

MATLAB 6.x 版 対応
MATLAB対応ライブラリ
ADSP324-41C / ADSP404-241C / ADSP674-341C
MATLAB対応U I L
ADSP324-49C / ADSP404-249C / ADSP674-349C
モデルコントロール
ADSP324-50 / ADSP404-250 / ADSP674-350

5. x. x 版 インストール・ガイド

中部電機株式会社

<< 目 次 >>

1	概要	2
2	全体の手順.....	4
3	インストール手順.....	5
3.1	DSPとDSP用コンパイラセットアップ	5
3.2	MATLABのインストール.....	5
3.3	旧版MATLAB対応ライブラリのアインストール	5
3.4	MATLAB対応ライブラリのインストール	6
4	異なる種類のDSPの同時利用	11
5	5.02.00 版以降にADSP 324-00Aを使用する際の注意事項	12
6	MATLAB対応ライブラリ実行環境について	13

1 概要

本書は、MATLAB対応ライブラリ（マルチ）「ADSP324-41C/ADSP404-241C/ADSP674-341C」及び、MATLAB対応UIL「ADSP324-49C/ADSP404-249C/ADSP674-349C」及び、モデルコントロール「ADSP324-50/ADSP404-250/ADSP674-350」（共に5.02.00版）のインストール方法について説明しています。本書が説明している内容は、DSPのセットアップを含めた全体の手順概略と、本製品の具体的なインストール手順です。DSPのセットアップ、コンパイラのセットアップ等については、個々の製品の説明書が必要になりますので、本製品以外に以下の製品の説明書等をお手元に用意してください。

ADSP324-41C/ADSP324-49C/ADSP324-50の場合

- DSPボード「ADSP324-00A スタートアップガイド」
- 「ADSP324-41C ライセンスシート」
- 「ADSP324-49C ライセンスシート」
- 「ADSP324-50 ライセンスシート」
- 「ADSP324-41C ユーザーズ・マニュアル」
- 「ADSP324-49C ユーザーズ・マニュアル」
- 「ADSP324-50/ADSP404-250/ADSP674-350 チュートリアル」
- DSP用コンパイラ CPL320FLT-MS の説明書
- MATLAB/SIMULINK/RTWの説明書

ADSP404-241C/ADSP404-249C/ADSP404-250の場合

- DSPボード「ADSP404-00 スタートアップガイド」
- 「ADSP404-241C ライセンスシート」
- 「ADSP404-249C ライセンスシート」
- 「ADSP404-250 ライセンスシート」
- 「ADSP404-241C ユーザーズ・マニュアル」
- 「ADSP404-249C ユーザーズ・マニュアル」
- 「ADSP324-50/ADSP404-250/ADSP674-350 チュートリアル」
- DSP用コンパイラ CPL320FLT-MS の説明書
- MATLAB/SIMULINK/RTWの説明書

ADSP674-341C/ADSP674-349C/ADSP674-350の場合

- DSPボード「ADSP674-00 スタートアップガイド」
- 「ADSP674-341C ライセンスシート」
- 「ADSP674-349C ライセンスシート」
- 「ADSP674-350 ライセンスシート」
- 「ADSP674-341C ユーザーズ・マニュアル」
- 「ADSP674-349C ユーザーズ・マニュアル」
- 「ADSP324-50/ADSP404-250/ADSP674-350 チュー
トリアル」
- DSP用コンパイラ CPL320FLT-MS の説明書
- MATLAB/SIMULINK/RTWの説明書

2 全体の手順

本製品が使える状態までセットアップする手順（及び参照する説明書）は以下の通りです。

- 1、DSPとDSP用コンパイラセットアップ（スタートアップガイド）
- 2、MATLABのインストール（MATLABの説明書、及び本書）
- 3、旧版のMATLAB対応ライブラリのアンインストール
- 4、ADSPXX4-X41C/X49C/X50のインストール（本書）

1～2項については、各資料を参照のうえ、セットアップを行ってください。

3～4項については、次ページ以後の説明に従ってください。

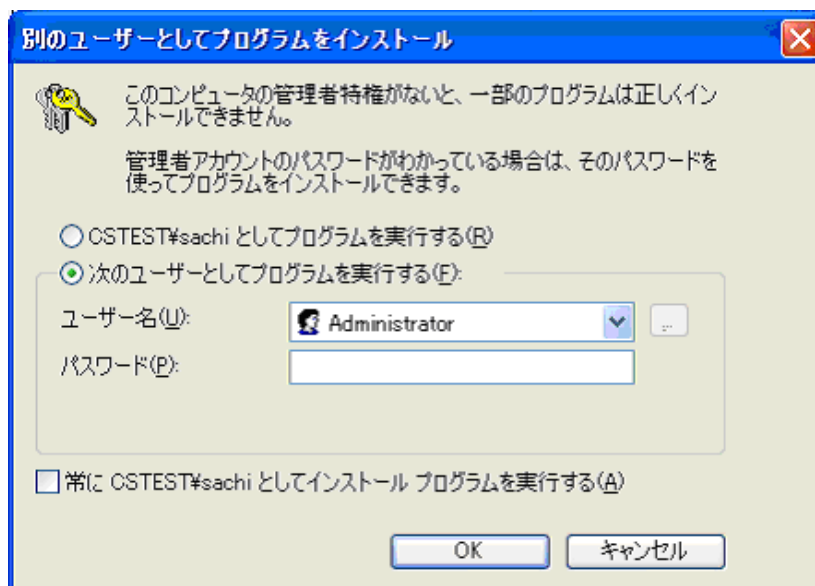
“2、MATLABインストール後”MATLABを起動し以下の設定を確認して下さい。
メニューバーより→ファイル→設定→「一般」→“Toolbox キャッシュを行う”のチェックをはずす。
設定後一旦MATLABを閉じてから“3、”以降の処理へ移ってください。

《注意》

インストールは、管理者権限を持つユーザにて実行してください。管理者権限のないユーザにて実行した場合は正常にインストールされませんので、一度アンインストールしてから再度インストールを実行してください。

管理者権限のないユーザにてインストールを開始すると以下のメッセージが表示される場合があります。ユーザの切り替えをして、再度インストールをしてください。

インストール先フォルダは、名称に日本語・空白を含まないフォルダへインストールしてください。例えば、“C:\¥program files”の下へインストールするのはNGです。



3 インストール手順

3.1 DSPとDSP用コンパイラセットアップ

DSPボードに添付している「スタートアップガイド」及び、DSP用コンパイラ（CPL320FLT-MS/CPL3206X-MS/HLL3206X-MS）の説明書を参照し、DSPのセットアップとコンパイラのインストールからDSPボードの動作確認まで行ってください。

既にインストールしてある場合は、再インストールは不要です。

3.2 MATLABのインストール

使用出来るMATLABは、6.0/6.1/6.5版です。

既にMATLAB6.0/6.1/6.5版がインストールしてある場合は、再インストールは不要です。

それ以外の版をお使いの場合や、まだMATLABがインストールされていない場合は、MATLABの取扱説明書を参照のうえMATLABのインストールを行ってください。

3.3 旧版MATLAB対応ライブラリのアンインストール

旧版の弊社MATLAB対応ライブラリ「ADSPXX4-41C/X49C/X50」をご利用の場合は、それらの製品をアンインストールしてください。

5.00.00版より前の場合

アンインストールは、Windowsのコントロールパネルにある、「アプリケーションの追加と削除」を起動し、下記の項目を選択し、「追加と削除ボタン」で削除します。

ADSP32X MATLAB 6.x Library ADSP324の場合

ADSP404 MATLAB 6.x Library ADSP404の場合

ADSP674 MATLAB 6.x Library ADSP674の場合

これら全てを削除して下さい。

5.00.00版以降の場合

アンインストールは、Windowsのコントロールパネルにある、「アプリケーションの追加と削除」を起動し、「MATLAB 対応ライブラリ for MATLAB6.x Ver x.xx.xx」の項目を選択し、「追加と削除ボタン」で削除します。

補足：上記のx.xx.xxはMATLAB対応ライブラリのバージョンを表します。

3.4 MATLAB対応ライブラリのインストール

最後に本製品をインストールします。

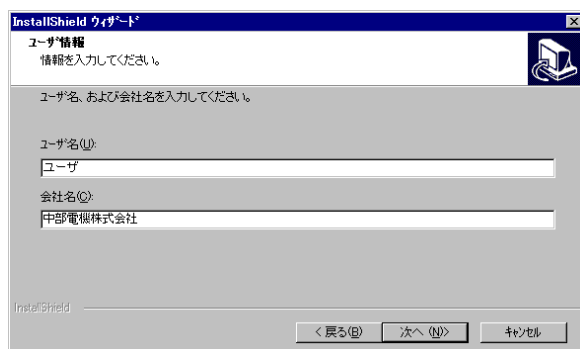
CD-ROMをCDドライブに挿入するとインストーラが自動的に起動します。起動しない場合は、エクスプローラ等でCDの¥Setupを起動します。

【1】ようこそ



この画面が出たら、**次へ(N)** ボタンを押してください。

【2】ユーザ情報の入力



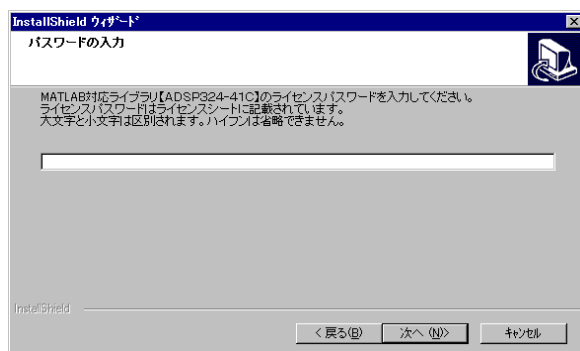
ユーザ情報を入力し**次へ(N)**ボタンを押してください。

【3】購入製品の確認



購入製品を確認し、チェックしてください。

【4】M L L I Bのライセンスコードの入力



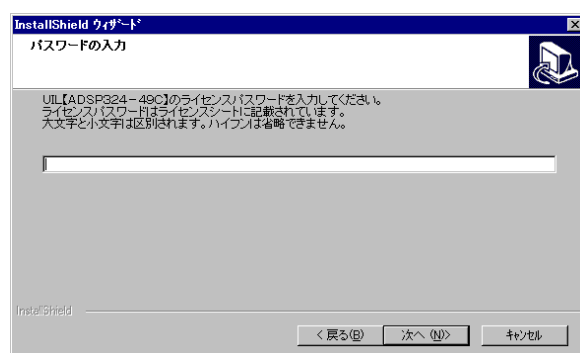
次に、“M L L I Bのライセンスコードの入力” 画面が出ます。

MATLAB対応ライブラリ（ADSP324-41C、ADSP404-241C、ADSP674-341C）のライセンスシートに記載されているパスワードを入力してください。

パスワードの大文字小文字は区別されます。正確に入力してください。また4桁毎に入っている — 記号も省略出来ません。

正しく入力したら、**次へ(N)** ボタンを押してください。

【5】U I Lのパスワードの入力



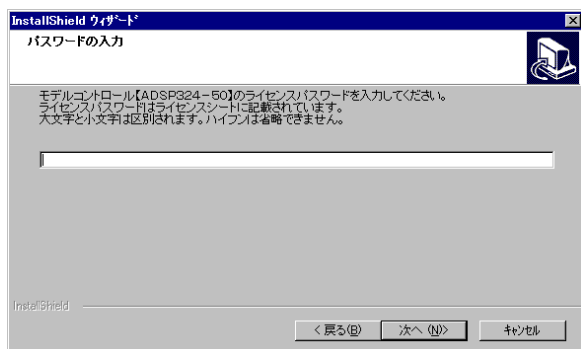
U I Lを購入された場合のみ、“U I Lのパスワードの入力” 画面が出ます。

MATLAB対応U I L（ADSP324-49C、ADSP404-249C、ADSP674-349C）のライセンスシートに記載されているパスワードを入力してください。

パスワードの大文字小文字は区別されます。正確に入力してください。また4桁毎に入っている — 記号も省略出来ません。

正しく入力したら、**次へ(N)** ボタンを押してください。

【6】モデルコントロールのパスワードの入力



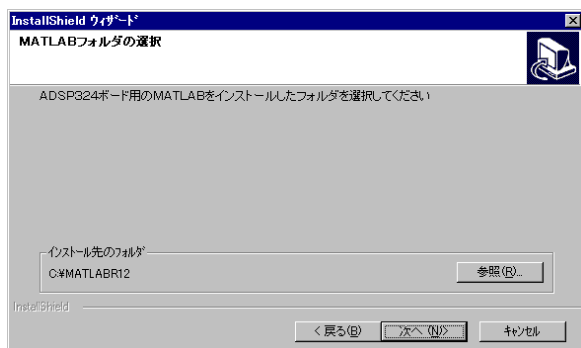
モデルコントロールを購入された場合のみ、“モデルコントロールのパスワードの入力”画面が出ます。

MATLAB 対応モデルコントロール（ADSP324-50、ADSP404-250、ADSP674-350）のライセンスシートに記載されているパスワードを入力してください。

パスワードの大文字小文字は区別されます。正確に入力してください。また 4 桁毎に入っている - 記号も省略出来ません。

正しく入力したら、**次へ(N)** ボタンを押してください。

【7】MATLABのフォルダーの選択

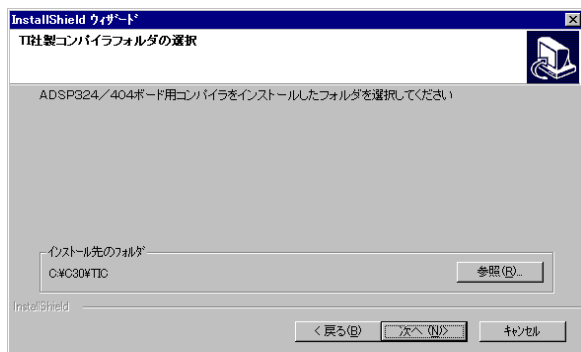


MATLABがインストールされているディレクトリ（フォルダー）を指定します。

MATLABをインストールしたディレクトリ（フォルダー）が表示されているディレクトリと異なる場合は、**参照(R)** ボタンを押しそのディレクトリ名を設定してください。

MATLABのディレクトリーが正しければ **次へ(N)** ボタンを押します。

【8】Cコンパイラのフォルダーの選択



続いて、DSPの開発環境（TI社製Cコンパイラ）をインストールしたフォルダーを指定します。

DSP用Cコンパイラをインストールしたフォルダーが表示と異なる場合は、**参照(R)** ボタンを押し、そのフォルダー名を設定してください。

DSPの開発環境のフォルダーが正しければ、**次へ(N)** ボタンを押してください。

【9】プログラムフォルダーの選択

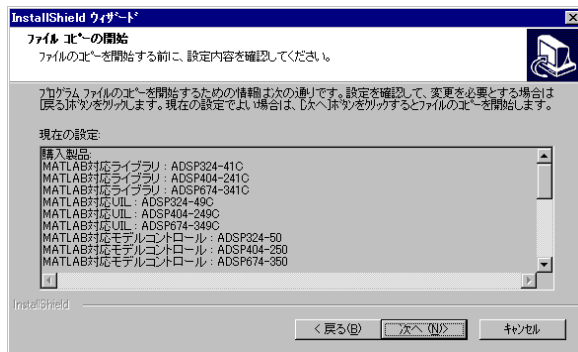


次に、ショートカットを追加するプログラムフォルダーの選択をします。

表示されているフォルダー以外のフォルダーにショートカットを追加する場合は、そのフォルダー名を選択又は入力します。

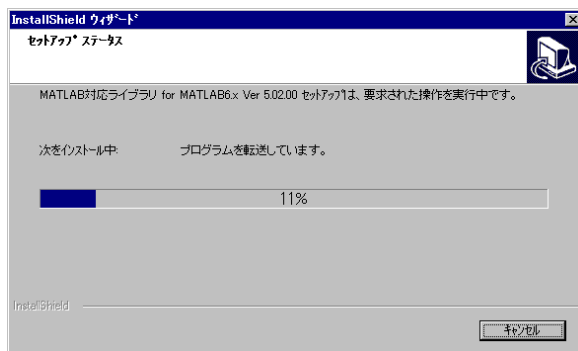
内容を確認したら、**次へ(N)** ボタンを押してください。

【10】ファイルコピーの開始



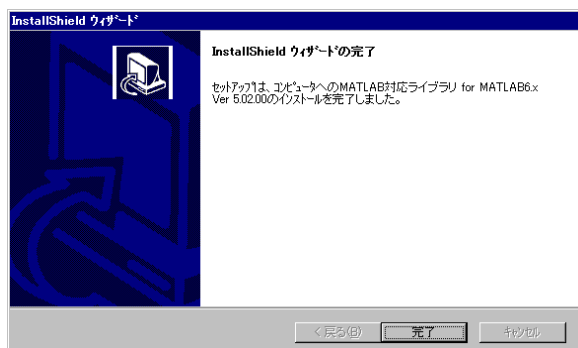
ファイルコピーの開始画面が表示されます。表示内容を確認してください。
内容が誤っているようであれば、**戻る(B)** ボタンで元に戻って再設定できます。
内容を確認したら、**次へ(N)** ボタンを押してください。

【11】セットアップの実行



セットアップステータスが表示され、ファイルのコピーが自動的に行われます。
画面の表示及びプログレスバーがインストール作業の進行と共に変化します。

【12】ウィザードの完了



最後に、“InstallShield ウィザードの完了”、が表示されます。
完了ボタンを押してください。これでインストールは完了です。

4 異なる種類のDSPの同時利用

MATLAB 対応ライブラリ／MATLAB 対応U I L／モデルコントロールは、以下の表に示す種類のDSPボードに対応しています。

D S P ボード	MATLAB 対応L I B	MATLAB 対応U I L	モデルコントロール
ADSP324-00A	ADSP324-41C	ADSP324-49C	ADSP324-50
ADSP404-00	ADSP404-241C	ADSP404-249C	ADSP404-250
ADSP674-00	ADSP674-341C	ADSP674-349C	ADSP674-350

同じ種類のDSPは1つのML L I B／ML U I L／モデルコントロールで動作させる事が出来ます。

2種類以上の異なる種類のDSPを使用するは、それぞれのDSP用のソフトウェアをインストールする必要があります。その場合の注意点について述べます。

複数の製品をインストールする場合以下の注意が必要です。

- ◎MATLABを複数それぞれのDSP用にインストールしてください。1つのMATLAB上に2種類以上のDSP用の環境をインストールする事は出来ません。
- ◎作業フォルダーを区別してください。同じフォルダーで異なるDSP用のコードを作成する事は出来ません。
- ◎ブロック線図の共用は可能です。同じ版のMATLAB間であれば、例えばADSP324-00A用に作成したブロック線図を、ADSP674-00用のフォルダーにコピーし、そこでADSP674-00用のコードを作成する事は可能です。
- ◎コンパイラの共用は ADSP324-00A と ADSP404-00 との間では同じコンパイラを共用できます。コンパイラは個別にインストールする必要はありません。

5 5.02.00 版以降にADSP324-00Aを使用する際の注意事項

ADSP324-00Aの場合にリアルタイムモニタ、UIL、モデルコントロールで使用するDLL（ダイナミック・リンク・ライブラリ）はUSB接続との互換性を重視し、ISA接続でのパソコン-DSP間の通信速度が低下しています。DSPの動作速度には関係ありませんが、過去にISA接続で使用したパソコン側（RTWでのモニタ、UILを使用したVBプログラムなど）の動作速度が間に合わない場合はISA専用DLL（前バージョンのもの）を使用して下さい。今後、新規に作成する場合は共用DLLを使用して下さい。切り替え方法は スタート～プログラム～MATLAB対応ライブラリ～DLL切替～「ISA専用DLLへ切り替え」をクリックして下さい。共用DLLへ戻す場合は上記同様～「共用DLLへ切り替え（標準）」をクリックして下さい。

USB接続にするには別売のUSB接続ボード ADSP324-146 が必要です。

この作業は WindowsNT/2000/XP の場合は Administrator 権限で行って下さい

この問題はスタンドアローン時には関係ありません。

ADSP404-00はUSB接続には未対応です。

ADSP674-00は当初より互換性があります。

接続に関する詳細は ADSP324-00A スタートアップガイド、ADSP674-00 スタートアップガイドを参照して下さい。

最新版のマニュアル、サポートソフトはホームページより無料でダウンロードできます。

中部電機ホームページURL(<http://www.chubu-el.co.jp>)

6 MATLAB対応ライブラリ実行環境について

コンパイラのバージョンによって、コンパイラの実行に必要な以下の環境変数の設定が動的に実行するようになっていきます。この設定をする実行ファイルは”コンパイラのインストールフォルダ¥dosrun.bat”にあります。MATLABを起動するにあたって必要に応じて先にこのBATファイルを実行し環境設定をしてください。

サンプル)

- ① 起動用BATファイルを作成する。

BATファイル内容(コンパイラがc:¥CCStudio_v3.1, MATLABがc:¥matlab71にインストールされている時)・・・

```
call c:¥CCStudio_v3.1¥dosrun.bat  
call c:¥matlab71¥bin¥win32¥matlab.exe  
exit
```

- ② ショートカットにより①で作成したBATファイルを起動する。

BATファイルが c:¥bat¥execML71.bat の場合



○MATLAB, SIMULINK, Real-Time Workshop, は米国 The Math Works Inc. の製品です。

○コンパイラ CPL320FLT-MS は Texas Instruments 社の製品です。

◎本書の内容は著作権法により保護されています。本書の内容を無断で複製・転載することは出来ません。

◎本書及び本製品の内容は、製品の向上等の為予告なく変更される場合があります。悪しからずご了承頂けますようお願いいたします。

◎本書・本製品に関するご質問・ご要望は下記までお願いいたします。

◎落丁・乱丁はお取り替えいたします。

MATLAB 6.x 版 対応
MATLAB 対応ライブラリ
MATLAB 対応 U I L
モデルコントロール
5. x . x 版 インストールガイド

第 1 版 2001. 03. 21

第 7 版 2003. 12. 04

中部電機株式会社 技術部 C S G

TEL 0532-61-9566

FAX 0532-63-1081

E-mail csg@chubu-el.co.jp

URL <http://www.chubu-el.co.jp>

R Tモデル自動生成

<< 目次 >>

1	概要	2
2	使用上の注意事項	2
2.1	バージョンアップ後のコード生成	2
2.2	モデル名称の制約	2
2.3	D S P 用コンパイラ版番号	3
3	S I M U L I N K ブロックの制約	4
3.1	simulink Sources ～ Pulse Generator	4
3.2	simulink Sources ～ Signal Generator	4
3.3	simulink Sources ～ From File	4
3.4	simulink Sources ～ From Workspace	4
3.5	simulink Sinks ～	4
3.6	simulink Nonlinear ～ MATLAB Fcn	5
3.7	simulink Nonlinear ～ Algebraic Constraint	5
3.8	simulink Nonlinear ～ Fcn	5
3.9	Simulink Extras ～ Additional Discrete ～ Discrete Zero-Pole	6
3.10	Simulink Extras ～ Flip Flops ～ D Flip Flop	6
4	MATLAB 6.x での作成手順	7
4.1	作成に必要なファイル	7
4.2	オプションの設定	7
4.3	実行形式の選択	10
4.4	作成されるファイル	12
4.5	実行手順（リアルタイムモニタの場合）	13
4.6	実行手順（自動実行の場合）	13
5	ステップサイズについて	14
5.1	SIMULINK との関係	14
5.2	ステップサイズのデフォルト	14
6	速度向上について	15
6.1	実行速度の相違	15
6.2	高速化するには	15
6.3	M A P 変更ツール	15
6.4	メモリー不足エラー例	18

1 概要

MATLAB のツールボックスのひとつ、Real-Time Workshop を用い、C コードを生成し、DSP ボード上で実行することができます。また、本製品のリアルタイムモニタ又を使用すれば、ただちに DSP 上で実行出来、さらに、その結果をファイルに残すことができます。

ここでは、SIMULINK 上でのシミュレーションのモデル作成は完了しているものとし、これを元にリアルタイムモデルを生成する手順について説明します。

2 使用上の注意事項

2.1 バージョンアップ後のコード生成

本製品のバージョンアップを行った場合、SIMULINK ブロック線図から RT モデルを作成する際に、一旦、モデルと同じフォルダーにあるサブフォルダー “ブロック線図名_grt_rtw” を削除してからコード生成を行ってください。

2.2 モデル名称の制約

RT モデルの名称 (ブロック線図のファイル名) には、以下に示す制約があります。モデル命名の際には、充分注意をしてください。

- 名称の先頭は、英字で始めてください。

良い例 : a3bc.mdl

悪い例 : 3abc.mdl (理由=数字で始まっている)

- 名称で使う文字は英数字と `_` 記号のみとしてください。

良い例 : a32_abc.mdl

悪い例 : a(#1).mdl (理由=かっこ、#等の記号を使っている)

- 英字の大文字と小文字は同じものとして取り扱われます。

2.3 DSP用コンパイラ版番号

使用出来るDSP用コンパイラは以下の通りです。

◎ CPL3206X-MS 1.1 版(Code Composer Studio 対応版)

◎ CCS3206X-MS 1.1 版

3 SIMULINKブロックの制約

SIMULINK 標準のブロックライブラリの一部は、R Tモデルに於いて利用出来ないか、又は利用上の注意が必要です。この章ではこれらについて解説しています。

3.1 simulink Sources ～ Pulse Generator

3.2 simulink Sources ～ Signal Generator

Pulse Generator／Signal Generator は、D S Pが取り扱う時刻変数の精度の関係で、実際の時間（又は SIMULINK で実行した仮想時刻）より、1 ステップサイズ分、前後する場合があります。

D S Pは変数を全て単精度実数（32Bit=8E24）で取り扱います。この精度は、1 0進数換算で有効桁数7桁程度です。例えば 0.003 はD S Pでは、0.00299999979324 となります。この誤差は避けられません。

3.3 simulink Sources ～ From File

R T Wにてコード生成をした時点のファイルの内容が、生成されるCソースコードにデータテーブルの形で組み込まれます。組み込まれたデータはその状態で実行形式コード内に固定され、R Tモデルの実行時点のファイル内容とは無関係となります。

ファイル内容を変更し、その結果をR Tモデルに反映させるには、再度R T Wでコード生成が必要です。

データが実行コードに組み込まれる関係上、その長さは、D S Pのメモリー容量を超える事は出来ません。

3.4 simulink Sources ～ From Workspace

R T Wにてコード生成をした時点のワークスペース変数の内容が、生成されるCソースコードにデータテーブルの形で組み込まれます。組み込まれたデータはその状態で実行形式コード内に固定され、R Tモデルの実行時点のワークスペース内容とは無関係となります。ワークスペース内容を変更し、その結果をR Tモデルに反映させるには、再度R T Wでコード生成が必要です。

データが実行コードに組み込まれる関係上、その長さは、D S Pのメモリー容量を超える事は出来ません。

3.5 simulink Sinks ～

Sinks グループの Stop Simulation を除く他のブロック（Scope 等）は、R T Wでは利用できません。（MATLAB4.2c では Stop Simulation に加え Scope と ToWorkspace は利用可能）。

Scope はR T M O Nの変数モニター機構を使って代用が可能です。具体的には、Scope へ

接続されている信号の出力元のブロック名称を指標して変数表示を行います。

3.6 simulink Nonlinear ～ MATLAB Fcn

R T Wでは利用出来ません。sin 、 cos 等、一部の関数は、Fcn ブロックで代用が可能です。

3.7 simulink Nonlinear ～ Algebraic Constraint

R T Wでは利用出来ません。

3.8 simulink Nonlinear ～ Fcn

利用は可能ですが、式の定義に注意が必要です。

- (1) C 言語の標準ランタイムライブラリでサポートされていない関数は利用出来ません。
利用出来る関数／演算子は、以下の通りです。

利用出来る関数	C 言語展開	利用出来る演算子	C 言語展開
sin	sin	>	>
cos	cos	<	<
tan	tan	= =	= =
asin	asin	!=	!=
acos	acos	>=	>=
atan	atan	<=	<=
atan2	atan2	&&	&&
sinh	sinh		
cosh	cosh	()	()
tanh	tanh	+	+
pow	pow	-	-
exp	exp	/	/
log	log	*	*
log10	log10	!	!
sqrt	sqrt		
floor	floor		
ceil	ceil		
abs	fabs		
mod	fmod		
rem	fmod		
sgn	sgn (注 1)		
hypot	hypot		

注 1：標準ランタイムには含まれませんが、R T W用拡張ライブラリが対応しています。

- (2) 式中に定数を記述する場合で、分数の分子・分母に整数を置く場合は、整数であっても少数形式で記述してください。例えば、1／2を記述する場合は、
1／2.0 又は、1.0／2、又は、1.0／2.0 等としてください。

R T WではF c nブロックの Expression に記述された式を、殆どそのままC言語ソ

ースに展開します。よって単に $1/2$ と記述すると、C 言語の文法規約上、整数計算が行われ、 $1/2$ は少数以下切り捨てられ 本来 0.5 が欲しい所が、= 0 となってしまう、計算結果に重大な誤りを招きます。

3.9 Simulink Extras ～ Additional Discrete ～ Discrete Zero-Pole

Discrete Zero-Pole (with initial states) は、Simulink の純正ブロック自体が Bug を孕んでいます。以下の手順で対策してください。

- (1) MATLAB Command window から Simulink を開く。
- (2) Blocksets & Toolboxes をマウス左ダブルクリックし、
Library: Blocksets_and_Toolboxes を開く。
- (3) 同 Simulink Extras をマウス左ダブルクリックし、Library: Simulink_Extras を開く。
- (4) 同 Additional Discrete をマウス左ダブルクリックし、
Library: Simulink_Extras/Additional Discrete を開く
- (5) 同 Discrete Zero-Pole (with initial states) をマウスで選択する。
- (6) メニューから Edit ～ Unlock Library を選択する。
- (7) さらに、Edit ～ Look Under Mask を選択し、Library: Simulink_Extras/Additional Discrete/Discrete Zero-Pole (with initial states) を開く。
- (8) 同、Discrete State Space をダブルクリックし、Discrete State Space ダイアログを開く。
- (9) エディットボックス D: の内容が Dd となっているので、これを D に変更する。
- (10) 同ダイアログの Close ボタンを押す。
- (11) 先の (7) で開いたウィンドウを閉じる。
- (12) 先の (4) で開いたウィンドウを閉じる。
- (13) (3) で開いた Library: Simulink_Extras のメニューから、File ～ Save で変更結果を保存する。

3.10 Simulink Extras ～ Flip Flops ～ D Flip Flop

D 入力=1 の状態で、CLK の立ち上がり時、!CLR=0 が同時に発生する状況では、SIMULINK の実行結果と RT モデルの実行結果に相違が現れます。

SIMULINK では CLK が優先され Q=1 となり、RT モデルでは、!CLR が優先され Q=0 となります。

4 MATLAB 6.x での作成手順

4.1 作成に必要なファイル

リアルタイムモデルの作成に必要なファイルは、リアルタイムモデルを記述したブロック線図です。ブロック線図名を **EXAMPLE** とするとブロック線図をファイルに保存すると **EXSAMPLE.MDL** というファイルとして保存されます。

4.2 オプションの設定

モデルのブロック線図を開き、ブロック線図のメニューから シミュレーション(S) の シミュレーションパラメータ(M)をマウス左ボタンでクリックし、選択します。すると、**Simulation Parameter** ダイアログウィンドウが開きます。(メニュー ツール(O) の **Real-TimeWorkShop(W)** の **Option(O)** で開いたものと同じダイアログです)。リアルタイムモデルのDSP内での実行条件(実行時可変な一部パラメータを除く)の設定はここで行ないます。

Simulation parameters: ダイアログの ソルバ タブをマウス左クリックします。

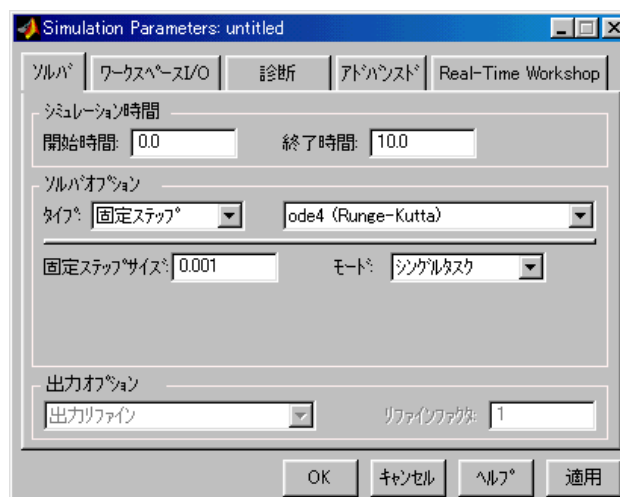
ソルバオプションを固定ステップに変更します。これはリアルタイムモデルでは一定周期での計算しか行えない事を意味しています。

続いて、その右のリストボックスの中から積分則を選択します。モデルが連続系の状態変数を持たない場合は、離散(連続状態無し)を選択、それ以外の場合はリアルタイムモデルでデフォルトとして利用したいと思う積分則を選択します。

さらにすぐ下の 固定ステップサイズ: に具体的なステップサイズ値を設定します。単位は秒です。図 4-1 ではステップサイズが 1 mSecに設定されています。ここで設定した積分則とステップサイズは、リアルタイムモニターでその設定をデフォルトとして取り出す事が出来ます。

