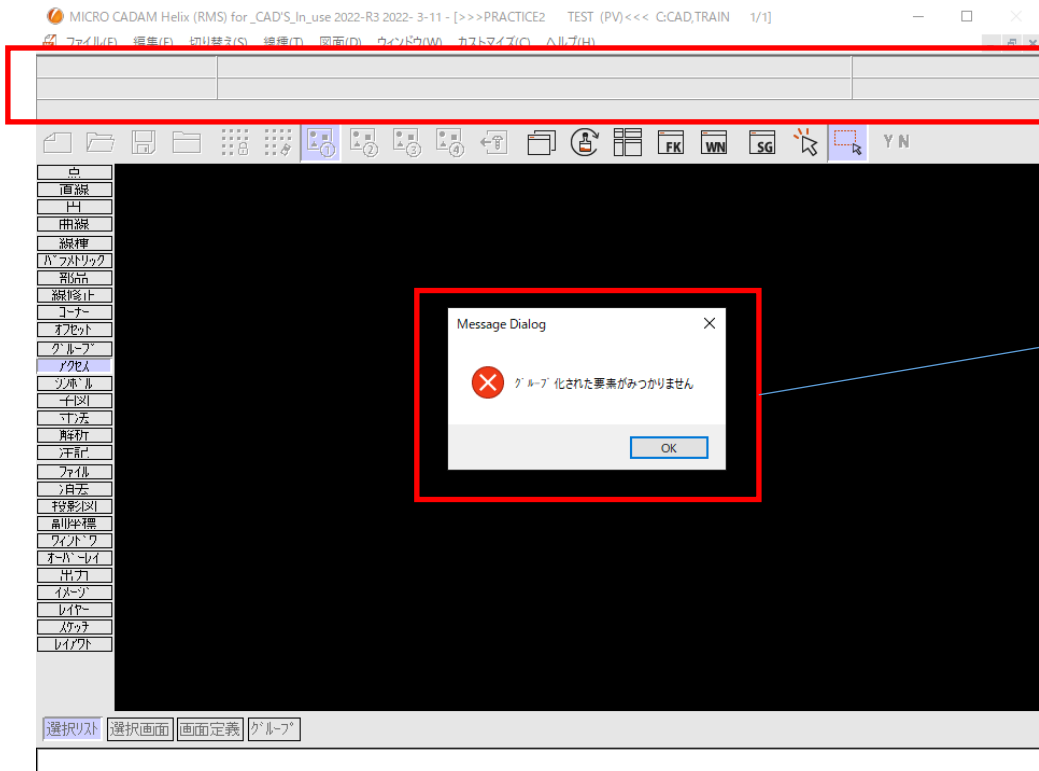
第2章 リソースファイルの作成

1. メッセージ構造(1/3)

作成プログラムではエラーが発生した場合などにメッセージ・ボックスでメッセージを表示します。

ACCESSで表示できるメッセージには次の2種類があります。

1. メッセージを表示する領域へ表示するもの
2. メッセージ・ボックスとして表示するもの（エラー・警告・情報表示用）



メッセージ表示領域

メッセージ・ボックス

いずれもメッセージの文字、表示方法、種類などをあらかじめリソースとしてメッセージ定義ファイルに定義しておく必要があります。
リソースは「メニュー・メッセージ作成支援ツール」を使用して作成します。

「メニュー・メッセージ作成支援ツール」によりリソースを定義すると次の3つのファイルが作成されます。

- ・インクルード・ファイル (XXXXXXXX.DFN) コンパイル時に使用

名称と番号の対応が書かれています。

メニュー・メッセージを取り扱う場合、ブロックという単位で扱います。効率を考慮して1つの場面(例えば、ファンクションごと)に必要なメニューやメッセージを集めてブロックにします。

ブロックごとに「#ifdef ~ #endif」文が書かれており、コンパイル時に必要なブロックだけを参照でき、効率的です。

通常、プログラムを作成する場合、番号と対になっている名称を使ってコーディングします。ACCESS関数は、このファイルを介して「メニュー番号」、「メッセージ番号」などを名称で参照できます。

```
#define BLOCK_PROG1 1
#define BLOCK_PROG2 2

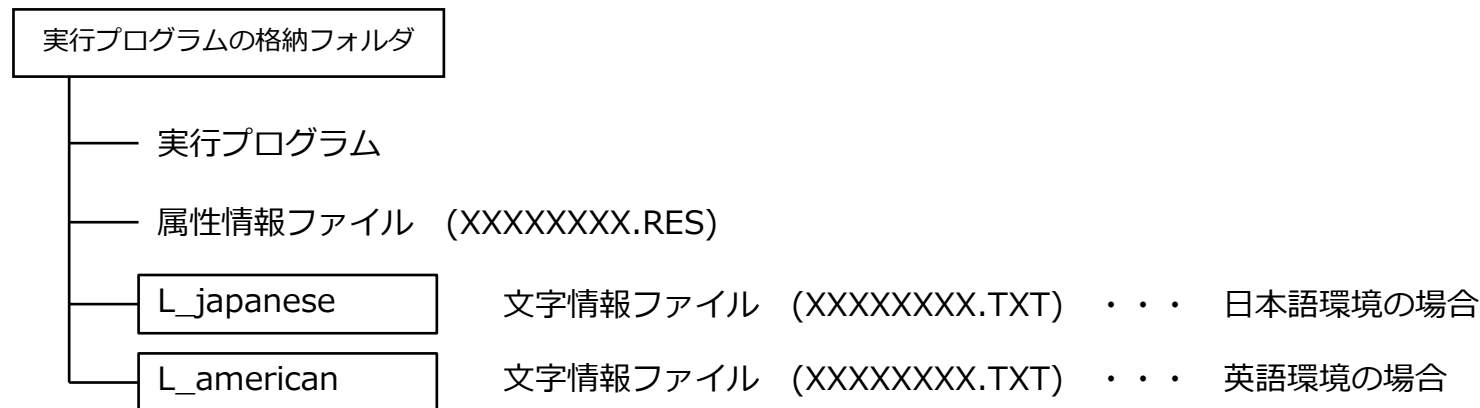
#ifdef INCL_BLOCK_PROG1
#define APL_MESG_1 1
#define APL_MESG_2 2
#define APL_MESG_3 3
  .
#endif
#ifdef INCL_BLOCK_PROG2
  .
#endif
```

ブロック名

ブロック識別子

メッセージ名

- 属性情報ファイル (XXXXXXXXX.RES) プログラム実行時に使用
メニュー、メッセージなどの形状、機能、動作などの情報が書かれています。
プログラムと同じフォルダに格納します。
- 文字情報ファイル (XXXXXXXXX.TXT) プログラム実行時に使用
実際に表示するメニュー・メッセージなどの文字列の情報が書かれています。
ここで定義する文字列には、日本語環境、英語環境の2種類があります。
プログラムと同じフォルダの言語ごとのフォルダの下に格納します。



2. 作成プログラムのメッセージ仕様

作成プログラムのメッセージは次の通りです。

| メッセージ(日本語) | メッセージ(英語) | 表示方法 | 種類 |
|----------------------|---------------------|------------|-----|
| 「グループ化された要素が見つかりません」 | "Not Found Group" | メッセージ・ボックス | エラー |
| 「メモリが足りません」 | "Out of Memory" | メッセージ・ボックス | 警告 |
| 「ファイルを出力しました」 | "File Output" | メッセージ・ボックス | 警告 |
| 「処理を終了します」 | "End the Process" | メッセージ・ボックス | 警告 |
| 「ファイルの出力に失敗しました」 | "File Output Error" | メッセージ・ボックス | 警告 |

これらのメッセージを1つのブロックとして定義します。(今回は以下の名称で定義します)

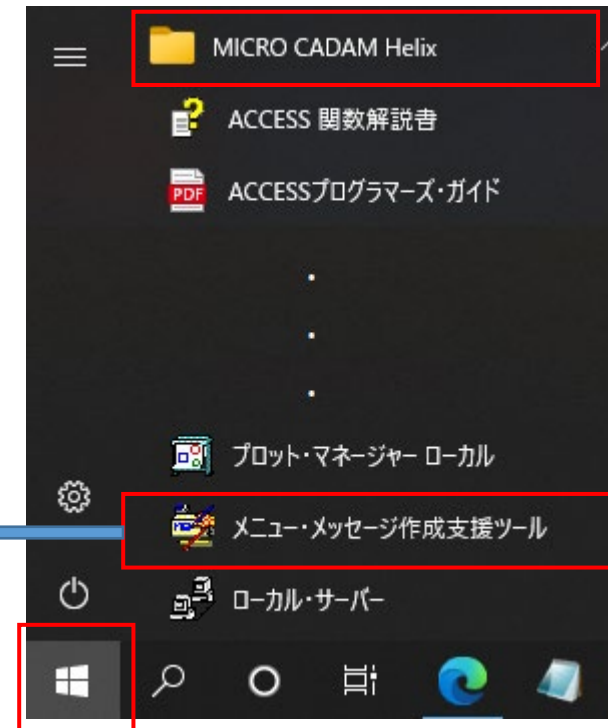
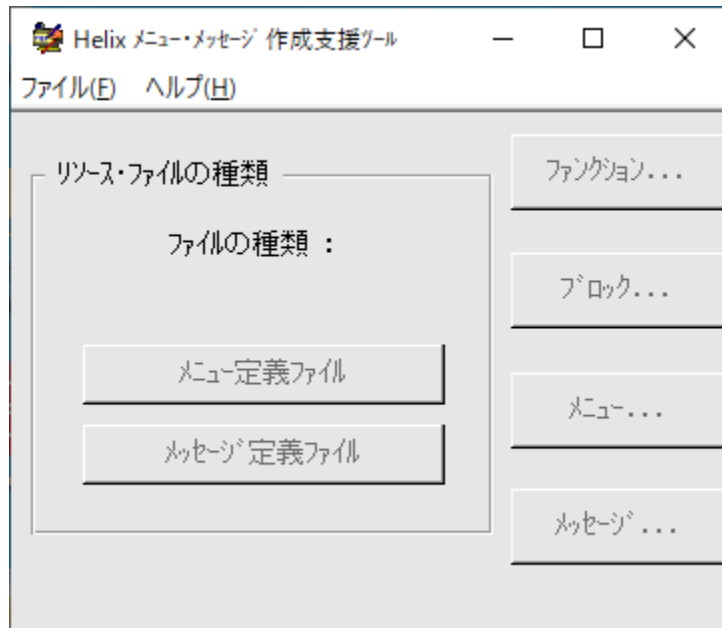
ブロック名 : BLK_MSG_SP8
 ブロック識別子 : INCL_BLK_MSG_SP8
 メッセージ名 : MSG_NOT_FOUND_GROUP
 MSG_OUT_OF_MEMORY
 MSG_FILE_OUTPUT
 MSG_END_PROCESS
 MSG_FILE_OUTPUT_ERROR

3. メニュー・メッセージ作成支援ツールによる作成(1/5)

