第14章　カスタム・プログラミング

これまでにMATLABのブロックを組み合わせてモデルを制作し，シミュレーションを行い，プログラムをビルドして，パソコンへダウンロードして実行しました．

MATLABに適当なブロックが見当たらない場合，独自のブロックを制作してMATLABのブロックと同様に使用することができます．

ユーザが制作するブロックをカスタム・ブロックといいます．カスタム・ブロックの作り方については，参考文献2において詳しく述べました．

ここでは，特にxPC Targetのカスタム・ブロックの作り方について述べます．

14.1　カスタム・ブロック

これまでにMATLABのブロックを使ってモデルを制作し，プログラムをビルドし，パソコンへダウンロードして実験を行いました．

The MathWorks社は各種のブロックを開発し，それを自社プロダクトへ組み込んで販売しているので，必要なプロダクトがあればそれを購入することによって開発の工期を短縮することが可能になります．

しかし，問題によっては必要なブロックが入手できないという状況も考えられます．

また，すでに何らかのプログラム資産があり，それを利用したいという場合もあります．

このような場合はカスタム・ブロックを使います．

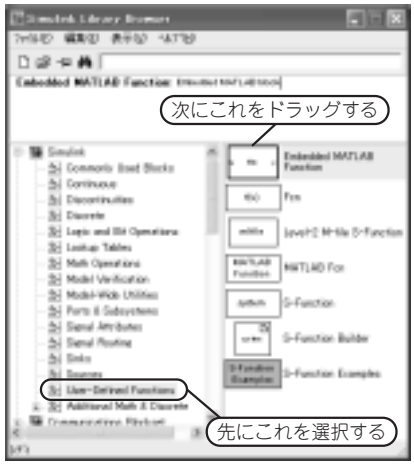
Real-Time Workshopにおけるカスタム・ブロックの使用法については，参考文献2において詳しく述べました．

ここではxPC Targetにおけるカスタム・ブロックの使い方について述べます．

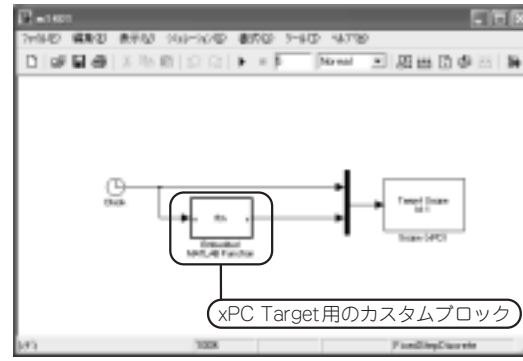
最初に，MATLABのMファイルを使ってカスタム・ブロックを作ります．

モデルm1401としてディレクトリm1401へ格納します．

［Simulink Library Browser］において**画面14.1**に示すように［Simulink］→［User-Defined Functions］とクリックして，［Embedded MATLAB Function］をモデル・ウインドウへドラッグ・アンド・ドロップします．



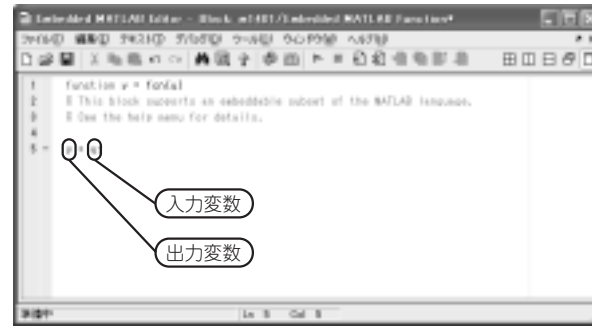
画面14.1　［Embedded MATLAB Function］



画面14.2　モデルm1401

表14.1　モデルm1401を作る

ブロック	場所
Clock	［Simulink］→［Sources］
Mux	［Simulink］→［Commonly Used Blocks］
Scope (xPC)	［xPC Target］→［Misc］



画面14.3　［Embedded MATLAB Function］の編集

xPC Target においてMファイルのカスタム・ブロックを作るときは，［Embedded MATLAB Function］ブロックを使用します．

続いて，［Simulink Library Browser］から**表14.1**に示すブロックをドラッグ・アンド・ドロップして，**画面14.2**に示すようにモデルを作ります．