その右の モード:はシングルタスクを選択してください。

図 4-1 積分アルゴリズム等設定後



次に、Real-Time Workshop タブをマウス左クリックし、以下を確認します。

システムターゲットファイル: chubu.tlc

テンプレート Make ファイル: a674.tmf (Large モデル時 a674b.tmf)

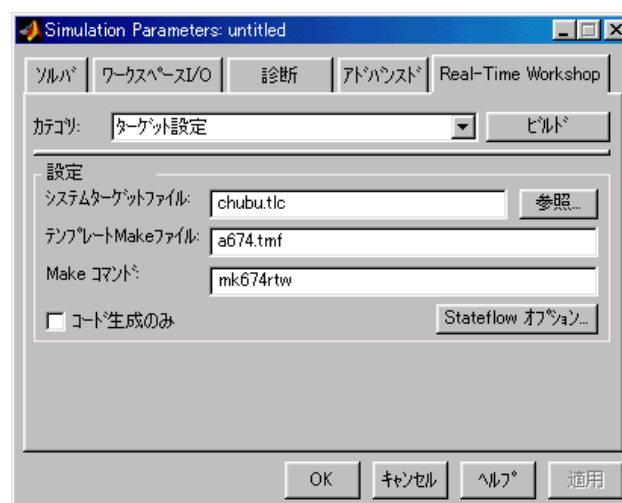
Make コマンド: mk674rtw (Large モデル時 mk674rtwb)

デフォルトは Matlab 対応ライブラリ Ver5.2.0 以降は Large モデルです。

図 4-2 は、これらの設定が終わった例です。

通常、新規に作成したばかりのブロック線図にはこの設定がデフォルトとして設定されています。この設定は、リアルタイムモデル用のタイマードリブンモデルを生成する場合の設定です。それ以外の実行形式モデルの生成方法は 4.3 実行形式の選択 (10p) を参照してください。

図 4-2 R TWオプション (設定後)



設定を変更した場合は **適用** ボタンを押して設定を確定してください。

出来ればこの段階で、上記設定を保存する意味で、一旦モデルを保存する事をお勧めします。ブロック線図メニューから **ファイル(F)** → **名前を付けて保存(A)** 又は、**ファイル(F)** → **上書き保存(S)** を実行します。

コード生成は、同ダイアログの **ビルド** ボタンを押してください。

MATLAB コマンドウィンドウに、#### で始まるメッセージが何行か表示されます。

(一部####以外で始まる行が含まれる場合がありますが、これは異常ではありません。)

これらの行は全て R TW 関連のメッセージです。コード生成に続きコンパイル・リンク作業が行われ、以下が表示されればコード生成は完了です。

Successful completion of RTW build procedure for model: XXXXX

(注 XXXXX は作成するモデルの名称です。)

ブロック線図作成後初めてコード生成をする場合と2回目以後の再度コード生成する場合とでは表示される行数が異なります。これは、ブロック線図変更に無関係のソースファイルは2度目のコンパイル作業を省略し、必要最小限のソースファイルのみコンパイルしている為です。

注) 新規以外で作成されたモデルをコード化する場合、弊社ホームページの Q&A「MATLAB対応ライブラリでよくあるQ&A」を参照してください。

4.3 実行形式の選択

R T WによるD S P用リアルタイムモデルは、その実行方法とモデル計算のきっかけとなるイベントが選択可能です。

実行方法は、リアルタイムモニター又はU I Lによる方法“パラメータ可変モデル”と、自動的に実行開始する“スタンドアロンモデル”が可能です。

パラメータ可変モデル は、ステップサイズ、積分アルゴリズム、ゲインブロック等のパラメータ等が実行時に可変で、データロギング、リアルタイムなモニター等が行えます。未完成のモデルを評価したり、パラメータを種々変更してテストする場合等に有用です。

スタンドアロンモデル は、パラメータ等はR T Wにてコード化される時点の設定で固定され、実行時には可変出来ない代わりに、プログラムをD S Pにロードするとただちに実行が開始されます。これを応用し、ROM化（オプションハードウェアが必要）する事により、電源投入と同時に自動的に動作を開始するスタンドアロンモデルを作る事ができます。

表 4.3-1 実行方法別モデル機能一覧

名称	自動実行	ROM 化	ロギング	パラメータ変更	モニター
パラメータ可変	×	×	○	○	○
スタンドアロン	○	○	×	×	×

モデル計算のきっかけとなるイベントは、タイマーイベント、外部割り込みイベントが選択出来ます。

タイマーイベント は、最も一般的で、D S Pのタイマーを用いて一定周期のタイマーイベントを発生させ、これにより、ブロック線図の計算を行います。この周期がR T Wによるコード生成時の Step size に相当します。

外部割り込みイベント は、ブロック線図の計算を外部からの割り込みイベントをきっかけとして行います。これにより、外部C P Uや外部ハードウェアに同期したモデル計算が可能となります。SIMULINKの可変ステップサイズと似た動作をしますが、SIMULINKの場合は時間刻みの増減の基準が状態変数の微係数の大小であるのに対し、イベントドリブンでは時間刻みが“外部割り込みイベントの発生”に完全に依存します。

実行方法とイベントは、コード生成の際に **R T W Option** ダイアログへの設定で切り替えます。

表 4.3-2 実行形式・イベント切り替え一覧

イベント	実行方法	上段：Template make file
		下段：Make command
タイマー	パラメータ可変	a674.tmf
		mk674rtw
	スタンドアロン	a674.tmf
		mk674rtw ALONE=ON
外部割り込み	パラメータ可変	a674.tmf
		mk674rtw EVENTD=ON
	スタンドアロン	a674.tmf
		mk674rtw EVENTD=ON ALONE=ON

☆ Large モデル時 a674.tmf → a674b.tmf

mk674rtw → mk674rtwb を使用してください。

デフォルトは、Matlab 対応ライブラリ Ver5.2.0 以降は Large モデルです。

4.4 作成されるファイル

Real-Time Workshopで作成作業を行なうと、MATLABのカレントディレクトリに、“ブロック線図名_grt_rtw”という名称のサブディレクトリが生成され、その中に、表 4.4-1に示すファイルが作成されます。作成作業を行なう前のカレントフォルダーに、これらと同名のファイルがあると、上書きにより以前の内容は失われてしまいます。カレントフォルダーにユーザのファイルを置く場合には注意が必要です。

作業は順を追って自動的に行なわれます。最終的に実行に必要なファイルは、EXAMPLE.OUT です。

表 4.4-1 自動生成されるファイル一覧表

番号	生成ファイル名	作成主	備考
1	EXAMPLE.C EXAMPLE.H EXAMPLE_COMMON.H EXAMPLE_EXPORT.H EXAMPLE_PRM.H EXAMPLE_REG.H EXAMPLE.RTW	Real-Time Workshop	主となるソースファイル
2	EXAMPLE.MK EXAMPLE.LNK EXAMPLE.CMD EXSAMPLE.OPT EXSAMPLE.ITB	Buil command	実行ファイルの作成に必要なファイル
3	*.OBJ	CL6X.EXE	オブジェクトファイル
4	EXAMPLE.OUT EXAMPLE.MAP	LNK6X.EXE	実行形式ファイル 番地情報ファイル

注：EXAMPLE は、ユーザが命名したモデルの名称です。

4.5 実行手順（リアルタイムモニタの場合）

リアルタイムモニタの場合は、Windows のプログラクマネージャで、リアルタイムモニタを起動し、これによりロード・実行等を行ないます。この方法は画面上でパラメータや、起動停止等をマニュアルックに操作するので、自動起動等を行なえませんが、最も柔軟性に富んだ方です。

詳細は、リアルタイムモニタの章を参照してください。

4.6 実行手順（自動実行の場合）

自動実行タイプの場合は、以下に示す 2 つのケースについて説明します。

- 1、リアルタイムモニターを用いてサブDSPにロードする場合。
- 2、ROM化する場合。

何れの場合も、リアルタイムモデルはスタンドアロンモデルとして作成されていなければいけません。

リアルタイムモニターを用いてサブDSPにロードする場合 は、コード生成作業は全て完了していますので、後はリアルタイムモニターの“サブDSP”機能でリアルタイムモデルの実行ファイルを指定します。

ROM化する場合 は、DSP 付属ソフトウェアの付属コマンド **UTY674** を用いてROM化します。詳しくは DSP ボードのソフトウェア・ユーザーズ・マニュアルを参照してください。

5 ステップサイズについて

5.1 SIMULINK との関係

R T WによるR Tモデルを生成する場合、実時間モデルでは、可変時間刻みは実現出来ません。実時間モデルでは、過去の時刻にさかのぼって計算をする事が出来ないからです。

可変時間刻み（Variable-Step）みは、微係数を監視して、傾きが小さい場合は時間刻みを粗くして計算省略により時間を短縮すると共に、傾きが大きい場合は時間刻みを細かくして計算精度を維持する意図を併せ持っています。微係数の傾きが大きい場合は自動的に時間刻みが細かくなりますが、固定刻みの場合はこれが行われなない為、誤差が増大する可能性があります。

従って、最終的に実時間モデルを作成する予定がある場合、SIMULINK での評価の段階で、固定刻み（Solver Type を Fixed-Step）にして評価する必要があります。

5.2 ステップサイズのデフォルト

R T M O Nでは、R T Wで生成したR Tモデルのデフォルトステップサイズを取り出す機能を持っています。この機能で取り出せる情報は、モデルの内容により、若干異なります。

この機能で取り出せる情報と、SIMULINK（R T Wとの関係）は、以下の表の様になっています。

モデルの内容	R T M O N のデフォルト
連続時間系のブロックのみ	R T W の Simulation parameter の Fixed-step
連続時間系の離散時間系ブロック	一番小さいサンプルタイム
離散時間系のブロックのみ	(同上)

6 速度向上について

6.1 実行速度の相違

DSPボードADSP674-00では、使用しているDSPチップの特性上、プログラムの配置により、実行速度が異なります。ボード上に、SBSRAM、SDRAMと、チップ内部にオンチップRAMがあります。最も早いのはオンチップRAMですが、チップ上に実装されているオンチップRAMは容量に限りがあります。オンチップRAMより外部のSBSRAMの方が容量が大きいので、大きなプログラムは外部に置きます。

6.2 高速化するには

本ソフトでは、ユーザが作るプログラム（ブロック線図）の大きさが判らない為、標準では全て外部のSBSRAMに実装します。この状態で、希望とする速度が満たせられれば良いのですが、満たせられない場合、プログラム等の配置をオンチップRAMに変更します。

配置を変えるには以下の手順を踏みます。

- 1、MAP変更ツールを使い、配置指定を変更します。
- 2、RTWでコード生成をします。
- 3、メモリーが不足する旨のエラーが出たら1へ戻り、内部に割り付けた領域を外部に戻します。
- 4、エラーが出なければ完了です。

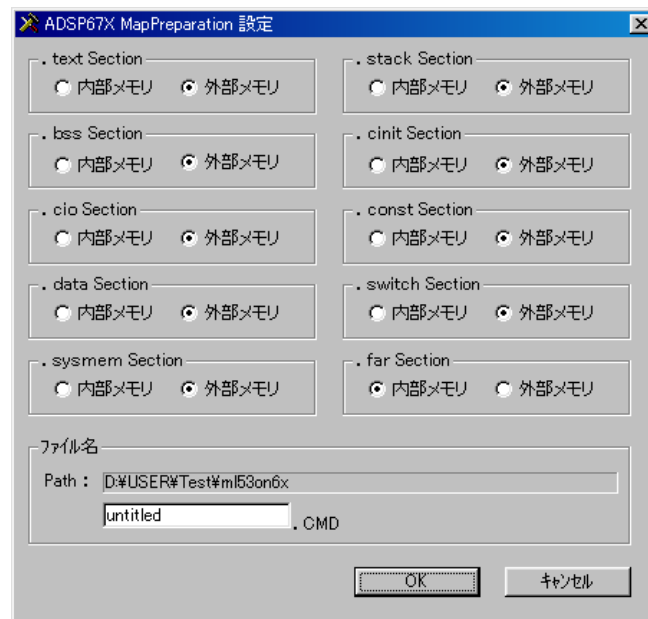
6.3 MAP変更ツール

MAP変更ツールは、I/Oブロックライブラリのブロック線図に有る、“MAPエディットボタン”（下図）を目的とするブロック線図に貼りつけ、これをダブルクリックします。



DSP Memory Map

次のような画面が開きます。



本ソフトではDSPに実装されるユーザプログラムを、それを構成する要素（セクション）毎に分けて、そのセクション毎に配置する場所をチップの内部か外部か選択できる様になっています。

.text Section 等のグループは、個々のセクションを表しています。

ファイル名のグループは、これらの情報を保存するフォルダーとファイル名を表しています。この名称は、MAP変更ツールをブロック線図から呼び出した場合は変更してはいけません。又、このツールにより自動生成されたファイル（上の例では `untitled.cmd`）は、直接変更してはいけません。

配置の変更は、各セクションのグループのラジオボタンで、内部メモリーか外部メモリーかを選択し、**OK** ボタンを押します。設定された内容が書き込まれます。

変更後の内容は次回このツール起動時に表示されます。現在の配置を確認する目的でこのツールを起動した場合は、**キャンセル** ボタンを押して下さい。

設定（又は表示）が終わったらMAT変更ツールを終了してください。MAP変更ツールを起動したままでは、MATLAB／SIMULINKの操作は出来ません。

各要素のおよその意味は以下の表の通りです。

＜セクション名と意味＞

番号	セクション名	意味	効果	割付先
1	.text	プログラムの命令コード	◎	OCPMEM
2	.bss	大域変数、static な局所変数	◎	OCDMEM
3	.cio			
4	.data	ASM 用データ領域		
5	.sysmem	ヒープ領域	◎	
6	.stack	システムスタック	◎	
7	.cinit	C 言語用初期化データ		
8	.const	大域定数、static な局所定数データ		
9	.switch	分岐テーブル	◎	
10	.far	Far 参照用アドレステーブル	◎	

実行時間に大きく影響を及ぼすと思われるものは表の効果の覧に◎を付けてあります。

6.4 メモリー不足エラー例

メモリーが不足すると、RTW によるコード生成の最後の段階で、MATLAB Command Window にエラーメッセージが表示されます。

```
>> cannot allocate .text in OCPMEM (page 0)
>> cannot allocate .bss in OCDMEM (page 0)
>> cannot allocate .cinit in OCDMEM (page 0)
>> cannot allocate .system in OCDMEM (page 0)
>> cannot allocate .far in OCDMEM (page 0)
>> errors in input - untitled.out not built
NMAKE : fatal error U1077: 'e:\ti\C6000\CGTOOLS\bin\lnk6x.exe' : リターン コード
'0x1'
Stop.
```

>>cannot allcate の後ろに割り付けできなかったセクション名(.text 等)とその割付先 (OCDMEM 等) が表示されています。

エラーが出た場合の対策は、表示されたセクション、又は、同じ割付先 (OCPMEM／OCDMEM) の別のセクションを外部メモリーに割り当てます。

R Tモデル自動生成
(MATLAB6.x 対応版)

第 1 版 2001.03.24

第 3 版 2003.06.11

中部電機株式会社 C S 事業部

TEL 0532-61-9566

FAX 0532-63-1081

MATLAB 6.x 用
I/Oデバイスライブラリ

<< 目次 >>

1	ハードウェアのセットアップ	6
1.1	I/Oボードアドレスの設定.....	6
2	MATLAB 6.x版用ライブラリ	7
2.1	ブロックライブラリの概要	7
2.2	ブロックライブラリの開き方.....	7
2.3	ADSP674-00 デバイスドライバブロック	8
2.3.1	概要	8
2.3.2	ホストDP入力デバイスドライバブロック.....	10
2.3.3	ホストDP出力デバイスドライバブロック.....	11
2.3.4	ホストDP入出力デバイスの概念.....	12
2.3.5	SIO入力デバイスドライバブロック.....	13
2.3.6	SIO出力デバイスドライバブロック.....	15
2.3.7	SIO初期化デバイスドライバブロック.....	16
2.3.8	サーキュラーバッファ・デバイスドライバブロック.....	17
2.4	ADSP674-00 On Board PIO ウィンドウ.....	19
2.4.1	ADSP674-00 ビット単位入力デバイスドライバブロック	21
2.4.2	ADSP674-00 ビット単位出力デバイスドライバブロック	22
2.4.3	ADSP674-00 終了時出力指定ビット単位出力デバイスドライバブロック ...	23
2.4.4	ADSP674-00 パラレル入力デバイスドライバブロック	24
2.4.5	ADSP674-00 パラレル出力デバイスドライバブロック	25
2.4.6	ADSP674-00 終了時出力指定パラレル出力デバイスドライバブロック.....	26
2.4.7	ADSP674-00 ポート入力デバイスドライバブロック.....	27
2.4.8	ADSP674-00 ポート出力デバイスドライバブロック.....	28
2.4.9	ADSP674-00 終了時出力指定 ポート出力デバイスドライバブロック.....	29
2.5	ADSP674-00 On Board PIO AD&DA ウィンドウ	30
2.5.1	ADSP674-00 A/D入力デバイスドライバブロック.....	31
2.5.2	ADSP674-00 D/A出力デバイスドライバブロック.....	32
2.5.3	ADSP674-00 終了時出力指定D/A出力デバイスドライバブロック.....	33
2.6	ADSP32X-03/53 デバイスドライバブロック	34
2.6.1	概要	34
2.6.2	チャンネル番号とボードとの対応.....	36
2.6.3	1チャンネルA/D入力デバイスドライバブロック.....	37
2.6.4	多チャンネルA/D入力デバイスドライバブロック.....	38

2.6.5	1チャンネルD/A出力デバイスドライバブロック	39
2.6.6	多チャンネルD/A出力デバイスドライバブロック	40
2.6.7	終了時出力指定 1チャンネルD/A出力デバイスドライバブロック	41
2.6.8	終了時出力指定 多チャンネルD/A出力デバイスドライバブロック	42
2.7	ADSP32X-06/56デバイスドライバブロック	43
2.7.1	概要	43
2.7.2	ポート/ビット番号とボードとの対応	46
2.7.3	ビット単位入力デバイスドライバブロック	48
2.7.4	ポート単位入力デバイスドライバブロック	49
2.7.5	16Bit port単位入力デバイスドライバブロック	50
2.7.6	8Bit port単位入力デバイスドライバブロック	51
2.7.7	ビット単位出力デバイスドライバブロック	52
2.7.8	ポート単位出力デバイスドライバブロック	53
2.7.9	16Bit port単位出力デバイスドライバブロック	54
2.7.10	8Bit port単位出力デバイスドライバブロック	55
2.7.11	複数ビット入力デバイスドライバブロック	56
2.7.12	複数ビット出力デバイスドライバブロック	57
2.7.13	終了時出力指定 ビット単位出力デバイスドライバブロック	58
2.7.14	終了時出力指定 ポート単位出力デバイスドライバブロック	59
2.7.15	終了時出力指定 16Bit port単位出力デバイスドライバブロック	60
2.7.16	終了時出力指定 8Bit port単位出力デバイスドライバブロック	61
2.8	ADSP32X-11/61デバイスドライバブロック	62
2.8.1	概要	62
2.8.2	ビット番号、ポート/チャンネル番号とボードとの対応	64
2.8.3	ビット単位、絶縁入力デバイスドライバブロック	66
2.8.4	ビット単位、絶縁出力デバイスドライバブロック	67
2.8.5	プリセット付きカウンタデバイスドライバブロック	68
2.8.6	カウンタプリセットデバイスドライバブロック	69
2.8.7	終了時出力指定 ビット単位、絶縁出力デバイスドライバブロック	70
2.8.8	複数ビット同時、絶縁入力デバイスドライバブロック	71
2.8.9	複数ビット同時、絶縁出力デバイスドライバブロック	72
2.9	ADSP32X-13/63デバイスドライバブロック	73
2.9.1	概要	73
2.9.2	チャンネル番号とボードとの対応	75
2.9.3	PGAバイパス	76
2.9.4	1チャンネルA/D入力デバイスドライバブロック	77

2.9.5	多チャンネルA/D入力デバイスドライバブロック	78
2.9.6	1チャンネルA/D入力 (MP付き) デバイスドライバブロック	79
2.9.7	多チャンネルA/D入力 (MP付き) デバイスドライバブロック	80
2.9.8	1チャンネルD/A出力デバイスドライバブロック	81
2.9.9	多チャンネルD/A出力デバイスドライバブロック	82
2.9.10	終了時出力指定 1チャンネルD/A出力デバイスドライバブロック ...	83
2.9.11	終了時出力指定 多チャンネルD/A出力デバイスドライバブロック ...	84
2.10	ADSP324-141 デバイスドライバブロック	85
2.10.1	概要	85
2.10.2	チャンネル番号とボードとの対応	87
2.10.3	1チャンネルA/D入力デバイスドライバブロック	88
2.10.4	多チャンネルA/D入力デバイスドライバブロック	89
2.10.5	1チャンネルD/A出力デバイスドライバブロック	90
2.10.6	多チャンネルD/A出力デバイスドライバブロック	91
2.10.7	終了時出力指定 1チャンネルD/A出力デバイスドライバブロック	92
2.10.8	終了時出力指定 多チャンネルD/A出力デバイスドライバブロック ...	93
2.11	ADSP324-143 デバイスドライバブロック	94
2.11.1	概要	94
2.11.2	コネクタとボードとのメモリマップ	95
2.11.3	CN12 Input デバイスドライバブロック	96
2.11.4	CN13 Input デバイスドライバブロック	97
2.11.5	CN12 Output デバイスドライバブロック	98
2.11.6	CN13 Output デバイスドライバブロック	99
2.11.7	DSP間DP通信アービトレーション	100
2.11.8	DSPの前段と後段の意味	101
2.12	ADSP324-145 デバイスドライバブロック	102
2.12.1	概要	102
2.12.2	コネクタとボードとのメモリマップ	103
2.12.3	WatchDog デバイスドライバブロック	105
2.12.4	Pulse output デバイスドライバブロック	106
2.12.5	Counter Input デバイスドライバブロック	107
2.12.6	Counter Input with Change reset flag デバイスドライバブロック ..	108
2.12.7	Counter preset デバイスドライバブロック	109
2.12.8	Capture(S) デバイスドライバブロック	110
2.12.9	Capture(3) デバイスドライバブロック	111
2.12.10	Pwm Output L [S] デバイスドライバブロック	112

2.12.11	Pwm Output L (3) デバイスドライバブロック	113
2.12.12	Pwm Output L (S) with Freq デバイスドライバブロック	114
2.12.13	Pwm Output L (3) with Freq デバイスドライバブロック	115
2.12.14	PI0 bit In [32] デバイスドライバブロック	116
2.12.15	PI0 bit In [16] デバイスドライバブロック	117
2.12.16	PI0 paralell in [32] デバイスドライバブロック	118
2.12.17	PI0 paralell in [16] デバイスドライバブロック	119
2.12.18	PI0 bit out [32] デバイスドライバブロック	120
2.12.19	PI0 bit out [16] デバイスドライバブロック	121
2.12.20	PI0 bit out with Terminate [32] デバイスドライバブロック	122
2.12.21	PI0 bit out with Terminate [16] デバイスドライバブロック	123
2.12.22	PI0 paralell out [32] デバイスドライバブロック	124
2.12.23	PI0 paralell out [16] デバイスドライバブロック	125
2.12.24	PI0 parallel out with Terminate [32] デバイスドライバブロック ..	126
2.12.25	PI0 paralell out with Terminate [16] デバイスドライバブロック ..	127
2.12.26	From GOT デバイスドライバブロック	128
2.12.27	From GOT[8] デバイスドライバブロック	129
2.12.28	From GOT[16] デバイスドライバブロック	130
2.12.29	From GOT32 デバイスドライバブロック	131
2.12.30	From GOT32 [8] デバイスドライバブロック	132
2.12.31	From GOT32 [16] デバイスドライバブロック	133
2.12.32	To GOT デバイスドライバブロック	134
2.12.33	To GOT[8] デバイスドライバブロック	135
2.12.34	To GOT[16] デバイスドライバブロック	136
2.12.35	To GOT32 デバイスドライバブロック	137
2.12.36	To GOT32 [8] デバイスドライバブロック	138
2.12.37	To GOT32 [16] デバイスドライバブロック	139
2.13	TM 3 2 D A デバイスドライバブロック	140
2.13.1	概要	140
2.13.2	1 チャンネル D/A 出力 デバイスドライバブロック	142
2.13.3	多チャンネル D/A 出力 デバイスドライバブロック	143
2.14	Data I/O By Excel ブロック	144
2.14.1	概要	144
2.14.2	From Excel ブロック	145
2.14.3	To Excel ブロック	147

3	他のDSPとの互換性	148
3.1	他のDSPとの互換性	148

1 ハードウェアのセットアップ

1.1 I/Oボードアドレスの設定

DSPボードはI/Oボードの種類を問わず最大4台のI/Oボードを接続できるように設計されています。本製品はそれに対応し、合計4台までの各種I/Oボードを接続した場合を想定しています。各I/Oボードはアドレスで識別しますので、I/Oアドレスの設定を適切に行なわなければいけません。

まず、各I/Oボードに個々のボードを識別する為の番号を、その種類毎に1台目から割り付ける事とします。例えば4種類のボードを各1台ずつ接続したとすると、何れも1台目となります。又、同じ種類のボードを複数接続する場合は、ボードの種類別に番号を飛ばさないように1台目から順に割り付けます。こうして割り付けた各ボードに対して 表 1.1.1 の様にボードアドレスを設定して下さい。ボードアドレスの具体的設定方法は各I/Oボードのハードウェア・ユーザーズ・マニュアルを参照して下さい。

表 1.1.1 ボードアドレス設定表

	I/Oボードの種類			
	ADSP324-03	ADSP324-06	ADSP324-11	ADPS324-13
1 台目	900000h	900080h	900100h	900180h
2 台目	900010h	900090h	900110h	900190h
3 台目	900020h	9000A0h	900120h	9001A0h
4 台目	900030h	9000B0h	900130h	9001B0h

	TM32DA	ADSP324-141	ADSP324-143	ADPS324-145
1 台目	900200h	900280h	908000h	901000h
2 台目	900220h	9002A0h	90C000h	901200h
3 台目	900240h	9002C0h	910000h	901400h
4 台目	900260h	9002E0h	914000h	901600h

2 MATLAB 6.x版用ライブラリ

2.1 ブロックライブラリの概要

本製品のブロックライブラリは、SIMULINKの標準ブロックや、オプションのブロックセットと同様に、ブロック線図に貼り付けて使用します。こうしたブロック線図をReal-Time Workshopでコード化し、DSP上で実行する際にこれらブロックにより、外部I/Oとデータ交換が出来ます。

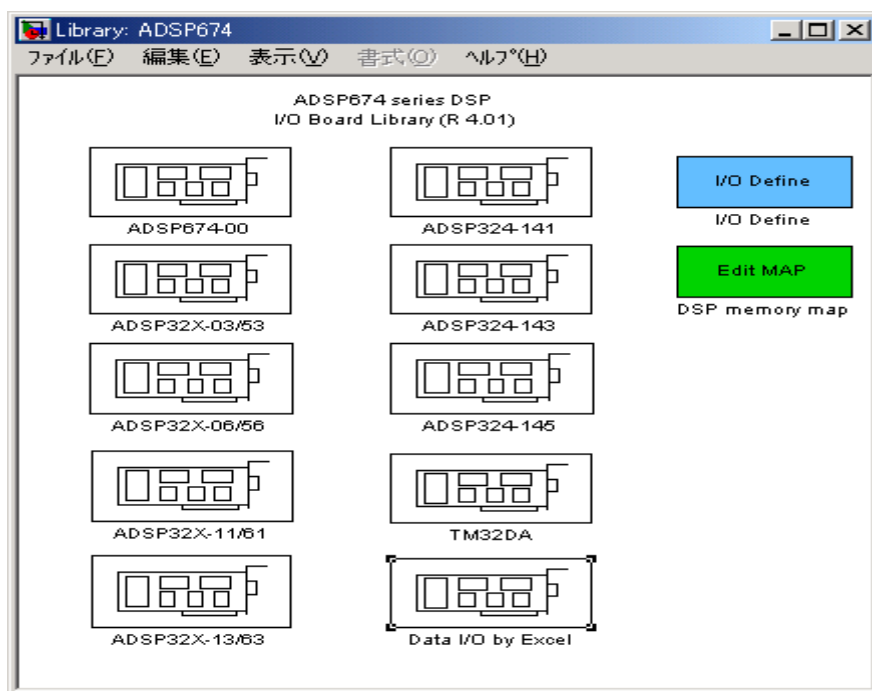
2.2 ブロックライブラリの開き方

ブロックライブラリは、MATLABコマンドウィンドウに次のコマンドを入力します。

```
>> adsp674
```

すると、図 2.2-1 ADSP674 ウィンドウに示すブロックライブラリが開きます。

図 2.2-1 ADSP674 ウィンドウ



ADSP674 ウィンドウには、現在標準で供給されているDSP用のI/Oボード及びDSPボード本体のI/Oに関するデバイスドライバをボード毎に分類して収納してあります。

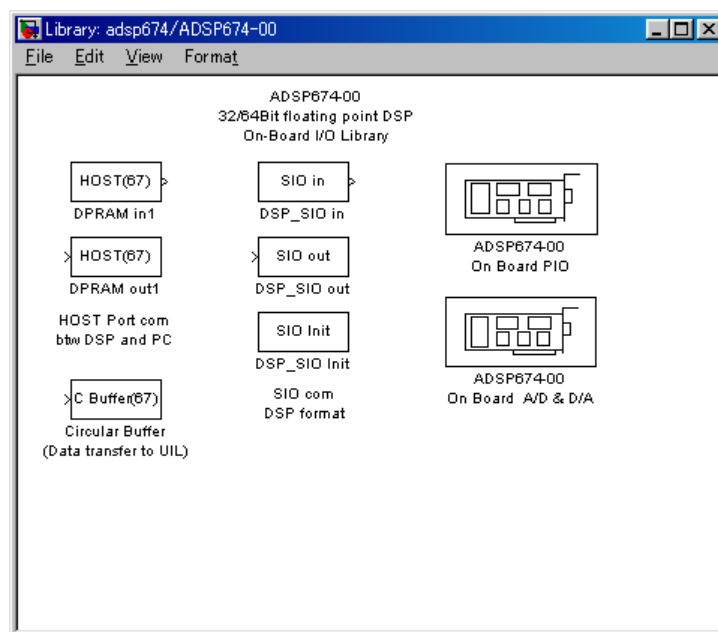
各I/Oボードの型番を記したアイコン（ADSP32X-03/53等）が登録されています。これらアイコンをダブルクリックすることにより、I/Oボードに対応したサブブロックのウィンドウを開きます。また、各モデル毎にI/O Defineブロックが必要です。このブロックを1モデルに1ブロック使用してください。内容は、実装するI/Oボードの枚数を設定するブロックです。このブロックの内容によりI/Oをアクセス制御します。

2.3 ADSP674-00 デバイスドライバブロック

2.3.1 概要

図 2.3-1に、ADSP674-00 (DSPボード) 用デバイスドライバウィンドウを開いた所を示します。

図 2.3-1 ADSP674-00 ウィンドウ



この中には DSPボード上の I/O (DPRAM等含む) とのインターフェースを取るためのデバイスドライバブロックが登録されています。

HOST(67) DSPとホストとの間にあるDPRAMを使い、ホストとの間で非同期にデータ交換をする為のブロックです。

C Buffer(67) DSPのサーキュラーバッファ機構を使った高速データバッファリングによるホストへのデータ送信を行うブロックです。高速に処理させたいブロック線図で使用する事により、データをバッファリングする為の負担を低減させることが出来ます。バッファリングしたデータは、UILを用いたユーザアプリケーションへデータで取り出す事が出来ます。

SIO in/out/init DSPのSIOを使ったシリアル送受信を行うブロックです。DSPボード間での通信に使用します。

ADSP674-00 on Board PIO

D S P ボード上の P I O をアクセスするブロックを収納してある
ADSP674-00 On Bard PIO ウィンドウを開きます。

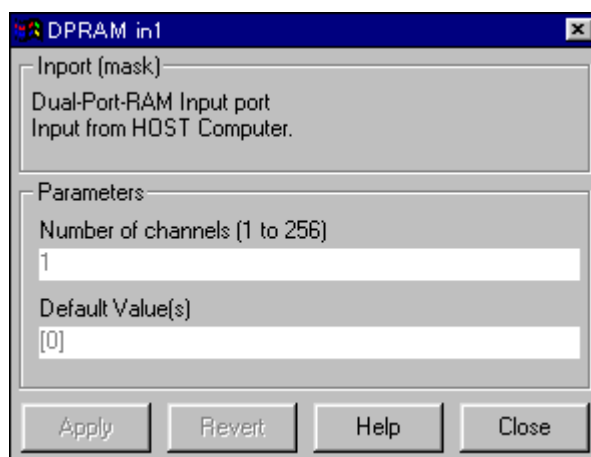
ADSP674-00 on Board AD & DA

D S P ボード上のアナログ I O (A D 及び D A) をアクセスするブ
ロックを収納してある ADSP674-00 On Bard AD & DA ウィンドウを
開きます。