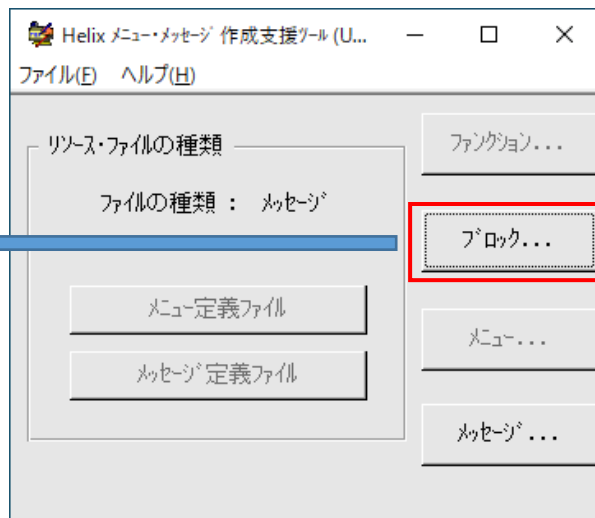
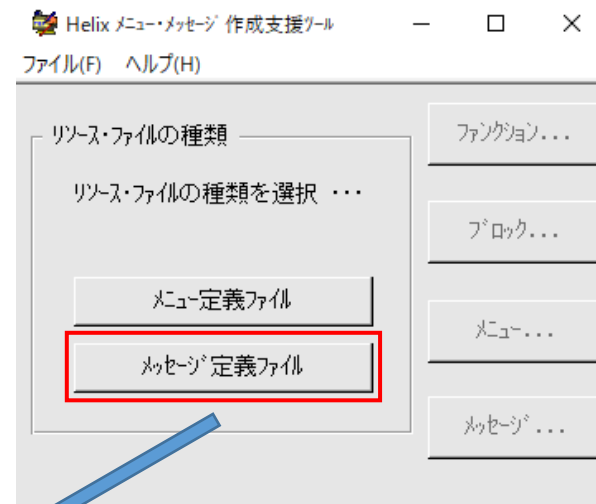
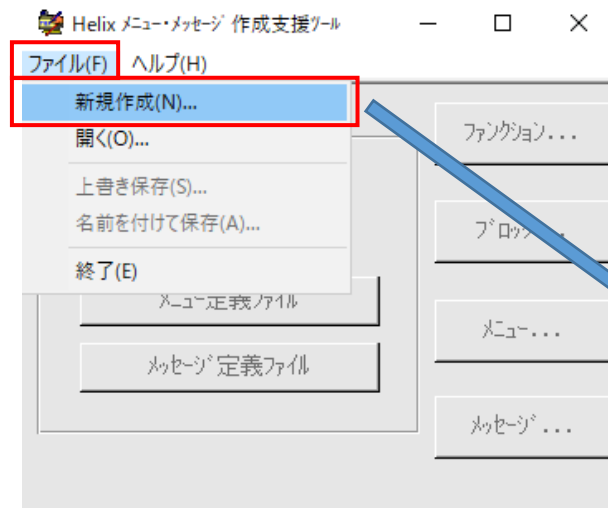
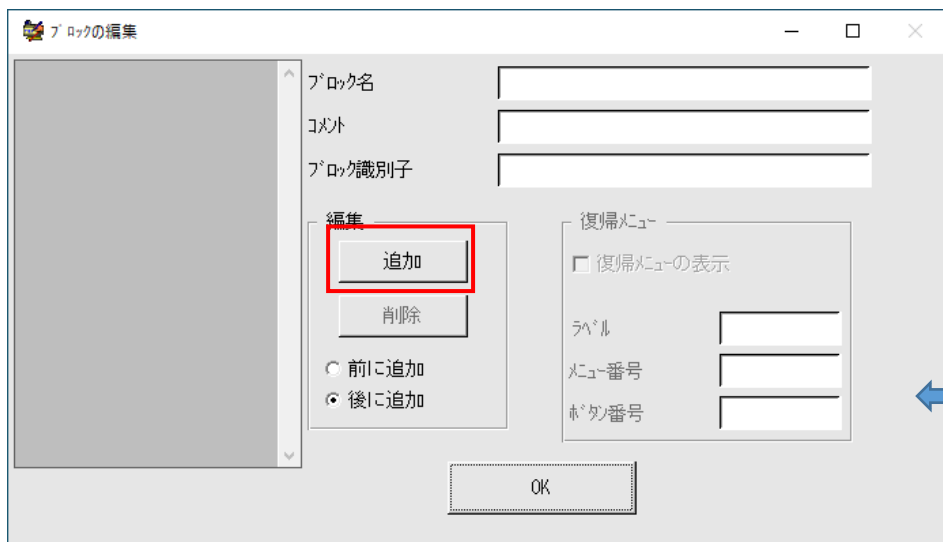
「メニュー・メッセージ作成支援ツール」を使用して、メッセージ定義ファイルを作成します。

1. 「メニュー・メッセージ作成支援ツール」を起動します。

Windowsの[スタート]ボタンから[MICRO CADAM Helix]を選択、
[メニュー・メッセージ作成支援ツール]をクリック



2. メニュー・バー[ファイル]から[新規作成]を選択します。
3. [メッセージ定義ファイル]を押します。
4. [ブロック]を押します。
「ブロックの編集」画面が表示されます。
5. [追加]を押します。



メニュー・メッセージを取り扱う場合の単位
(1つの場面に必要なメニューやメッセージを
集めてブロックとします)

6. ブロックを定義します。

ブロックには次の項目があります。

- ・ブロック番号(「ブロック名」の項目で定義)
- ・コメント
- ・ブロック識別子

| | |
|---------|--------------------|
| ブロック名 | : BLK_MSG_SP8 |
| コメント | : <SP8> |
| ブロック識別子 | : INCL_BLK_MSG_SP8 |

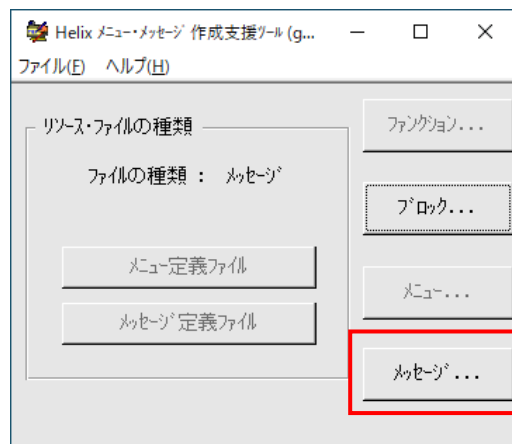
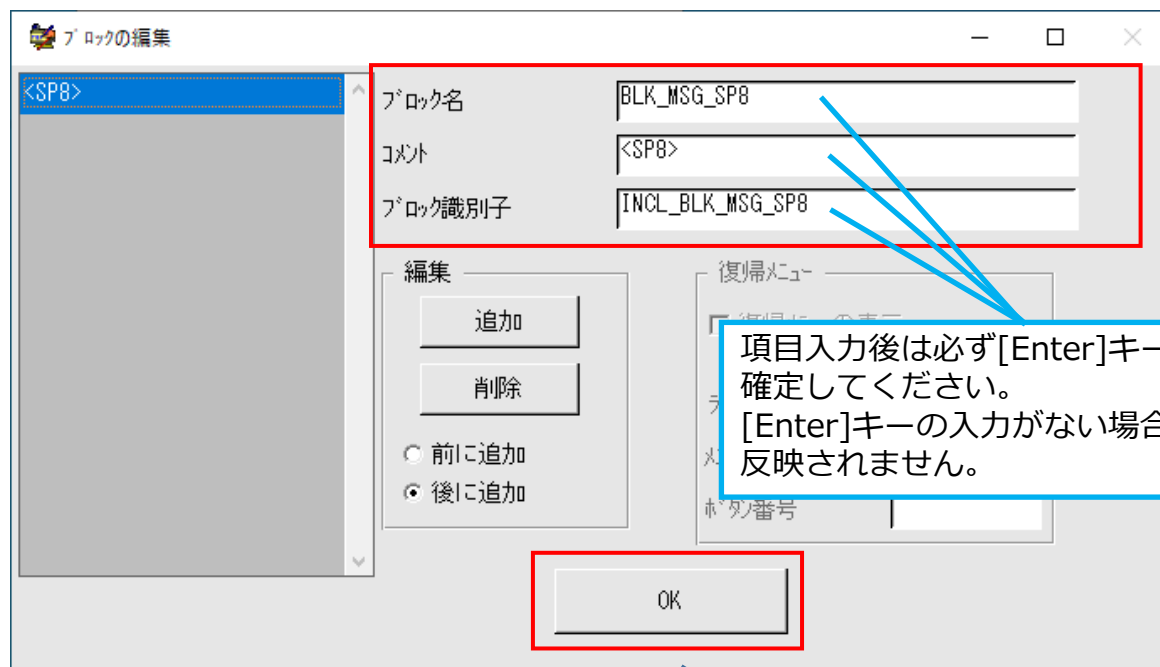
各項目入力後は確定のため[Enter]

ACCESS関数は、「ブロック番号」を使って制御します。
また、「ブロック識別子」は、コンパイルのとき使用します。

7. [OK]を押します。

8. [メッセージ]を押します。

「メッセージの編集」画面が表示されます。



9. 6. で定義したブロック<SP8>を選択します。

10. メッセージを定義します。

①[追加]を押す

②メッセージの表示方法を指定

[表示領域]の[メッセージ・ボックス]選択

[種類]の[エラー]選択

③メッセージ名、メッセージ番号を入力

メッセージ名 : MSG_NOT_FOUND_GROUP
メッセージ番号 : 1 (表示値のまま)

各項目入力後は確定のため[Enter]

④メッセージ(「ラベル」項目)を入力

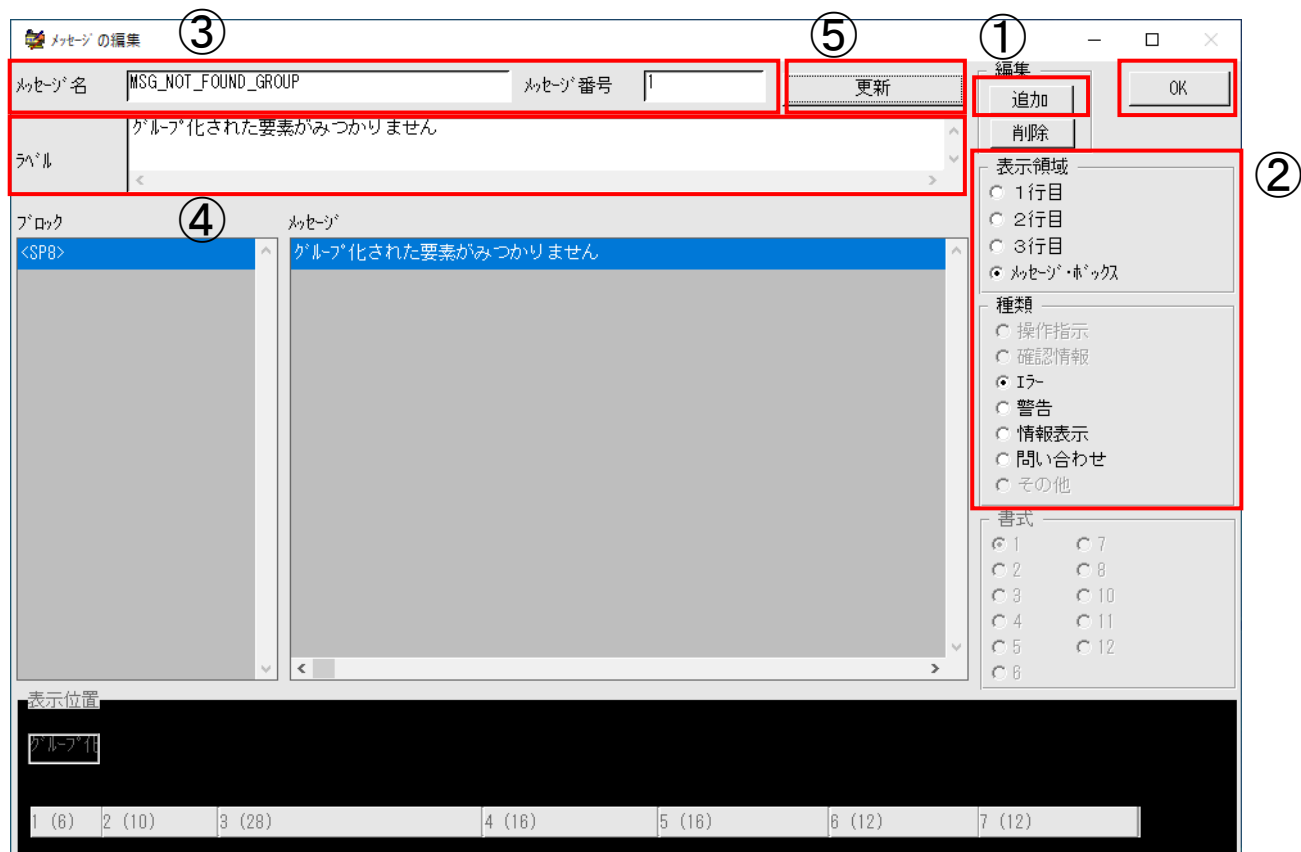
まず、日本語のメッセージを指定します。

メッセージ : グループ化された要素が見つかりません

⑤[更新]