［Embedded MATLAB Function］ブロックをダブル・クリックします．

画面14.3の編集画面が開きます．

ここでuは入力変数，yは出力変数です．

修正なしではチェックにならないので，デフォルトの，

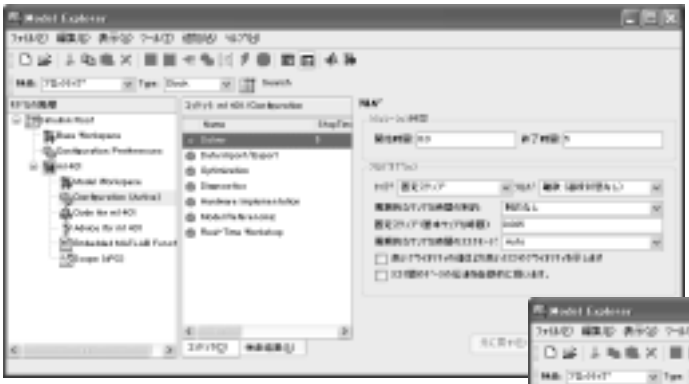
```
y＝u
```

を，

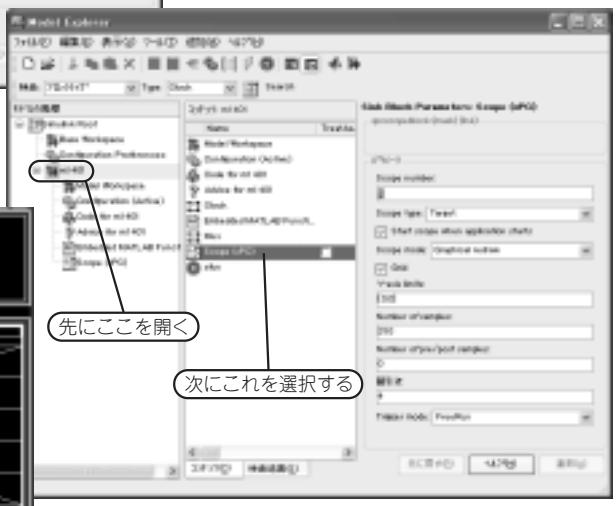
```
y＝5－u
```

と変更します．

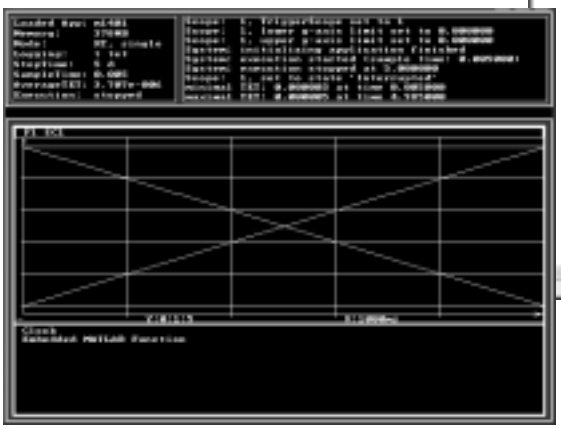
変更したら，必ず編集ウインドウのツール・バー上の［保存］ボタンをクリックして修正した内容を保存します．



画面14.4　[Solver]の設定



画面14.5　[Scope(xPC)]の設定



画面14.6　実行結果

[Model Explorer]を開いて[Solver]を**画面14.4**に示すように設定します．

[Scope(xPC)]は**画面14.5**に示すように設定します．

シミュレーション時間を5秒，間引き係数を4，サンプル時間を0.005としたので，

$$\frac{5}{0.005\times 4}=250$$

となります．シミュレーション終了時に[Scope(xPC)]はグラフを表示します．この250はグラフのプロット数です．

ターゲット2を電源をONにしてローダの状態で待機させます．

モデルをビルドして，ターゲット2へダウンロードして実行します．

結果を**画面14.6**に示します．

[Clock]ブロックの出力は右上がりの直線，[Embedded MATLAB Function]ブロックの出力は右下

がりの直線です.

期待した結果が得られました.

［Embedded MATLAB Function］ブロックの処理を記述する際にMATLABのスクリプト言語を使用します.

宇宙船のモデルにおいて使用した［Timing］ブロックを，［Embedded MATLAB Function］ブロックを用いて作ります.

モデルをm1402としてディレクトリm1402へ格納します.

画面14.7にモデルm1402を示します.

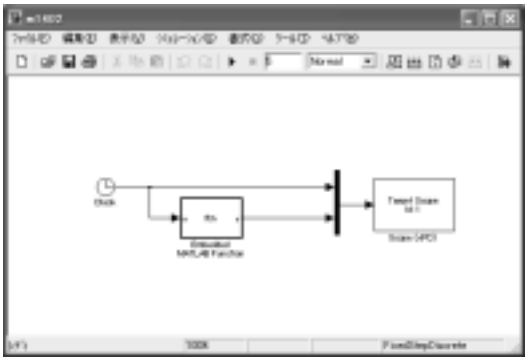
［Embedded MATLAB Function］ブロックの編集画面を**画面14.8**に示します.