2.3.2 ホストD P入力デバイスドライバブロック

ホストD P入力デバイスドライバブロックは、D S Pとホストとの間にあるD P R A Mを使い、ホストーD S P間で非同期にデータ交換を行う為のブロックです。主に、U I Lを使って作成したホスト側アプリケーションで、比較的ゆっくりと変化をするデータや、時々しか変更しないパラメータ等を、ブロック線図上のデバイスと直接交換する場合等に使用します。通信の概要は、“2.3.4 ホストD P入出力デバイスの概念” を参照してください。

図 2.3-2 HOST DPRAM in



- Number of Channels (1 to 256)

ホストから受け取るデータのチャンネル数を設定します。データはここで設定した幅のベクトルとしてこのブロックから出力されます。

- Default Value(s)

ホストが未だ一度もデータを送信していない段階でこのブロックから出力される値を設定します。第一引数 (Number of Channels) で指定した点数の幅を持つ定数ベクトルを与えます。

このブロックが通信に使用する領域は、D S Pから見たメモリーマップの、3 8 0 0 H番地から連続する2 5 6ワードを使っています。3 8 0 0 H番地がベクトルの最初の要素です。データ形式は、D S Pの実数フォーマットですので、U I L以外のアプリケーションで通信する場合はホスト側でI E E Eとの間のデータ変換が必要です。

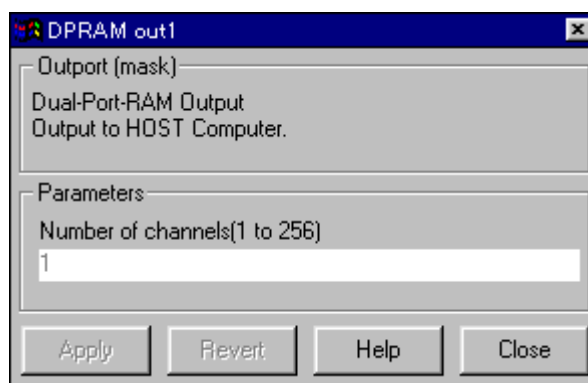
複数のデータを渡す場合、データの同時性は保証されません。つまり、ホストとD S Pのアクセスがほぼ同時に起こった場合、データの前半と後半で1時刻分ずれる可能性があります。

このブロックは1つのブロック線図に1個しか配置できません。

2.3.3 ホストD P出力デバイスドライバーブロック

ホストD P出力デバイスドライバーブロックは、D S Pとホストとの間にあるD P R A Mを使い、ホストーD S P間で非同期にデータ交換を行う為のブロックです。主に、U I Lを使って作成したホスト側アプリケーションで、比較的ゆっくりと変化をするデータや、時々しか変更しないパラメータ等を、ブロック線図上のデバイスと直接交換する場合等に使用します。通信の概要は、“2.3.4 ホストD P入出力デバイスの概念” を参照してください。

図 2.3-3 HOST DPRAM out



- Number of Channels (1 to 256)

ホストへ送るデータのチャンネル数を設定します。データはここで設定した幅のベクトルにまとめてからこのブロックに入力します。

このブロックが通信に使用する領域は、D S Pから見たメモリーマップの、3 C 0 0 H番地から連続する2 5 6ワードを使っています。3 C 0 0 H番地がベクトルの最初の要素です。データ形式は、D S Pの実数フォーマットですので、U I L以外のアプリケーションで通信する場合はホスト側でI E E Eとの間のデータ変換が必要です。

複数のデータを渡す場合、データの同時性は保証されません。つまり、ホストとD S Pのアクセスがほぼ同時に起こった場合、データの前半と後半で1時刻分ずれる可能性があります。

このブロックは1つのブロック線図に1個しか配置できません。

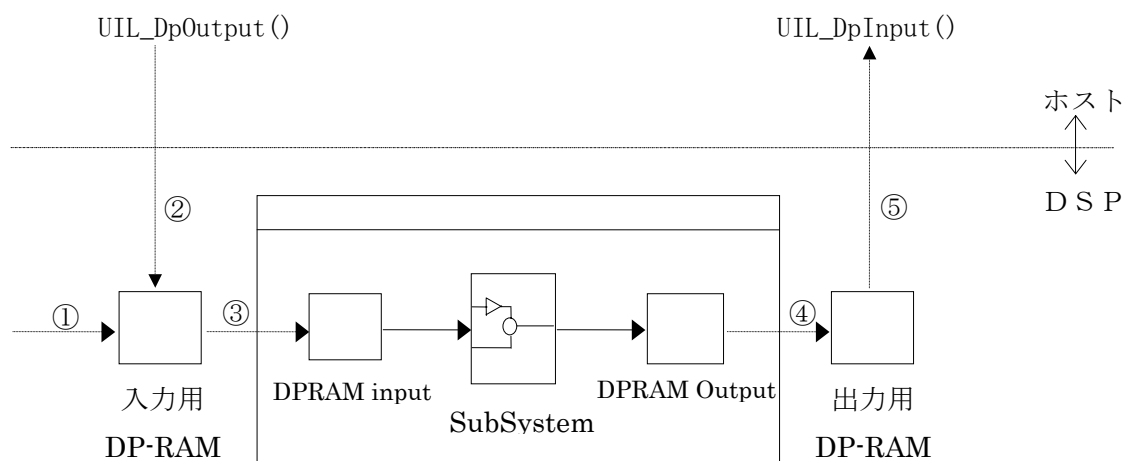
2.3.4 ホストDP入出力デバイスの概念

ホストDPRAMインターフェースは、DSPボードに実装されているデュアルポートRAM（DPRAM^{*1参考}）を使い、ホストのユーザインターフェースライブラリとDSP内のRTWによるリアルタイムモデルとの間で、非同期なデータ交換を可能にするものです。この機能を用いて相互にデータ転送が可能です。

この機能の実装は次の2つの作業で可能となります。

- 1、ホスト側のユーザプログラムにUILライブラリをリンク。
- 2、DSP側のSIMULINKブロック線図にDPライブラリを配置。

次の図はDP-RAMインターフェースのデータ交換の流れを記した図です。



- ①リアルタイムモデルの初期化時にセットされます。
- ②ホストプログラムから関数呼び出し時にセットされます。
- ③RTモデルのステップサイズ毎に読み出しされます。
- ④RTモデルのステップサイズ毎に書き込まれます。
- ⑤ホストプログラムから関数呼び出し時点のデータが読み出されます。

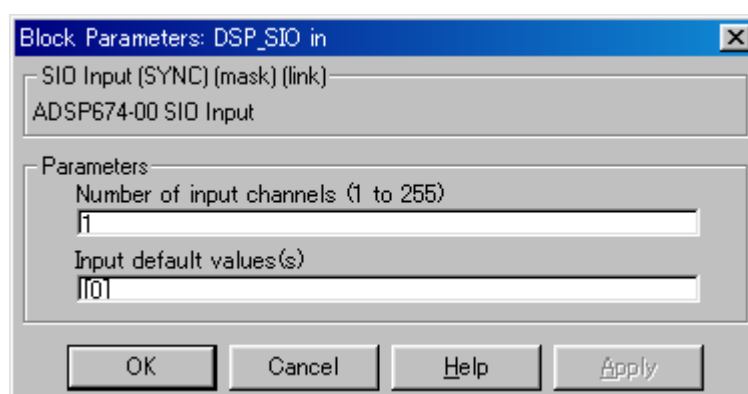
* 1 参考：デュアルポートRAM（DPRAM）はDSPボードに標準実装されている特殊なメモリーで、DSPとホストコンピュータの間で非同期、かつ、双方向に通信が可能です。

2.3.5 SIO入力デバイスドライバブロック

SIO入力デバイスドライバブロックは、DSPのSIOポートを使い、他のDSPボードと通信します。

このブロックを使用するには、SIO初期化ブロックを同じブロック線図に配置する必要があります。

図 2.3-4 SIO in



- Number of input Channels (1 to 255)
受け取るデータのチャンネル数を設定します。データはここで設定した幅のベクトルとしてこのブロックから出力されます。
- Default Value(s)
相手が未だ一度もデータを送信していない段階でこのブロックから出力される値を設定します。第一引数 (Number of Channels) で指定した点数の幅を持つ定数ベクトルを与えます。

このブロックから入力されるデータは、送信元DSPが送信した最も新しいデータです。複数チャンネルのデータを受け取る場合、チャンネル間のデータの同時性は保証されません。

受け取る側と相手側から送信されて来るデータのチャンネル数の整合は、確認されません。もし、相手DSPが送信したデータの点数が Number of input Channels の設定を超える場合、超えた分のデータは破棄されます。逆に、Number of input Channels で指定された点数のデータが送られない場合は、不足するチャンネルからは、Default Value(s) で設定したデフォルト値を出力します。例えば受信側が5チャンネルで送信側が3チャンネルの場合、先頭側の3チャンネルには相手から送られたデータが、そして残りは Default Value(s) で設定した値が入力されます。

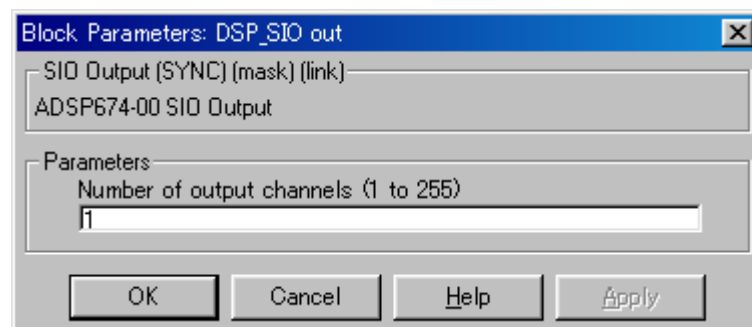
データはベクトルで出力されますので、個々のチャンネルのスカラーデータが必要な場合は、Demux 等で分解して利用します。

2.3.6 S I O出力デバイスドライバブロック

S I O出力デバイスドライバブロックは、DSPのS I Oポートを使い、他のDSP（ADSP67400）に対してデータを送信します。

このブロックを使用するには、S I O初期化ブロックを同じブロック線図に配置する必要があります。

図 2.3-5 SIO out



- Number of output Channels (1 to 255)

送るデータのチャンネル数を設定します。データはここで設定した幅のベクトルとをこのブロックに入力します。

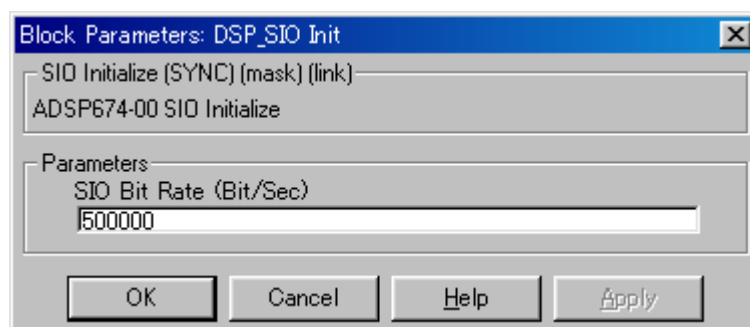
このブロックに出力されるデータは、S I Oポートに自動的に送信されます。

このブロックへの出力データはベクトルで与えます。複数のスカラーデータを出力する場合は、Mux ブロック等でベクトルにまとめてから出力します。

2.3.7 S I O 初期化デバイスドライバブロック

S I O 初期化デバイスドライバブロックは、S I O 入力・出力デバイスドライバブロックを使う場合に、S I O ポートを初期化する為使います。

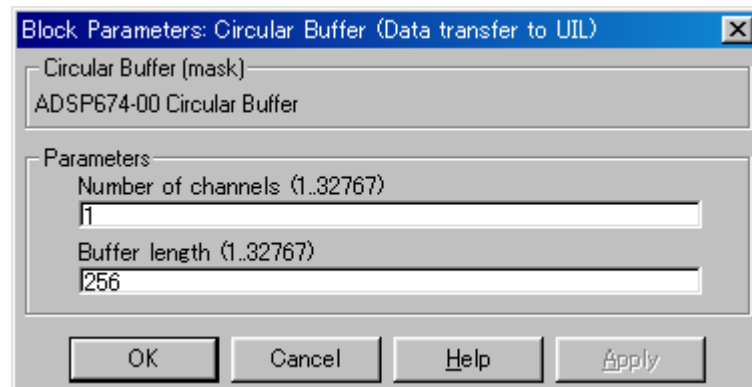
図 2.3-6 SIO Init



2.3.8 サークュラーバッファ・デバイスドライバブロック

D S Pのメモリー内に高速にデータ保管が行えるサーキュラーバッファを構築し、ここにデータを貯える為のブロックです。貯えたデータはU I L（オプション）にて取り出します。

図 2.3-7 C Buffer(67)



- Number of Channels (1 .. 32767)

このブロックへの入力データの点数を指定します。入力データはベクトルで与えますので、このベクトルの幅と同じです。

- Buffer length (1 .. 32767)

各データの保存する長さを指定します。

データバッファはD S P内のメモリーに確保されます。その概ねの長さは、

$$\text{バッファ長さ} = (\text{データ点数} + 1) \times (\text{データの長さ} + \alpha)$$

α はバッファの長さの最小単位が2のN乗である事から、その最小単位に切り上げる為の係数です。この領域は、実行時にヒープ領域から確保されますので、十分な長さのヒープ領域が必要です。

バッファが確保出来たかどうかは、U I Lで確認できます。バッファが確保出来なかった場合は、データの保管は行われません。その場合、データ保管する長さを減らす等の再試行は、ブロック線図の変更とR T Wの再実行により行ってください。

バッファは、サーキュラーバッファと呼ばれる、輪状につながったエンドレスのバッファを構築しています。バッファ内のデータが満杯になった段階で、新たなデータは最古のデータに上書きされ、最新の Buffer length 点のデータが、バッファに残ります。

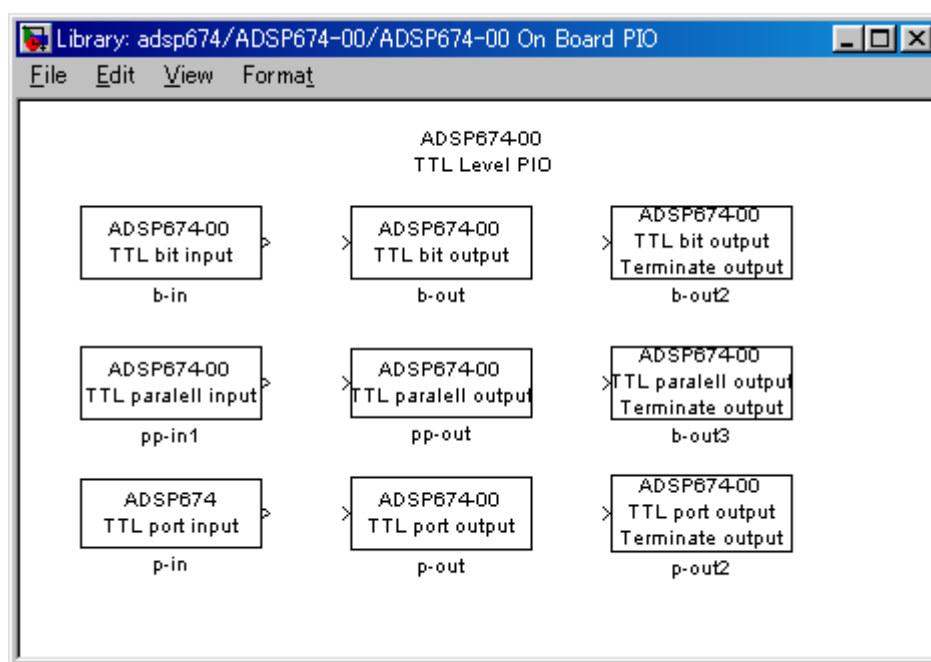
データはU I Lにて取り出します。現在何点のデータが格納されているか等の情報は、U

I Lで取り出すことができます。

2.4 ADSP674-00 On Board PIO ウィンドウ

ADSP674-00 ウィンドウの ADSP674-00 On Board PIO をダブルクリックして開くウィンドウです。ここには DSP ボード上に実装されている PIO ポートをアクセスする IO デバイスドライバが収納されています。

図 2.4-1 ADSP674-00 On Board PIO



- **ADSP674-00 TTL bit input**
1 ビット単位で入力を行うブロックです。
- **ADSP674-00 TTL bit output**
1 ビット単位で出力を行うブロックです。
- **ADSP674-00 TTL bit output Terminate output**
1 ビット単位で出力を行うブロックです。モデルの動作を終了する際に予め指定した状態を出力します。
- **ADSP674-00 TTL parallel input**
全ビットを平行に入力するブロックです。
- **ADSP674-00 TTL parallel output**
全ビットを平行に出力するブロックです。

- **ADSP674-00 TTL parallel output Terminate output**
全ビットをパラレルに出力するブロックです。モデルの動作を終了する際に予め指定した状態を出力します。
- **ADSP674-00 TTL port input**
ポート単位で入力を行うブロックです。
- **ADSP674-00 TTL port output**
ポート単位で出力を行うブロックです。
- **ADSP674-00 TTL port output Terminate output**
ポート単位で入力を行うブロックです。モデルの動作を終了する際に予め指定した状態を出力します。

表 2.4.1 ビット番号とADSP674-00ボードPIOとの対応

入出力信号名	ビット番号 注 1	パラレル 注 2	ポート単位 注 3
EXTIN7/ EXTOUT7	7	最後	2^7
EXTIN6/ EXTOUT6	6	最後の 1 つ前	2^6
EXTIN5/ EXTOUT5	5	・	2^5
EXTIN4/ EXTOUT4	4	・	2^4
EXTIN3/ EXTOUT3	3	・	2^3
EXTIN2/ EXTOUT2	2	・	2^2
EXTIN1/ EXTOUT1	1	先頭の次	2^1
EXTIN0/ EXTOUT0	0	先頭	2^0

注 1 : ビット単位の入出力の場合、各ブロックの初期化ダイアログで指定するビット番号に相当します。

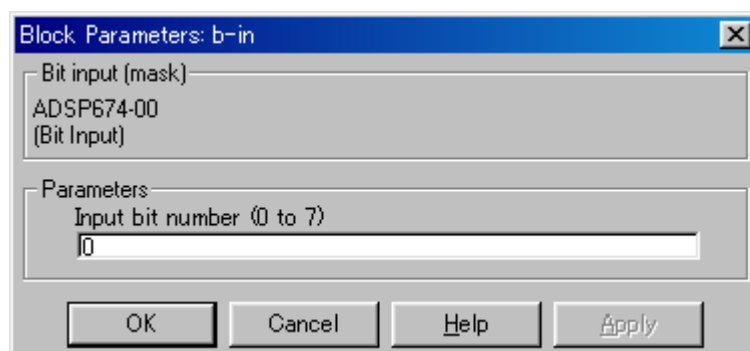
注 2 : パラレル入出力の場合、ブロック線図との間で受け渡すベクトルの各要素と入出力信号との関係は表の通りとなります。終了処理付きの場合の終了値も同様です。

注 3 : ポート単位の入出力の場合、各ブロックとブロック線図との間で受け渡すデータは、各ビットをここに示した重みで加算された値となります。

2.4.1 ADSP674-00 ビット単位入力デバイスドライバブロック

ビット単位入力デバイスドライバブロックは、指定されたポートの指定されたビットを入力し、結果をスカラーで出力します。このブロックから出力される値は、ボードへの入力レベルが LOW の時 0、HIGH の時 1 です。入力信号が接続されていないビットは 1 になります。

図 2.4-2 ADSP674-00 TTL bit input



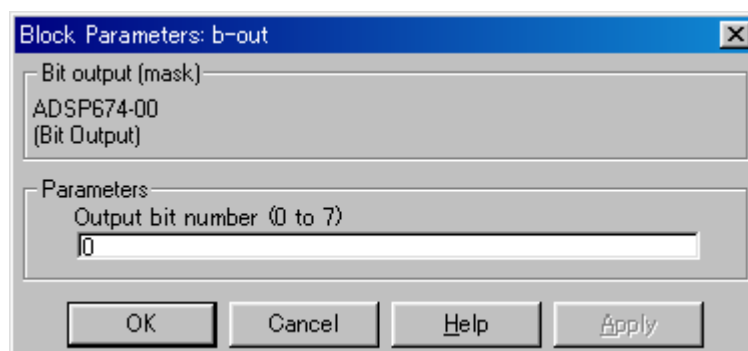
- Input bit number (0 to 7)

入力の対象となるビット番号を指定します。ビット番号とハードウェアとの対応は、表 2.4.1 (20ページ) を参照してください。

2.4.2 ADSP674-00 ビット単位出力デバイスドライバブロック

ビット単位出力デバイスドライバブロックは、スカラーの入力を受取り、指定されたポートの指定されたビットへ出力します。ボードから出力される信号レベルは、このブロックへの入力レベルが 0 の時に LOW、1 の時 HIGH です。指定以外のビットの状態は保存されます。ボードの出力の初期値は全ビット共 LOW です。

図 2.4-3 ADSP674-00 TTL bit output



- Output bit number (0 to 7)

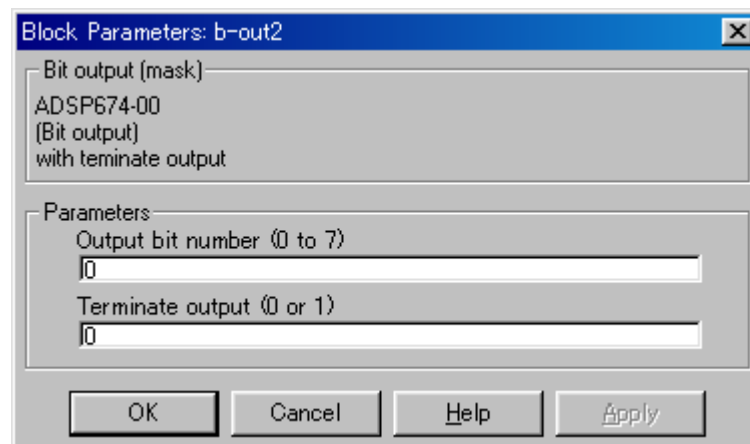
入力の対象となるビット番号を指定します。ビット番号とハードウェアとの対応は、表 2.4.1 (20ページ) を参照してください。

2.4.3 ADSP674-00 終了時出力指定ビット単位出力デバイスドライバブロック

終了時出力指定付きビット単位出力デバイスドライバブロックは、スカラーの入力を受け取り、指定されたポートの指定されたビットへ出力します。ボードから出力される信号レベルは、このブロックへの入力レベルが0の時に LOW、1の時 HIGH です。指定以外のビットの状態は保存されます。ボードの出力の初期値は全ビット共 LOW です。

このブロックは実時間モデルが停止する際に、予め設定した値を出力します。

図 2.4-4 ADSP674-00 TTL bit output with Terminate output

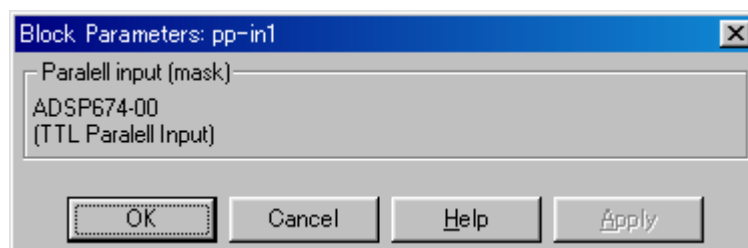


- Output bit number (0 to 7)
入力の対象となるビット番号を指定します。ビット番号とハードウェアとの対応は、表 2.4.1 (20ページ) を参照してください。
- Terminate output (0 or 1)
実時間モデルが停止する際に出力して欲しい状態を設定します。0でLOW, 1でHIGHが出力されます。

2.4.4 ADSP674-00 パラレル入力デバイスドライバブロック

パラレル入力デバイスドライバブロックは、PIO ポートのデータを読み込み、ビットの数と同じ点数のベクトルとして入力します。ボードに入力される信号レベルが LOW の時、これに該当する信号は 0 となります。入力が接続されていない信号の値は不定です。

図 2.4-5 ADSP674-00 TTL parallel input



このブロックの初期化ダイアログには設定すべき項目はありません。

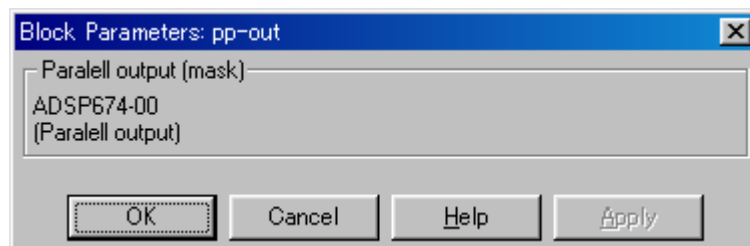
入力信号のビット位置とハードウェアとの対応は、表 2.4.1 (20ページ) を参照してください。

注意：ビット単位出力デバイスドライバブロックとは同時に使用できません。同時に使用した場合、このブロックで出力した結果は、ビット単位出力デバイスドライバブロックにより失われます。

2.4.5 ADSP674-00 パラレル出力デバイスドライバブロック

パラレル出力デバイスドライバブロックは、ブロック線図から出力ポートにのビット数と同じ点数のベクトルを受け取り、個々の要素に対応するビットに出力します。対応する要素の値が 0 の場合は LOW を、それ以外の場合は HIGH を出力します。

図 2.4-6 ADSP674-00 TTL parallel output



このブロックの初期化ダイアログには設定すべき項目はありません。

出力信号のビットの位置とハードウェアとの対応は、表 2.4.1 (20ページ) を参照してください。

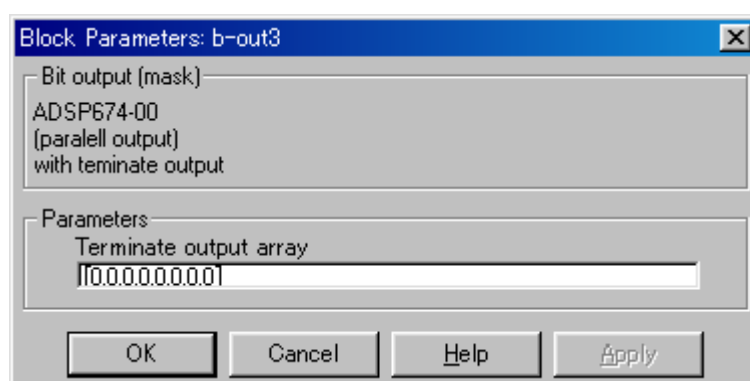
注意：ビット単位出力デバイスドライバブロックとは同時に使用できません。同時に使用した場合、このブロックで出力した結果は、ビット単位出力デバイスドライバブロックにより失われます。

2.4.6 ADSP674-00 終了時出力指定パラレル出力デバイスドライバブロック

終了時出力指定パラレル出力デバイスドライバブロックは、ブロック線図から出力ポートにのビット数と同じ点数のベクトルを受け取り、個々の要素に対応するビットに出力します。対応する要素の値が0の場合はLOWを、それ以外の場合はHIGHを出力します。

このブロックは、実時間モデルが停止する際に予め指定した値を出力します。

図 2.4-7 ADSP674-00 TTL parallel output



- Terminate output array

実時間モデルが停止する際に出力して欲しい値をベクトルで指定します。

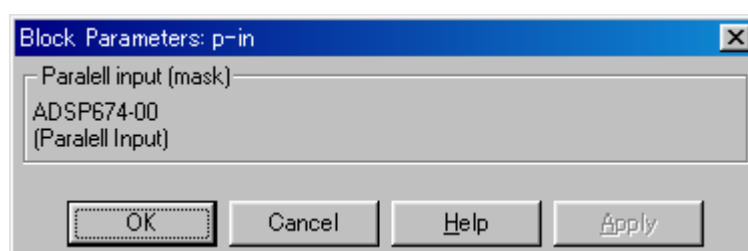
出力信号のビットの位置とハードウェアとの対応は、表 2.4.1 (20ページ) を参照してください。

注意：ビット単位出力デバイスドライバブロックとは同時に使用できません。同時に使用した場合、このブロックで出力した結果は、ビット単位出力デバイスドライバブロックにより失われます。

2.4.7 ADSP674-00 ポート入力デバイスドライバブロック

ポート入力デバイスドライバブロックは、PIO ポートのデータを読み込み、ビットの重みをかけて加算された値を入力します。即ち PIO 入力をバイナリ値として読み込みます。ボードに入力される信号レベルが LOW の時、これに該当するビットは0となります。入力が接続されていない信号の値は不定です。

図 2.4-8 ADSP674-00 TTL port input



このブロックの初期化ダイアログには設定すべき項目はありません。

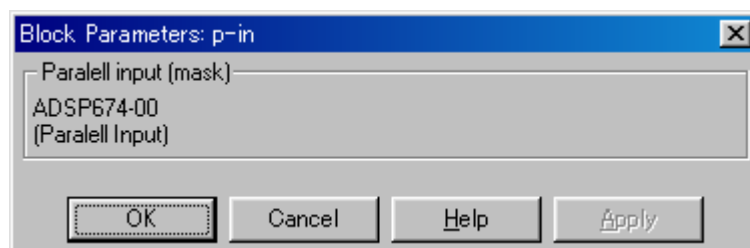
入力信号のビットの重みとハードウェアとの対応は、表 2.4.1 (20ページ) を参照してください。

2.4.8 ADSP674-00 ポート出力デバイスドライバブロック

ポート出力デバイスドライバブロックは、ブロック線図から受け取った値をバイナリ値に変換し、PIO ポートのへ出力します。バイナリに変換後のビットが0の場合、ボードに出力される信号レベルはLOW となります。

ラップラウンドによるクリッピングは行っていません。ブロック線図上で値の上下限を規制してください。

図 2.4-9 ADSP674-00 TTL port output



このブロックの初期化ダイアログには設定すべき項目はありません。

出力信号のビットの重みとハードウェアとの対応は、表 2.4.1 (20ページ) を参照してください。

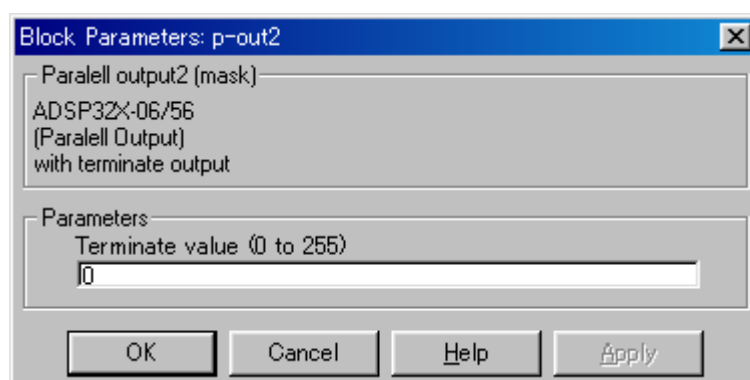
2.4.9 ADSP674-00 終了時出力指定 ポート出力デバイスドライバブロック

終了時出力指定ポート出力デバイスドライバブロックは、ブロック線図から受け取った値をバイナリ値に変換し、PIO ポートのへ出力します。バイナリに変換後のビットが0の場合、ボードに出力される信号レベルはLOW となります。

ラップラウンドによるクリッピングは行っていません。ブロック線図上で値の上下限を規制してください。

このブロックは、実時間モデルが停止する際に予め指定した値を出力します。

図 2.4-10 ADSP674-00 TTL port output terminate output



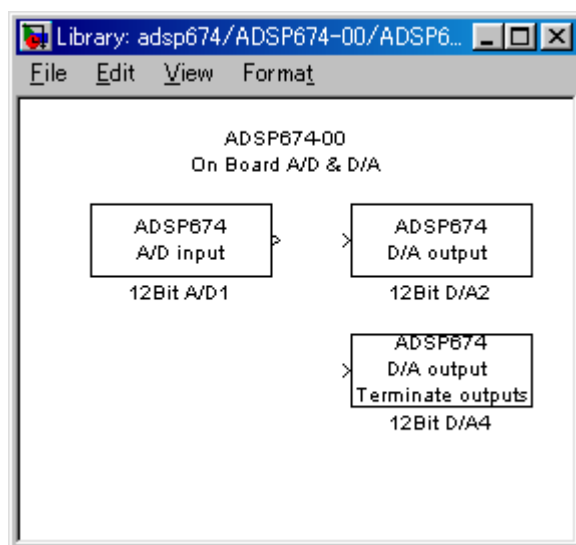
- Terminate value (0 to 255)

実時間モデルが停止する際に出力して欲しい値を10進数で設定します。

出力信号のビットの重みとハードウェアとの対応は、表 2.4.1 (20ページ) を参照してください。

2.5 ADSP674-00 On Board PIO AD&DA ウィンドウ
ADSP674-00 ウィンドウの ADSP674-00 On Board AD & DA をダブルクリックして開くウィンドウです。ここには DSP ボード上に実装されている AD や DA ポートをアクセスする IO デバイスドライバーが収納されています。

図 2.5-1 ADSP674-00 On Board AD & DA

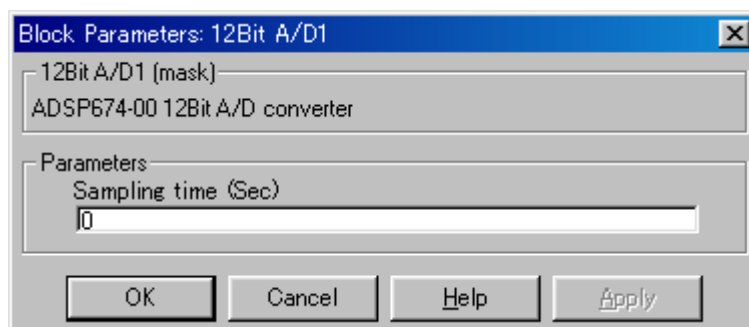


- **ADSP674-00 AD input**
AD から入力を行うブロックです。
- **ADSP674-00 DA output**
DA への出力を行うブロックです。
- **ADSP674-00 DA output Terminate output**
DA への出力を行うブロックです。モデルの動作を終了する際に予め指定した状態を出力します。