11. 残りのメッセージ(P.11を参照)を定義するため10. を繰り返します。(残りは「警告」メッセージのため、[種類]で「警告」を選択します。)

12. [OK]を押します。メッセージの定義を終了し、起動直後の状態に戻ります。



- 1 3. メニュー・バー[ファイル]から[名前を付けて保存]を選択します。
- 1 4. 「Save As」画面が表示されるので保存場所とファイル名を指定して[保存]を押します。

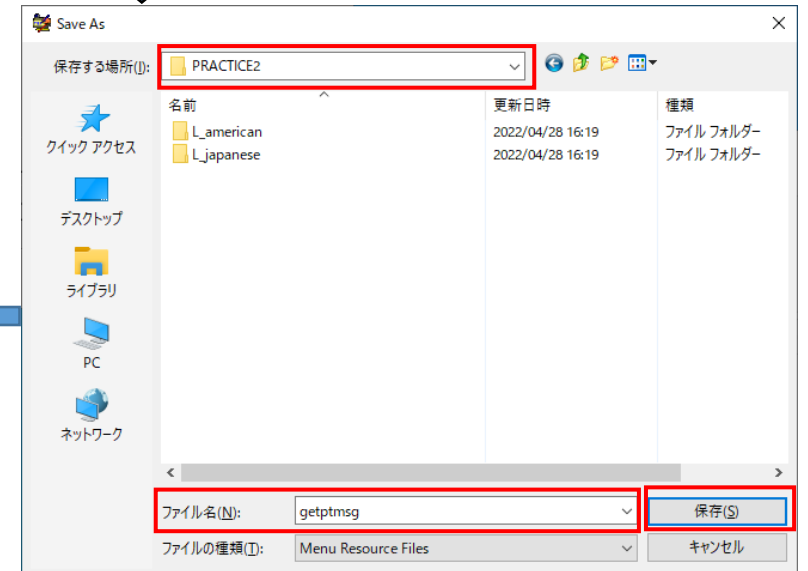
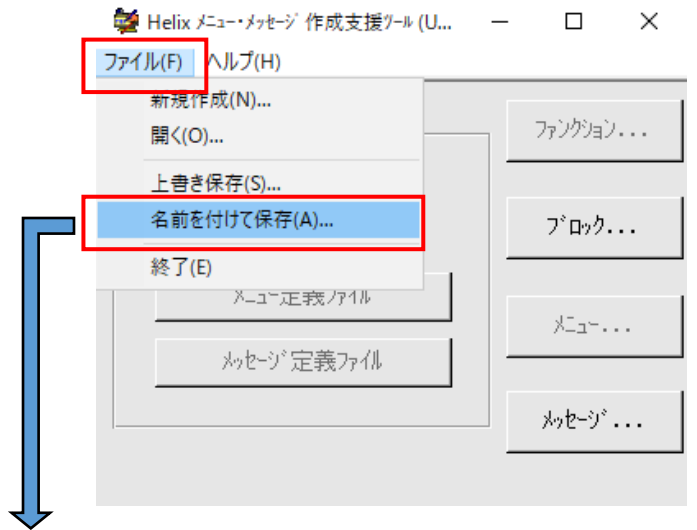
保存場所 : c:\mchelix\HDD\ACCESS\PRACTICE2
ファイル名 : getptmsg

- 1 5. 「言語の選択(別名保存)」画面が表示されるので[japanese]を選択して[OK]を押します。
- 1 6. c:\mchelix\HDD\ACCESS\PRACTICE2の下に
getptmsg.resとgetptmsg.dfnが¥L_japaneseの下に
getptmsg.txtが作成されていることを
確認してください。

- 1 7. 英語環境の文字情報ファイルを作成します。

- ① ¥L_japaneseのgetptmsg.txtを
¥L_americanにコピーします。
- ② コピーしたgetptmsg.txtをエディタで開き、

P.11を参照して日本語メッセージに対応する英語メッセージに置き換え、ファイルします。



第3章 プログラムのコーディングと実行

1. 作成プログラムの全コード紹介

リソースファイルの準備はできましたので、次はコードの記述です。

作成プログラムの全コードは[こちら](#)です。

処理の流れは次の通りです。

ソースファイル内での対応箇所

- | | |
|------------------|----|
| 1. 初期化処理 | 1 |
| 2. メッセージの設定 | 2 |
| 3. グループ化要素数の取得 | 3 |
| 4. グループ化要素の取得 | 4 |
| 5. 点要素の検索 | 5 |
| 6. 点要素のハイライト処理 | 6 |
| 7. CSVファイルの指定 | 7 |
| 8. 点要素の通常表示 | 8 |
| 9. CSVファイルへの書き出し | 9 |
| 10. 終了処理 | 10 |

この後、コードを少しずつ組み込みながらコンパイル・リンクして、その都度動作を確認しながら進めます。

2. メインプログラム(初期化・終了処理)作成(1/3) **1** **10**

本頁より、各コードについて解説します。

1. プログラム作成のために準備したフォルダ(c:\mchelix\HDD\ACCESS\PRACTICE2)に"GETPOINT.C"という名前でテキスト・ファイルを作成してください。
2. エディタで"GETPOINT.C"を開いてください。ソースコードを記述していきます。
3. MC Helixとの連携を図るためメインに相当する関数の名前は"ac2userp"としてください。
4. ACCESSの関数群を使用するためには初期化処理が必要です。まず最初に"MC_bubegn()"関数を呼び出してください。オプションは対話モードなので"1"を指定します。バージョンはMC Helixのバージョンとして"52"を指定します。入力変数のチェックは"0"を指定します。精度は2Dモジュールのモードを引き継ぐため"0"を指定します。

関数群の引数の意味や仕様については『ACCESS 関数解説書』を参照しながらコーディングしてください。

```
void    ac2userp(void)
{
    return;
}
```

```
iflgar[0] = 52L;    ...バージョン
iflgar[1] = 0L;    ...入力変数のチェック
iflgar[2] = 0L;    ...精度
retc = MC_bubegn ( 1L, iflgar, ioutar, 10L );
if ( retc ) goto exit;
```

4.オプション

4.エラー発生箇所識別番号

エラー発生箇所識別番号は、「bu」で始まる名前の関数の多くにあります。同じ関数を何回も呼び出している場合などにどの関数呼び出しでエラーが発生したか特定するために利用できます。

今回は"10"を指定します。(値は任意の整数値です)

戻り値が0以外の場合、初期化に失敗したということですので処理を終了します。

5. プログラムを終了する時には最後に必ず終了処理を行う必要があります。

"MC_buend()"関数を呼び出してください。

オプションは"1"を指定してください。

6. ACCESSの関数を使用していますので、その関数宣言が必要です。

"accfnc.h"というインクルード・ファイルが提供されていますので、必ずインクルードしてください。

```
#include "accfnc.h" — 6.関数宣言

void ac2userp(void)
{
    .
    retc = MC_bubegn ( 1L, iflgar, ioutar, 10L );
    if ( retc ) goto exit;
    .
    .
    .
exit:
    MC_buend ( 1L, 10L );
    return; — 5.オプション
}
```

7. 初期化・終了処理はコーディングできましたが、もう1つプログラム実行中にエラーが発生した場合に役立つ関数を呼び出します。“MC_bumode()”です。

発生したエラー情報や各関数の入出力パラメータ情報を
トレース情報として取得できるようになります。

オプションはモードを設定するので“1”を指定します。

トレース・モードは「MCADAM5.SYS」のキーワードの指定に従うよう“2”を指定します。

未使用パラメータには“0”を指定しておきます。

精度は倍精度“12”を指定します。

```

retc = MC_bubegn ( 1L, iflgar, ioutar, 10L );

modear [0] = 2L;   ...トレース・モード
modear [1] = 0L;   ...未使用
modear [2] = 12L;  ...精度
retc = MC_bumode ( 1L, modear, 10L );
if ( retc ) goto exit;
}

exit:
MC_buend ( 1L, 10L );

return;
}
    
```

7.オプション

トレース・モードが“2”の時は環境設定ユーティリティーで「MCADAM5.SYS」の「その他」シートのキーワード「ACCMSLVL」に設定したトレースのレベルに従います。
(このキーワードは初期状態では設定されていないので環境設定ユーティリティーで追加指定してご使用ください。)

- 0 : メッセージ・ファイルは出力しない
- 1 : エラーが発生した場合にエラー情報だけを出力する
- 2 : 引数の値とエラーが発生したときはエラー情報も出力する

トレース情報は、c:¥MCADAMの下のメッセージ・ファイル「ACCMSnnn.XXX」に出力されます。(nnnは000~999)
ただし、トレース・モードをオンにすると、メッセージ・ファイルはかなりの量が出力され、実行速度低下の原因となります。
デバッグのときだけオンにして、デバッグ終了後はオフにして使用することをお勧めします。

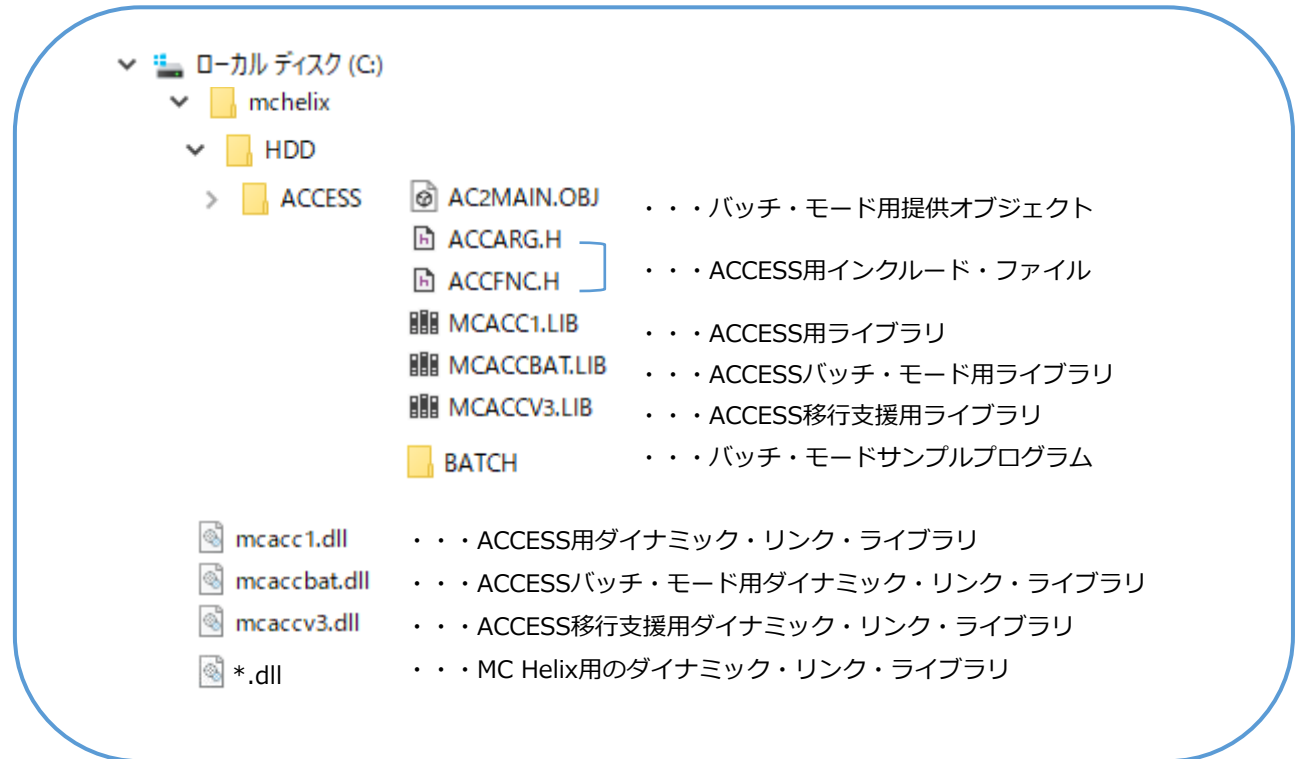
8. ソースファイル(GETPOINT.C)を保存します。

3. コンパイル・リンク方法(1/7)

