日付:	2018/08/17
著者:	CarMaker サービスチーム、Kaustub Ragunathan
バージョン番号:	CM-7.0.1

グローバル C コード変数を User Accessible Quantity として追加する

CarMaker では、シンプルなシミュレーション時でも 1,000 を超える変数が UAQ (User Accessible Quantity)として利用されます。そして、そういった変数の中でも、計算されているものの、UAQ として示されない変数も多数存在します。これらの変数は、デフォルトで C コードレベルで利用できます。以下に示す例では、これらの変数にアクセスし、アクセスした変数から UAQ を作成する方法を説明します。作成した UAQ は結果ファイルに保存したり、IPGControl で表示したりできます。

使用事例

Free Space Sensor では多数の UAQ が計算されますが、以下の 2 つの信号の計算を例に挙げます。

Sensor.FSpace.<name>.Segm.<Nr>.ds.x

Sensor.FSpace.<name>.Segm.<Nr>.ds.y

これらの信号は、特定のセグメントで識別された物標の最近傍点とセンサとの間の距離(x 方向および y 方向)を返します。ただし、これらの信号はセンサ座標系で計算されます。ここでは、慣性座標系 Fr0 でこれらの信号が必要であると仮定しましょう(コントローラアルゴリズムで必要とすると仮定)。

CarMaker では、C コードレベルで自動的に信号が計算されるため、こうしたニーズに簡単に対応できます。変換行列を使用してこれらの信号をセンサ座標系から慣性座標系に変換する必要はありません。次は、これらの信号に UAQ レベルでアクセスします。

これらの信号は、Free Space Sensor に関連するヘッダファイル内で宣言されます。そのヘッダファイルは Sensor_FSpace.h という名前が付けられており、/IPG/hil/<CarMaker version>/include>CarMaker インストールフォルダ>「Vehicle」フォルダ内に格納されています。求めている変数は P_0[0]と P_0[1]であり、tSegment 構造体で見つけることができます。この構造体は、別の tFSpaceSensor 構造体の一部です。この構造体は以下のように宣言されています。

*extern tFSpaceSensor *FSpaceSensor;*

したがって、この名前で直接アクセスできます。以下は、User.c ファイル(CarMaker プロジェクトフォルダ> src フォルダ内)に入力する必要のあるコード例です。コードの最後には、簡単な説明を記述します。

```
1: ------------------------------------------------------------
2: 行「int UserCalcCalledByAppTestRunCalc = 0;」の後に以下の行を追加
3: ------------------------------------------------------------
4:
5: #include <Vehicle/Sensor FSpace.h>
6:
7: double **FSS x0 = NULL;
8: double **FSS y0 = NULL;
9: int i, nFSS, j;
10:
11: ------------------------------------------------------------
```

関数「int User TestRun Start atEnd (void)」に以下の行を追加

```
------------------------------------------------------------
nFSS = iGetIntOpt(SimCore.Vhcl.Inf, "Sensor.FSpace.N", 0.0);

FSS x0 = (double **)malloc (sizeof(double *)*nFSS);
FSS y0 = (double **)malloc (sizeof(double *)*nFSS);

for (i=0; i<=(nFSS-1); i++) {

  FSS x0[i] = (double *)malloc (sizeof(double)*FSpaceSensor[i].nTotSegm);
  FSS y0[i] = (double *)malloc (sizeof(double)*FSpaceSensor[i].nTotSegm);

}

for (i=0; i<=(nFSS-1); i++) {

     for (j=0; j<=(FSpaceSensor[i].nTotSegm-1); j++) {

         FSS x0[i][j]=0;
         FSS y0[i][j]=0;

    }
}

tDDefault *df = DDefaultCreate(NULL);

for (i=0; i<=(nFSS-1); i++) {

     for (j=0; j<=(FSpaceSensor[i].nTotSegm-1); j++) {

        char sbuf1 [160];
        sprintf(sbuf1, "Sensor%01d.", i);
        char *FSSPos = sbuf1;
        char sbuf [160];
        DDefPrefix (df,"FSpace pos.%s",FSSPos);

        sprintf(sbuf, "X 0.%01d", j);
        DDefDouble4 (df, sbuf,"m", &FSS x0[i][j], DVA None);

        sprintf(sbuf, "Y 0.%01d", j);
        DDefDouble4 (df, sbuf,"m", &FSS y0[i][j], DVA None);
    }

}
------------------------------------------------------------
```