MATLABのスクリプトはC言語に似ているのですが，同じではありません.

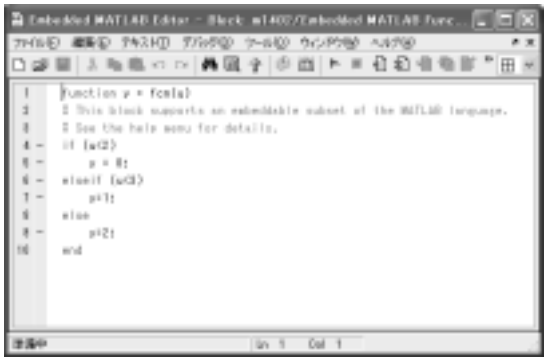
if文の最後にend記号が必要です．画面で確認してください.

モデルに関する設定はm1401と同じです.

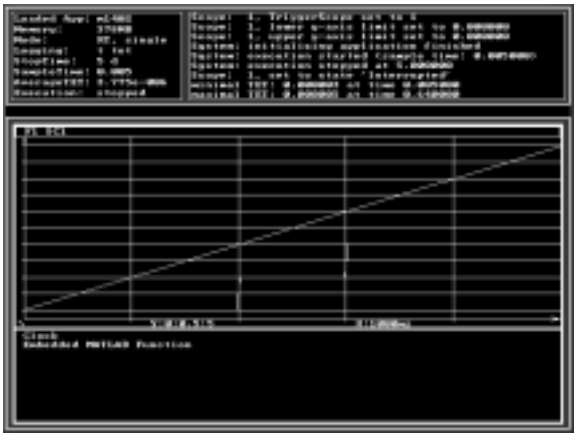
モデルをビルドして実行すると**画面14.9**となります.



画面14.7　モデルm1402



画面14.8　［Embedded MATLAB Function］の編集画面



画面14.9　実行の結果

［Embedded MATLAB Function］の出力はグラフのグリッドと一致するので，本書の印刷でははっきり見えないかもしれません．

多くの関数を組み合わせて数学的な処理を行う場合は，Simulinkのブロックを使うよりMATLABのスクリプトによって処理を記述するとモデルは簡単になります．

■ 14.2　C言語のカスタム・ブロック

カスタム・ブロックを使用する主な目的は，おそらくC言語のプログラムを追加することがメインになります．

まず最初に簡単なモデルを作り，C言語のカスタム・ブロックを作る際の手順を示します．

モデルをm1403としてディレクトリm1403へ格納します．

画面14.1（P.312）に示した［Simulink Library Browser］の［User-Defined Functions］のカテゴリから，今度は［S-Function］ブロックをドラッグ・アンド・ドロップします．

そのほかのブロックはこれまで使用したものです．

モデルを**画面14.10**に示します．

［Model Explorer］において［Solver］と［Real-Time Workshop］の設定は，これまでの設定と同じです．

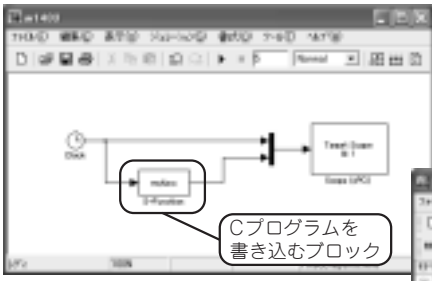
［Scope（xPC）］の設定はこれまでの設定と同じです．

画面14.11に示すように，［Model Explorer］において［S-Function］の設定ダイアログを開きます．

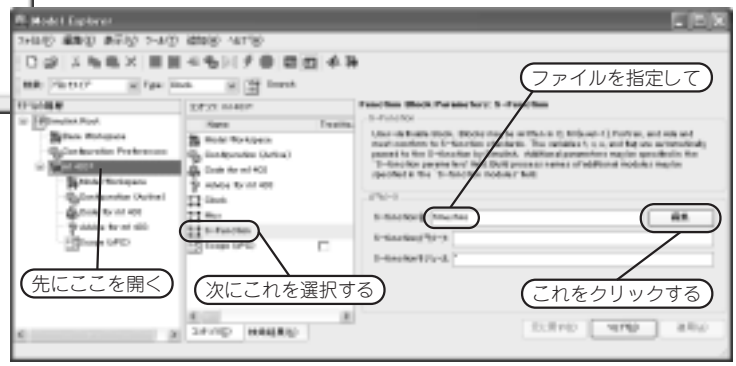
画面に示したように［S-Function名］のところへ，

```
timestwo
```