メインプログラムがコーディングできたところで一度コンパイル・リンクしてみましょう。

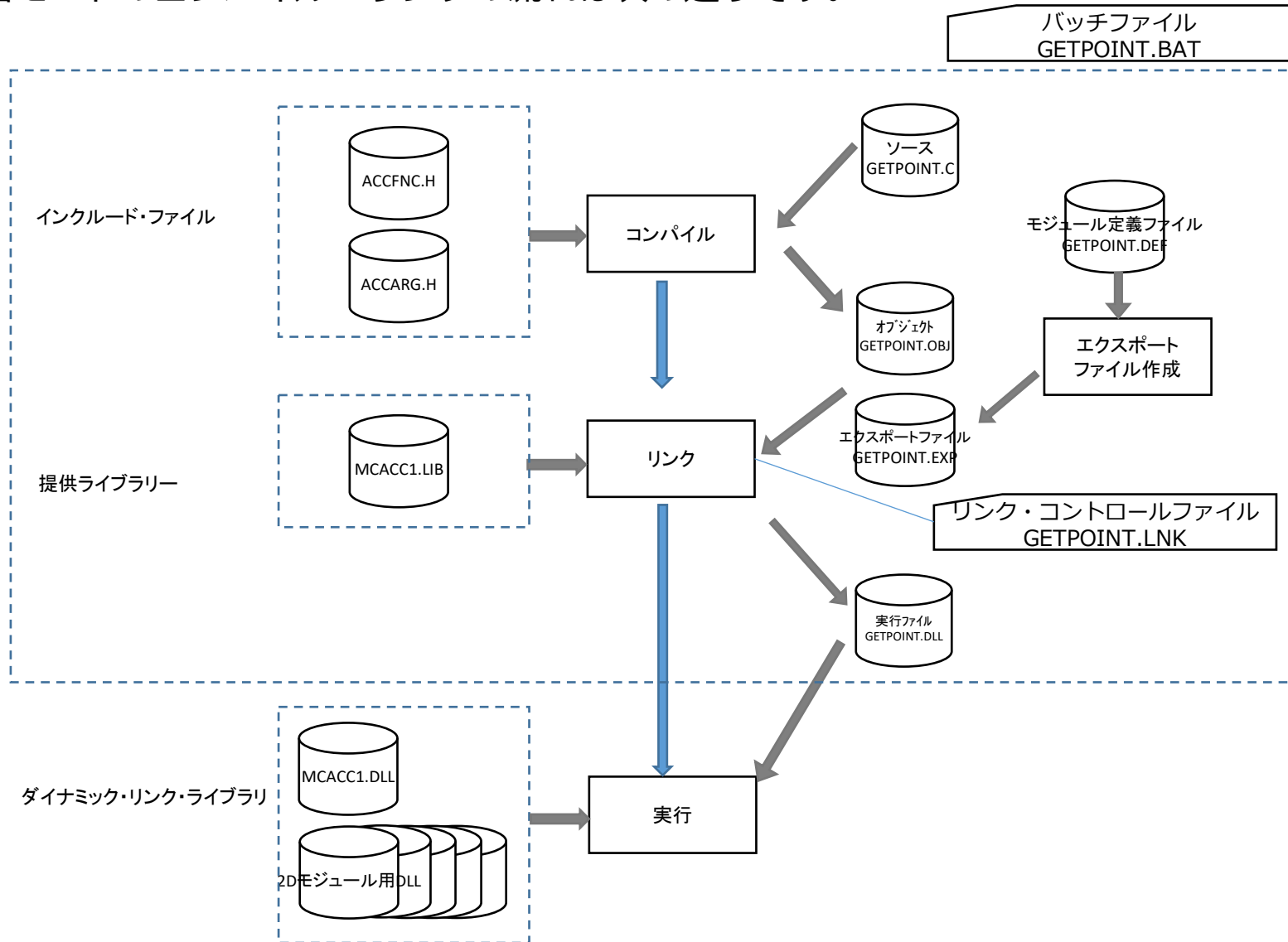
ACCESSプログラムのリンク・コンパイルに必要なファイルはc:\\$mhelix\\$HDDの下に導入されています。

これらのうち必要なファイルを使用してコンパイル・リンクします。



| | | |
|----------------|--------------|------------------------------------|
| ローカル ディスク (C:) | | |
| mhelix | | |
| HDD | | |
| ACCESS | | |
| | AC2MAIN.OBJ | ・・・ バッチ・モード用提供オブジェクト |
| | ACCARG.H | ・・・ ACCESS用インクルード・ファイル |
| | ACCFNC.H | |
| | MCACC1.LIB | ・・・ ACCESS用ライブラリ |
| | MCACCBAT.LIB | ・・・ ACCESSバッチ・モード用ライブラリ |
| | MCACCV3.LIB | ・・・ ACCESS移行支援用ライブラリ |
| | BATCH | ・・・ バッチ・モードサンプルプログラム |
| | mcacc1.dll | ・・・ ACCESS用ダイナミック・リンク・ライブラリ |
| | mcaccbat.dll | ・・・ ACCESSバッチ・モード用ダイナミック・リンク・ライブラリ |
| | mcaccv3.dll | ・・・ ACCESS移行支援用ダイナミック・リンク・ライブラリ |
| | *.dll | ・・・ MC Helix用のダイナミック・リンク・ライブラリ |

対話モードのコンパイル・リンクの流れは次の通りです。



コンパイル・リンクはコマンドプロンプトからバッチファイルを実行することで行います。

まず、バッチファイル、リンク・コントロールファイルとモジュール定義ファイルを準備します。

これ以降、64bit版モジュールを作成する方法を解説します。

1. サンプルとして提供されている対話モードプログラム作成用のバッチファイル、リンク・コントロールファイルとモジュール定義ファイルをソースファイルと同じフォルダにコピーします。

c:¥mchelix¥HDD¥ACCESS¥SAMPLE1の下の

MAKEPLYC.BAT ————

MAKEPLYC.LNK ————

MAKEPLYC.DEF ————

をc:¥mchelix¥HDD¥ACCESS¥PRACTICE2にコピーします。

ファイル名をプログラム名に合わせてGETPOINT.BAT、GETPOINT.LNK、GETPOINT.DEFにリネームします。

チュートリアルダウンロード用サンプル ①
「グループ選択した点の座標をCSVに書き出す」を
ダウンロード・展開されている場合は
そのフォルダから該当するファイルをコピーして
いただくとリネーム、プログラム名の変更は必要
ありません。

拡張子「.lnk」のファイルはショートカットファイルとみなされ、エクスプローラーでは拡張子が表示されません。
編集のためファイルを開く場合、ファイルをエディタにドラッグ&ドロップしてください。
リネームする場合、拡張子以外の部分(表示されている部分)のみを変更してください。

2. バッチファイル内のプログラム名(MAKEPLYC)を作成プログラム名に合わせて"GETPOINT"に変更します。

Visual Studio のビルド環境を準備するバッチファイルの呼び出しを追加します。

"strcpy"関数など低いセキュリティレベルの古い関数に対する警告を表示しないため

"/D_CRT_SECURE_NO_WARNINGS"を付加します。(任意です)

GETPOINT.BAT

```
@ECHO OFF
CALL "C:¥Program Files (x86)¥Microsoft Visual Studio 14.0¥VC¥vcvarsall.bat" amd64
SET ORG_INCLUDE=%INCLUDE%
SET ORG_LIB=%LIB%
SET ACCESS_PATH=C:¥MCHelix¥HDD¥ACCESS
SET LIB=%ACCESS_PATH%;%LIB%;
SET INCLUDE=%ACCESS_PATH%;%INCLUDE%;
cl /c /W3 /J /MD /DWIN32 /D_NTSDK /D_CRT_SECURE_NO_WARNINGS GETPOINT.C
LIB /MACHINE:X64 /OUT:GETPOINT.LIB /DEF:GETPOINT.DEF > NUL
LINK @GETPOINT.LNK
MT -MANIFEST GETPOINT.DLL.MANIFEST -OUTPUTRESOURCE:GETPOINT.DLL;2
SET INCLUDE=%ORG_INCLUDE%
SET LIB=%ORG_LIB%
SET ORG_INCLUDE=
SET ORG_LIB=
SET ACCESS_PATH=
```

Visual Studio のビルド環境を準備するバッチファイル呼び出しバージョンやインストール時の指定により場所は異なります。64ビットモジュールを作成するためのオプションを指定して呼び出します。

ACCESS用ファイルの導入されている場所を設定

ライブラリとインクルード・ファイルの参照場所にACCESS用ファイルの場所を追加

コンパイル

エクスポートファイル作成

コントロールファイルを使用してのリンク

マニフェスト埋め込み

3. リンク・コントロールファイル内のプログラム名(MAKEPLYC)を作成プログラムに合わせて"GETPOINT"に変更します。

GETPOINT.LNK

```
/DLL
/manifest
/OUT:GETPOINT.DLL    ...実行モジュール名
/MAP:GETPOINT.MAP   ...リスト・ファイル名
GETPOINT.OBJ        ...オブジェクト・ファイル
GETPOINT.EXP        ...エクスポート・ファイル名
MCACC1.LIB          ...ライブラリ名
MSVCRT.LIB OLDNAMES.LIB KERNEL32.LIB GDI32.LIB USER32.LIB
COMDLG32.LIB WSOCK32.LIB
```

4. モジュール定義ファイル内のプログラム名(MAKEPLYC)を作成プログラムに合わせて"GETPOINT"に変更します。

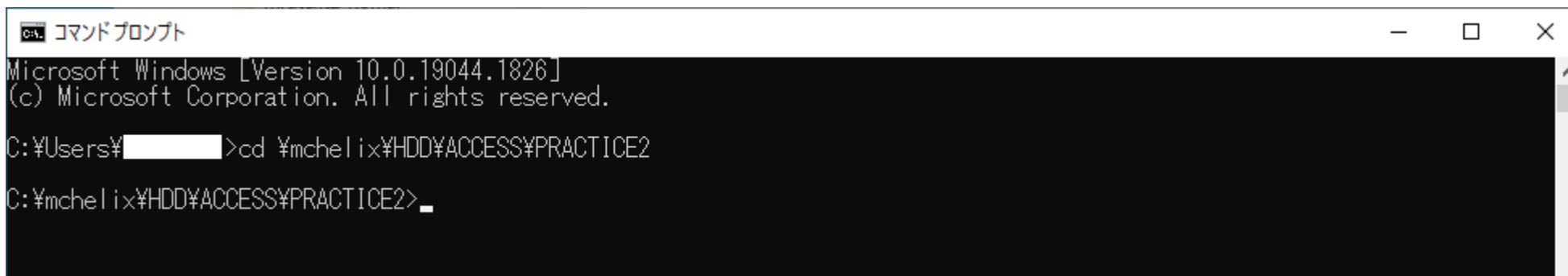
GETPOINT.DEF

```
LIBRARY GETPOINT
EXPORTS
    ac2userp
```