関数「int User_Calc (double dt)」に以下の行を追加

```
------------------------------------------------------------

if (SimCore.State == SCState_Simulate) {

   for (i=0; i<=(nFSS-1); i++) {

        for (j=0; j<=(FSpaceSensor[i].nTotSegm-1); j++) {

        FSS_x0[i][j] = FSpaceSensor[i].Segm[j].P_0[0];
        FSS y0[i][j] = FSpaceSensor[i].Segm[j].P 0[1];

       }
   }
}
------------------------------------------------------------
```

関数「void User Cleanup (void)」に以下の行を追加

```
------------------------------------------------------------

if ( FSS x0!= NULL)
   free (FSS x0);

if ( FSS y0!= NULL)
```

```
80:    free (FSS y0);
```

C コードを順に見ていきましょう。

5 行目から 10 行目では、User.c ファイルにヘッダファイルをインクルードし、後に使用するいくつかの変数を初期化しています。重要なのは、2 つの変数 FSS_x0 と FSS_y0 が double 型の 2D 配列として宣言されていることです。このようにする理由を簡単に説明します。

14 行目では、車両に追加された Free Space Sensor の数を読み取っています。

16 行目から 24 行目にかけて、2D 配列 NxM (N はセンサの数、M はセグメントの総数)のメモリを動的に確保しています。これにより、C コードは完全にスケーラブルになります。ハードコード化されていないため、一部のセンサでしか動作しないということがありません。これが、変数 FSS_x0 と FSS_y0 を固定次元の静的な配列ではなく、ポインタへのポインタとして定義した理由です。

28 行目から 34 行目にかけて、2 つの 2D 配列変数 FSS_x0 と FSS_y0 の各要素にデフォルト値である 0 を割り当てています。

36 行目から 55 行目にかけて、CarMaker でよく使用される関数「DDefDouble」を使って UAQ を作成しています。しかし、UAQ をさらに明快かつわかりやすくするために、UAQ の名前にプレフィックスを割り当てます。この割り当てには、「DDefPrefix」という関数を使用します。for ループを使用すれば、変数それぞれに対する UAQ を簡単に作成できます。水平方向に 40 セグメント、垂直方向に 50 セグメントの Free Space Sensor が 4 つあれば、合計で 16,000 個(4×40×50×2)の UAQ が生成されることになります。UAQ の名称は以下の形式となります。

FSpace_pos.Sensor<sensorNo>.X_0.<SegmentNo>

FSpace_pos.Sensor<sensorNo>.Y_0.<SegmentNo>

例:

FSpace_pos.Sensor0.X_0.12
FSpace_pos.Sensor0.Y_0.12

60 行目から 71 行目にかけて、2 重の for ループ(センサの数に応じたメインループと、セグメントごとのインナーループ)により、ヘッダファイルの変数 P_0[0]と P_0[1]の値をローカル変数 FSS_x0 と FSS_y0 に代入します。

優れたプログラミングを行うために、変数で占有されているメモリをシミュレーションプログラムが終了する前にクリアすることをお勧めします。これは 76～80 行目で実行しています。

これらの変更を行った後、User.c ファイルを保存し、CarMaker の実行ファイルを再構築する必要があります。このプロセスについては、『Programmer's Guide』の 1.3.2 章を参照してください。実行ファイルを生成したら、CarMaker の GUI を開き、メニューオプションの[Application] > [Configuration / Status]に移動し、生成したばかりの CarMaker 実行ファイルを選択します。これで、Free Space Sensor を搭載した車両を含む任意の TestRun を選択できます。シミュレーションを開始し、IPGControl を開くと、図「16,000 個の UAQ を示す FSpace_pos カテゴリ」の赤枠で示すように、新たに追加されたすべての信号が表示されます。