2.5.1 ADSP674-00 A/D入力デバイスドライバブロック

A/D入力デバイスドライバブロックは、DSP ボード上のA/Dから一括してデータを入力します。入力の対象となるA/Dチャンネルは、このブロックからの出力は、2の幅を持つベクトルとなります。ベクトルの最初の要素がチャンネル番号0からのデータです。このブロックからの出力データの単位は電圧値[V]です。センサー等の物理単位への変換はデータ入力後 Gain ブロックを使って行なってください。

図 2.5-2 ADSP674-00 A/D input



- Sampling time (Sec)

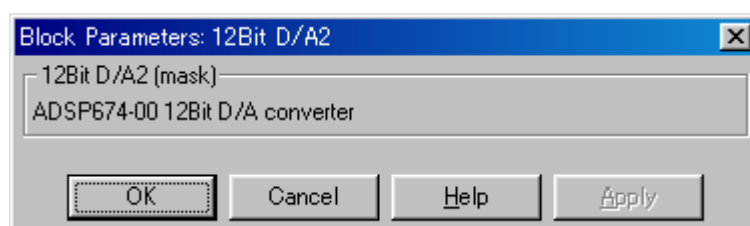
サンプリングの時間間隔を指定します。単位は秒です。ここで指定したサンプリングタイムと実時間モデルのステップサイズが異なる場合は、ここで指定されたサンプリングタイムでA/D変換が行われます。特殊な場合として、ここに0を入力すると、実時間モデルのステップサイズ毎にA/D変換が行われます。

2.5.2 ADSP674-00 D/A出力デバイスドライバブロック

D/A出力デバイスドライバブロックは、幅2のベクトルデータを受取り、D/Aへ一括して出力します。このブロックへの入力ベクトルの幅は必ず2でなければいけません。このブロックへ入力されるベクトルの、最初の要素がチャンネル番号0のD/Aへ出力されます。

このブロックへの入力データの単位は電圧値[V]です。物理単位からの変換は Gain ブロック等を使って予め行なってください。

図 2.5-3 ADSP674-00 D/A output



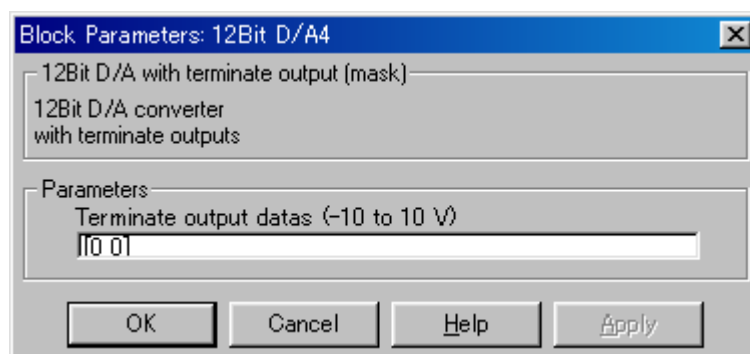
このブロックの初期化ダイアログには設定すべき項目はありません。

2.5.3 ADSP674-00 終了時出力指定D/A出力デバイスドライバブロック

終了時出力指定D/A出力デバイスドライバブロックは、幅2のベクトルデータを受取り、D/Aへ一括して出力します。このブロックへの入力ベクトルの幅は必ず2でなければいけません。このブロックへ入力されるベクトルの、最初の要素がチャンネル番号0のD/Aへ出力されます。

このブロックはモデルが停止する際に、予め指定した電圧値を出力します。このブロックへの入力データの単位は電圧値[V]です。物理単位からの変換は Gain ブロック等を使って予め行なってください。

図 2.5-4 ADSP674-00 D/A output



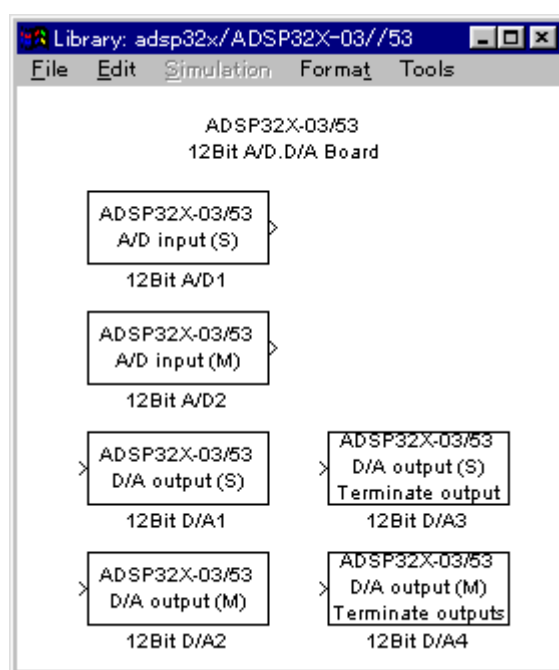
- Terminate output datas (-10 to 10V)
リアルタイムモデルが終了する際にD/Aから出力する値をベクトルで設定します。単位は[Volt]です。

2.6 ADSP32X-03/53 デバイスドライバブロック

2.6.1 概要

図 2.6-1に、ADSP32X-03/53（12ビットA/D・D/Aボード）用デバイスドライバウィンドウを開いた所を示します。

図 2.6-1 ADSP32X-03/53 ドライバウィンドウ



この中にはADSP32X-03/53とのインターフェースを取るための4つのデバイスドライバブロックが登録されています。

- A/D input (S) 1チャンネルのA/D入力デバイスドライバブロックです。
- A/D input (M) 多チャンネルのA/D入力デバイスドライバブロックです。
- D/A input (S) 1チャンネルのD/A出力デバイスドライバブロックです。
- D/A input (M) 多チャンネルのD/A出力デバイスドライバブロックです。
- D/A input (S) Terminate output
終了時の出力指定が行える、1チャンネルのD/A出力デバイスドライバブロックです。
- D/A input (M) Terminate output
終了時の出力指定が行える、多チャンネルのD/A出力デバイスドライバブロックです。

1 チャンネルの入力（出力）デバイスドライバは、任意のチャンネルからデータを入力（出力）できます。特定のチャンネルのみをアクセスしたい場合や、アクセスしたいチャンネルが連続するチャンネルでない場合に有効です。

多チャンネルの入力（出力）デバイスドライバは、チャンネル番号 0 から連続する任意の数のチャンネルからデータを入力（出力）できます。複数のチャンネルをベクトルとして取り扱う場合や、入力（出力）の為のオーバーヘッドを低減する目的等で連続するチャンネルを一括してアクセスする場合等に利用します。

終了時出力指定の行えるデバイスドライバーは、実時間モデルが停止する際に、予め初期化ダイアログで指定した値を出力します。リアルタイムモニター、U I L、ユーザ定義関数等で強制停止させた場合にも終了時出力は行われます。それ以外のブロックは実時間モデルが停止する直前の値を出力し続けます。

1 チャンネル版と多チャンネル版は一度に取り扱うチャンネルの数やチャンネル番号の指定が異なるのみで、本質的には全く差異はありません。例えば、多チャンネル版で入力した複数チャンネルの A/D 入力データを Demux ブロックで個別の信号に分解したものと、1 チャンネル版で個々に入力したものととは全く同一として取り扱えます。D/A の場合も同様です。特に、A/D では、同じサンプリング時間を設定した複数のデバイスドライバを使用した場合のサンプリングの同時性は保証されています。1 チャンネル版と多チャンネル版のドライバを混在した場合でも同様です。

2.6.2 チャンネル番号とボードとの対応

ADSP32X-03/53は最大4台使用でき、1台のボードにA/D、D/Aが各4チャンネル実装されています。ADSP32X-03/53の各デバイスドライバブロックはこれらのチャンネルを0～15の連続した論理チャンネル番号として取り扱います。論理チャンネル番号とハードウェアとの対応は、表 2.6.1の通りです。ボードの1～4台目の区別についてはハードウェアのセットアップ（6ページ）を参照してください。

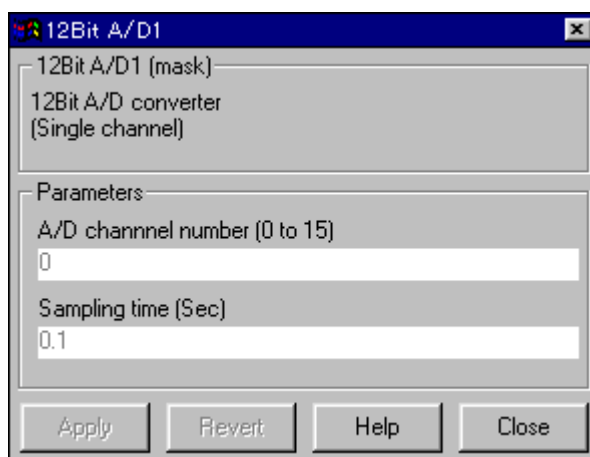
表 2.6.1 チャンネル番号とADSP32X-03/53ボードとの対応

ボード	IN/OUT0	IN/OUT1	IN/OUT2	IN/OUT3
1 台目	0	1	2	3
2 台目	4	5	6	7
3 台目	8	9	10	11
4 台目	12	13	14	15

2.6.3 1 チャンネルA/D入力デバイスドライバブロック

1 チャンネルA/D入力デバイスドライバブロックは、指定されたチャンネル番号のA/Dからデータを入力します。このブロックからの出力はスカラーとなります。このブロックからの出力データの単位は電圧値 [V] です。センサー等の物理単位への変換はデータ入力後 Gain ブロックを使って行なってください。

図 2.6-2 ADSP32X-03/53 A/D input (S)



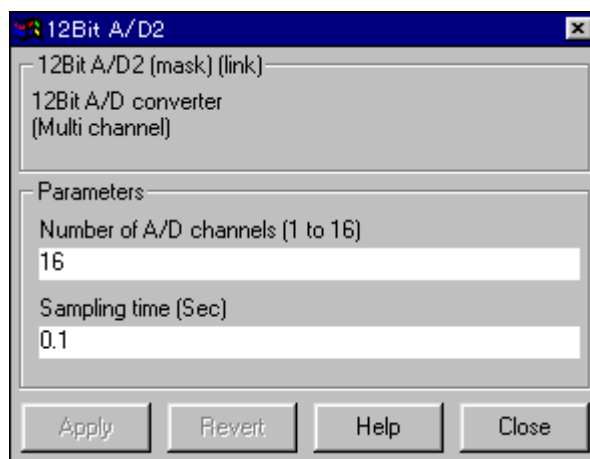
- A/D channel number (0 to 15)
入力するA/Dのチャンネル番号を0～15の整数で指定します。チャンネル番号とハードウェアとの対応は、表 2.6.1 (36ページ) を参照してください。
- Sampling time (Sec)
サンプリングの時間間隔を指定します。単位は秒です。ここで指定したサンプリングタイムと実時間モデルのステップサイズが異なる場合は、ここで指定されたサンプリングタイムでA/D変換が行われます。特殊な場合として、ここに0を入力すると、実時間モデルのステップサイズ毎にA/D変換が行われます。

2.6.4 多チャンネルA/D入力デバイスドライバブロック

多チャンネルA/D入力デバイスドライバブロックは、複数チャンネルのA/Dから一括してデータを入力します。入力の対象となるA/Dチャンネルは、チャンネル番号0から連続するチャンネルで、先頭チャンネル番号は0で固定となっています。このブロックからの出力は、指定されたチャンネル数の幅を持つベクトルとなります。ベクトルの最初の要素がチャンネル番号0からのデータです。

このブロックからの出力データの単位は電圧値 [V] です。センサー等の物理単位への変換はデータ入力後 Gain ブロックを使って行なってください。

図 2.6-3 ADSP32X-03/53 A/D input (M)



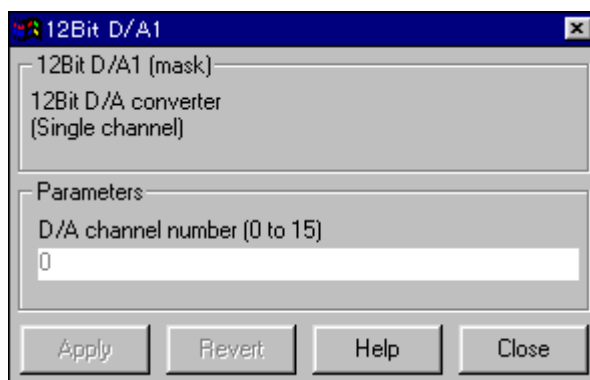
- Number of A/D channels (1 to 16)
入力するA/Dのチャンネル数を1～16の整数で指定します。チャンネル番号とハードウェアとの対応は、表 2.6.1 (36ページ) を参照してください。
- Sampling time (Sec)
サンプリングの時間間隔を指定します。単位は秒です。ここで指定したサンプリングタイムと実時間モデルのステップサイズが異なる場合は、ここで指定されたサンプリングタイムでA/D変換が行われます。特殊な場合として、ここに0を入力すると、実時間モデルのステップサイズ毎にA/D変換が行われます。

2.6.5 1チャンネルD/A出力デバイスドライバブロック

1チャンネルD/A出力デバイスドライバブロックは、スカラーデータを受取り、指定されたチャンネル番号のD/Aへ出力します。

このブロックへの入力データの単位は電圧値[V]です。物理単位からの変換は Gain ブロック等を使って予め行なってください。

図 2.6-4 ADSP32X-03/53 D/A output (S)



- D/A channel number (0 to 15)

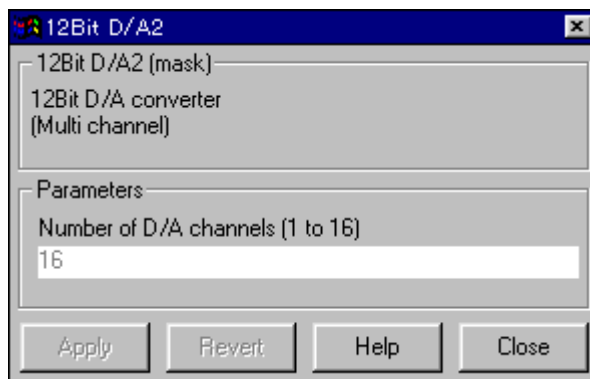
出力するD/Aのチャンネル番号を0～15の整数で指定します。チャンネル番号とハードウェアとの対応は、表 2.6.1 (36ページ) を参照してください。

2.6.6 多チャンネルD/A出力デバイスドライバブロック

多チャンネルD/A出力デバイスドライバブロックは、ベクトルデータを受取り、複数チャンネルのD/Aへ一括して出力します。出力の対象となるD/Aチャンネルは、チャンネル番号0から連続するチャンネルで、先頭チャンネル番号は0で固定となっています。このブロックへの入力ベクトルの幅と、このブロックに指定したチャンネル数とは一致していなければいけません。このブロックへ入力されるベクトルの、最初の要素がチャンネル番号0のD/Aへ出力されます。

このブロックへの入力データの単位は電圧値[V]です。物理単位からの変換は Gain ブロック等を使って予め行なってください。

図 2.6-5 ADSP32X-03/53 D/A output (M)



- Number of A/D channels (1 to 16)

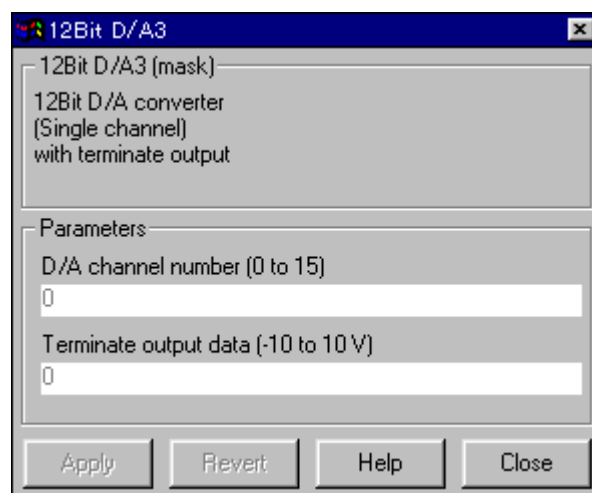
出力するD/Aのチャンネル数を1～16の整数で指定します。チャンネル番号とハードウェアとの対応は、表 2.6.1 (36ページ) を参照してください。

2.6.7 終了時出力指定 1 チャンネルD/A出力デバイスドライバブロック

終了時出力指定付き 1 チャンネルD/A出力デバイスドライバブロックは、スカラーデータを受取り、指定されたチャンネル番号のD/Aへ出力します。このブロックは実時間モデルが停止する際に、予め指定した電圧値を出力します。

このブロックへの入力データの単位は電圧値[V]です。物理単位からの変換は Gain ブロック等を使って予め行なってください。

図 2.6-6 ADSP32X-03/53 D/A output (S) Terminate output



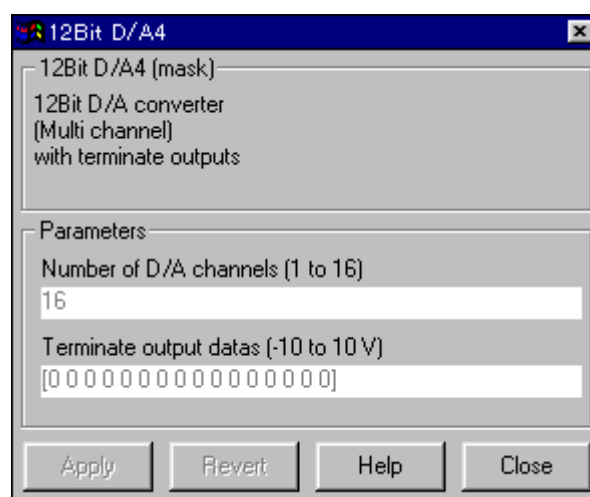
- D/A channel number (0 to 15)
出力するD/Aのチャンネル番号を0～15の整数で指定します。チャンネル番号とハードウェアとの対応は、表 2.6.1 (36ページ) を参照してください。
- Terminate output data (-10 to 10V)
リアルタイムモデルが終了する際にD/Aから出力する値を設定します。単位は[Volt]です。

2.6.8 終了時出力指定 多チャンネルD/A出力デバイスドライバブロック

終了時出力指定付き多チャンネルD/A出力デバイスドライバブロックは、ベクトルデータを受取り、複数チャンネルのD/Aへ一括して出力します。先頭チャンネル番号は0で固定となっています。このブロックへの入力ベクトルの幅と、このブロックに指定したチャンネル数とは一致していなければいけません。このブロックへ入力されるベクトルの、最初の要素がチャンネル番号0のD/Aへ出力されます。このブロックはモデルが停止する際に、予め指定した電圧値を出力します。

このブロックへの入力データの単位は電圧値[V]です。物理単位からの変換は Gain ブロック等を使って予め行なってください。

図 2.6-7 ADSP32X-03/53 D/A output (M) Terminate output



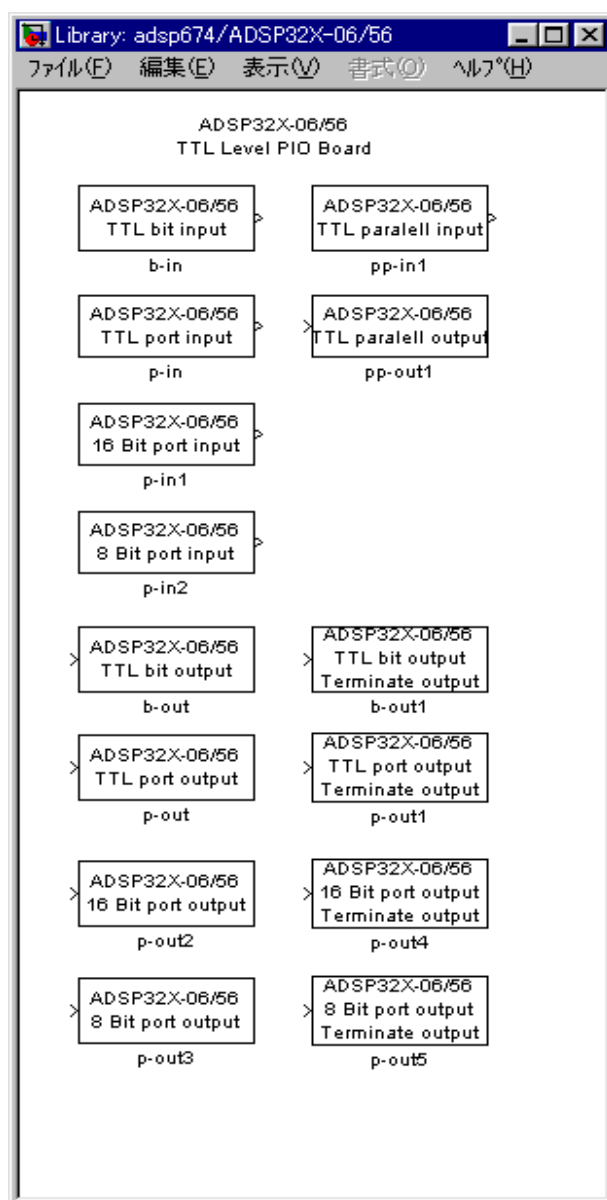
- Number of D/A channels (1 to 16)
出力するD/Aのチャンネル数を1～16の整数で指定します。先頭チャンネル番号は0で固定です。チャンネル番号とハードウェアとの対応は、表 2.6.1 (36ページ) を参照してください。
- Terminate output data (-10 to 10V)
リアルタイムモデルが終了する際にD/Aから出力する値をベクトルで設定します。単位は[Volt]です。

2.7 ADSP32X-06/56 デバイスドライバブロック

2.7.1 概要

図 2.7-1 に、ADSP32X-06/56 (TTL レベル PIO ボード) 用デバイスドライバウィンドウを開いた所を示します。

図 2.7-1 ADSP32X-06/56 ドライバウィンドウ



この中には、ADSP32X-06/56 とのインターフェースを取るための 14 個のデバイスドライバブロックが登録されています。

- TTL bit input 1 ビット単位の入力デバイスドライバです。
- TTL port input ポート単位の入力デバイスドライバです。
- TTL 16Bit input 1 6 ビット単位の入力デバイスドライバです。
- TTL 8Bit input 8 ビット単位の入力デバイスドライバです。
- TTL bit output ビット単位の実力デバイスドライバです。
- TTL port output ポート単位の実力デバイスドライバです。
- TTL 16Bit output 1 6 ビット単位の実力デバイスドライバです。
- TTL 8Bit output 8 ビット単位の実力デバイスドライバです。
- TTL paralell input 複数ビットの同時入力デバイスドライバです。
- TTL paralell output 複数ビットの同時出力デバイスドライバです。
- TTL bit output Terminate output
 終了時の出力指定が行える、ビット単位の実力デバイスドライバです。
- TTL port output Terminate output
 終了時の出力指定が行える、ポート単位の実力デバイスドライバです。
- TTL 16Bit output Terminate output
 終了時の出力指定が行える、1 6 ビット単位の実力デバイスドライバです。
- TTL 8Bit output Terminate output
 終了時の出力指定が行える、8 ビット単位の実力デバイスドライバです。

ビット単位の実力デバイスドライバは、任意のポートの任意のビットをランダムにアクセスできます。外部機器で2 値化されたデータ、操作スイッチ、リミットセンサー等のデジタル信号を取り扱うことができます。

ポート単位の実力デバイスドライバは、ポートを3 2 ビット符号付き整数として取り扱います。データ形式はストレートバイナリで、 $-2^{31} \sim 2^{31}-1$ の範囲のデータとなります。外部機器でデジタル化されたデータを取り扱うことができます。

複数ビットの実力デバイスドライバは、複数のビットを同時に入出力します。ビット単位の実力デバイスドライバーを複数並べ、Mux (Demux) にてベクトル/スカラーの変換を行ったのと同等の処理が行えますが、ビットの数が多い場合は、このドライバーの方がビット単位の実力デバイスドライバーを複数並べるより高速に入出力が行えます。

終了時出力指定の行えるデバイスドライバーは、実時間モデルが停止する際に、予め初期化ダイアログで指定した値を出力します。リアルタイムモニター、U I L、ユーザ定義関数等で強制停止させた場合にも終了時出力は行われます。それ以外のブロックは実時間モデルが停止する直前の値を出力し続けます。

2.7.2 ポート／ビット番号とボードとの対応

ADSP32X-06／56は最大4台使用できます。ADSP32X-06／56にはI／Oポートが1ポートあり、各デバイスドライバブロックはこれらポートに論理ポート番号（論理ビット番号）を割り付けて取り扱います。ADSP32X-06／56の各デバイスドライバブロックが取り扱うポート番号、ビット番号とハードウェアとの対応は、表 2.7.1 及び 表 2.7.2 の通りです。

ボードの1～4台目の区別についてはハードウェアのセットアップ（6ページ）を参照してください。

表 2.7.1 ポート番号とADSP32X-06／56ボードとの対応

ボード	入出力信号名	ポート番号		
		8ビット単位	16ビット単位	ポート単位
1 台目	ID/OD00～07	0	0	0
	ID/OD08～15	1		
	ID/OD16～23	2	1	
	ID/OD24～31	3		
2 台目	ID/OD00～07	4	2	1
	ID/OD08～15	5		
	ID/OD16～23	6	3	
	ID/OD24～31	7		
3 台目	ID/OD00～07	8	4	2
	ID/OD08～15	9		
	ID/OD16～23	10	5	
	ID/OD24～31	11		
4 台目	ID/OD00～07	12	6	3
	ID/OD08～15	13		
	ID/OD16～23	14	7	
	ID/OD24～31	15		

表 2.7.2 ビット番号とADSP32X-06/56ボードとの対応

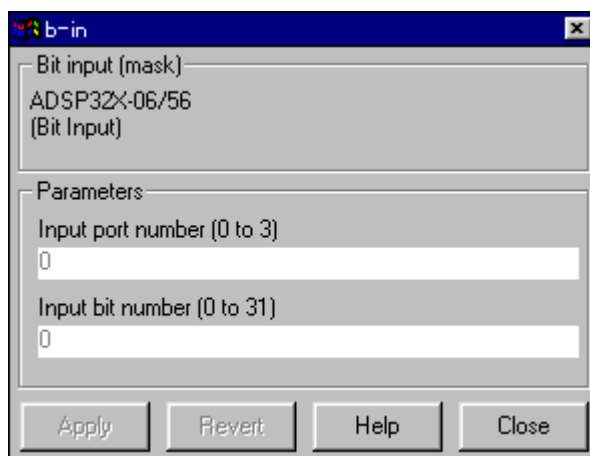
入出力信号名	ビット番号／ポート単位			
	1ビット単位	8ビット単位	16ビット単位	ポート単位
ID/OD31	31	符号	符号	符号
ID/OD30	30	2^6	2^{14}	2^{30}
ID/OD29	29	2^5	2^{13}	2^{29}
ID/OD28	28	2^4	2^{12}	2^{28}
ID/OD27	27	2^3	2^{11}	2^{27}
ID/OD26	26	2^2	2^{10}	2^{26}
ID/OD25	25	2^1	2^9	2^{25}
ID/OD24	24	2^0	2^8	2^{24}
ID/OD23	23	符号	2^7	2^{23}
ID/OD22	22	2^6	2^6	2^{22}
ID/OD21	21	2^5	2^5	2^{21}
ID/OD20	20	2^4	2^4	2^{20}
ID/OD19	19	2^3	2^3	2^{19}
ID/OD18	18	2^2	2^2	2^{18}
ID/OD17	17	2^1	2^1	2^{17}
ID/OD16	16	2^0	2^0	2^{16}
ID/OD15	15	符号	符号	2^{15}
ID/OD14	14	2^6	2^{14}	2^{14}
ID/OD13	13	2^5	2^{13}	2^{13}
ID/OD12	12	2^4	2^{12}	2^{12}
ID/OD11	11	2^3	2^{11}	2^{11}
ID/OD10	10	2^2	2^{10}	2^{10}
ID/OD09	9	2^1	2^9	2^9
ID/OD08	8	2^0	2^8	2^8
ID/OD07	7	符号	2^7	2^7
ID/OD06	6	2^6	2^6	2^6
ID/OD05	5	2^5	2^5	2^5
ID/OD04	4	2^4	2^4	2^4
ID/OD03	3	2^3	2^3	2^3
ID/OD02	2	2^2	2^2	2^2
ID/OD01	1	2^1	2^1	2^1
ID/OD00	0	2^0	2^0	2^0

注：各ボード共通

2.7.3 ビット単位入力デバイスドライバブロック

ビット単位入力デバイスドライバブロックは、指定されたポートの指定されたビットを入力し、結果をスカラーで出力します。このブロックから出力される値は、ボードへの入力レベルが LOW の時 0、HIGH の時 1 です。入力信号が接続されていないビットは 1 になります。ボードが実装されていないポートの全てのビットは不定です。

図 2.7-2 ADSP32X-06/56 TTL bit input

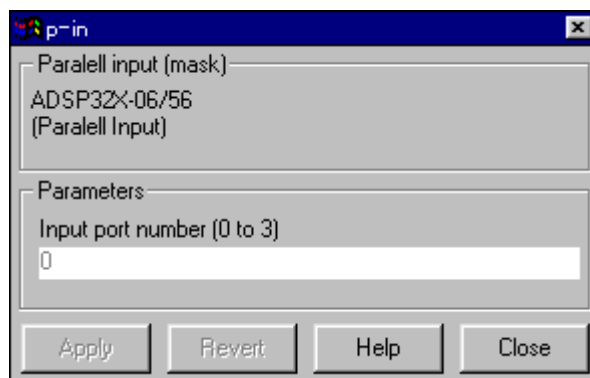


- Input port number (0 to 3)
入力の対象となるビットが含まれるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表 2.7.1 (46ページ) を参照してください。
- Input bit number (0 to 31)
入力の対象となるビット番号を指定します。ビット番号とハードウェアとの対応は、表 2.7.2 (47ページ) を参照してください。

2.7.4 ポート単位入力デバイスドライバブロック

ポート単位入力デバイスドライバブロックは、指定されたポートを入力し、結果をスカラーで出力します。このブロックから出力される値は、ボードの各ビットへの入力レベルが LOW の時 0、HIGH の時 1 とした、符号付き 32 ビット整数とみなした値です。ボードが実装されていないポートの値は不定です。

図 2.7-3 ADSP32X-06/56 TTL port input



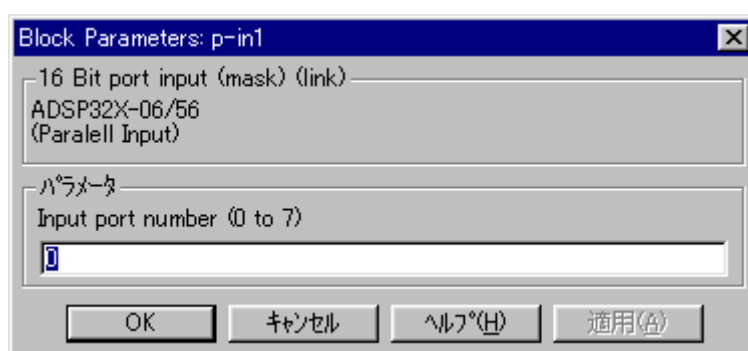
- Input port number (0 to 3)

入力の対象となるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表 2.7.1 (46ページ) を参照してください。

2.7.5 16Bit port 単位入力デバイスドライバブロック

16ビット単位入力デバイスドライバブロックは、指定されたポートを入力し、結果をスカラーで出力します。このブロックから出力される値は、ボードの各ビットへの入力レベルが LOW の時 0、HIGH の時 1 とした、符号付き 16ビット整数とみなした値です。入力が接続されていないポート及びボードが実装されていないポートの値は不定です。

図 2.7-4 ADSP32X-06/56 16Bit port input



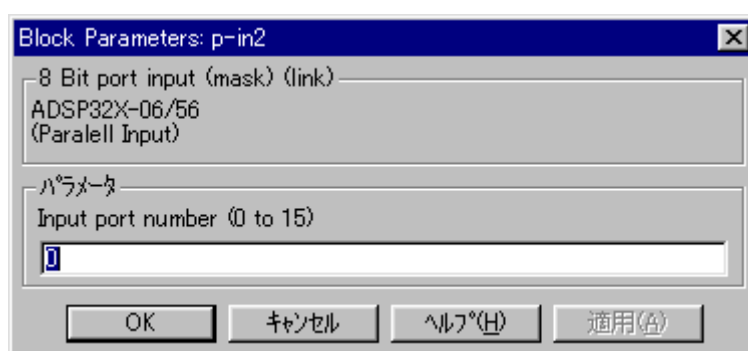
- Input port number (0 to 7)

入力の対象となるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表 2.7.1 (46ページ) を参照してください。

2.7.6 8Bit port 単位入力デバイスドライバブロック

8ビット単位入力デバイスドライバブロックは、指定されたポートを入力し、結果をスカラーで出力します。このブロックから出力される値は、ボードの各ビットへの入力レベルが LOW の時 0、HIGH の時 1 とした、符号付き 8 ビット整数とみなした値です。入力が接続されていないポート及びボードが実装されていないポートの値は不定です。