5. コマンド・プロンプトを起動します。

6. ソースファイルがあるフォルダに移動します。

コマンドプロンプトで"cd %mchelix%HDD%ACCESS%PRACTICE2"と入力して[Enter]



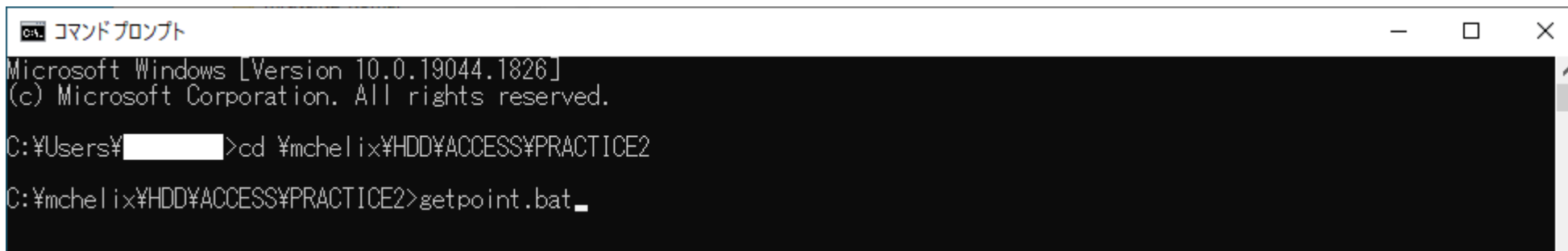
```
コマンドプロンプト
Microsoft Windows [Version 10.0.19044.1826]
(c) Microsoft Corporation. All rights reserved.

C:\Users\%>cd %mchelix%HDD%ACCESS%PRACTICE2

C:\mchelix%HDD%ACCESS%PRACTICE2>_
```

7. バッチファイルを実行します。

コマンドプロンプトで"getpoint.bat"と入力して[Enter]



```
コマンドプロンプト
Microsoft Windows [Version 10.0.19044.1826]
(c) Microsoft Corporation. All rights reserved.

C:\Users\%>cd %mchelix%HDD%ACCESS%PRACTICE2

C:\mchelix%HDD%ACCESS%PRACTICE2>getpoint.bat_
```

8. コンパイル・リンクが実行されます。

エラーが発生した場合はコーディングを修正し、再度コンパイル・リンクを行います。

ソースファイルと同じフォルダに"GETPOINT.DLL"が作成されていることを確認してください。

(ここでは正常にコンパイル・リンクできることを確認するだけでモジュールの実行は行いません。)

The screenshot shows a Windows Command Prompt window on the left and a File Explorer window on the right. The Command Prompt displays the execution of the 'getpoint.bat' script, which runs the Microsoft C/C++ Optimizing Compiler and the Incremental Linker. The linker output shows the creation of 'GETPOINT.DLL' and other files like 'GETPOINT.DLL.manifest', 'GETPOINT.exp', 'GETPOINT.ilc', 'GETPOINT.lib', 'GETPOINT.map', 'GETPOINT.pdb', 'getpoint.obj', 'vc140.pdb', 'getpoint', 'getpoint.bat', 'getpoint.c', 'getpoint.def', 'getptmsg.dfn', and 'getptmsg.res'. The File Explorer window shows the 'PRACTICE2' folder containing the compiled files, with 'GETPOINT.DLL' highlighted in a red box.

```
Microsoft Windows [Version 10.0.19044.1826]
(c) Microsoft Corporation. All rights reserved.

C:\Users\%username%>cd %mchelix%\HDD\ACCESS\PRACTICE2

C:\mchelix%\HDD\ACCESS\PRACTICE2>getpoint.bat
Microsoft(R) C/C++ Optimizing Compiler Version 19.00.24215.1 for x64
Copyright (C) Microsoft Corporation. All rights reserved.

GETPOINT.C
Microsoft (R) Incremental Linker Version 14.00.24215.1
Copyright (C) Microsoft Corporation. All rights reserved.

/DLL
/manifest
/OUT:GETPOINT.DLL
/MAP:GETPOINT.MAP
GETPOINT.OBJ
GETPOINT.EXP
MCACC1.LIB
MSVCRT.LIB OLDNAMES.LIB KERNEL32.LIB GDI32.LIB USER32.LIB
COMDLG32.LIB WSOCK32.LIB
Microsoft (R) Manifest Tool version 10.0.10011.16384
Copyright (c) Microsoft Corporation 2012.
All rights reserved.

C:\mchelix%\HDD\ACCESS\PRACTICE2>
```

| 名前 | 更新日時 | 種類 | サイズ |
|-----------------------|------------------|-----------------------|--------|
| GETPOINT.DLL | 2022/05/12 9:23 | アプリケーション拡張 | 37 KB |
| GETPOINT.DLL.manifest | 2022/05/12 9:23 | MANIFEST ファイル | 1 KB |
| GETPOINT.exp | 2022/05/12 9:23 | Exports Library File | 1 KB |
| GETPOINT.ilc | 2022/05/12 9:23 | Incremental Linker... | 319 KB |
| GETPOINT.LIB | 2022/05/12 9:23 | Object File Library | 2 KB |
| GETPOINT.MAP | 2022/05/12 9:23 | Linker Address Map | 26 KB |
| GETPOINT.pdb | 2022/05/12 9:23 | Program Debug D... | 396 KB |
| getpoint.obj | 2022/05/12 9:23 | 3D Object | 23 KB |
| vc140.pdb | 2022/05/12 9:23 | Program Debug D... | 84 KB |
| getpoint | 2022/05/12 9:23 | ショートカット | 1 KB |
| getpoint.bat | 2022/05/12 9:22 | Windows バッチファ... | 1 KB |
| getpoint.c | 2022/05/11 16:41 | C Source | 6 KB |
| getpoint.def | 2021/12/23 0:42 | Export Definition ... | 1 KB |
| getptmsg.dfn | 2021/12/23 0:42 | DFN ファイル | 1 KB |
| getptmsg.res | 2021/12/23 0:42 | Compiled Resourc... | 1 KB |
| L_american | 2022/05/11 11:01 | ファイル フォルダー | |
| L_japanese | 2022/05/10 15:29 | ファイル フォルダー | |

4. メッセージの設定 2 10

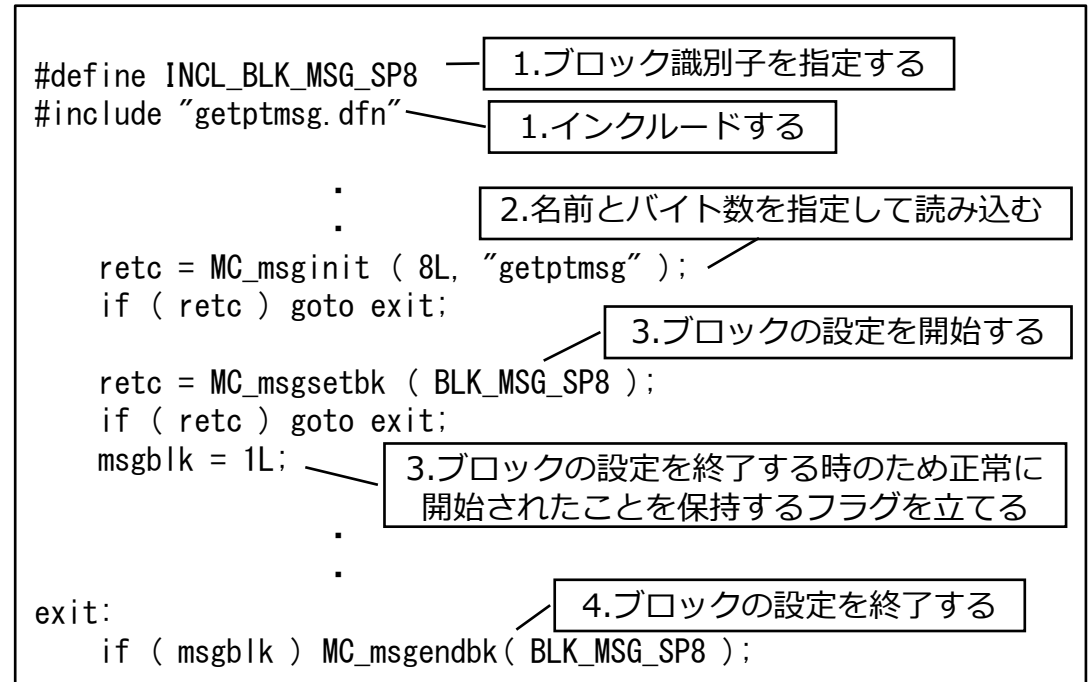
「メニュー・メッセージ作成支援ツール」により作成したメッセージ定義ファイルを利用してメッセージを表示する準備を行います。

インクルード・ファイル： getptmsg.dfn

ブロック識別子： INCL_BLK_MSG_SP8

ブロック名： BLK_MSG_SP8

1. ブロック識別子を指定してインクルードファイルをインクルードします。
2. メッセージ定義ファイルを、名前と名前のバイト数を指定してMC_msginitで読み込みます。
3. 使用するメッセージが登録されているブロックをMC_msgsetbkで指定してメッセージ・ブロックの設定を開始します。
4. 終了処理部分でメッセージ・ブロックの設定を終了します。
5. コンパイル・リンクしてエラーがないことを確認します。



メッセージ関係の関数を使用する前に必ずMC_msginitを実行するよう記述します。この関数が2回以上実行された場合、新しくメッセージ定義ファイルを読み込み、前回読み込まれていたファイルの内容と置き換えます。MC_msgsetbkでブロックを指定するとこれ以降MC_msgendbkを実行するまでの間にメッセージ関係の関数で設定を行うメッセージはこのブロックのものが使用されます。

2Dモジュールによりグループ化されている要素の数を取得し、グループ化された要素が存在しない場合はメッセージボックスによりエラーメッセージを表示します。

1. 2Dモジュールでは全ビューを対象にグループ化されている場合がありますが、今回の作成プログラムの処理は現行ビュー・子図内の要素に対してのみ行いたいため、現行ビュー・子図を取得します。もし、現行ビューが全ビュー(1000)となっている場合はPVを現行ビューとみなすよう値を"0"に変更します。
2. MC_butgrpによりグループ化されている要素の数を取得します。数取得のみのため、オプションは"900"です。モデル参照番号はACCESSで図面を制御するための固有の番号です。対話モードの場合、2Dモジュールで表示していた図面にはモデル参照番号"1"が付けられます。表示していた図面に対して処理を行うため、"1"を指定します。
3. 他の入力引数は未使用のため"0"を指定しておきます。

```

      :
      :
MC_bucds( 2L, 1L, 1L, &save_ivudet, &number, 10 );
if ( save_ivudet == 1000L ) save_ivudet = 0L;

istart = 0L;
istop = 0L;
mptrln = 0L;
MC_butgrp ( 900L, 1L, istart, istop, mptrln, mptrar,
            &numptr, nptrar, 5L );

if ( numptr == 0 )
{
    MC_msgbox ( MSG_NOT_FOUND_GROUP, 1L, 1L, &prsbtn );
    goto exit;
}
      :
      :
```