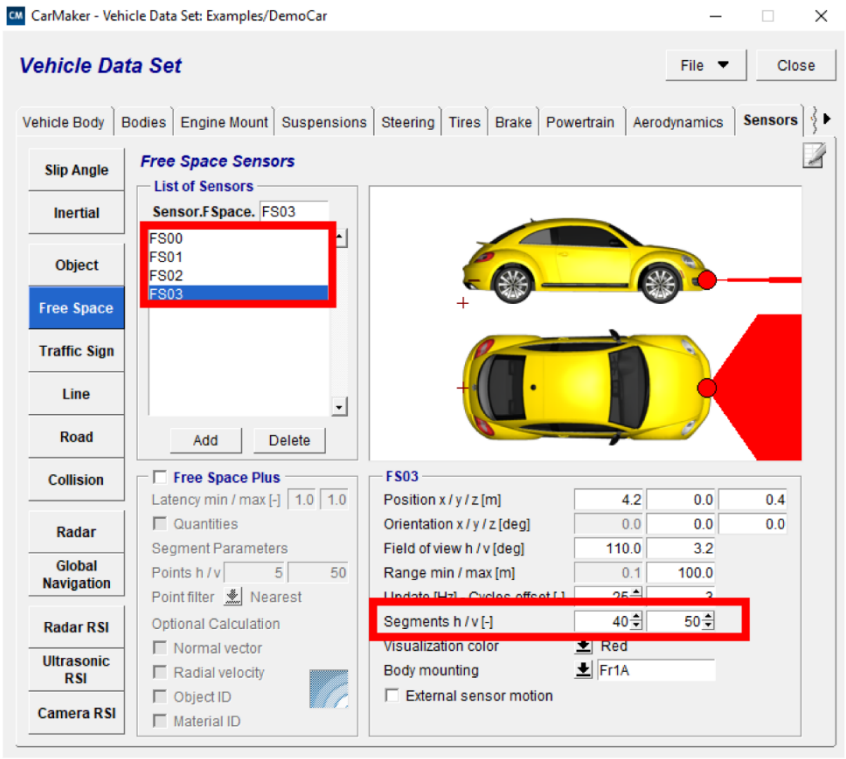

図 1: それぞれが 40 × 50 のセグメントを持つ 4 つの Free Space Sensor

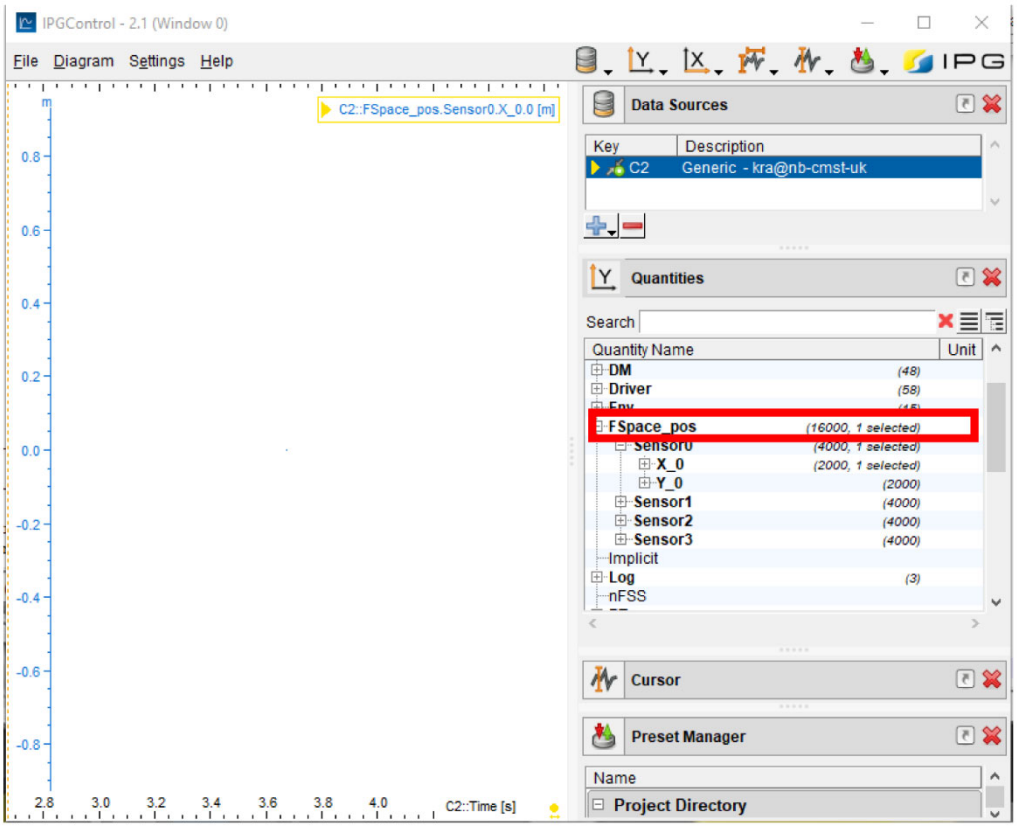

図 1: 16,000 個の UAQ を示す FSpace_pos カテゴリ