図 2.7-5 ADSP32X-06/56 8Bit port input



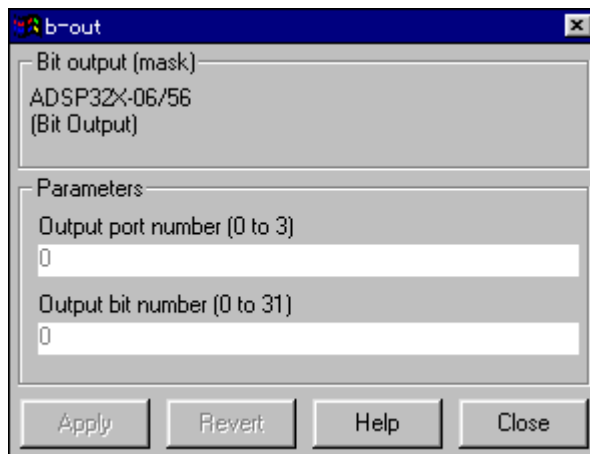
- Input port number (0 to 15)

入力の対象となるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表 2.7.1 (46ページ) を参照してください。

2.7.7 ビット単位出力デバイスドライバブロック

ビット単位出力デバイスドライバブロックは、スカラーの入力を受取り、指定されたポートの指定されたビットへ出力します。ボードから出力される信号レベルは、このブロックへの入力レベルが 0 の時に LOW、1 の時 HIGH です。指定以外のビットの状態は保存されます。ボードの出力の初期値は全ビット共 LOW です。

図 2.7-6 ADSP32X-06/56 TTL bit output



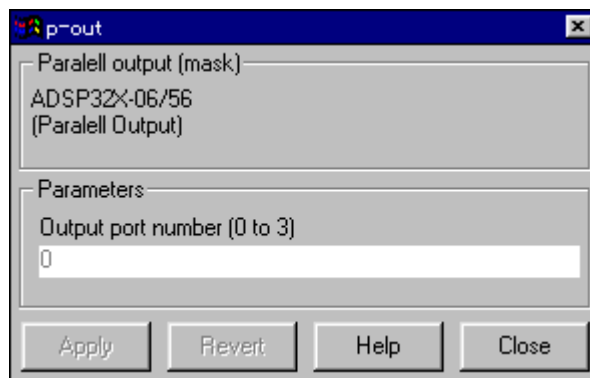
- Output port number (0 to 3)
出力の対象となるビットが含まれるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表 2.7.1 (46ページ) を参照してください。
- Output bit number (0 to 31)
出力の対象となるビット番号を指定します。ビット番号とハードウェアとの対応は、表 2.7.2 (47ページ) を参照してください。

注意：ポート単位出力デバイスドライバブロックとは同じポートを指標できません。同じポートを指標した場合、このブロックで出力した結果は、ポート単位出力デバイスドライバブロックにより失われます。

2.7.8 ポート単位出力デバイスドライバブロック

ポート単位出力デバイスドライバブロックは、スカラーの入力を受取り、符号付き 32 ビット整数に変換した後、指定されたポートへ出力します。ボードから出力される信号レベルは、変換後の整数の各ビットが 0 の時に LOW、1 の時 HIGH です。

図 2.7-7 ADSP32X-06/56 TTL port output



- Output port number (0 to 3)

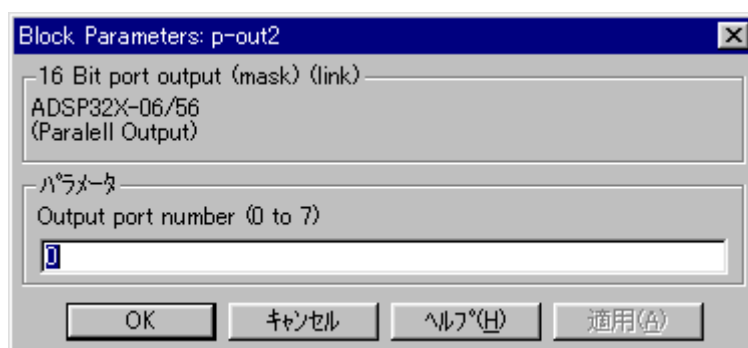
出力先ポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表 2.7.1 (46ページ) を参照してください。

注意：ビット単位出力デバイスドライバブロックとは同じポートを指標できません。同じポートを指標した場合、このブロックで出力した結果は、ビット単位出力デバイスドライバブロックにより失われます。

2.7.9 16 Bit port 単位出力デバイスドライバブロック

16ビット単位出力デバイスドライバブロックは、スカラーの入力を受取り、範囲制限を行った上で符号付き16ビット整数に変換した後、指定されたポートへ出力します。ボードから出力される信号レベルは、変換後の整数の各ビットが0の時に LOW、1の時 HIGH です。このブロックが出力できる値は、 $-2^{15} \sim 2^{15} - 1$ の範囲です。この範囲を超える値を出力しようとする、値が取り扱える範囲に収まる様に、+側は+の、-側は-の最大値（絶対値が最大の値）に制限されます。従って出力データのラップラウンドは発生しません。

図 2.7-8 ADSP32X-06/56 16 Bit port output



- Output port number (0 to 7)

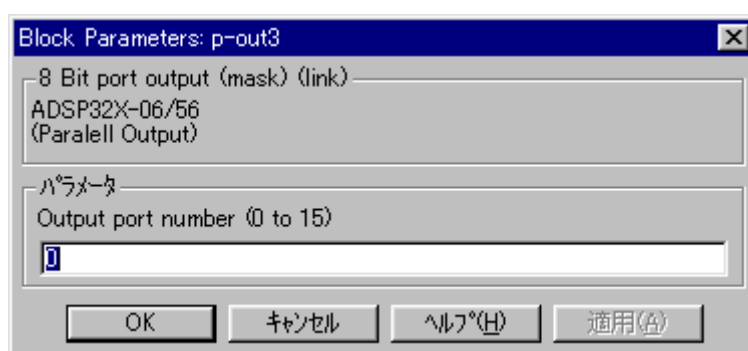
出力先ポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表 2.7.1 (46ページ) を参照してください。

注意：他の出力デバイスドライバブロックとは同じポートを指標できません。同じポートを指標した場合、このブロックで出力した結果は失われ、出力状態は予測不能となります。

2.7.10 8 Bit port 単位出力デバイスドライバブロック

8 ビット単位出力デバイスドライバブロックは、スカラーの入力を受取り、範囲制限を行った上で符号付き 8 ビット整数に変換した後、指定されたポートへ出力します。ボードから出力される信号レベルは、変換後の整数の各ビットが 0 の時に LOW、1 の時 HIGH です。このブロックが出力できる値は、 $-2^7 \sim 2^7 - 1$ の範囲です。この範囲を超える値を出力しようとする、値が取り扱える範囲に収まる様に、+側は+の、-側は-の最大値（絶対値が最大の値）に制限されます。従って出力データのラップラウンドは発生しません。

図 2.7-9 ADSP32X-06/56 8 Bit port output



- Output port number (0 to 15)

出力先ポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表 2.7.1 (46ページ) を参照してください。

注意：他の出力デバイスドライバブロックとは同じポートを指標できません。同じポートを指標した場合、このブロックで出力した結果は失われ、出力状態は予測不能となります。

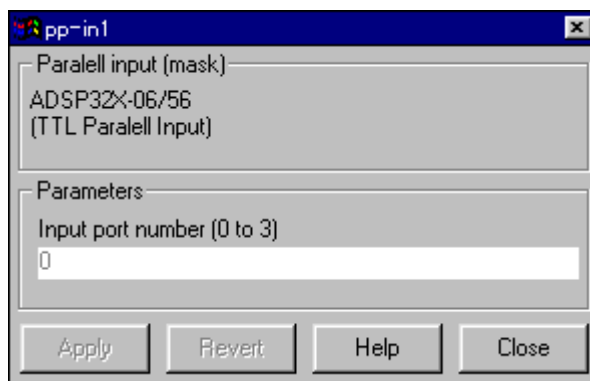
2.7.11 複数ビット入力デバイスドライバブロック

複数ビット入力デバイスドライバブロックは、指定されたポートを入力し、結果をベクトルで出力します。このブロックから出力されるベクトルは幅32のベクトルです。ベクトルの先頭の要素がビット番号0、次がビット番号1の順で、最後の要素がビット番号31に対応します。ベクトルの各要素の値は、ボードの各ビットへの入力レベルが LOW の時0、HIGH の時1です。

ポート入力デバイスドライバはとの相違点は、ポート入力はTTL入力の1ポートを1つのデータと見なすのに対し、このブロックは1つ1つのビットの状態が個々の信号（0か1）として入力できます。

ボードが実装されていない場合の値は不定です。

図 2.7-10 ADSP32X-06/56 TTL parallel input



- Input port number (0 to 3)

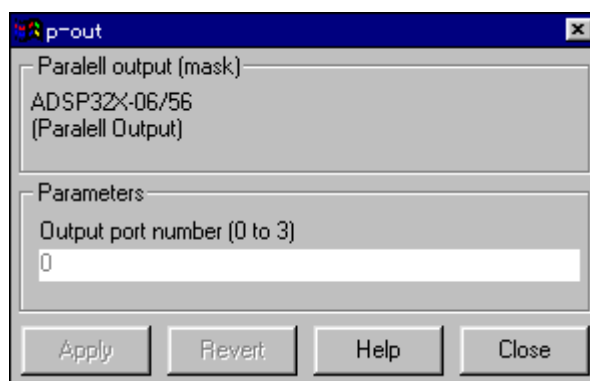
入力の対象となるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表 2.7.1 (46ページ) を参照してください。

2.7.12 複数ビット出力デバイスドライバブロック

複数ビット出力デバイスドライバブロックは、ベクトルの入力を受取り、各要素に対応するビットにHIGH又はLOWを出力します。各要素の値が0の時にはLOWを、0以外の時にはHIGHを出力します。このブロックへの入力は幅32のベクトルを与えます。ベクトルの先頭の要素がビット番号0、次がビット番号1の順で、最後の要素がビット番号31に対応します。

ポート出力デバイスドライバはとの相違点は、ポート出力はそのブロックへの入力をバイナリに変換し、各ビットに対応するTTLポートのビットにHIGH／LOWを出力しますが、このブロックは入力ベクトルの個々の要素の状態がTTLポートの個々のビットにHIGH／LOWで出力されます。

図 2.7-11 ADSP32X-06/56 TTL port output



- Output port number (0 to 3)

出力先ポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表 2.7.1 (46ページ) を参照してください。

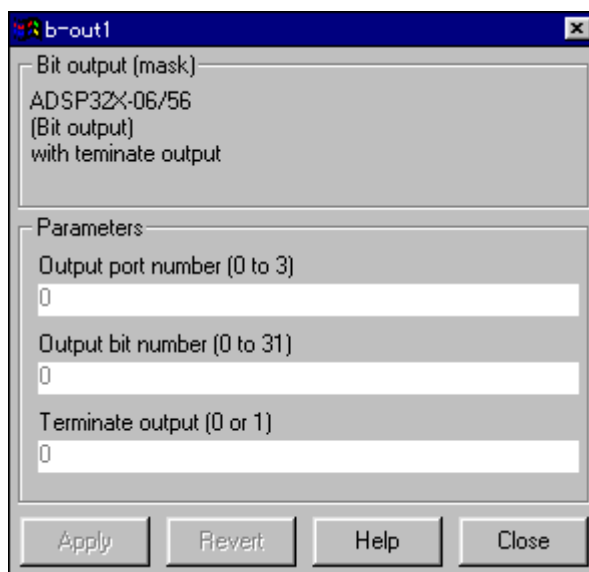
注意：ビット単位出力デバイスドライバブロックとは同じポートを指標できません。同じポートを指標した場合、このブロックで出力した結果は、ビット単位出力デバイスドライバブロックにより失われます。

2.7.13 終了時出力指定 ビット単位出力デバイスドライバブロック

終了時出力指定付きビット単位出力デバイスドライバブロックは、スカラーの入力を受け取り、指定されたポートの指定されたビットへ出力します。ボードから出力される信号レベルは、このブロックへの入力レベルが0の時に LOW、1の時 HIGH です。指定以外のビットの状態は保存されます。ボードの出力の初期値は全ビット共 LOW です。

このブロックは実時間モデルが停止する際に、予め設定した値を出力します。

図 2.7-12 ADSP32X-06/56 TTL bit output Terminate output



- Output port number (0 to 3)
出力の対象となるビットが含まれるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表 2.7.1 (46ページ) を参照してください。
- Output bit number (0 to 31)
出力の対象となるビット番号を指定します。ビット番号とハードウェアとの対応は、表 2.7.2 (47ページ) を参照してください。
- Output bit number (0 or 1)
実時間モデルが停止する際に出力して欲しい状態を設定します。0でLOW, 1でHIGHが出力されます。

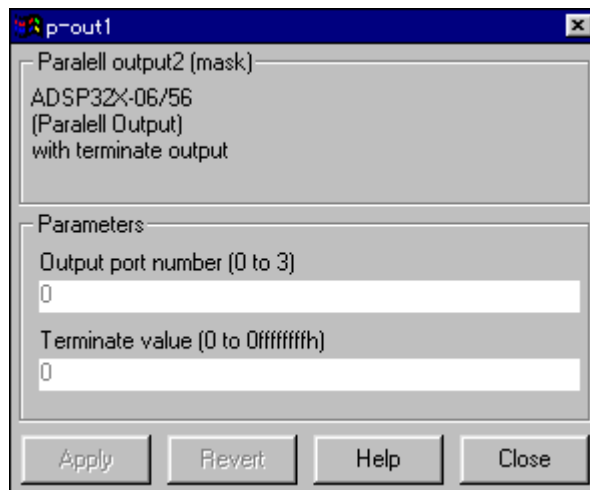
注意：ポート単位出力デバイスドライバブロックとは同じポートを指標できません。同じポートを指標した場合、このブロックで出力した結果は、ポート単位出力デバイスドライバブロックにより失われます。

2.7.14 終了時出力指定 ポート単位出力デバイスドライバブロック

終了時出力指定付きポート単位出力デバイスドライバブロックは、スカラーの入力を受け取り、符号付き 32 ビット整数に変換した後、指定されたポートへ出力します。ボードから出力される信号レベルは、変換後の整数の各ビットが 0 の時に LOW、1 の時 HIGH です。

このブロックは、実時間モデルが停止する際に予め指定した値を出力します。

図 2.7-13 ADSP32X-06/56 TTL port output Terminate output



- Output port number (0 to 3)
出力先ポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表 2.7.1 (46ページ) を参照してください。
- Terminate value (0 to 0ffffffh)
実時間モデルが停止する際に出力して欲しい値を 16 進数で設定します。

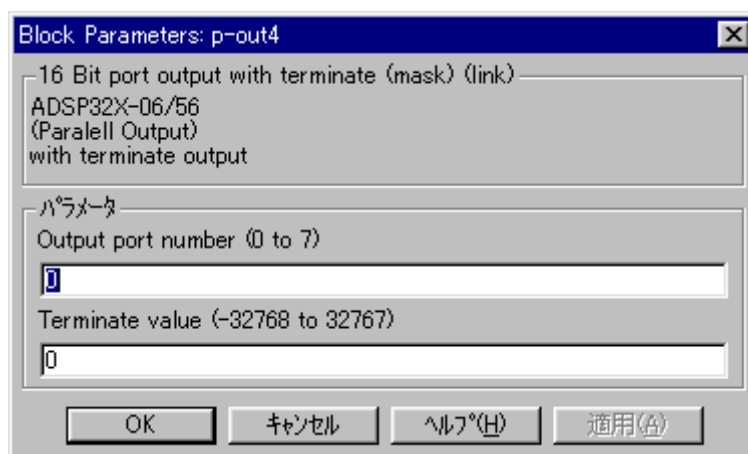
注意：ビット単位出力デバイスドライバブロックとは同じポートを指標できません。同じポートを指標した場合、このブロックで出力した結果は、ビット単位出力デバイスドライバブロックにより失われます。

2.7.15 終了時出力指定 16 Bit port 単位出力デバイスドライバブロック

終了時出力指定付き 16 ビット単位出力デバイスドライバブロックは、スカラーの入力を受取り、範囲制限を行った上で符号付き 16 ビット整数に変換した後、指定されたポートへ出力します。ボードから出力される信号レベルは、変換後の整数の各ビットが 0 の時に LOW、1 の時 HIGH です。このブロックが出力できる値は、 $-2^{15} \sim 2^{15} - 1$ の範囲です。この範囲を超える値を出力しようとする、値が取り扱える範囲に収まる様に、+側は+の、-側は-の最大値（絶対値が最大の値）に制限されます。従って出力データのラップラウンドは発生しません。

このブロックは、実時間モデルが停止する際に予め指定した値を出力します。

図 2.7-14 ADSP32X-06/56 16 Bit port output Terminate output



- Output port number (0 to 7)
出力先ポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表 2.7.1 (46ページ) を参照してください。
- Terminate value (0 to 0xffffffffh)
実時間モデルが停止する際に出力して欲しい値を 16 進数で設定します。

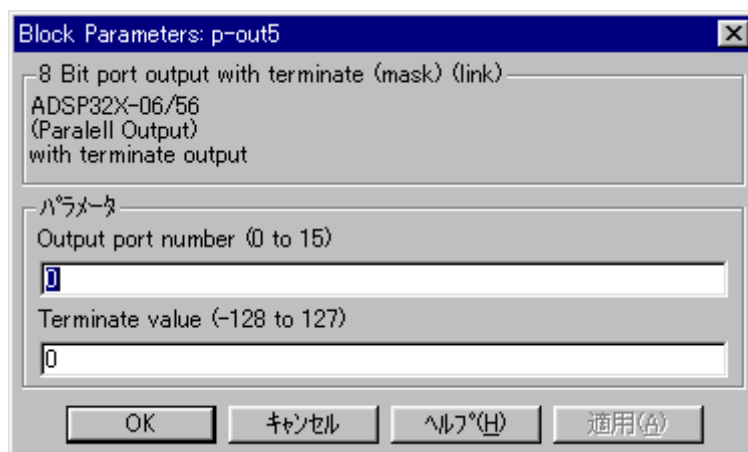
注意：他の出力デバイスドライバブロックとは同じポートを指標できません。同じポートを指標した場合、このブロックで出力した結果は失われ、出力状態は予測不能となります。

2.7.16 終了時出力指定 8Bit port 単位出力デバイスドライバブロック

終了時出力指定付き 8 ビット単位出力デバイスドライバブロックは、スカラーの入力を受取り、範囲制限を行った上で符号付き 8 ビット整数に変換した後、指定されたポートへ出力します。ボードから出力される信号レベルは、変換後の整数の各ビットが 0 の時に LOW、1 の時 HIGH です。このブロックが出力できる値は、 $-2^7 \sim 2^7 - 1$ の範囲です。この範囲を超える値を出力しようとする、値が取り扱える範囲に収まる様に、+側は+の、-側は-の最大値（絶対値が最大の値）に制限されます。従って出力データのラップラウンドは発生しません。

このブロックは、実時間モデルが停止する際に予め指定した値を出力します。

図 2.7-15 ADSP32X-06/56 8 Bit port output Terminate output



- Output port number (0 to 15)
出力先ポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表 2.7.1 (46ページ) を参照してください。
- Terminate value (0 to 0xffffffffh)
実時間モデルが停止する際に出力して欲しい値を 16 進数で設定します。

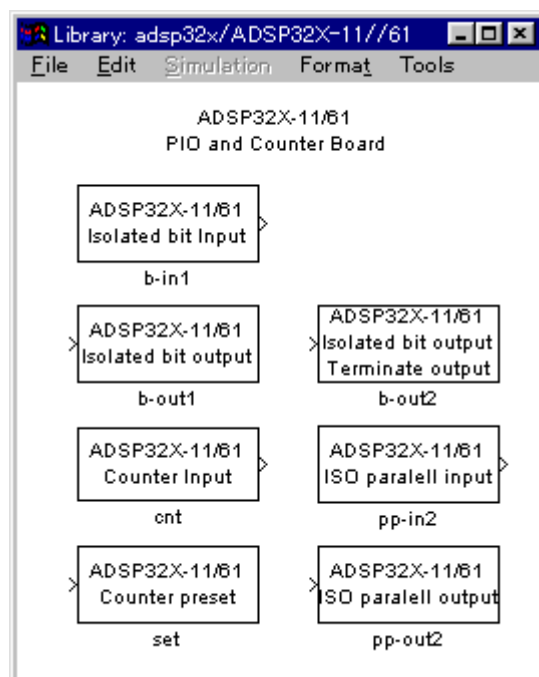
注意：他の出力デバイスドライバブロックとは同じポートを指標できません。同じポートを指標した場合、このブロックで出力した結果は失われ、出力状態は予測不能となります。

2.8 ADSP32X-11/61 デバイスドライバブロック

2.8.1 概要

図 2.8-1に、ADSP32X-11/61 (PIO&カウンタボード) 用デバイスドライバウィンドウを開いた所を示します。

図 2.8-1 ADSP32X-11/61 ドライバウィンドウ



この中には、ADSP32X-11/61 とのインターフェースを取るためのデバイスドライバブロックが登録されています。

- Isolated bit input ビット単位の絶縁入力デバイスドライバブロックです。
- Isolated bit output ビット単位の絶縁出力デバイスドライバブロックです。
- Counter input プリセット付きカウンターデバイスドライバです。
- Counter preset カウンタープリセットデバイスドライバブロックです。
- Isolated bit output Terminate output
終了時の出力指定が行えるビット単位の絶縁出力デバイスドライバブロックです。
- Isolated parallell input
複数ビット同時の絶縁入力デバイスドライバブロックです。
- Isolated parallell output
複数ビット同時の絶縁出力デバイスドライバブロックです。

ビット単位の入出力デバイスドライバは、ADSP32X-11/61 ボードの任意の絶縁入出力ポートの任意のビットに信号を入出力します。アクセスは全てビット単位で行なわれます。

プリセット付きカウンターデバイスドライバは、カウンターをアクセスするデバイスドライバで、初期化時にカウンタを規定の値にセットすることもできます。

カウンタプリセットデバイスドライバは、任意のカウンタを、任意の時刻にプリセットするドライバです。

終了時出力指定の行えるデバイスドライバーは、実時間モデルが停止する際に、予め初期化ダイアログで指定した値を出力します。リアルタイムモニター、UIL、ユーザ定義関数等で強制停止させた場合にも終了時出力は行われます。それ以外のブロックは実時間モデルが停止する直前の値を出力し続けます。

複数ビットのデバイスドライバは、複数のビットを同時に入出力します。ビット単位のデバイスドライバーを複数並べ、Mux (Demux) にてベクトル/スカラーの変換を行ったのと同等の処理が行えますが、ビットの数が多い場合は、このドライバーの方がビット単位のデバイスドライバーを複数並べるより高速に入出力が行えます。

2.8.2 ビット番号、ポート／チャンネル番号とボードとの対応

ADSP32X-11/61は最大4台使用できます。ADSP32X-11/61ボードにはカウンターが4チャンネルと絶縁I/Oポートが1ポート実装されています。ADSP32X-11/61の各デバイスドライバブロックは、これらのポートを0から始まる連続する論理番号で取り扱います。各デバイスドライバブロックが取り扱うポート番号、ビット番号とハードウェアとの対応は、表 2.8.1 及び 表 2.8.2 の通りです。ボードの1～4台目の区別についてはハードウェアのセットアップ（6ページ）を参照してください。

表 2.8.1 ポート／チャンネル番号とADSP32X-11/61との対応

ボード	カウンタチャンネル番号				絶縁ポート番号
	カウンタ0	カウンタ1	カウンタ2	カウンタ3	
1 台目	0	1	2	3	0
2 台目	4	5	6	7	1
3 台目	8	9	10	11	2
4 台目	12	13	14	15	3

表 2.8.2 ビット番号とADSP 32X-11/61との対応

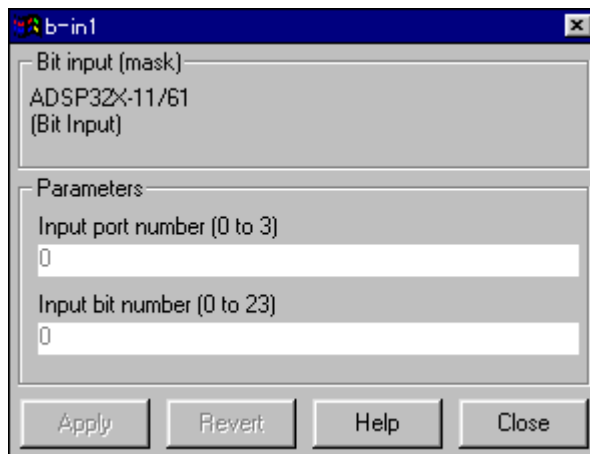
ビット番号	絶縁入力信号名	絶縁出力信号名
0	I0_00	00_00
1	I0_01	00_01
2	I0_02	00_02
3	I0_03	00_03
4	I0_04	00_04
5	I0_05	00_05
6	I0_06	00_06
7	I0_07	00_07
8	I0_10	00_10
9	I0_11	00_11
1 0	I0_12	00_12
1 1	I0_13	00_13
1 2	I0_14	00_14
1 3	I0_15	00_15
1 4	I0_16	00_16
1 5	I0_17	00_17
1 6	I0_20	—
1 7	I0_21	—
1 8	I0_22	—
1 9	I0_23	—
2 0	I0_24	—
2 1	I0_25	—
2 2	I0_26	—
2 3	I0_27	—

注 1 : 各絶縁入出力ポート共通

2.8.3 ビット単位、絶縁入力デバイスドライバブロック

ビット単位絶縁入力デバイスドライバブロックは、指定されたポートの指定されたビットを入力し、結果をスカラーで出力します。このブロックから出力される値は、ボードへの入力信号が OFF の時 0、ON の時 1 です。入力信号が接続されていないビットは 0 になります。ボードが実装されていないポートの全てのビットは不定です。

図 2.8-2 ADSP32X-11/61 Isolated bit input

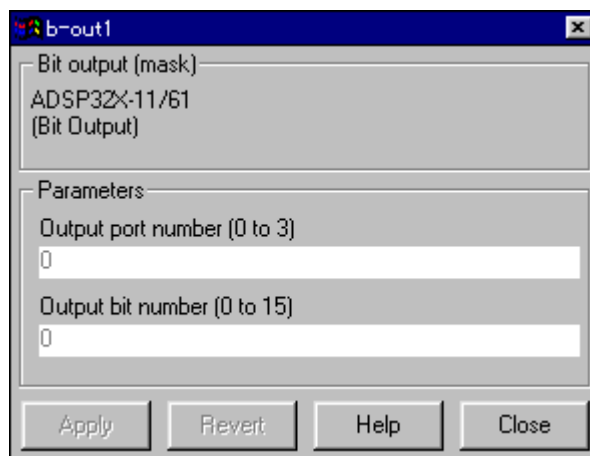


- Input port number (0 to 3)
入力の対象となるビットが含まれるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表 2.8.1 (64ページ) を参照してください。
- Input bit number (0 to 23)
入力の対象となるビット番号を指定します。指定できるビット番号は 0 ~ 23 です。ビット番号とハードウェアとの対応は、表 2.8.2 (65ページ) を参照してください。

2.8.4 ビット単位、絶縁出力デバイスドライバブロック

ビット単位絶縁出力デバイスドライバブロックは、スカラーの入力を受取り、指定されたポートの指定されたビットへ出力します。ボードから出力される信号（出力トランジスタ）は、このブロックへの入力レベルが0の時に OFF、1の時 ON です。指定以外のビットの状態は保存されます。ボードの出力の初期値は全ビット共 OFF です。

図 2.8-3 ADSP32X-11/61 Isolated bit output

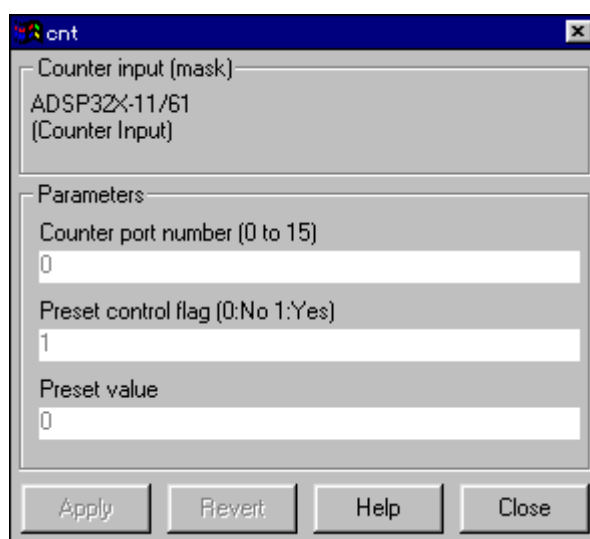


- Output port number (0 to 3)
出力の対象となるビットが含まれるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表 2.8.1（64ページ）を参照してください。
- Output bit number (0 to 15)
出力の対象となるビット番号を指定します。指定できるビット番号は、0～15です。ビット番号とハードウェアとの対応は、表 2.8.2（65ページ）を参照してください。

2.8.5 プリセット付きカウンタデバイスドライバブロック

プリセット付きカウンタデバイスドライバブロックは、任意のチャンネル番号のカウンターの値を読み込みます。このブロックの出力はスカラーです。ADSP32X-11/61 ボードの指定されたカウンターは符号付き 32 ビット整数として読み込まれ、スカラーに変換されて出力されます。プリセットの指定を行なった場合、初期化の際に指定した値にプリセットされます。プリセットは初期化の際に 1 回だけ行なわれます。

図 2.8-4 ADSP32X-11/61 Counter input



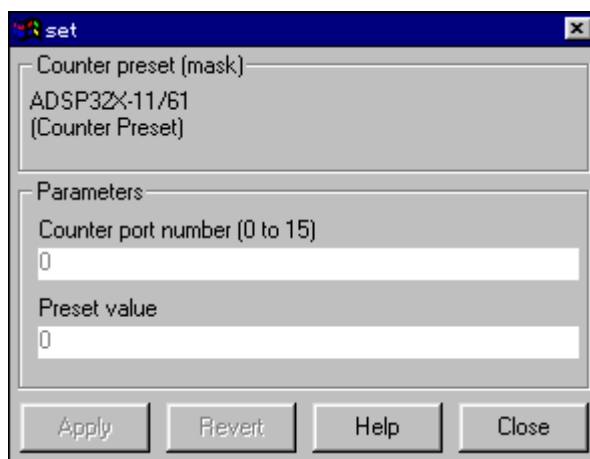
- Counter port number (0 to 15)
入力の対象となるカウンタポート番号を指定します。ポート番号とハードウェアとの対応は、表 2.8.1 (64ページ) を参照してください。
- Preset control flag (0:No 1:Yes)
初期化時にカウンタ値のプリセットを行なうか否かを決定します。0 又は 1 を設定してください。0 を設定した場合はプリセットは行ないません。1 を設定した場合は Preset value に設定した値がカウンタに初期化の際（シミュレーションの開始時）に 1 度だけ設定されます。
- Preset value
Preset control flag に 1 を設定した場合、ここに設定した値がカウンターの初期値として採用されます。Preset control flag に 1 を設定した場合のみ有効です。設定できる値は、符号付き 32 ビット整数（ストレートバイナリ）で表わせる範囲（ $-2^{31} \sim 2^{31}-1$ ）です。

2.8.6 カウンタプリセットデバイスドライバブロック

カウンタプリセットデバイスドライバブロックは、任意のチャンネル番号のカウンタを任意の時期にプリセットするデバイスドライバです。プリセット付きカウンタデバイスドライバブロックでは、初期化の際に1度しかプリセットできませんが、このブロックを使用すれば任意の時期に値をプリセットできます。

このブロックには1つのスカラー入力があり、ここに0又は1の値を入力することによりプリセットタイミングを制御します。0が入力されている期間はカウンタはカウント動作を行ないます。1が入力されるとカウント動作は停止し Preset value で予め設定した値がカウンタにセットされます。

図 2.8-5 ADSP32X-11/61 Counter preset



- Counter port number (0 to 15)

入力の対象となるカウンタポート番号を指定します。ポート番号とハードウェアとの対応は、表 2.8.1 (64ページ) を参照してください。

- Preset value

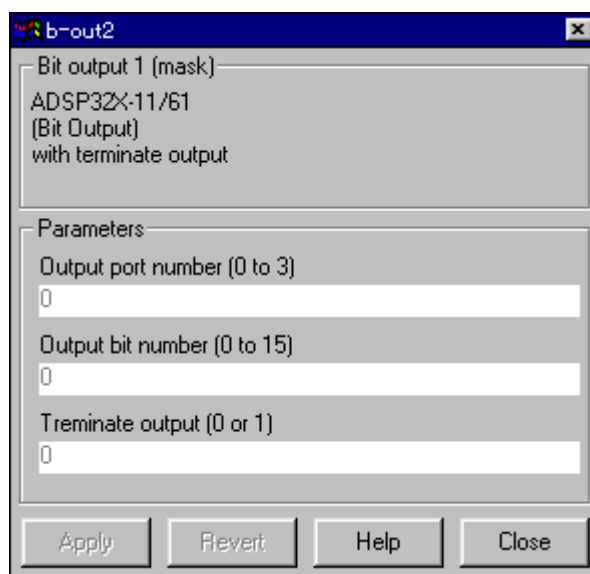
ここに設定した値がカウンタにセットされます。設定できる値は、符号付き32ビット整数（ストレートバイナリ）で表わせる範囲（ $-2^{31} \sim 2^{31}-1$ ）です。