1. 現行ビュー・子図を取得する

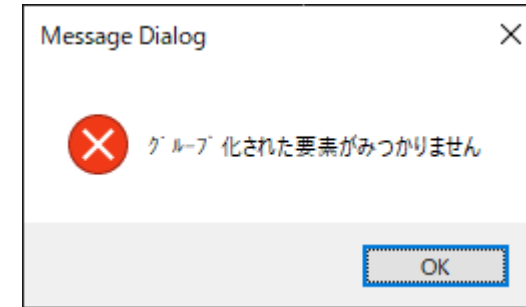
1. 現行ビューが全ビューになっている場合、PVを現行ビューとみなす

3. 未使用引数は"0"を指定

2. 要素数のみを取得する

4. メッセージ名で指定

4. グループ化されている要素数が0の場合、MC_msgboxを使用してメッセージ・ボックスでエラーメッセージを表示します。
 メッセージ番号は「メニュー・メッセージ作成支援ツール」でメッセージ名として設定した値を指定します。



| メッセージ名 | メッセージ(日本語) | 種類 |
|---------------------|----------------------|-----|
| MSG_NOT_FOUND_GROUP | 「グループ化された要素が見つかりません」 | エラー |

表示するボタンは「OK」のみとしますので、ボタンのタイプは"1"を指定します。

表示しているボタンは1個で、そのボタンをデフォルト・ボタンとしますので"1"を指定します。

デフォルト・ボタンとは、改行キーを押したときに、そのボタンをマウスで選択したときと同じ動きをするボタンのことをいいます。
 ボタン番号は左から数えて何番目のボタンであるのか、その番号のことをいいます。

5. コンパイル・リンクしてエラーがないことを確認してください。

6. ポップアップ・メニューへの登録と実行(1/3)

今まで作成したプログラムを実行してみましょう。

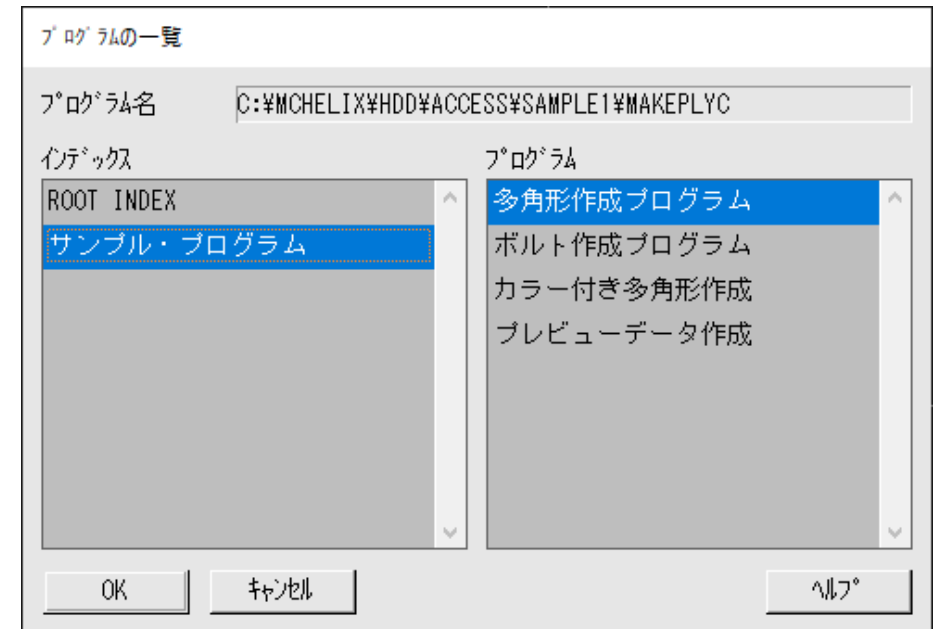
対話モードのプログラムはMC Helixから起動するために登録が必要です。登録方法はいくつかありますが、今回はファンクション<アクセス>のポップアップ・メニューを利用します。

ポップアップ・メニューとは、プログラム名の一覧表のことです。

ファンクション<アクセス>の【選択リスト】を選択してこの一覧表を表示し、項目を選択して目的のプログラムを起動します。

ポップアップ・メニューはプログラム名を記入したテキスト・ファイル(リスト・ファイルといいます)を作成し、指定のフォルダに登録することにより作成できます。

MC Helixインストール時に提供されるACCESSのサンプル・プログラムを起動するためのリスト・ファイルがC:\mchelix\HDD\Sampleに「ACCESS.LST」という名前で格納されています。このファイルをC:\MCADAMの下にコピーしてください。
ファンクション<アクセス>の【選択リスト】を選択するとポップアップ・メニューが表示されます。



サンプルとして提供されている「ACCESS.LST」に今回作成したプログラムを追記することで実行できるようにします。

1. C:¥MCADAM¥の下にコピーした「ACCESS.LST」をエディタで開きます。

```
1 ACCESS USER PROGRAM LIST
2 IDX00
3 サンプル・プログラム IDX01
4 RCサイズ計算プログラム PGM C:¥MHELIX¥HDD¥RCSIZE¥MCRCSIZE
5 IDX01
6 多角形作成プログラム PGM C:¥MHELIX¥HDD¥ACCESS¥SAMPLE1¥MAKEPLYC
7 ボルト作成プログラム PGM C:¥MHELIX¥HDD¥ACCESS¥SAMPLE2¥MAKEBLTC
8 カラー付き多角形作成 PGM C:¥MHELIX¥HDD¥ACCESS¥SAMPLE3¥MAKEPLY2
9 プレビューデータ作成 PGM C:¥MHELIX¥HDD¥ACCESS¥SAMPLE4¥MCPVIEW
[EOF]
```

2. 作成プログラムの情報を追記します。

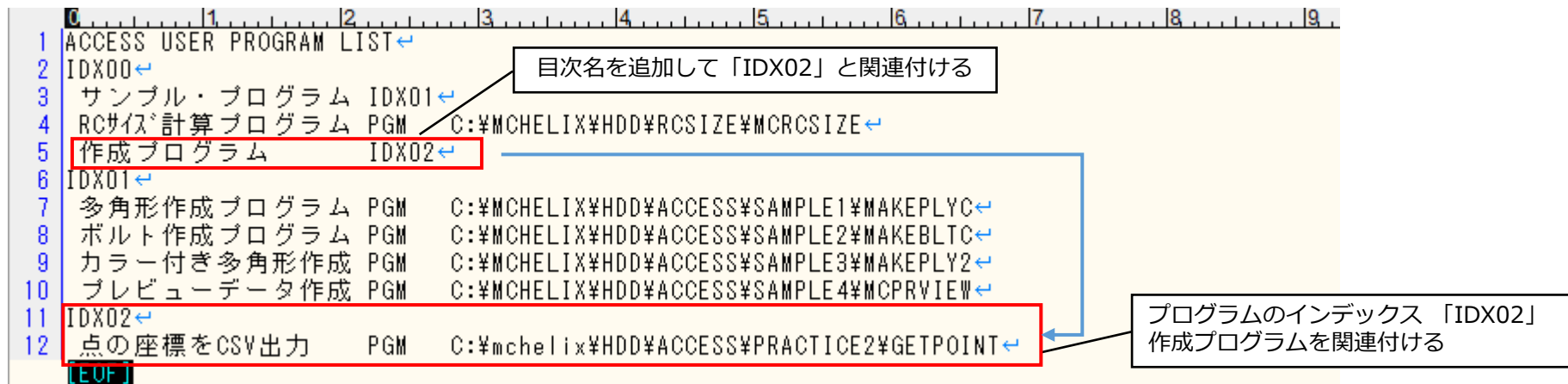
[IDX00] 目次名「作成プログラム」 区分：「IDX02」としてプログラムのリストと関連付ける

[IDX02] 項目名：「点の座標をCSV出力」

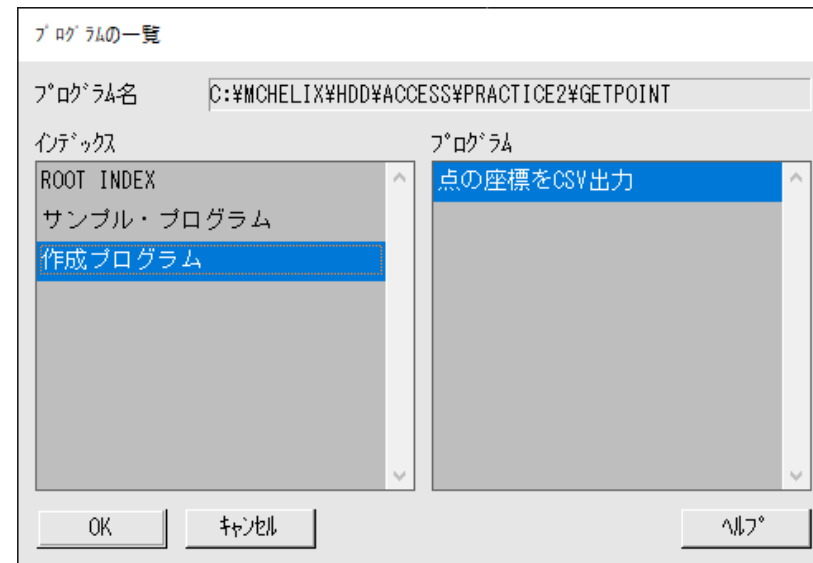
区分：「PGM」 プログラムと関連付けます。

プログラム・パス : 「C:\%mhelix%\HDD\ACCESS\PRACTICE2\GETPOINT」

作成プログラムの格納フォルダとプログラム名(拡張子なし)を指定します。



3. 「ACCESS.LST」を上書き保存します。
4. MC Helixを起動して新規図面を開き、<アクセス>【選択リスト】を選択します。
5. 「プログラムの一覧」が表示されますのでインデックスから「作成プログラム」を選択します。
6. [OK]ボタンを押します。
グループ化されている要素が存在しないためエラーを示すメッセージボックスが表示されることを確認してください。



7. グループ化要素の取得とハイライト表示(1/2)

4 5 6

グループ化されている要素を取得し、その中から点を抽出して現行ビュー・子図内の要素の場合、出力対象とします。出力対象点がない場合はエラーメッセージを表示し、対象点がある場合は点をハイライト表示(要素を強調色で表示)します。

1. グループ化されている要素の個数分の領域を確保します。領域確保に失敗した場合、エラーメッセージを表示します。
2. MC_butgrpでグループ化されている要素のポインターを取得します。
3. 取得したポインター列の中からMC_bufnd2により点を抽出します。
4. 点が現行ビュー・子図の要素の場合、出力対象として、そのポインターを保持します。
5. グループ化された要素のポインター列の最後まで抽出を繰り返します。

```
nptrar = malloc ( numptr * sizeof(long) );
ptptrar = malloc ( numptr * sizeof(long) );
if ( nptrar == NULL || ptptrar == NULL )
{
    MC_msgbox ( MSG_OUT_OF_MEMORY, 1L, 1L, &prsbtn );
    goto exit;
}

MC_butgrp ( 5L, 1L, istart, istop, mptrln, mptrar,
            &numptr, nptrar, 5L );

/* Check Point Elements */
istndx = 1L;
itype = 100L;
ptnum = 0L;
iatno = 0L;
for ( ; ; )
{
    retc = MC_bufnd2 ( 1L, 1L, numptr, nptrar, istndx, &itype,
                    iatno, &index, &matr, 20L );

    if ( retc != 0 ) break;
    istndx = index+1;

    if ( MC_bivdt ( 1L, nptrar[index-1] ) != save_ivudet ) continue;

    ptptrar[ptnum] = nptrar[index-1];
    ptnum++;
}
```