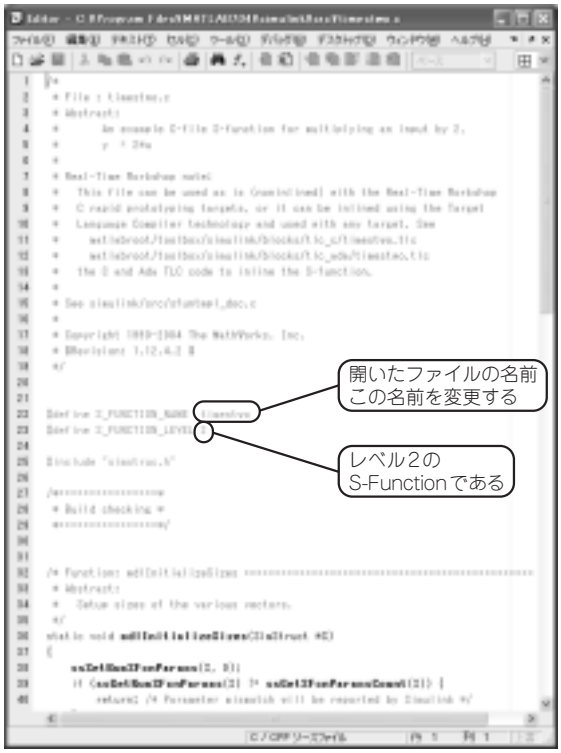
と書き込み，右側の［編集］ボタンをクリックします（画面を開いた直後は，systemとなっている）．



画面14.10　m1303



画面14.11　S-Functionの設定



画面14.12　timestwo.cの編集画面

画面14.12に示すようにtimestwo.cの編集画面が開きます．

このファイルはMATLABに用意されているS-Functionのサンプル・プログラムです．

システムに影響を与えてはいけないので，エディタのメニューから［ファイル］→［別名で保存］とクリックします．

システムのディレクトリが表示されるので，現在のディレクトリ，すなわちm1403へ，

```
myfunc
```

という名前で保存します．

画面14.12に示したように，

```
#define S_FUNCTION_NAME timestwo
```

を，

```
#define S_FUNCTION_NAME myfunc
```

と変更します（**画面14.12**は変更前の画面）．

変更したらツール・バーの［保存］ボタンをクリックして変更を保存します．

［Model Explorer］の画面に戻って（**画面14.11**）［S-Function名］のtimestwoを，

```
myfunc
```

と変更して，［適用］ボタンをクリックします．

myfuncは，私がつけた名前なので変更することは自由です．

要約すると，MATLABが用意している，

timestwo.c

というサンプル・ファイルをシステムのディレクトリから引っ張り出して，これを，

myfunc.c

という名前で自分のディレクトリへコピーし，それにともなって必要な変更したということです．

このような手続きを踏まないとシステムのファイルを壊してしまうことがあるので，十分に注意します．

また，ゼロからスタートしてS-Functionのプログラムを書くのは，エラーの確率が大きいので，避けたほうが良いと思います．

名前を変更しただけですが，とにかくmyfunc.cができました．

プログラムの内容はtimestwo.cそのままなので，入力を2倍して出力するCのプログラムです．

myfunc.cの内容を**リスト14.1**に示します（ただし，オリジナル・ファイルの空白行は削除した）．

リスト14.1　myfunc.cの内容

```
/*
 * File : timestwo.c
 * Abstract:
 *       An example C-file S-function for multiplying an input by 2,
 *       y  = 2*u
 *
 * Real-Time Workshop note:
 *   This file can be used as is (noninlined) with the Real-Time Workshop
 *   C rapid prototyping targets, or it can be inlined using the Target
 *   Language Compiler technology and used with any target. See
 *     matlabroot/toolbox/simulink/blocks/tlc_c/timestwo.tlc
 *     matlabroot/toolbox/simulink/blocks/tlc_ada/timestwo.tlc
 *   the C and Ada TLC code to inline the S-function.
 *
 * See simulink/src/sfuntmpl_doc.c
 *
 * Copyright 1990-2004 The MathWorks, Inc.
 * $Revision: 1.12.4.2 $
 */
#define S_FUNCTION_NAME  myfunc
#define S_FUNCTION_LEVEL 2
#include "simstruc.h"
/*================*
 * Build checking *
 *================*/
/* Function: mdlInitializeSizes ===============================================
```