2.8.7 終了時出力指定 ビット単位、絶縁出力デバイスドライバブロック

終了時出力指定付きビット単位絶縁出力デバイスドライバブロックは、スカラーの入力を受取り、指定されたポートの指定されたビットへ出力します。ボードから出力される信号（出力トランジスタ）は、このブロックへの入力レベルが0の時に OFF、1の時 ON です。指定以外のビットの状態は保存されます。ボードの出力の初期値は全ビット共 OFF です。

このブロックは、実時間モデルが終了する際に予め設定しておいた値を出力する事ができます。

図 2.8-6 ADSP32X-11/61 Isolated bit output Terminate output

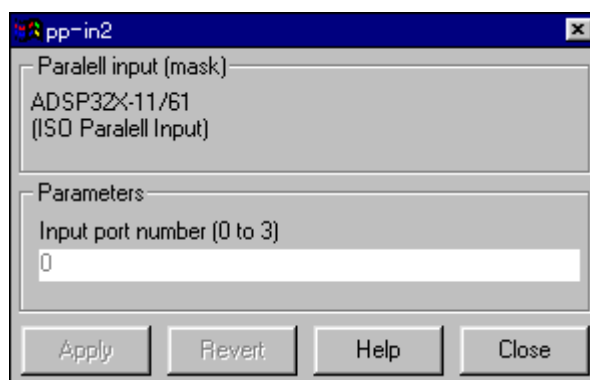


- Output port number (0 to 3)
出力の対象となるビットが含まれるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表 2.8.1（64ページ）を参照してください。
- Output bit number (0 to 15)
出力の対象となるビット番号を指定します。指定できるビット番号は、0～15です。ビット番号とハードウェアとの対応は、表 2.8.2（65ページ）を参照してください。
- Terminale output (0 or 1)
実時間モデルが終了する際に出力したい値を設定します。

2.8.8 複数ビット同時、絶縁入力デバイスドライバブロック

複数ビット同時絶縁入力デバイスドライバブロックは、指定されたポートの全ビットを入力し、結果をベクトルで出力します。このブロックから出力されるベクトルは、幅が24のベクトルで、先頭の要素がビット0、次がビット1の順で、最後の要素がビット23に対応しています。各要素はボードへの入力信号がOFFの時0、ONの時1です。入力信号が接続されていないビットは0になります。ボードが実装されていないポートの全てのビットは不定です。

図 2.8-7 ADSP32X-11/61 ISO parallel input



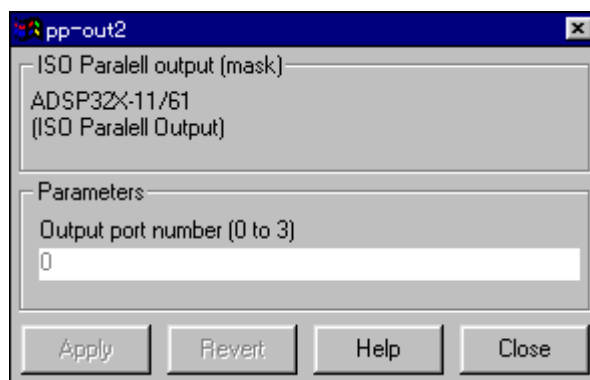
- Input port number (0 to 3)

入力の対象となるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表 2.8.1 (64ページ) を参照してください。

2.8.9 複数ビット同時、絶縁出力デバイスドライバブロック

複数ビット同時絶縁出力デバイスドライバブロックは、ベクトルの入力を受取り、指定されたポートの各ビットへ出力します。入力するベクトルは幅 16 のベクトルで、先頭の要素が出力ビット 0、次が出力ビット 1 の順で、最後の要素が出力ビット 15 に相当します。ボードから出力される信号（出力トランジスタ）は、このブロックへの入力が 0 の時に OFF、0 以外の時 ON です。

図 2.8-8 ADSP32X-11/61 ISO parallel output



- Output port number (0 to 3)

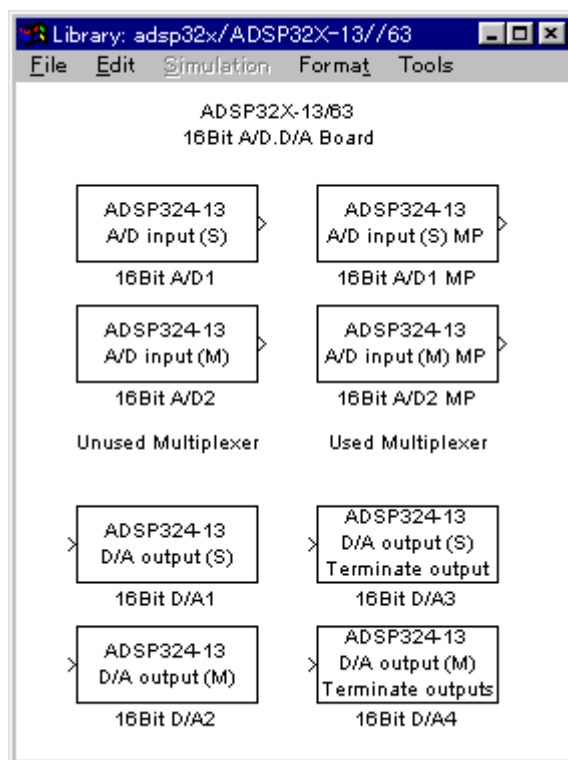
出力の対象となるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表 2.8.1（64ページ）を参照してください。

2.9 ADSP32X-13/63 デバイスドライバブロック

2.9.1 概要

図 2.9-1に、ADSP32X-13/63（16ビットA/D・D/Aボード）用デバイスドライバウィンドウを開いた所を示します。

図 2.9-1 ADSP32X-13/63 ドライバウィンドウ



この中にはADSP32X-13/63とのインターフェースを取るためのデバイスドライバブロックが登録されています。

- A/D input (S) 1チャンネルのA/D入力デバイスドライバブロックです。マルチプレクサー（オプション）が無い場合に利用出来ます。
- A/D input (M) 多チャンネルのA/D入力デバイスドライバブロックです。マルチプレクサー（オプション）が無い場合に利用出来ます。
- A/D input (S) MP 1チャンネルのA/D入力デバイスドライバブロックです。マルチプレクサー（オプション）が有る場合に利用出来ます。
- A/D input (M) MP 多チャンネルのA/D入力デバイスドライバブロックです。マルチプレクサー（オプション）が有る場合に利用出来ます。
- D/A input (S) 1チャンネルのD/A出力デバイスドライバブロック

です。

- D/A input (M) 多チャンネルのD/A出力デバイスドライバブロックです。
- D/A input (S) Terminate output
終了時出力指定付きの1チャンネルのD/A出力デバイスドライバブロックです。
- D/A input (M) Terminate output
終了時出力指定付きの多チャンネルのD/A出力デバイスドライバブロックです。

1チャンネルの入力（出力）デバイスドライバは、任意のチャンネルからデータを入力（出力）できます。特定のチャンネルのみをアクセスしたい場合や、アクセスしたいチャンネルが連続するチャンネルでない場合に有効です。

多チャンネルの入力（出力）デバイスドライバは、チャンネル番号0から連続する任意の数のチャンネルからデータを入力（出力）できます。複数のチャンネルをベクトルとして取り扱う場合や、入力（出力）のオーバーヘッドを低減する目的等で連続するチャンネルを一括してアクセスする場合等に利用します。

終了時出力指定の行えるデバイスドライバーは、実時間モデルが停止する際に、予め初期化ダイアログで指定した値を出力します。リアルタイムモニター、U I L、ユーザ定義関数等で強制停止させた場合にも終了時出力は行われます。それ以外のブロックは実時間モデルが停止する直前の値を出力し続けます。

1チャンネル版と多チャンネル版は一度に取り扱うチャンネルの数やチャンネル番号の指定が異なるのみで、本質的には全く差異はありません。例えば、多チャンネル版で入力した複数チャンネルのA/D入力データをDemuxブロックで個別の信号に分解したものと、1チャンネル版で個々に入力したものととは全く同一として取り扱えます。D/Aの場合も同様です。特に、A/Dでは、同じサンプリング時間を設定した複数のデバイスドライバを使用した場合のサンプリングの同時性は保証されています。1チャンネル版と多チャンネル版のドライバを混在した場合でも同様です。

2.9.2 チャンネル番号とボードとの対応

ADSP32X-13/63 ボードは最大4台使用できます。ADSP32X-13/63 ボードにはA/Dが2チャンネル（マルチプレクサーオプションを増設した場合は16チャンネル）と、D/Aが2チャンネル実装されています。ADSP32X-13/63の各デバイスドライバブロックは、これらのチャンネルを0から始まる連続する論理チャンネル番号として取り扱います。ADSP32X-13/63の各デバイスドライバブロックが取り扱うチャンネル番号とハードウェアとの対応は、表 2.9.1 及び 表 2.9.2の通りです。

ボードの1～4台目の区別についてはハードウェアのセットアップ（6ページ）を参照してください。

チャンネル番号とADSP32X-13/63 ボードとの対応

表 2.9.1 マルチプレクサが無い場合

ボード	信号名	
	IN/OUT00	IN/OUT01
1 台目	0	1
2 台目	2	3
3 台目	4	5
4 台目	6	7

表 2.9.2 マルチプレクサが有る場合

ボード	信号名							
	IN00	IN01	...	IN07	IN10	IN11	...	IN17
1 台目	0	1	...	7	8	9	...	15
2 台目	16	17	...	23	24	25	...	31
3 台目	32	33	...	39	40	41	...	47
4 台目	48	49	...	55	56	57	...	63

2.9.3 P G Aバイパス

マルチプレクサーオプション (ADSP324-12) を追加した場合、A/D変換は、マルチプレクサーを切り替えながら行う為、変換に要する時間が長くなります。これは、A/Dチャンネルの切り替えの際に、ボードに実装されている P G A (Programable Gain Amp) や S/H (Sample & Hold) の応答時間 (セトリングタイム) を待機している為です。この時間の大半は P G Aが占めているため、P G Aを取り除く (バイパスさせる) 事により大幅に短縮できます。(約 3 0 4 μ 秒)

P G Aをバイパスするには、以下の点の注意が必要です。

- (1) 入力がシングルエンドになります。(標準は差動入力です。)
- (2) ハードウェアの改造が必要です。
- (3) ハードウェアの改造を行わずに、ブロックの PGA amp パラメータを 0 にすると、正しい値が取り込めません。

詳細については、メーカーにお問い合わせください。

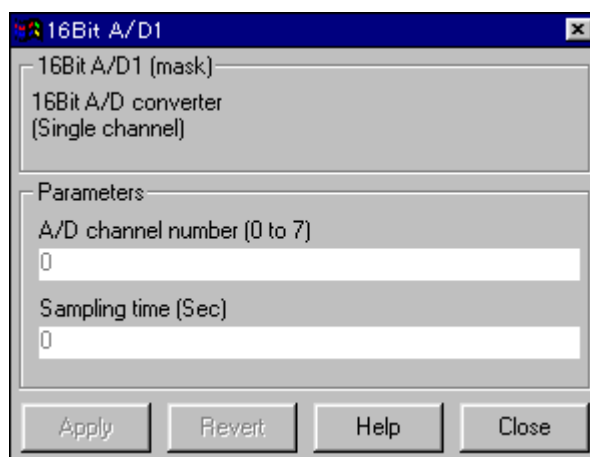
2.9.4 1 チャンネルA/D入力デバイスドライバブロック

1 チャンネルA/D入力デバイスドライバブロックは、指定されたチャンネル番号のA/Dからデータを入力します。このブロックからの出力はスカラーとなります。

このブロックからの出力データの単位は電圧値 [V] です。センサー等の物理単位への変換はデータ入力後 Gain ブロックを使って行なってください。

このデバイスドライバーはマルチプレクサー (オプション) が無い場合のみ利用可能です。

図 2.9-2 ADSP32X-13/63 A/D input (S)



- A/D channel number (0 to 7)

入力するA/Dのチャンネル番号を0～7の整数で指定します。チャンネル番号とハードウェアとの対応は、表 2.9.1 (75ページ) を参照してください。

- Sampling time (Sec)

サンプリングの時間間隔を指定します。単位は秒です。ここで指定したサンプリングタイムと実時間モデルのステップサイズが異なる場合は、ここで指定されたサンプリングタイムでA/D変換が行われます。特殊な場合として、ここに0を入力すると、実時間モデルのステップサイズ毎にA/D変換が行われます。

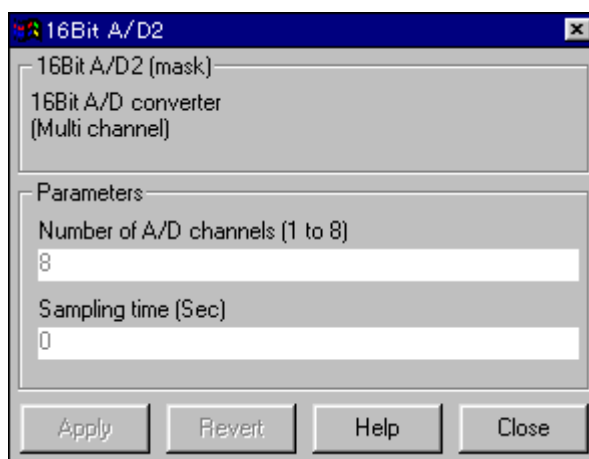
2.9.5 多チャンネルA/D入力デバイスドライバブロック

多チャンネルA/D入力デバイスドライバブロックは、複数チャンネルのA/Dから一括してデータを入力します。入力の対象となるA/Dチャンネルは、チャンネル番号0から連続するチャンネルで、先頭チャンネル番号は0で固定となっています。このブロックからの出力は、指定されたチャンネル数の幅を持つベクトルとなります。ベクトルの最初の要素がチャンネル番号0からのデータです。

このブロックからの出力データの単位は電圧値 [V] です。センサー等の物理単位への変換はデータ入力後 Gain ブロックを使って行なってください。

このデバイスドライバはマルチプレクサー（オプション）が無い場合のみ利用可能です。

図 2.9-3 ADSP32X-13/63 A/D input (M)



- Number of A/D channels (1 to 8)
入力するA/Dのチャンネル数を1～8の整数で指定します。チャンネル番号とハードウェアとの対応は、表 2.9.1（75ページ）を参照してください。
- Sampling time (Sec)
サンプリングの時間間隔を指定します。単位は秒です。ここで指定したサンプリングタイムと実時間モデルのステップサイズが異なる場合は、ここで指定されたサンプリングタイムでA/D変換が行われます。特殊な場合として、ここに0を入力すると、実時間モデルのステップサイズ毎にA/D変換が行われます。

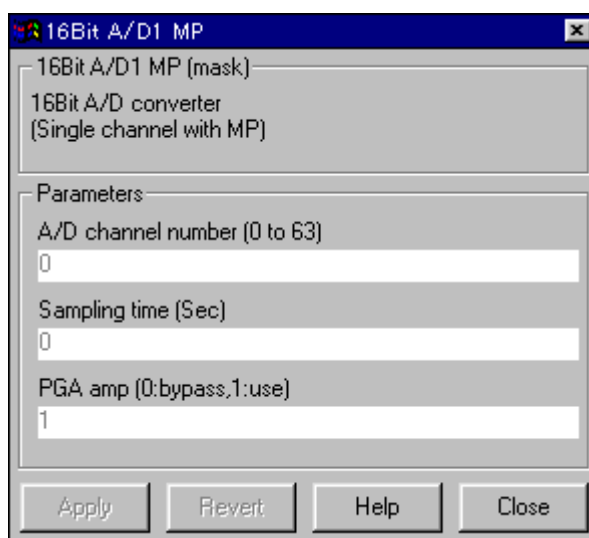
2.9.6 1チャンネルA/D入力（MP付き）デバイスドライバブロック

1チャンネルA/D入力（MP＝マルチプレクサ付き）デバイスドライバブロックは、指定されたチャンネル番号のA/Dからデータを入力します。このブロックからの出力はスカラーとなります。

このブロックからの出力データの単位は電圧値〔V〕です。センサー等の物理単位への変換はデータ入力後 Gain ブロックを使って行なってください。

このデバイスドライバーはマルチプレクサー（オプション）が有る場合のみ利用可能です。

図 2.9-4 ADSP32X-13/63 A/D input (S) MP



- A/D channel number (0 to 63)
入力するA/Dのチャンネル番号を0～63の整数で指定します。チャンネル番号とハードウェアとの対応は、表 2.9.2（75ページ）を参照してください。
- Sampling time (Sec)
サンプリングの時間間隔を指定します。単位は秒です。ここで指定したサンプリングタイムと実時間モデルのステップサイズが異なる場合は、ここで指定されたサンプリングタイムでA/D変換が行われます。特殊な場合として、ここに0を入力すると、実時間モデルのステップサイズ毎にA/D変換が行われます。
- PGA amp (0:bypass, 1:use)
ボード上のPGAを使用しているか否を設定します。通常は1を設定してください。AD変換の高速化の為、PGAのバイパス（2.9.3参照）を実施した場合のみ0としてください。

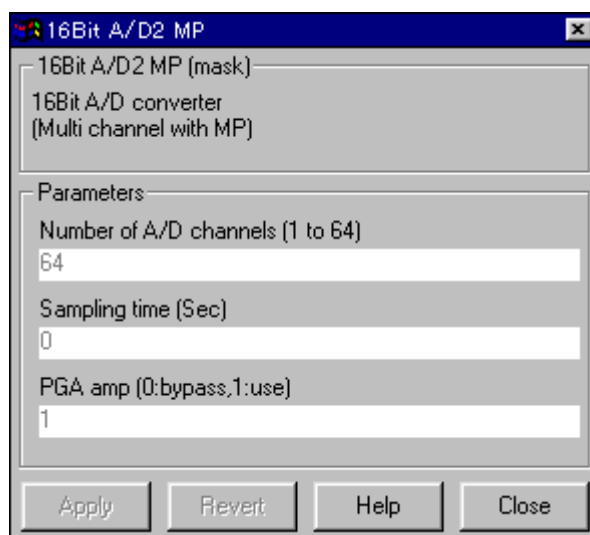
2.9.7 多チャンネルA/D入力（MP付き）デバイスドライバブロック

多チャンネルA/D入力（MP＝マルチプレクサ付き）デバイスドライバブロックは、複数チャンネルのA/Dから一括してデータを入力します。入力の対象となるA/Dチャンネルは、チャンネル番号0から連続するチャンネルで、先頭チャンネル番号は0で固定となっています。このブロックからの出力は、指定されたチャンネル数の幅を持つベクトルとなります。ベクトルの最初の要素がチャンネル番号0からのデータです。

このブロックからの出力データの単位は電圧値[V]です。センサー等の物理単位への変換はデータ入力後 Gain ブロックを使って行なってください。

このデバイスドライバはマルチプレクサー（オプション）が有る場合のみ利用可能です。

図 2.9-5 ADSP32X-13/63 A/D input (M) MP



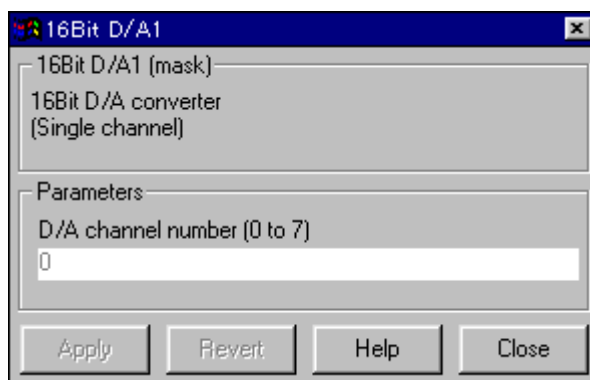
- Number of A/D channels (1 to 64)
入力するA/Dのチャンネル数を1～64の整数で指定します。チャンネル番号とハードウェアとの対応は、表 2.9.2（75ページ）を参照してください。
- Sampling time (Sec)
サンプリングの時間間隔を指定します。単位は秒です。ここで指定したサンプリングタイムと実時間モデルのステップサイズが異なる場合は、ここで指定されたサンプリングタイムでA/D変換が行われます。特殊な場合として、ここに0を入力すると、実時間モデルのステップサイズ毎にA/D変換が行われます。
- PGA amp (0:bypass, 1:use)
ボード上のPGAを使用しているか否を設定します。通常は1を設定してください。AD変換の高速化の為、PGAのバイパス（2.9.3参照）を実施した場合のみ0としてください。

2.9.8 1チャンネルD/A出力デバイスドライバブロック

1チャンネルD/A出力デバイスドライバブロックは、スカラーデータを受取り、指定されたチャンネル番号のD/Aへ出力します。

このブロックへの入力データの単位は電圧値[V]です。物理単位からの変換は Gain ブロック等を使って予め行なってください。

図 2.9-6 ADSP32X-13/63 D/A output (S)



- D/A channel number (0 to 7)

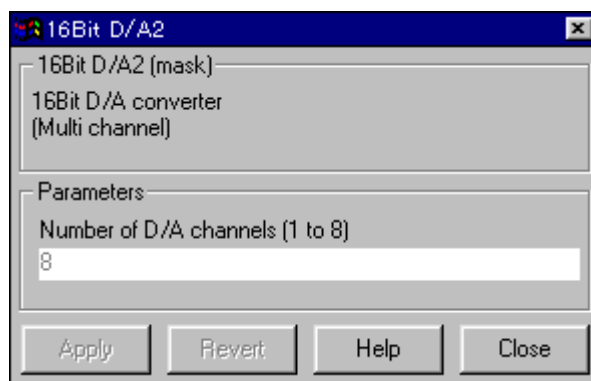
出力するD/Aのチャンネル番号を0～7の整数で指定します。チャンネル番号とハードウェアとの対応は、表 2.9.1 (75ページ) を参照してください。

2.9.9 多チャンネルD/A出力デバイスドライバブロック

多チャンネルD/A出力デバイスドライバブロックは、ベクトルデータを受取り、複数チャンネルのD/Aへ一括して出力します。出力の対象となるD/Aチャンネルは、チャンネル番号0から連続するチャンネルで、先頭チャンネル番号は0で固定となっています。このブロックへの入力ベクトルの幅と、このブロックに指定したチャンネル数とは一致していなければいけません。このブロックへの入力ベクトルの最初の要素がチャンネル番号0のD/Aへ出力されます。

このブロックへの入力データの単位は電圧値[V]です。物理単位からの変換は Gain ブロック等を使って予め行なってください。

図 2.9-7 ADSP32X-13/63 D/A output (M)



- Number of D/A channels (1 to 8)

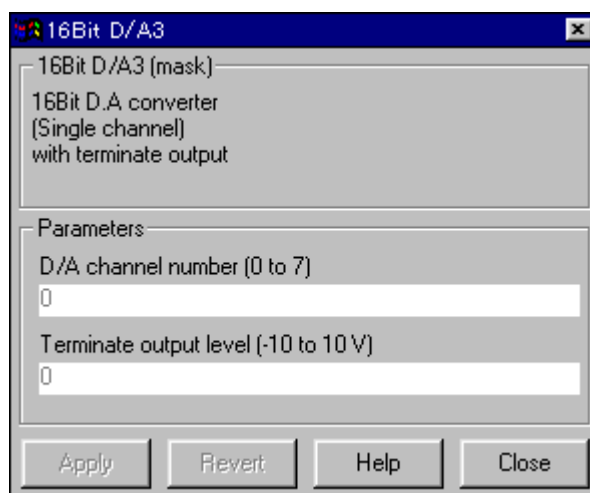
出力するD/Aのチャンネル数を1～8の整数で指定します。チャンネル番号とハードウェアとの対応は、表 2.9.1 (75ページ) を参照してください。

2.9.10 終了時出力指定 1チャンネルD/A出力デバイスドライバブロック

終了時出力指定付き 1チャンネルD/A出力デバイスドライバブロックは、スカラーデータを受取り、指定されたチャンネル番号のD/Aへ出力します。このブロックは実時間モデルが停止する際に、予め指定した電圧値を出力します。

このブロックへの入力データの単位は電圧値[V]です。物理単位からの変換は Gain ブロック等を使って予め行なってください。

図 2.9-8 ADSP32X-13/63 D/A output (S) Terminate output



- D/A channel number (0 to 7)
出力するD/Aのチャンネル番号を0～7の整数で指定します。チャンネル番号とハードウェアとの対応は、表 2.9.1 (75ページ) を参照してください。
- Terminate output data (-10 to 10V)
リアルタイムモデルが終了する際にD/Aから出力する値を設定します。単位は[Volt]です。

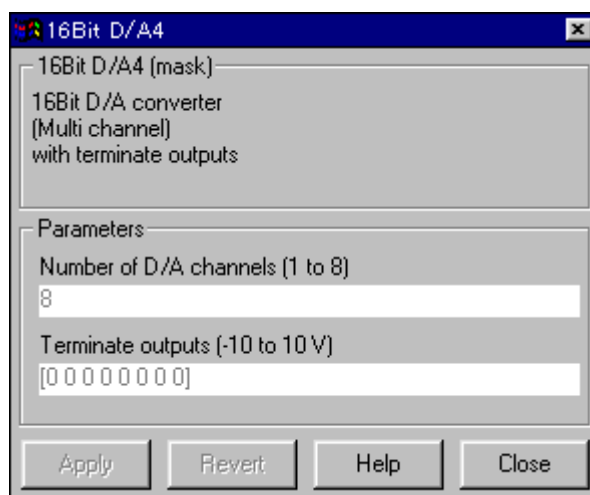
2.9.11 終了時出力指定 多チャンネルD/A出力デバイスドライバブロック

終了時出力指定付き 多チャンネルD/A出力デバイスドライバブロックは、ベクトルデータを受取り、複数チャンネルのD/Aへ一括して出力します。出力の対象となるD/Aチャンネルは、チャンネル番号0から連続するチャンネルで、先頭チャンネル番号は0で固定となっています。このブロックへの入力ベクトルの幅と、このブロックに指定したチャンネル数とは一致していなければいけません。このブロックへの入力ベクトルの最初の要素がチャンネル番号0のD/Aへ出力されます。

このブロックは実時間モデルが停止する際に、予め指定した電圧値を出力します。

このブロックへの入力データの単位は電圧値[V]です。物理単位からの変換は Gain ブロック等を使って予め行なってください。

図 2.9-9 ADSP32X-13/63 D/A output (M) Terminate output



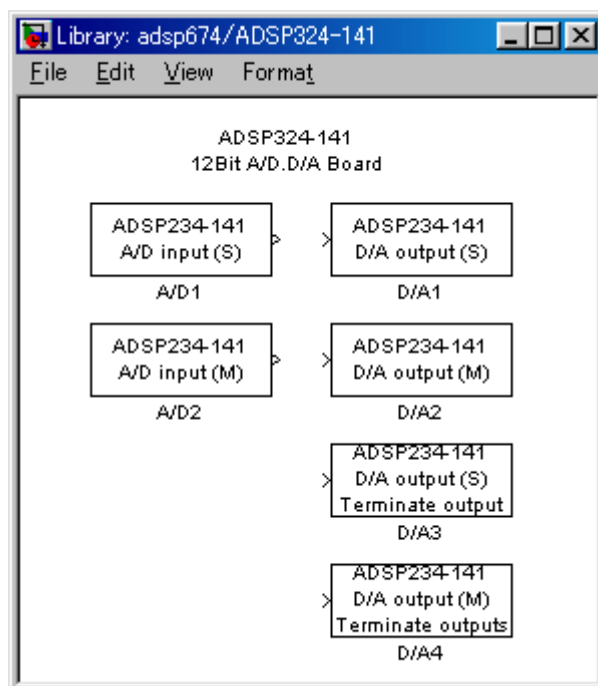
- Number of D/A channels (1 to 8)
出力するD/Aのチャンネル数を1～8の整数で指定します。チャンネル番号とハードウェアとの対応は、表 2.9.1 (75ページ) を参照してください。
- Terminate output data (-10 to 10V)
リアルタイムモデルが終了する際にD/Aから出力する値をベクトルで設定します。単位は[Volt]です。

2.10 ADSP324-141 デバイスドライバブロック

2.10.1 概要

図 2.10-1に、ADSP324-141（12ビットA/D・D/Aボード）用デバイスドライバウィンドウを開いた所を示します。

図 2.10-1 ADSP324-141 ドライバウィンドウ



この中にはADSP324-141とのインターフェースを取るための6つのデバイスドライバブロックが登録されています。

- A/D input (S) 1チャンネルのA/D入力デバイスドライバブロックです。
- A/D input (M) 多チャンネルのA/D入力デバイスドライバブロックです。
- D/A input (S) 1チャンネルのD/A出力デバイスドライバブロックです。
- D/A input (M) 多チャンネルのD/A出力デバイスドライバブロックです。
- D/A input (S) Terminate output
終了時の出力指定が行える、1チャンネルのD/A出力デバイスドライバブロックです。
- D/A input (M) Terminate output
終了時の出力指定が行える、多チャンネルのD/A出力デバイスドライバブロックです。

1 チャンネルの入力（出力）デバイスドライバは、任意のチャンネルからデータを入力（出力）できます。特定のチャンネルのみをアクセスしたい場合や、アクセスしたいチャンネルが連続するチャンネルでない場合に有効です。

多チャンネルの入力（出力）デバイスドライバは、チャンネル番号 0 から連続する任意の数のチャンネルからデータを入力（出力）できます。複数のチャンネルをベクトルとして取り扱う場合や、入力（出力）の為のオーバーヘッドを低減する目的等で連続するチャンネルを一括してアクセスする場合等に利用します。

終了時出力指定の行えるデバイスドライバーは、実時間モデルが停止する際に、予め初期化ダイアログで指定した値を出力します。リアルタイムモニター、U I L、ユーザ定義関数等で強制停止させた場合にも終了時出力は行われます。それ以外のブロックは実時間モデルが停止する直前の値を出力し続けます。

1 チャンネル版と多チャンネル版は一度に取り扱うチャンネルの数やチャンネル番号の指定が異なるのみで、本質的には全く差異はありません。例えば、多チャンネル版で入力した複数チャンネルの A/D 入力データを Demux ブロックで個別の信号に分解したものと、1 チャンネル版で個々に入力したものととは全く同一として取り扱えます。D/A の場合も同様です。特に、A/D では、同じサンプリング時間を設定した複数のデバイスドライバを使用した場合のサンプリングの同時性は保証されています。1 チャンネル版と多チャンネル版のドライバを混在した場合でも同様です。

2.10.2 チャンネル番号とボードとの対応

ADSP324-141は最大4台使用でき、1台のボードにA/D、D/Aが各16及び8チャンネル（オプション実装時12チャンネル）実装されています。ADSP324-141の各デバイスドライバブロックはこれらのチャンネルを連続した論理チャンネル番号として取り扱います。論理チャンネル番号とハードウェアとの対応は、表 2.10.1 及び 表 2.10.2 の通りです。ボードの1～4台目の区別についてはハードウェアのセットアップ（6ページ）を参照してください。

表 2.10.1 入力チャンネル番号とADSP324-141ボードとの対応

ボード	IN0	IN1	...	IN14	IN15
1 台目	0	1	...	14	15
2 台目	16	17	...	30	31
3 台目	32	33	...	46	47
4 台目	48	49	...	62	63

表 2.10.2 出力チャンネル番号とADSP324-141ボードとの対応

(D/A 標準実装の場合)

ボード	OUT0	OUT1	...	OUT6	OUT7
1 台目	0	1	...	6	7
2 台目	8	9	...	14	15
3 台目	16	17	...	22	23
4 台目	24	26	...	30	31

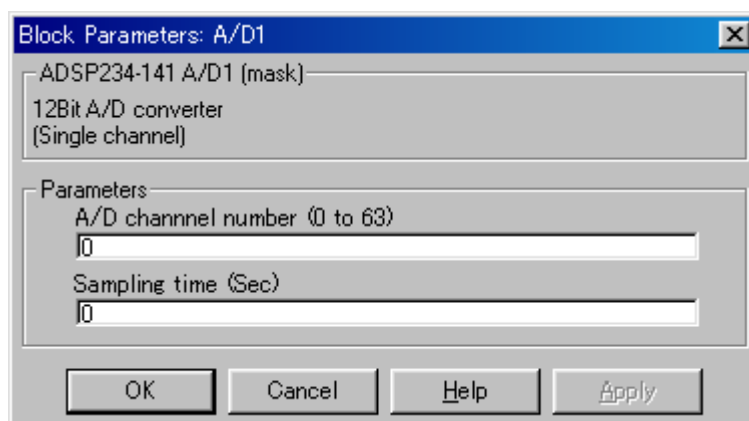
(D/A の CH 増設オプション有りの場合)

ボード	OUT0	OUT1	...	OUT10	OUT11
1 台目	0	1	...	10	11
2 台目	12	13	...	22	23
3 台目	24	25	...	34	35
4 台目	36	37	...	46	47

2.10.3 1チャンネルA/D入力デバイスドライバブロック

1チャンネルA/D入力デバイスドライバブロックは、指定されたチャンネル番号のA/Dからデータを入力します。このブロックからの出力はスカラーとなります。このブロックからの出力データの単位は電圧値 [V] です。センサー等の物理単位への変換はデータ入力後 Gain ブロックを使って行なってください。