1.個数分の領域確保

1.確保できなかった場合メッセージ表示

2.グループ化要素のポインター取得

3.検索を開始するインデックスを指定

3.点を抽出

3.MC_butgrpで取得したデータを渡す

5.見つかった点の次のポインターから検索を再開するようセット

4.見つかった点が現行ビュー・子図内の点かチェック

4.見つかった点を出力対象として保持

6. 出力対象の点がない場合、エラーメッセージを表示して終了します。
7. 出力対象の点がある場合、MC_budspeによりハイライト表示します。
オプションは"2"、モデル参照番号は"1"を指定します。
表示状態はハイライト表示で"1"を指定します。
処理は1要素ずつ行い、点の個数分繰り返します。

```

if ( ptnum == 0L )
{
    MC_msgbox ( MSG_NOT_FOUND_GROUP, 1L, 1L, &prsbtn );
    goto exit;
}

for ( ii=0; ii<ptnum; ii++ )
{
    MC_budspe ( 2L, 1L, 1L, &ptptrar[ii], 1L, &lenerr, ierrar, 1L );
}
    
```

対象要素をまとめて表示処理することもできますが、その場合は表示できなかった要素の情報が返される配列の領域を十分に確保してください。

8. コンパイル・リンクしてエラーがないことを確認してください。

もし、リンク時に「ファイル 'GETPOINT.DLL'を開くことができません。」というエラーメッセージが表示された場合はMC Helixを終了してから再度リンクするか他のACCESSプログラムを実行後に再度リンクしてください。
前回実行時にプログラムがメモリ上にロードされてそのままになっているため、リンクエラーとなっています。
プログラム実行後のプログラムのロード状態はMC_loadsaveで設定できます。

9. MC Helixを起動して、新規図面に点、円などの要素を作成し、それらをグループ化します。
<アクセス>【選択リスト】から作成プログラムを実行し、グループ化された点のみがハイライトすることを確認してください。

8. ファイル選択画面の表示と要素の通常表示(1/2) 7 8

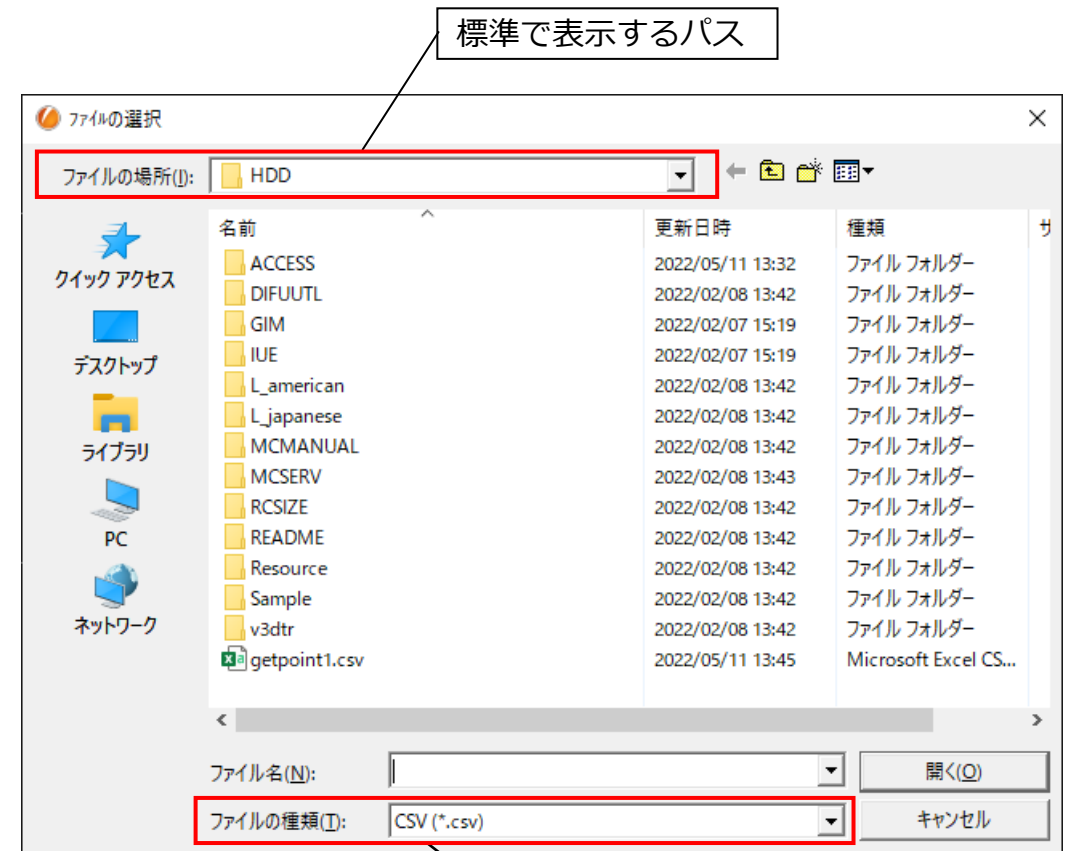
出力先のCSVファイルを指定するためファイル選択画面を表示します。

ファイル名選択後はハイライト表示されていた要素を通常表示に戻します。

ファイル選択画面は標準で表示するパス名、
フィルターとして表示する文字やフィルターとして
設定するワイルドカードなどを指定して表示します。

ファイル選択画面(MC_mmcdfl,MC_mmcdfl2)の他に
次のような選択画面も準備されています。

MC図面名形式のモデル選択画面(MC_mmcd,MC_mmcd2)
OSファイル形式のモデル選択画面(MC_mmcdos,MC_mmcdos2)
グループ、ユーザー選択画面(MC_mmcdgu)
ホスト名、ディレクトリー名選択画面(MC_mmcdoshd)



1. ファイル選択画面の標準で表示するパス名として作業フォルダのパス名を設定するため取得します。
2. CSVファイルのみとするためフィルターを設定します。
フィルターの文字列：「CSV (*.csv)」
フィルターのワイルドカード：「*.csv」
3. 新規ファイルも選択できるようオプションは"1"でMC_mmcdfl2を呼び出します。
4. ファイル名指定後は点要素のハイライト表示を通常表示に戻します。
5. ファイル選択画面で取得エラーが発生したり、「キャンセル」ボタンが押された場合はメッセージを表示して、処理を終了します。
6. コンパイル・リンクしてエラーがないことを確認してください。
7. MC Helixを起動して要素をグループ化してプログラムを実行してください。ファイル選択画面が表示されてCSVファイル名が指定できることを確認してください。

```

memset ( infl, 0x00, sizeof(infl) );
memset ( outfl, 0x00, sizeof(outfl) );

_getcwd ( infl, 512 ); — 1.作業フォルダを標準パスに設定

/* *.CSV */
memset ( filterstr, 0x00, sizeof(filterstr) );
strcpy ( filterstr, "CSV (*.csv)" ); — 2.フィルターの文字列

memset ( filtername, 0x00, sizeof(filtername) );
strcpy ( filtername, "*.csv" ); — 2.フィルターのワイルドカード

retc = MC_mmcdfl2 ( 1L, (long)strlen(infl), infl,
                  1L, filterstr, filtername, &outnum, outfl );
— 3.フィルターは1個 — 3.選択されたパス+ファイル名が返る

/* 点要素の通常表示 */
for ( ii=0; ii<ptnum; ii++ ) — 4.通常表示に戻す
{
    MC_budspe ( 2L, 1L, 1L, &ptptrar[ii], 0L,
               &lenerr, ierrar, 1L );
}

if ( retc != 0 ) — 5.エラー発生や「キャンセル」ボタンが
{                               押された場合はメッセージ表示して終了処理へ
    MC_msgbox ( MSG_END_PROCESS, 1L, 1L, &prsbtn );
    goto exit;
}
    
```


9. CSVファイル出力 9

ファイルを作成し、点の座標を取得して出力します。

1. ファイル選択画面で指定したパス+ファイル名でファイルをオープンします。
2. 対象となる点のX,Y座標を取得します。
3. 出力用文字列にX座標を設定します。
4. 出力用文字列にカンマを追加します。
5. 出力用文字列にY座標と改行コードを追加します。
6. 出力用文字列をファイルに出力します。
7. 点の個数分文字列設定、出力処理を繰り返してファイルをクローズします。
8. 「ファイル出力しました」のメッセージを表示します。
9. MC Helixを起動して要素をグループ化してプログラムを実行してください。ファイル選択画面でファイル名を指定してください。CSVファイルが作成され、点の座標値が出力されていることを確認してください。

以上でプログラムは完成です。次はデバッグ方法の解説です。

```

FilePtr = fopen( outfl, "wb" );
for ( ii=0; ii<ptnum; ii++ )
{
    pos = 0;
    memset ( csv_str, 0x00, sizeof(csv_str) );
    MC_gtpt ( 1L, ptprtar[ii], &xpt, &ypt );
    memset ( WorkBuf, 0x00, sizeof(WorkBuf) );
    sprintf ( WorkBuf, "%8.6f", xpt );
    memcpy ( &csv_str[pos], WorkBuf, strlen(WorkBuf) );
    pos += (long)strlen ( WorkBuf );
    csv_str[pos] = 0x2c;
    pos++;
    csv_str[pos] = 0x0d;
    pos++;
    csv_str[pos] = 0x0a;
    pos++;
    if ( fwrite ( csv_str, pos, 1, FilePtr ) == -1 )
    {
        fclose ( FilePtr );
        MC_msgbox ( MSG_FILE_OUTPUT_ERROR, 1L, 1L, &prsbtn );
        goto exit;
    }
    fclose ( FilePtr );
    MC_msgbox ( MSG_FILE_OUTPUT, 1L, 1L, &prsbtn );
}

```

1.指定したファイル名でオープン

2.点の座標取得

3.X座標文字列設定

4.カンマ追加

5.X座標と同様にY座標文字列設定

5.改行コード設定

6.ファイルに1行出力

7.出力失敗メッセージ表示

7.ファイルクローズ

8.出力メッセージ表示

10. デバッグ方法(1/4)