図 2.10-2 ADSP324-141 A/D input (S)



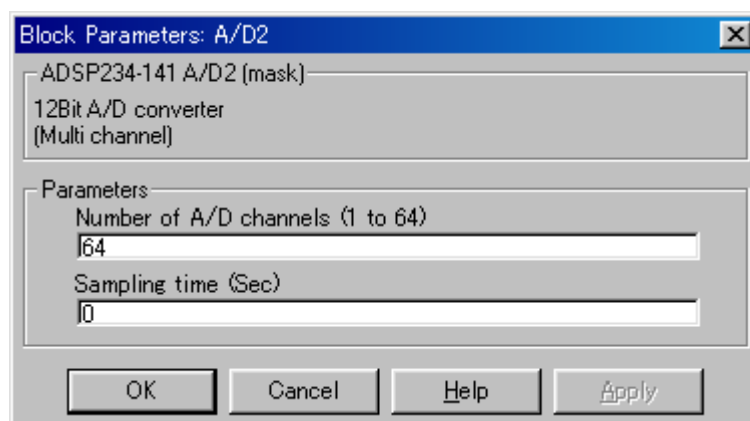
- A/D channel number (0 to 63)
入力するA/Dのチャンネル番号を0～63の整数で指定します。チャンネル番号とハードウェアとの対応は、表 2.10.1 (87ページ) を参照してください。
- Sampling time (Sec)
サンプリングの時間間隔を指定します。単位は秒です。ここで指定したサンプリングタイムと実時間モデルのステップサイズが異なる場合は、ここで指定されたサンプリングタイムでA/D変換が行われます。特殊な場合として、ここに0を入力すると、実時間モデルのステップサイズ毎にA/D変換が行われます。

2.10.4 多チャンネルA/D入力デバイスドライバブロック

多チャンネルA/D入力デバイスドライバブロックは、複数チャンネルのA/Dから一括してデータを入力します。入力の対象となるA/Dチャンネルは、チャンネル番号0から連続するチャンネルで、先頭チャンネル番号は0で固定となっています。このブロックからの出力は、指定されたチャンネル数の幅を持つベクトルとなります。ベクトルの最初の要素がチャンネル番号0からのデータです。

このブロックからの出力データの単位は電圧値 [V] です。センサー等の物理単位への変換はデータ入力後 Gain ブロックを使って行なってください。

図 2.10-3 ADSP324-141 A/D input (M)



- Number of A/D channels (1 to 64)

入力するA/Dのチャンネル数を1～64の整数で指定します。チャンネル番号とハードウェアとの対応は、表 2.10.1 (87ページ) を参照してください。

- Sampling time (Sec)

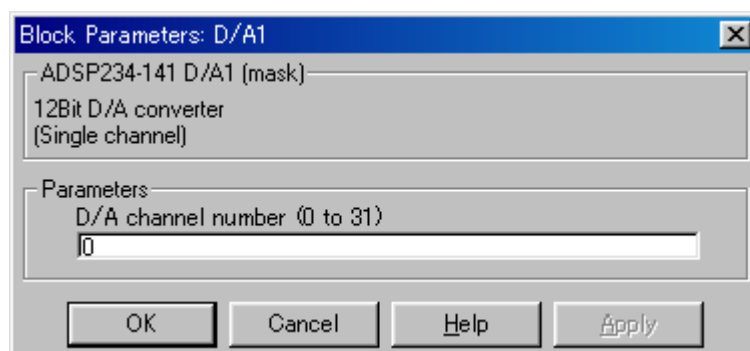
サンプリングの時間間隔を指定します。単位は秒です。ここで指定したサンプリングタイムと実時間モデルのステップサイズが異なる場合は、ここで指定されたサンプリングタイムでA/D変換が行われます。特殊な場合として、ここに0を入力すると、実時間モデルのステップサイズ毎にA/D変換が行われます。

2.10.5 1チャンネルD/A出力デバイスドライバブロック

1チャンネルD/A出力デバイスドライバブロックは、スカラーデータを受取り、指定されたチャンネル番号のD/Aへ出力します。

このブロックへの入力データの単位は電圧値[V]です。物理単位からの変換は Gain ブロック等を使って予め行なってください。

図 2.10-4 ADSP324-141 D/A output (S)



- D/A channel number (0 to 31)

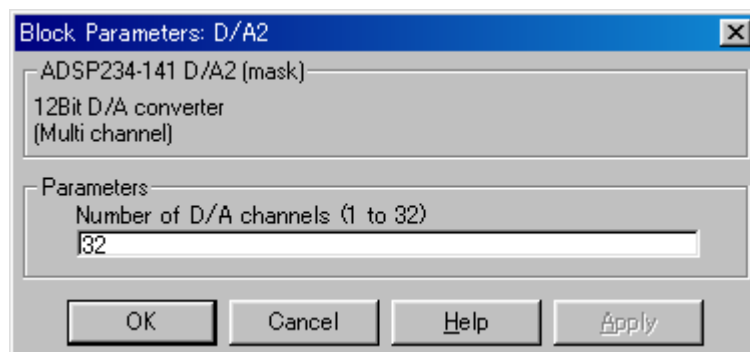
出力するD/Aのチャンネル番号を0～31の整数で指定します。チャンネル番号とハードウェアとの対応は、表 2.10.2 (87ページ) を参照してください。

2.10.6 多チャンネルD/A出力デバイスドライバブロック

多チャンネルD/A出力デバイスドライバブロックは、ベクトルデータを受取り、複数チャンネルのD/Aへ一括して出力します。出力の対象となるD/Aチャンネルは、チャンネル番号0から連続するチャンネルで、先頭チャンネル番号は0で固定となっています。このブロックへの入力ベクトルの幅と、このブロックに指定したチャンネル数とは一致していなければいけません。このブロックへ入力されるベクトルの、最初の要素がチャンネル番号0のD/Aへ出力されます。

このブロックへの入力データの単位は電圧値[V]です。物理単位からの変換は Gain ブロック等を使って予め行なってください。

図 2.10-5 ADSP324-141 D/A output (M)



- Number of A/D channels (1 to 32)

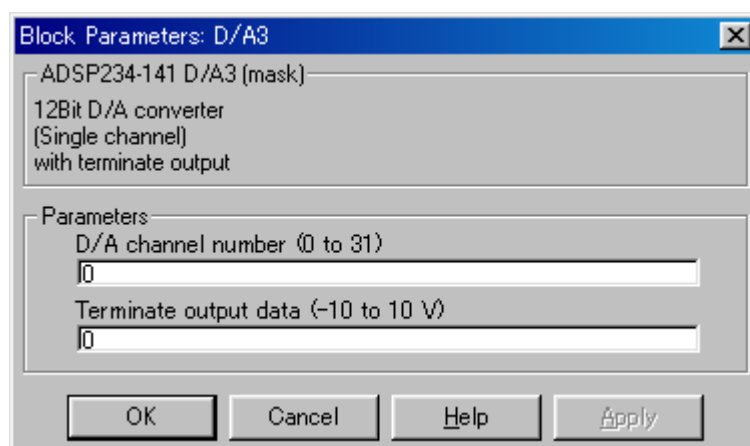
出力するD/Aのチャンネル数を1～32の整数で指定します。チャンネル番号とハードウェアとの対応は、表 2.10.2 (87ページ) を参照してください。

2.10.7 終了時出力指定 1 チャンネルD/A出力デバイスドライバブロック

終了時出力指定付き 1 チャンネルD/A出力デバイスドライバブロックは、スカラーデータを受取り、指定されたチャンネル番号のD/Aへ出力します。このブロックは実時間モデルが停止する際に、予め指定した電圧値を出力します。

このブロックへの入力データの単位は電圧値[V]です。物理単位からの変換は Gain ブロック等を使って予め行なってください。

図 2.10-6 ADSP324-141 output (S) Terminate output



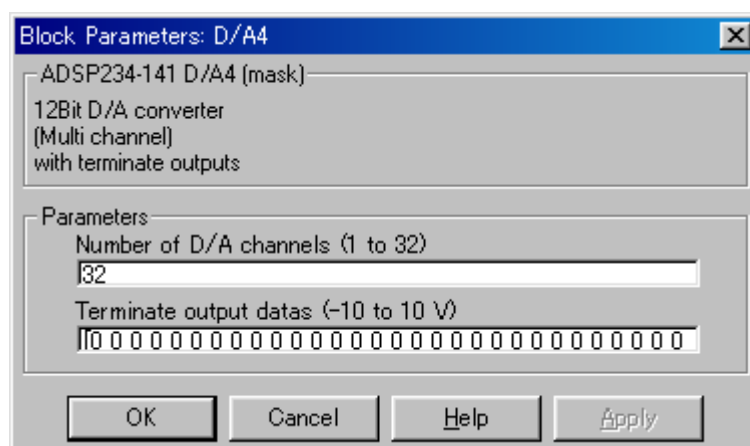
- D/A channel number (0 to 31)
出力するD/Aのチャンネル番号を0～31の整数で指定します。チャンネル番号とハードウェアとの対応は、表 2.10.2 (87ページ) を参照してください。
- Terminate output data (-10 to 10V)
リアルタイムモデルが終了する際にD/Aから出力する値を設定します。単位は[Volt]です。

2.10.8 終了時出力指定 多チャンネルD/A出力デバイスドライバブロック

終了時出力指定付き多チャンネルD/A出力デバイスドライバブロックは、ベクトルデータを受取り、複数チャンネルのD/Aへ一括して出力します。先頭チャンネル番号は0で固定となっています。このブロックへの入力ベクトルの幅と、このブロックに指定したチャンネル数とは一致していなければいけません。このブロックへ入力されるベクトルの、最初の要素がチャンネル番号0のD/Aへ出力されます。このブロックはモデルが停止する際に、予め指定した電圧値を出力します。

このブロックへの入力データの単位は電圧値[V]です。物理単位からの変換は Gain ブロック等を使って予め行なってください。

図 2.10-7 ADSP324-141 D/A output (M) Terminate output



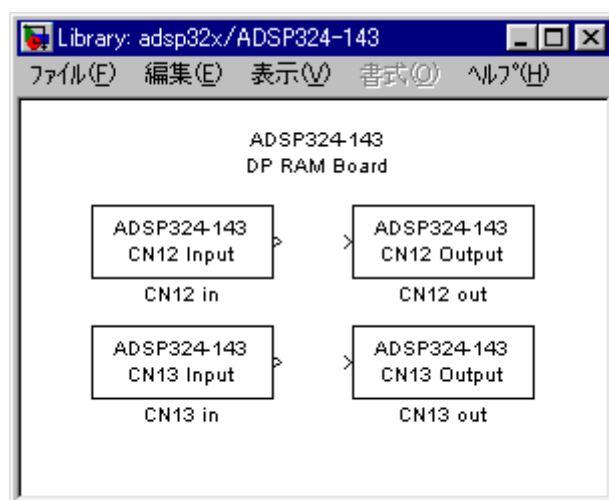
- Number of D/A channels (1 to 32)
出力するD/Aのチャンネル数を1～32の整数で指定します。先頭チャンネル番号は0で固定です。チャンネル番号とハードウェアとの対応は、表 2.10.2 (87ページ) を参照してください。
- Terminate output data (-10 to 10V)
リアルタイムモデルが終了する際にD/Aから出力する値をベクトルで設定します。単位は[Volt]です。

2.11 ADSP324-143 デバイスドライバブロック

2.11.1 概要

図 2.11-1に、ADSP324-143 (DP RAMボード) 用デバイスドライバウィンドウを開いた所を示します。

図 2.11-1 ADSP324-143 ドライバウィンドウ



この中にはADSP324-143とのインターフェースを取るための4つのデバイスドライバブロックが登録されています。

- CN12 input 前段 DSP との間にある DP RAMを使い、前段 DSP からデータを受け取るデバイスドライバブロックです。
- CN13 input 後段 DSP との間にある DP RAMを使い、後段 DSP からデータを受け取るデバイスドライバブロックです。
- CN12 output 前段 DSP との間にある DP RAMを使い、前段 DSP へデータを渡すデバイスドライバブロックです。
- CN13 output 後段 DSP との間にある DP RAMを使い、後段 DSP へデータを渡すデバイスドライバブロックです。

*前段・後段DSPについては、“DSPの前段と後段の意味”（101ページ）を参照してください。

2.11.2 コネクタとボードとのメモリマップ

ADSP324-143は、DSPボード1枚に対して最大4台使用できます。コネクタ番号とDSPボードのメモリ対応は、表 2.11.1の通りです。ボードの1～4台目の区別についてはハードウェアのセットアップ（6ページ）を参照してください。

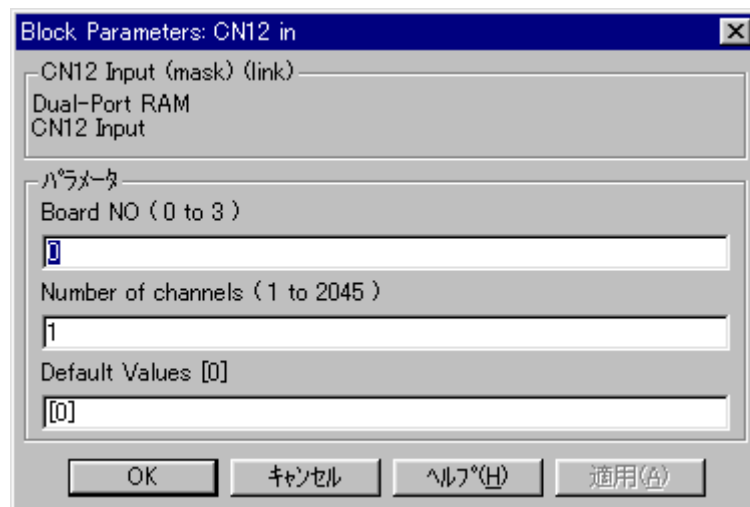
表 2.11.1 コネクタとADSP324-143ボードメモリマップ

ボード	ボード番号	C N 1 2	C N 1 3
1 台目	0	908000 h ～908FFF h	909000 h ～909FFF h
2 台目	1	90C000h～90CFFF h	90D000h～90DFFF h
3 台目	2	910000 h ～910FFF h	911000 h ～911FFF h
4 台目	3	914000 h ～914FFF h	915000 h ～915FFF h

2.11.3 CN12 Inputデバイスドライバブロック

CN12 Inputデバイスドライバは、コネクタ12を使用し、DSPとDSPを接続するパイプラインボード（オプション）を用いて、前段DSPとの間で非同期にデータ交換を行うためのブロックです。マルチプロセッシングを構成した場合、DSPボード間の通信をホストコンピュータのバスを介さずに直接DSPボード間で交信することができます。

図 2.11-2 ADSP324-143 CN12 Input



- Board NO (0 to 3)
ADSP324-143 のボード番号を 0 ～ 3 の整数で指定します。ボード番号とハードウェアとの対応は、表 2.11.1 (95ページ) を参照してください。
- Number of channels (1 to 2045)
相手 DSP から受け取るデータのチャンネル数を設定します。データはここで設定した幅のベクトルとしてこのブロックから出力されます。
- Default Values [0]
相手 DSP が未だ一度もデータを送信していない段階でこのブロックから出力される値を設定します。第二引数で指定した点数の幅をもつ定数ベクトル又は空ベクトル ([]) を与えます。空ベクトル時は、第二引数の幅のデータ 0 ベクトルを与えたことと同一になります。

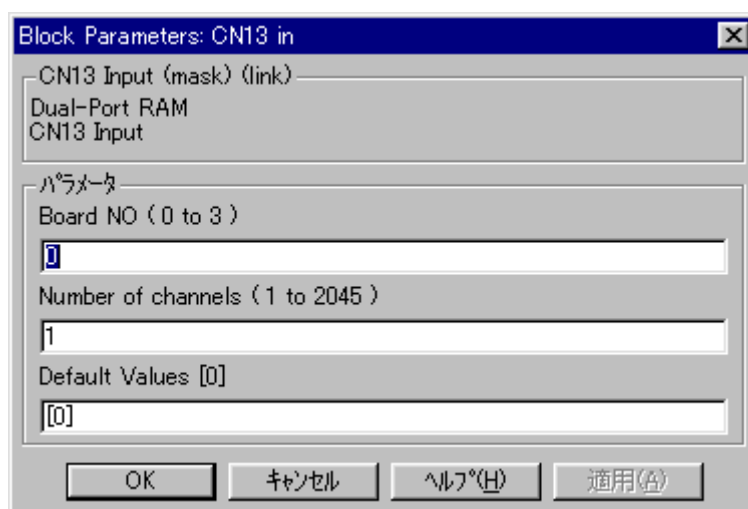
複数のデータを渡す場合の、データの同時性は保証されています。詳しくは、“DSP間DSP通信アービトラーション” (100ページ) を参照してください。

又、前段・後段DSPについては、“DSPの前段と後段の意味” (101ページ) を参照してください。

2.11.4 CN13 Inputデバイスドライバブロック

CN13 Inputデバイスドライバは、コネクタ13を使用し、DSPとDSPを接続するパイプラインボード（オプション）を用いて、後段DSPとの間で非同期にデータ交換を行うためのブロックです。マルチプロセッシングを構成した場合、DSPボード間の通信をホストコンピュータのバスを介さずに直接DSPボード間で交信することができます。

図 2.11-3ADSP324-143 CN13 Input



- Board NO (0 to 3)
ADSP324-143 のボード番号を 0 ～ 3 の整数で指定します。ボード番号とハードウェアとの対応は、表 2.11.1 (95ページ) を参照してください。
- Number of channels (1 to 2045)
相手 DSP から受け取るデータのチャンネル数を設定します。データはここで設定した幅のベクトルとしてこのブロックから出力されます。
- Default Values [0]
相手 DSP が未だ一度もデータを送信していない段階でこのブロックから出力される値を設定します。第二引数で指定した点数の幅をもつ定数ベクトル又は空ベクトル ([]) を与えます。空ベクトル時は、第二引数の幅のデータ 0 ベクトルを与えたことと同一になります。

複数のデータを渡す場合の、データの同時性は保証されています。詳しくは、“DSP間DSP通信アービトラーション” (100ページ) を参照してください。

又、前段・後段DSPについては、“DSPの前段と後段の意味” (101ページ) を参照してください。

2.11.5 CN12 Output デバイスドライバブロック

CN12 Output デバイスドライバは、コネクタ12を使用し、DSPとDSPを接続するパイプラインボード（オプション）を用いて、前段DSPとの間で非同期にデータ交換を行うためのブロックです。マルチプロセッシングを構成した場合、DSPボード間の通信をホストコンピュータのバスを介さずに直接DSPボード間で交信することができます。

図 2.11-4ADSP324-143 CN12 Output



- Board NO (0 to 3)
ADSP324-143 のボード番号を 0 ～ 3 の整数で指定します。ボード番号とハードウェアとの対応は、表 2.11.1 (95ページ) を参照してください。
- Number of channels (1 to 2045)
相手 DSP へ渡すデータのチャンネル数を設定します。データはここで設定した幅のベクトルをこのブロックに入力します。

複数のデータを渡す場合の、データの同時性は保証されています。詳しくは、“DSP間DSP通信アービトラレーション” (100ページ) を参照してください。

又、前段・後段DSPについては、“DSPの前段と後段の意味” (101ページ) を参照してください。

2.11.6 CN13 Output デバイスドライバブロック

CN13 Output デバイスドライバは、コネクタ13を使用し、DSPとDSPを接続するパイプラインボード（オプション）を用いて、後段DSPとの間で非同期にデータ交換を行うためのブロックです。マルチプロセッシングを構成した場合、DSPボード間の通信をホストコンピュータのバスを介さずに直接DSPボード間で交信することができます。

図 2.11-5 ADSP324-143 CN13 Output



- Board NO (0 to 3)
ADSP324-143 のボード番号を 0 ～ 3 の整数で指定します。ボード番号とハードウェアとの対応は、表 2.11.1 (95ページ) を参照してください。
- Number of channels (1 to 2045)
相手 DSP へ渡すデータのチャンネル数を設定します。データはここで設定した幅のベクトルをこのブロックに入力します。

複数のデータを渡す場合の、データの同時性は保証されています。詳しくは、“DSP間DSP通信アービトラレーション” (100ページ) を参照してください。

又、前段・後段DSPについては、“DSPの前段と後段の意味” (101ページ) を参照してください。

2.11.7 D S P 間 D P 通信アービトレーション

D S P 間の D P R A M 通信デバイスブロックによる通信は、データの同時性（ブロックから取り出したデータ（送ったデータ）が全て同一時刻である事）を保証しています。つまり、「データの前半が後半より 1 時刻（ステップサイズ）分、昔のデータであった」等ということが無いように、送る側と受ける側でフラグを用いたソフトウェア同期処理を行っています。

これは、複数の D S P が異なる周期で計算を行う、非同期処理間でデータを交換する為には必要な機構です。

本ソフトウェアでは、データの同時性は、以下の考え方によりアービトレーションしています。

1、書き込み側の要求が先に発生した場合

読み出し側は書き込み側が完了するまで待機して後読み出す。

2、読み出し側の要求が先に発生した場合

書き込み側は読み出し側が完了するまで待機して後書き込む。

3、両方の要求が同時に発生した場合

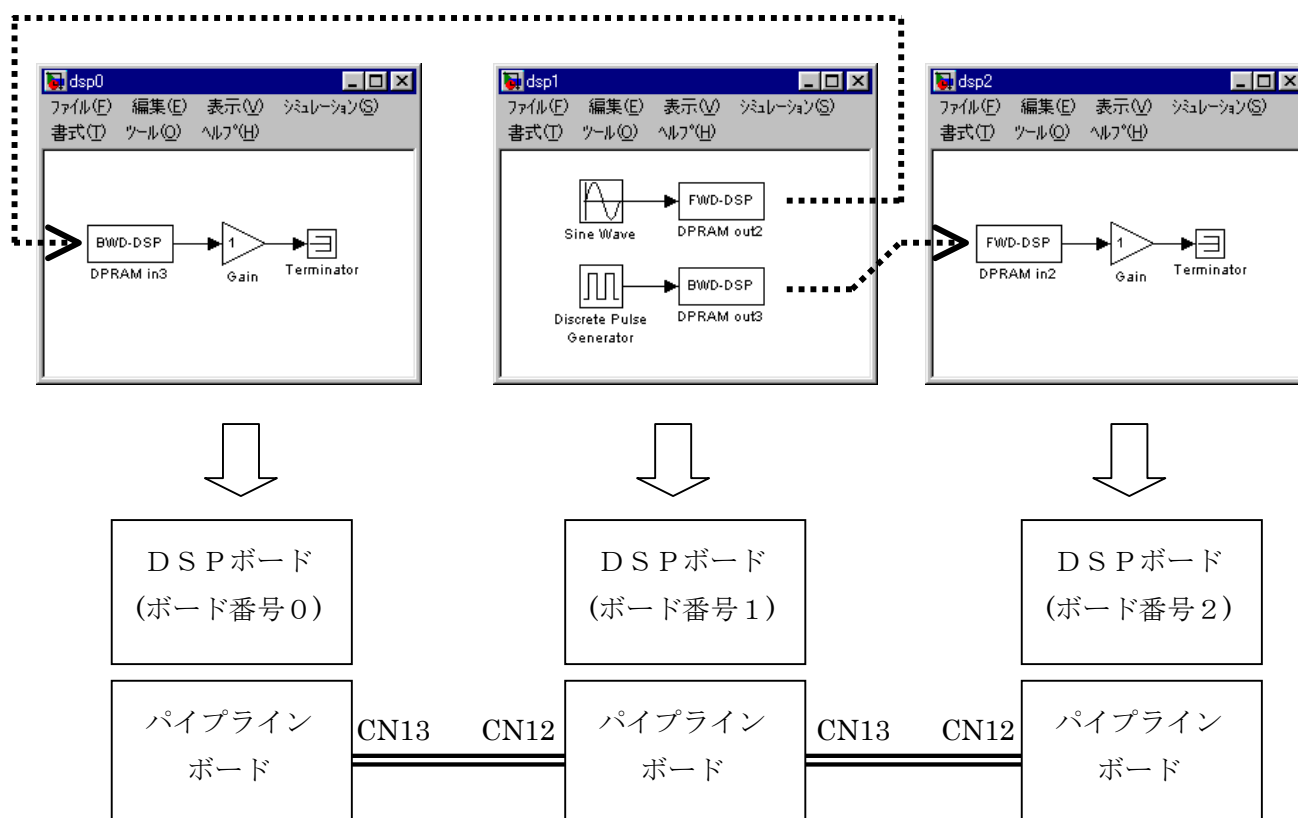
読み出し側は書き込み側が完了するまで待機して後読み出す。

上記 3 項は、同時の場合は、最新のデータを渡す為、読み出し側が待機します。

2.11.8 DSPの前段と後段の意味

DSP間のDPRAM通信デバイスブロックには、前段（FWD）と後段（BWD）があります。前段とは当該DSPボードに接続されているパイプラインボード（ADSP324-143等）のコネクターCN12の先に接続されたDSPボードであり、後段とは同CN13の先に接続されているDSPボードの事を表します。つまり、前段・後段の関係は、パイプラインボードの接続がどうなっているかにより決定され、個々のDSPボードのボード番号には全く関係ありません。

以下に簡単な例を示します。説明の便宜上各DSPにはボード番号0、1、2と番号を付けたとします。下の図では、真中のDSPボード（ボード番号1）から見て前段とは、同ボードに接続されたパイプラインボードのCN12に接続されたDSPボードですから、ボード番号0が前段になり、同様に後段は、同ボードに接続されたパイプラインボードのCN13に接続されたDSPボードですから、ボード番号2が後段になります。図のブロック線図を結ぶ破線矢印  は信号の流れを表しています。

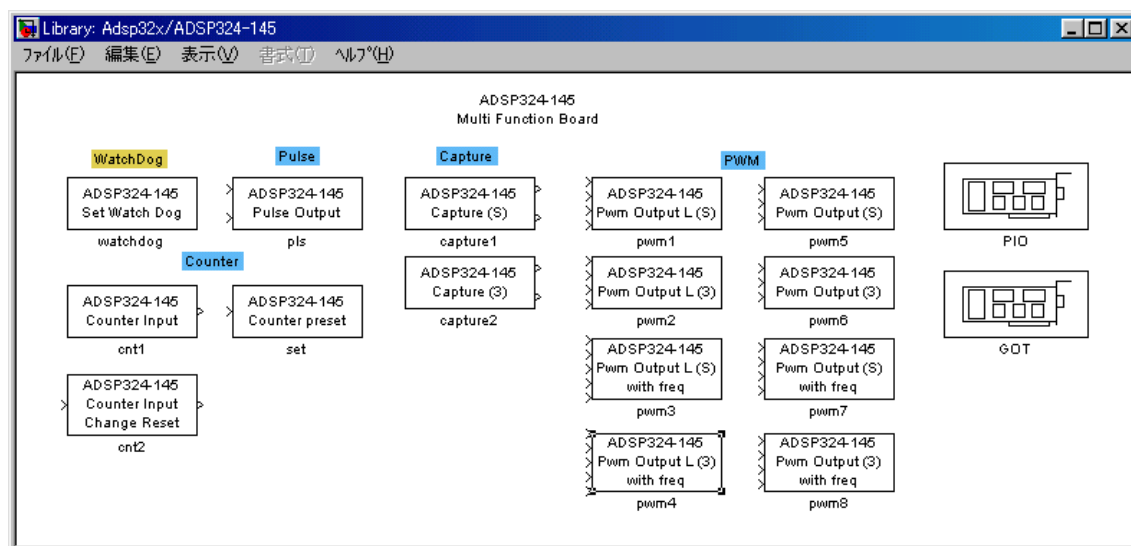


2.12 ADSP324-145 デバイスドライバブロック

2.12.1 概要

図 2.12-1に、ADSP324-145 用デバイスドライバウィンドウを開いた所を示します。

図 2.12-1 ADSP324-145 ドライバウィンドウ



この中にはADSP324-145とのインターフェースを取るための22のデバイスドライバブロックが登録されています。

- watch dog ウォッチドグタイマ
- Pulse Output パルス出力デバイスドライバです。
- Counter Input プリセット付カウンタデバイスドライバです。
- Counter preset カウンタプリセットドライバです。
- Capture[S] 単相モードのキャプチャデバイスドライバです。
- Capture[3] 3相モードのキャプチャデバイスドライバです。
- Pwm Output[S] 単相モードのPWM出力デバイスドライバです。
- Pwm Output[3] 3相モードのPWM出力デバイスドライバです。
- PIO Set Time PIO のハートリング 信号の ON/遅延時間設定デバイスドライバです。
- PIO bit In[x] ビット入力デバイスドライバです。
- PIO Paralell In[x] 複数ビット同時入力デバイスドライバです。
- PIO bit Out[x] ビット出力デバイスドライバです。
- PIO Paralell Out[x] 複数ビット同時出力デバイスドライバです。
- GOT GOT 操作用デバイスドライバです。

2.12.2 コネクタとボードとのメモリマップ

ADSP324-145は、DSPボード1枚に対して最大4台使用できます。コネクタ番号とDSPボードのメモリ対応は、表 2.12.1 の通りです。ボードの1～4台目の区別についてはハードウェアのセットアップ (6ページ) を参照してください。

表 2.12.1 コネクタとADSP324-145ボードメモリマップ

ボード	ボード番号	ADSP324-145	
		ADSP324-00A	ADSP674 シリーズ
1 台目	0	00901000h	03004000h
2 台目	1	00901200h	03004800h
3 台目	2	00901400h	03005000h
4 台目	3	00901600h	03005800h

表 2.12.2 ADSP324-145カウンタチャンネル番号対応

ボード	カウンタチャンネル番号				絶縁ポート番号
	カウンタ 0	カウンタ 1	カウンタ 2	カウンタ 3	
1 台目	0	1	2	3	0
2 台目	4	5	6	7	1
3 台目	8	9	10	11	2
4 台目	12	13	14	15	3

表 2.12.3 ADSP324-145キャプチャ・PWM・パルスチャンネル番号対応

ボード	キャプチャ・PWM・パルスチャンネル番号			
	CH0	CH1	CH2	CH3
1 台目	0	1	2	3
2 台目	4	5	6	7
3 台目	8	9	10	11
4 台目	12	13	14	15

表 2.12.4 ADSP324-145PIOポート番号対応

ボード	ポート0 OUT	ポート0 IN	ポート2 OUT	ポート2 IN
1 台目	0	1	2	3
2 台目	4	5	6	7
3 台目	8	9	10	11
4 台目	12	13	14	15

表 2.12.5 ADSP324-145ボードPIOとビット番号の対応

入出力信号名	ビット番号 注1	パラレル 注2	ポート単位 注3
SIG031 / SIG331	31	最後	2^{31}
SIG030 / SIG330	30	最後の1つ前	2^{30}
・	・	・	・
・	・	・	・
SIG006 / SIG306	6	・	2^6
SIG005 / SIG305	5	・	2^5
SIG004 / SIG304	4	・	2^4
SIG003 / SIG303	3	・	2^3
SIG002 / SIG302	2	・	2^2
SIG001 / SIG301	1	先頭の次	2^1
SIG000 / SIG300	0	先頭	2^0

注1：ビット単位の入出力の場合、各ブロックの初期化ダイアログで指定するビット番号に相当します。

注2：パラレル入出力の場合、ブロック線図との間で受け渡すベクトルの各要素と入出力信号との関係は表の通りとなります。終了処理付きの場合の終了値も同様です。

注3：ポート単位の入出力の場合、各ブロックとブロック線図との間で受け渡すデータは、各ビットをここに示した重みで加算された値となります。

表 2.12.6 PIOハンドリング方式

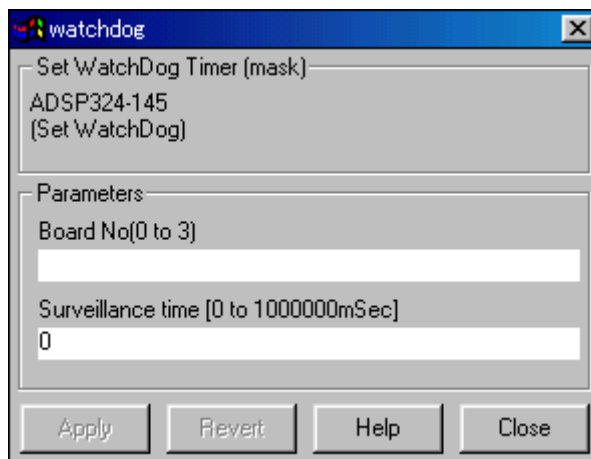
	ハンドリング方式
0	ハンドリングなし
1	ホールド方式
2	Strobe / Ack 方式
3	Write Strobe 方式