プログラムのデバッグには「Microsoft Visual Studio」を使用します。

デバッグするためにはデバッグ用のオプションを指定して、コンパイル・リンクする必要があります。

1. コンパイル時のオプションに「 /Od /Zi」を追加します。

プログラム作成用のバッチファイル内のコンパイル行に追加します。

```

:
:
SET LIB=%ACCESS_PATH%;%LIB%;
SET INCLUDE=%ACCESS_PATH%;%INCLUDE%;

cl /c /W3 /J /MD /Od /Zi /DWIN32 /D_NTSDK /D_CRT_SECURE_NO_WARNINGS GETPOINT.C

LIB /MACHINE:X64 /OUT:GETPOINT.LIB /DEF:GETPOINT.DEF > NUL

LINK @GETPOINT.LNK

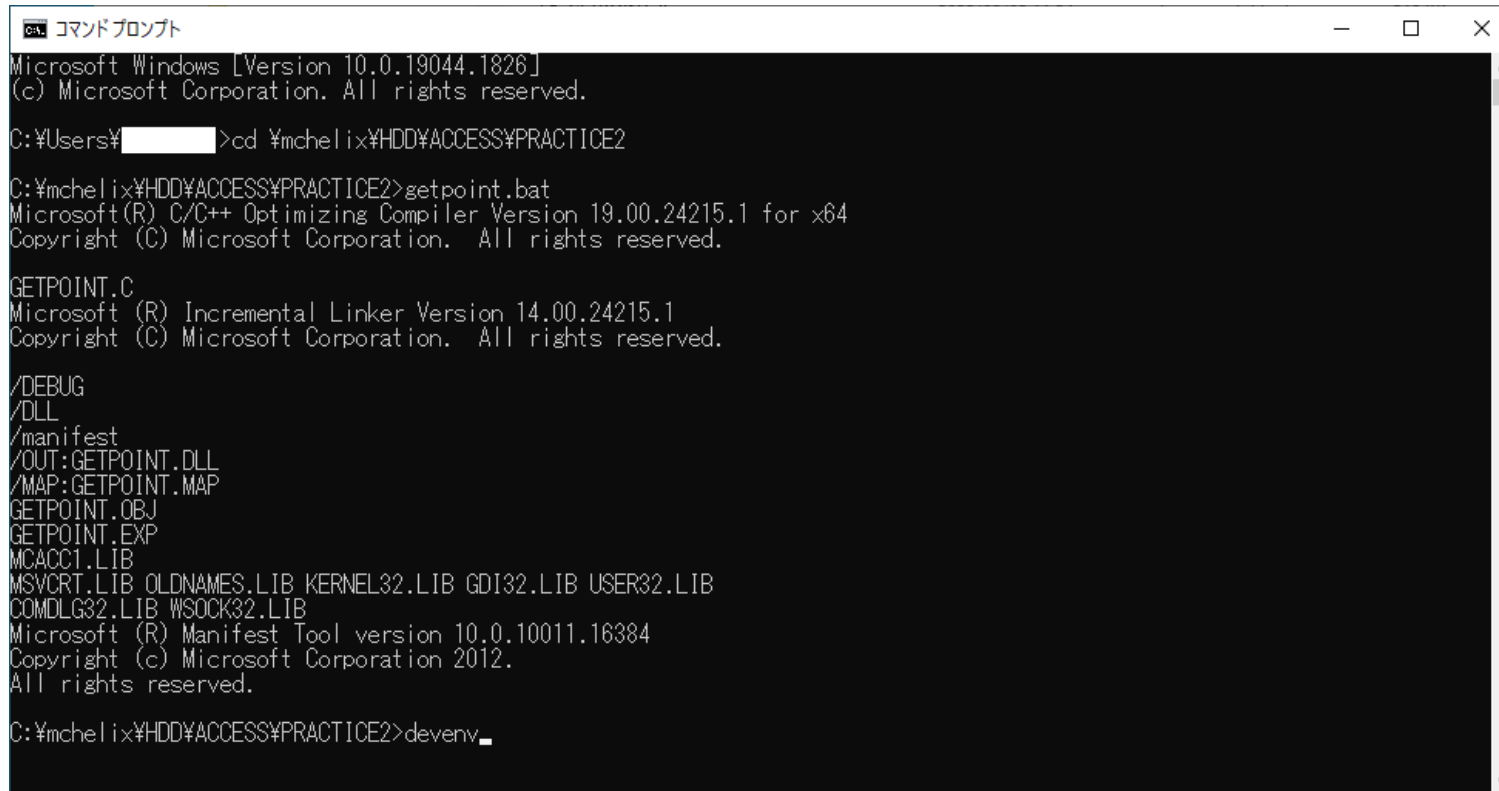
:
:
```

2. リンク・コントロールファイルに「/DEBUG」を追加します。

```
/DEBUG
/DLL
/manifest
.
.
```


3. コンパイル・リンク用のコマンドプロンプトからオプションを追加したバッチファイルを実行してコンパイル・リンクします。
4. 同じコマンドプロンプトから「Microsoft Visual Studio」を起動します。

コマンドプロンプトで“devenv”と入力して[Enter]



```
コマンドプロンプト
Microsoft Windows [Version 10.0.19044.1826]
(c) Microsoft Corporation. All rights reserved.

C:\Users\>cd %mchelix%HDD%ACCESS%PRACTICE2

C:\mchelix%HDD%ACCESS%PRACTICE2>getpoint.bat
Microsoft(R) C/C++ Optimizing Compiler Version 19.00.24215.1 for x64
Copyright (C) Microsoft Corporation. All rights reserved.

GETPOINT.C
Microsoft (R) Incremental Linker Version 14.00.24215.1
Copyright (C) Microsoft Corporation. All rights reserved.

/DEBUG
/DLL
/manifest
/OUT:GETPOINT.DLL
/MAP:GETPOINT.MAP
GETPOINT.OBJ
GETPOINT.EXP
MCACC1.LIB
MSVCRT.LIB OLDNAMES.LIB KERNEL32.LIB GDI32.LIB USER32.LIB
COMDLG32.LIB WSOCK32.LIB
Microsoft (R) Manifest Tool version 10.0.10011.16384
Copyright (c) Microsoft Corporation 2012.
All rights reserved.

C:\mchelix%HDD%ACCESS%PRACTICE2>devenv _
```

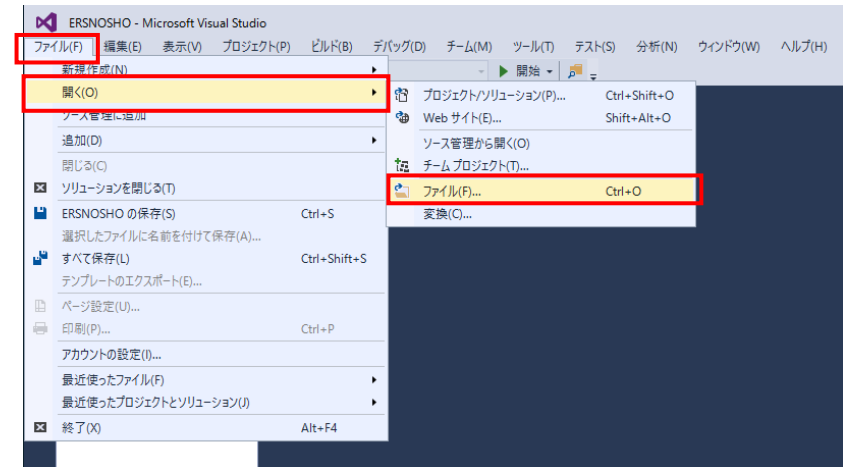
5. 「Microsoft Visual Studio」 が起動します。
メニュー[ファイル]の[開く]-[プロジェクト/ソリューション]を
選択します。

6. 「プロジェクトを開く」ダイアログが表示されるので、
「c:\%mhelix%HDD」から
実行プログラム「mcw.exe」を選択します。

7. メニュー[ファイル]の[開く]-[ファイル]を選択します。

8. 「ファイルを開く」ダイアログが表示されるので、
「c:\%mhelix%HDD%ACCESS%PRACTICE2」から
ソースファイル「GETPOINT.C」を選択します。

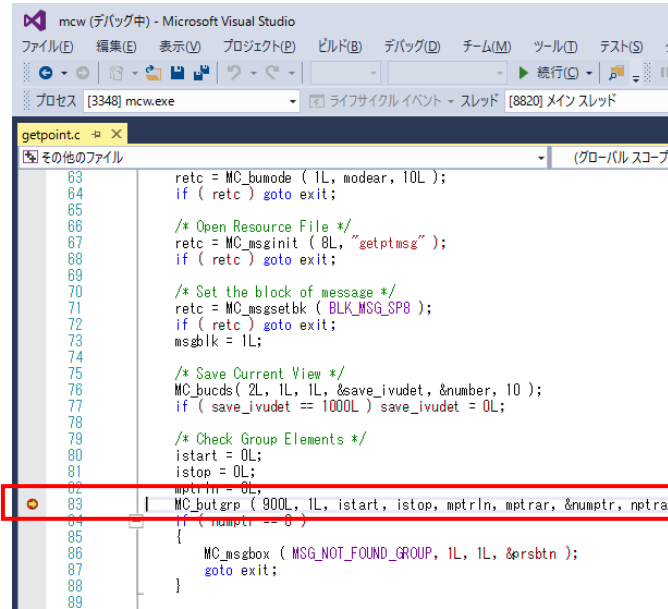
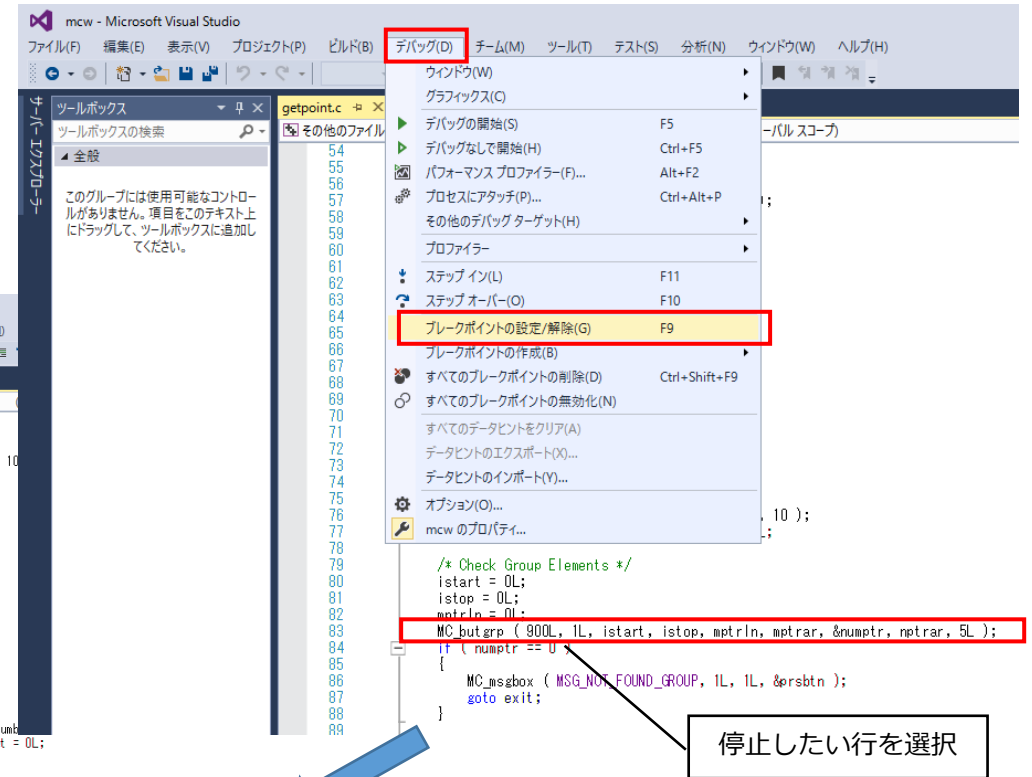
ソースコードが表示されます。



9. デバッグで停止したい行を選択して、メニュー[デバッグ]の [ブレークポイントの設定/解除]または[F9]でブレークポイントを設定します。

10. メニュー[デバッグ]の[デバッグ開始]またはツール・バーの [開始]で実行します。

11. MC Helixが起動しますので <アクセス>【選択リスト】から プログラムを実行します。



- 10. で[開始]後にダイアログ・ボックスで続行するか否かを問いかせてきたら[はい]を押して続行してください。
- 11. でプログラム実行後にブレークポイントで停止したらデバッグしてください。



※当資料内の文章・画像・商標等（以下、「データ」）に関する著作権とその他の権利は、弊社または原作者、その他の権利者のものです。企業等が非営利目的で使用する場合、個人的な使用を目的とする場合、その他著作権法により認められている場合を除き、データは弊社、原作者、その他の権利者の許諾なく使用することはできません。

※データ等のご利用またはご利用できなかったことによって生じた損害については、弊社は一切の責任を負わないものとし、いかなる損害も補償をいたしません。

※掲載されている内容は2022年8月時点のものです。内容は、事前の予告なしに変更することがあります。

MICRO CADAM、MICRO CADAM Helix は、株式会社CAD SOLUTIONSの商標です。
他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。