注1：ハンドリング方式の詳細は、“ADSP324-145ハードウェア・ユーザーズ・マニュアル” 機能説明（パラレル入出力 ハンドリング方式）を参照してください。

2. 12. 3 WatchDog デバイスドライバブロック

WatchDog デバイスドライバーは、ウォッチドグタイマを使用可能にします。

図 2. 12-2 ADSP324-145 WatchDog

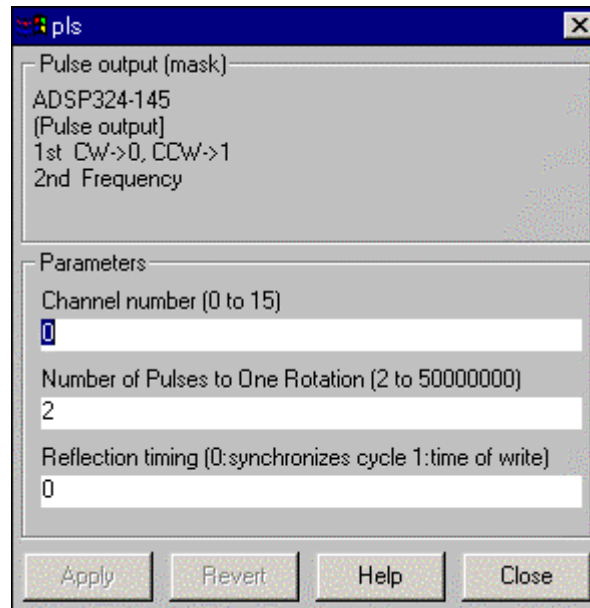


- Board NO (0 to 3)
ADSP324-145 のボード番号を 0 ～ 3 の整数で指定します。ボード番号とハードウェアとの対応は、表 2. 12. 1 (103 ページ) を参照してください。
- Surveillance time [0 to 1000000mSec]
ウォッチドグタイマの監視周期時間を 0 ～ 1000000 mSec で設定してください。

2.12.4 Pulse output デバイスドライバブロック

Pulse output デバイスドライバは、第一入力として、回転方向（0：正転＜CW＞、1 逆転＜CCW＞）、第二入力としてパルス周波数（0.01Hz～6MHz）を入力すると、指定チャンネルへパルス出力します。

図 2.12-3 ADSP324-145 Pulse output

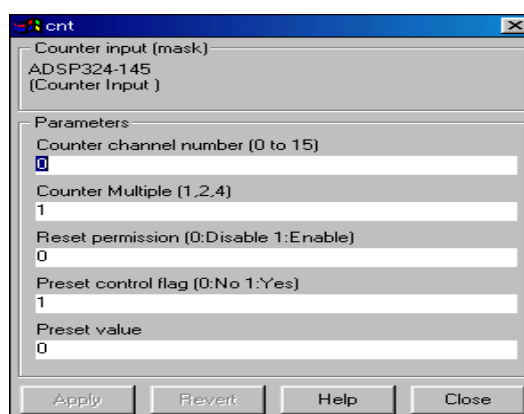


- Channel number (0 to 15)
ADSP324-145 のパルスチャンネル番号を 0～15 の整数で指定します。チャンネル番号とハードウェアとの対応は、表 2.12.3を参照してください。
- Number of Pulses to One Rotation [1 to 50000000]
1 回転のパルス数を指定します。パルス数とは、Z 相の出力を A・B 相パルスの何パルスに 1 回出力するかを設定します。
- Reflection timing [0: synchronizes cycle 1: time of write]
パルス設定値の反映タイミングを 0：周波数に同期、1：書込み時のいずれかを指定します。

2.12.5 Counter Input デバイスドライバブロック

Counter Input デバイスドライバブロックは、任意のチャンネル番号のカウンターの値を読み込みます。このブロックの出力はスカラーです。ADSP324-145 ボードの指定されたカウンターは符号付き 32 ビット整数として読み込まれ、スカラーに変換されて出力されます。プリセットの指定を行なった場合、初期化の際に指定した値にプリセットされます。プリセットは初期化の際に 1 回だけ行なわれます。よりセットを有効にした場合リセット条件が満たされた時カウンタがリセットされます。

図 2.12-4 ADSP324-145 Counter Input

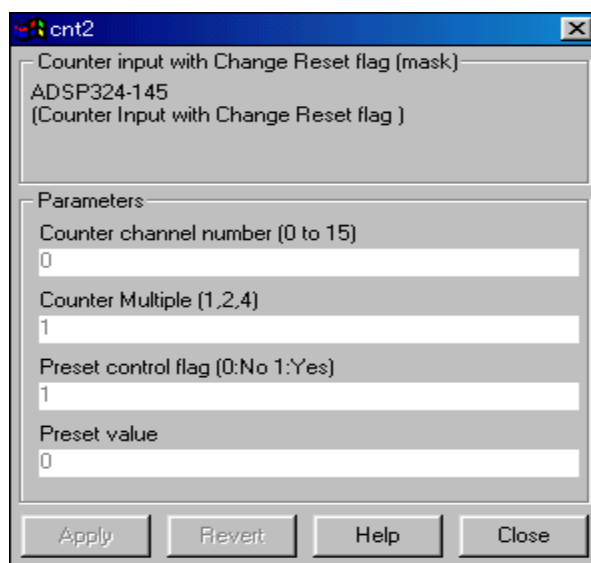


- Counter channel number (0 to 15)
入力の対象となるカウンタチャンネル番号を指定します。チャンネル番号とハードウェアとの対応は、表 2.12.2を参照してください。
- Counter Multiple (1,2,4)
カウンタの通倍を 1 : $\times 1$ 、2 : $\times 2$ 、4 : $\times 4$ のいずれかより選択してください。
- Reset permission (0:Disable 1: Enable)
カウンタリセット信号の許可を設定してください。(0 : 未許可、1 : 許可)
詳細は、ADSP324-145 ハードウェアマニュアルを参照してください。
- Preset controll flag (0:No 1:Yes)
初期化時にカウンタ値のプリセットを行なうか否かを決定します。0 又は 1 を設定してください。0 を設定した場合はプリセットは行ないません。1 を設定した場合は Preset value に設定した値がカウンタに初期化の際（シミュレーションの開始時）に 1 度だけ設定されます。
- Preset value
Preset controll flag に 1 を設定した場合、ここに設定した値がカウンタの初期値として採用されます。設定できる値は、符号付き 32 ビット整数（ストレートバイナリ）で表わせる範囲（ $-2^{31} \sim 2^{31}-1$ ）です。

2.12.6 Counter Input with Change reset flag デバイスドライバブロック

Counter Input with Change Reset flag デバイスドライバブロックは、任意のチャンネル番号のカウンターの値を読み込みます。リセット許可（0：未許可 1：許可）をスカラーで入力します。このブロックの出力はスカラーです。ADSP324-145 ボードの指定されたカウンターは符号付き 32 ビット整数として読み込まれ、スカラーに変換されて出力されます。プリセットの指定を行なった場合、初期化の際に指定した値にプリセットされます。プリセットは初期化の際に 1 回だけ行なわれます。

図 2.12-5 ADSP324-145 Counter Input with Change Reset flag



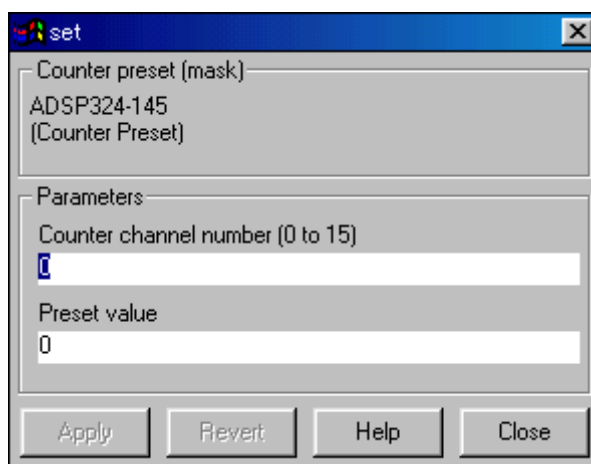
- Counter channel number (0 to 15)
入力の対象となるカウンタチャンネル番号を指定します。チャンネル番号とハードウェアとの対応は、表 2.12.2を参照してください。
- Counter Multiple (1,2,4)
カウンタの通倍を 1：×1、2：×2、4：×4 のいずれかより選択してください。
- Preset control flag (0:No 1:Yes)
初期化時にカウンタ値のプリセットを行なうか否かを決定します。0又は1を設定してください。0を設定した場合はプリセットは行ないません。1を設定した場合は Preset value に設定した値がカウンタに初期化の際（シミュレーションの開始時）に1度だけ設定されます。
- Preset value
Preset control flag に1を設定した場合、ここに設定した値がカウンタの初期値として採用されます。設定できる値は、符号付き32ビット整数（ストレートバイナリ）で表わせる範囲（ $-2^{31} \sim 2^{31}-1$ ）です。

2.12.7 Counter preset デバイスドライバブロック

Counter preset デバイスドライバブロックは、任意のチャンネル番号のカウンターを任意の時期にプリセットするデバイスドライバです。プリセット付きカウンタデバイスドライバブロックでは、初期化の際に1度しかプリセットできませんが、このブロックを使用すれば任意の時期に値をプリセットできます。

このブロックには1つのスカラー入力があり、ここに0又は1の値を入力することによりプリセットタイミングを制御します。0が入力されている期間はカウンタはカウント動作を行いません。1が入力されるとカウント動作は停止し Preset value で予め設定した値がカウンタにセットされます。

図 2.12-6 ADSP324-145 Counter preset

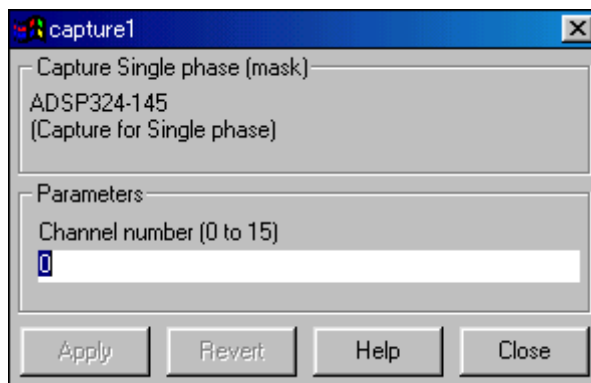


- Counter channel number (0 to 15)
対象となるカウンタチャンネル番号を指定します。チャンネル番号とハードウェアとの対応は、表 2.12.2を参照してください。
- Preset value
ここに設定した値がカウンタにセットされます。設定できる値は、符号付32ビット整数（ストレートバイナリ）で表せる範囲（ $-2^{31} \sim 2^{31} - 1$ ）です。

2. 12. 8 Capture(S) デバイスドライバブロック

Capture(S)デバイスドライバは、指定チャンネルの単相信号を計測し、第一出力に周期(周波数)、第二出力にUV ON時間(Duty比 100.0～100.0)を出力します。

図 2. 12-7 ADSP324-145 Capture [S]



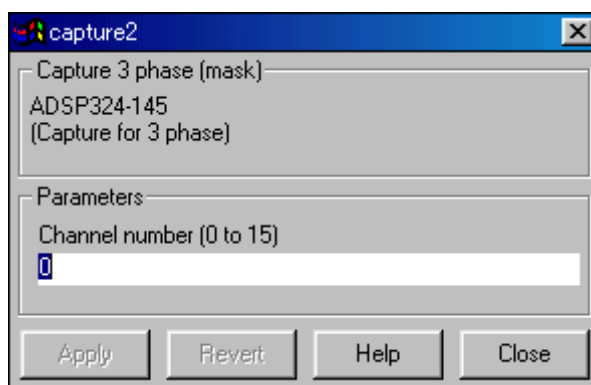
- Channel number (0 to 15)

ADSP324-145 のキャプチャチャンネル番号を 0 ～ 1 5 の整数で指定します。チャンネル番号とハードウェアとの対応は、表 2. 12. 3を参照してください。

2. 12. 9 Capture(3) デバイスドライバブロック

Capture(3) デバイスドライバは、指定チャネルの単相信号を計測し、第一出力に周期（周波数）、第二出力にON時間（Duty 比 100.0～-100.0）をUV相・VW相・WU相の順にベクトル出力します。

図 2. 12-8 ADSP324-145 Capture(3)



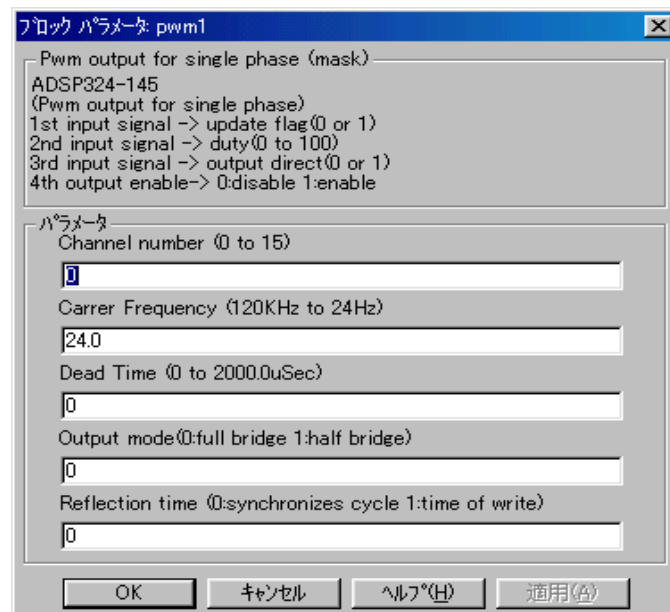
- Channel number (0 to 15)

ADSP324-145 のキャプチャチャネル番号を 0 ～ 15 の整数で指定します。チャネル番号とハードウェアとの対応は、表 2. 12. 3を参照してください。

2.12.10 Pwm Output L [S] デバイスドライバブロック

Pwm Output [S] デバイスドライバは、第一入力としてD u t y比の変更有無（1：変更する、0：変更しない）第二入力としてD u t y比（0.0～100.0%）、第三入力に出力方向（0：P側出力、1：N側出力）を入力すると、指定したチャンネルへ指定したキャリア周波数・デットタイムにて単相パルスを作成し、第四入力の出力許可により1を指定時は出力します。[Lなしブロックは第一入力がありません]

図 2.12-9 ADSP324-145 Pwm Output(S)

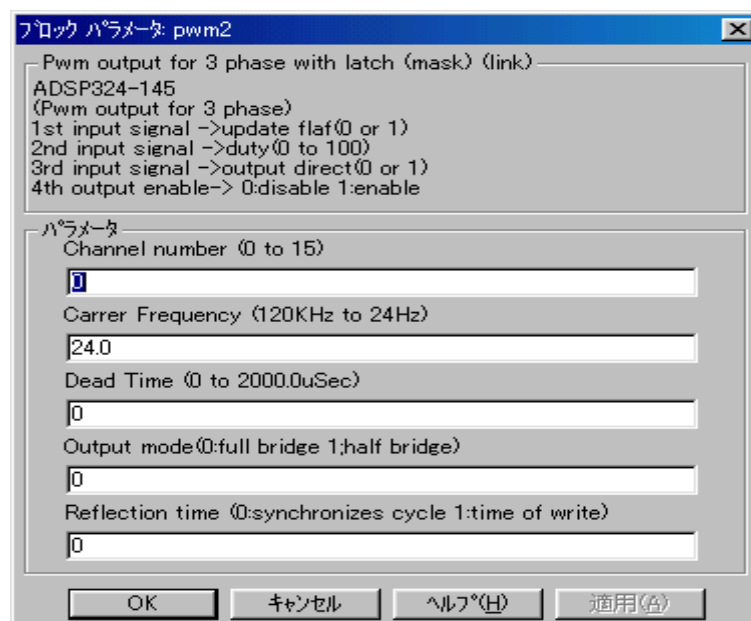


- Channel number (0 to 15)
ADSP324-145 の PWMチャンネル番号を 0～15 の整数で指定します。チャンネル番号とハードウェアとの対応は、表 2.12.3を参照してください。
- Carrer Frequency [120kHz to 24Hz]
キャリア周波数を 120KHz から 24Hz にて指定してください。
- Dead Time [0 to 2000.0 μSec]
デットタイムを 0～2000.0 μSec にて指定してください。
- Output Mode [0:full bridge 1:half bridge]
波形の出力モードを指定してください。
0：フルブリッジ出力 1：ハーフブリッジ出力
- Reflection time [0:synchronizes cycle 1:time of write]
設定値の反映タイミングを指定してください。
0：周期に同期（制御周期の切り替わりに同期して設定値を反映）
1：書きこみ時（モデルの制御周期時に設定値を反映）

2.12.11 Pwm Output L (3)デバイスドライバブロック

Pwm Output (3) デバイスドライバは、第一入力としてD u t y比の変更有無（1：変更する、0：変更しない）第二入力としてD u t y比（0.0～100.0%）をU相・V相・W相の順にベクトル入力、第三入力に出力方向（0：P側出力、1：N側出力）を入力すると、指定したチャンネルへ指定したキャリア周波数・デットタイムにてパルスを作成し、第四入力の出力許可により1を指定時は出力します。[Lなしブロックは第一入力がありません]

図 2.12-10 ADSP324-145 Pwm Output(3)

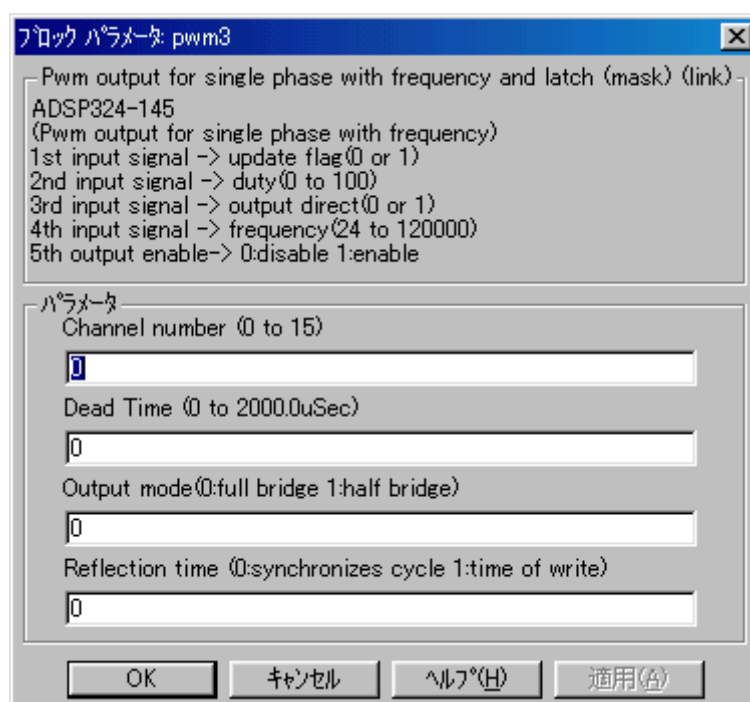


- Channel number (0 to 15)
ADSP324-145 の PWMチャンネル番号を 0～15 の整数で指定します。チャンネル番号とハードウェアとの対応は、表 2.12.3を参照してください。
- Carrer Frequency [120kHz to 24Hz]
キャリア周波数を 120KHz から 24Hz にて指定してください。
- Dead Time [0 to 2000.0 μSec]
デットタイムを 0～2000.0 μSec にて指定してください。
- Output Mode [0:full bridge 1:half bridge]
波形の出力モードを指定してください。
0：フルブリッジ出力 1：ハーフブリッジ出力
- Reflection time [0:synchronizes cycle 1:time of write]
設定値の反映タイミングを指定してください。
0：周期に同期（制御周期の切り替わりに同期して設定値を反映）
1：書きこみ時（モデルの制御周期時に設定値を反映）

2.12.12 Pwm Output L (S) with Freq デバイスドライバブロック

Pwm Output (S) with Freq デバイスドライバは、第一入力としてD u t y比の変更有無（1：変更する、0：変更しない）第二入力としてD u t y比（0.0～100.0%）、第三入力に出力方向（0：P側出力、1：N側出力）、第四入力としてキャリア周波数（120KHz～24Hz）を入力すると、指定したチャンネルへ指定したキャリア周波数・デットタイムにて単相パルスを作成し、第五入力の出力許可により1を指定時は出力します。[L なしブロックは第一入力がありません]

図 2.12-11 ADSP324-145 Pwm Output (S) with Freq

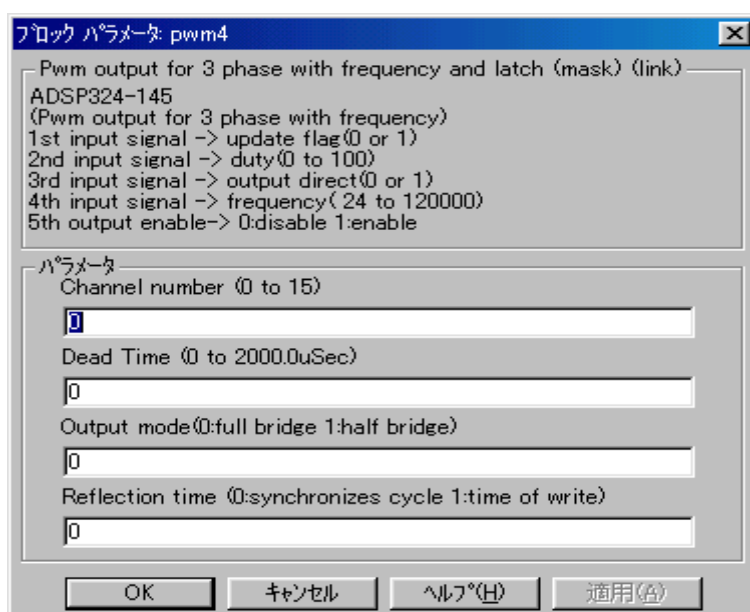


- Channel number (0 to 15)
ADSP324-145 の PWMチャンネル番号を 0～15 の整数で指定します。チャンネル番号とハードウェアとの対応は、表 2.12.3を参照してください。
- Dead Time [0 to 2000.0 μ Sec]
デットタイムを 0～2000.0 μ Sec にて指定してください。
- Output Mode [0:full bridge 1:half bridge]
波形の出力モードを指定してください。
0：フルブリッジ出力 1：ハーフブリッジ出力
- Reflection time [0:synchronizes cycle 1:time of write]
設定値の反映タイミングを指定してください。
0：周期に同期（制御周期の切り替わりに同期して設定値を反映）
1：書きこみ時（モデルの制御周期時に設定値を反映）

2.12.13 Pwm Output L (3) with Freq デバイスドライバブロック

Pwm Output (3) with Freq デバイスドライバは、第一入力としてD u t y比の変更有無（1：変更する、0：変更しない）、第二入力としてD u t y比（0.0～100.0%）をU相・V相・W相の順にベクトル入力、第三入力に出力方向（0：P側出力、1：N側出力）、第四入力としてキャリア周波数（120KHz～24Hz）を入力すると、指定したチャンネルへ指定したキャリア周波数・デットタイムにてパルスを作成し、第五入力の出力許可により1を指定時は出力します。[Lなしブロックは第一入力がありません]

図 2.12-12 ADSP324-145 Pwm Output (3) with Freq

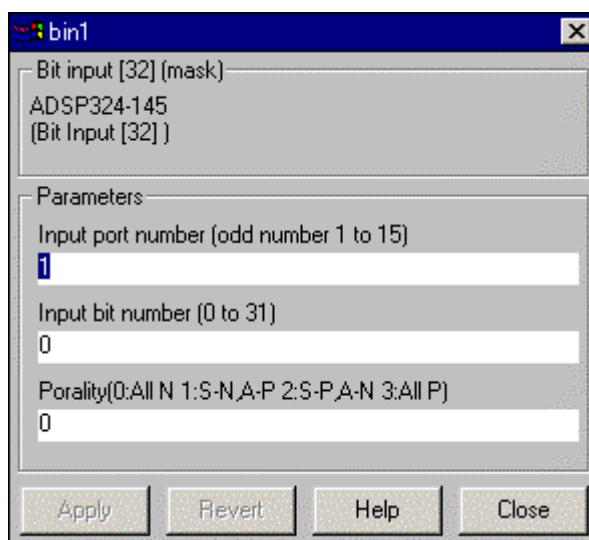


- Channel number (0 to 15)
ADSP324-145 の PWMチャンネル番号を 0～15 の整数で指定します。チャンネル番号とハードウェアとの対応は、表 2.12.3を参照してください。
- Dead Time [0 to 2000.0 μSec]
デットタイムを 0～2000.0 μSec にて指定してください。
- Output Mode [0:full bridge 1:half bridge]
波形の出力モードを指定してください。
0：フルブリッジ出力 1：ハーフブリッジ出力
- Reflection time [0:synchronizes cycle 1:time of write]
設定値の反映タイミングを指定してください。
0：周期に同期（制御周期の切り替わりに同期して設定値を反映）
1：書きこみ時（モデルの制御周期時に設定値を反映）

2.12.14 PIO bit In [32] デバイスドライバブロック

PIO bit In [32] デバイスドライバは、ビット単位入力ドライバです。指定されたポートの指定されたビットの値をスカラーで出力します。このポートは32ビット単位のアクセスになります。このブロックから出力される値は、ボードへの入力レベルは負論理選択時LOW:1・HIGH:0・未接続:1, 正論理選択時LOW:0・HIGH:1・未接続:1です。ボードが接続されていないポートの全てのビットは不定です。

図 2.12-13 ADSP324-145 bit In [32]

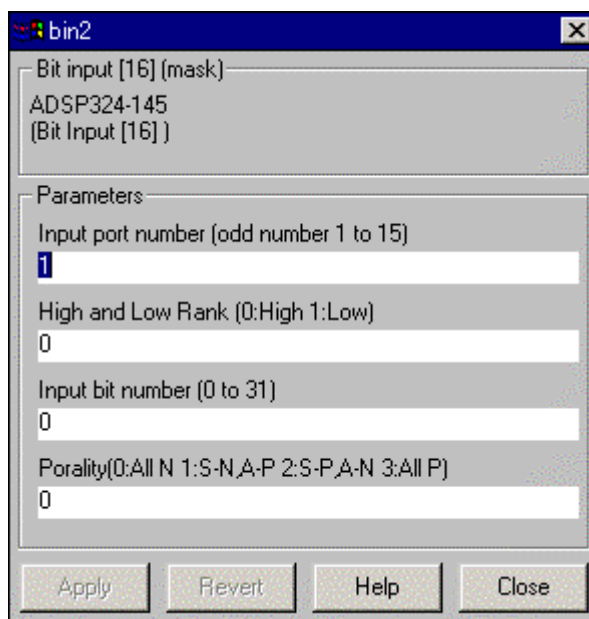


- Input port number (odd number in 1 to 15)
入力の対象となるビットが含まれるポートの番号を指定します。ポート番号とハードウェアの対応表は、表 2.12.4を参照してください。
- Input bit number (0 to 31)
入力の対象となるビット番号を指定します。ビット番号とハードウェアの対応表は、表 2.12.5を参照してください。
- Porality (0:All Negative 1:ACK=N STB=P 2:ACK=P STB=N 3: All Positive)
入力信号の極性を指定します。
(TTLレベルのみ有効、絶縁レベル時は指定時も無効)
0 : STB, ACK 共負論理 1 : STB 負論理、ACK 正論理
2 : STB 正論理、ACK 負論理 3 : STB, ACK 共正論理

2.12.15 PIO bit In [16] デバイスドライバブロック

PIO bit In [16] デバイスドライバーは、ビット単位入力ドライバです。指定されたポートの指定されたビットの値をスカラーで出力します。このポートは16ビット単位のアクセスになります。このブロックから出力される値は、ボードへの入力レベルは負論理選択時LOW:1・HIGH:0・未接続:1, 正論理選択時LOW:0・HIGH:1・未接続:1です。ボードが接続されていないポートの全てのビットは不定です。

図 2.12-14 ADSP324-145 bit In [16]

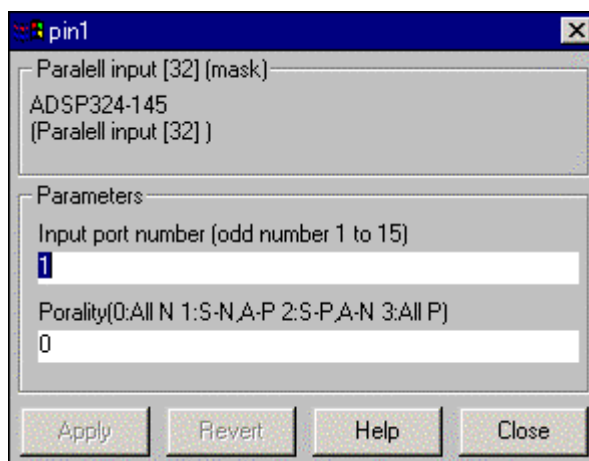


- Input port number (odd number in 1 to 15)
入力の対象となるビットが含まれるポートの番号を指定します。ポート番号とハードウェアの対応表は、表 2.12.4を参照してください。
- Higher and Low Rank(0: Higer 1: Low)
ポートでの16ビットの位置を0:上位又は1:下位で指定してください。
- Input bit number(0 to 15)
入力の対象となるビット番号を指定します。ビット番号とハードウェアの対応表は、表 2.12.5を参照してください。
- Porality (0:All Negative 1:ACK=N STB=P 2:ACK=P STB=N 3: All Positive)
入力信号の極性を指定します。
(TTLレベルのみ有効、絶縁レベル時は指定時も無効)
0: STB, ACK 共負論理 1: STB 負論理、ACK 正論理
2: STB 正論理、ACK 負論理 3: STB, ACK 共正論理

2.12.16 PIO parallell in [32] デバイスドライバブロック

PIO parallell in [32] デバイスドライバーは、複数ビット同時入力デバイスドライバブロックです。指定されたポートの全ビットを入力し、結果をベクトルで出力します。このポートは32ビット単位のアクセスになります。また出力されるベクトルは、幅が32のベクトルで、先頭の要素がビット0、次がビット1の順で、最後の要素がビット31に対応しています。各要素はボードへの入力信号が OFF の時0、ON の時1です。入力信号が接続されていないビットは0になります。ボードが実装されていないポートの全てのビットは不定です。

図 2.12-15 ADSP324-145 parallell in [32]



- Input port number (odd number in 1 to 15)

入力の対象となるビットが含まれるポートの番号を指定します。ポート番号とハードウェアの対応表は、表 2.12.4を参照してください。

- Porality (0:All Negative 1:ACK=N STB=P 2:ACK=P STB=N 3: All Positive)

入力信号の極性を指定します。

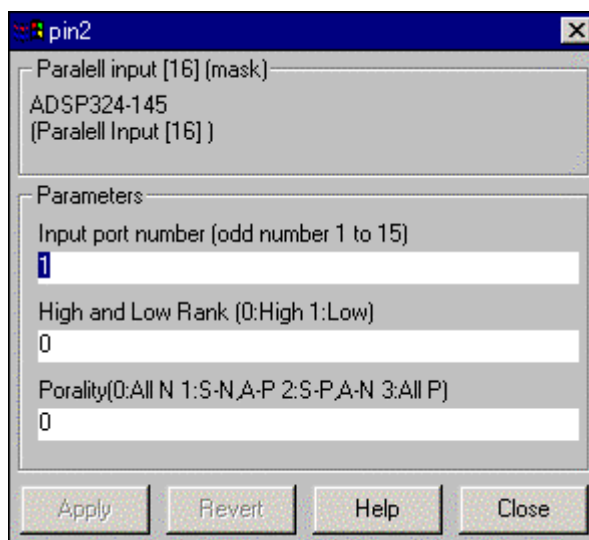
(TTLレベルのみ有効、絶縁レベル時は指定時も無効)

0 : STB, ACK 共負論理	1 : STB 負論理、ACK 正論理
2 : STB 正論理、ACK 負論理	3 : STB, ACK 共正論理

2.12.17 PIO parallell in [16] デバイスドライバブロック

PIO parallell in [16] デバイスドライバーは、複数ビット同時入力デバイスドライバブロックです。指定されたポートの指定位置（上位/下位）16ビットを入力し、結果をベクトルで出力します。このポートは16ビット単位のアクセスになります。また出力されるベクトルは、幅が16のベクトルで、先頭の要素がビット0、次がビット1の順で、最後の要素がビット15に対応しています。各要素はボードへの入力信号がOFFの時0、ONの時1です。入力信号が接続されていないビットは0になります。ボードが実装されていないポートの全てのビットは不定です。

図 2.12-16 ADSP324-145 parallell in [16]

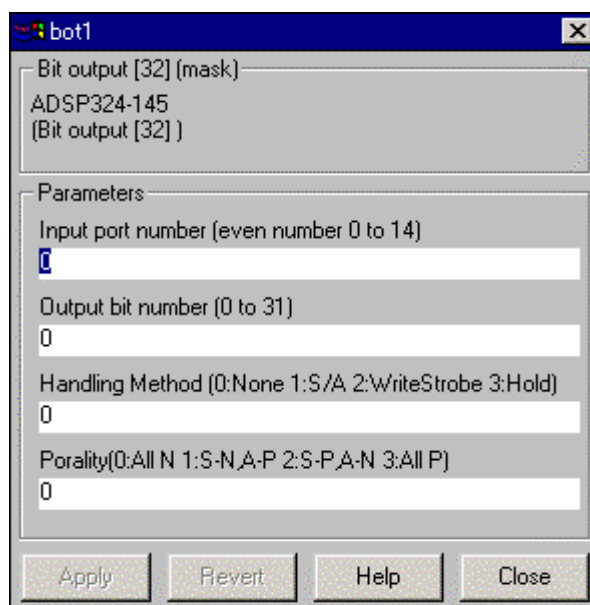


- Input port number (odd number in 1 to 15)
入力の対象となるビットが含まれるポートの番号を指定します。ポート番号とハードウェアの対応表は、表 2.12.4を参照してください。
- Higher and Low Rank(0: Higer 1: Low)
ポートでの16ビットの位置を0：上位又は1：下位で指定してください。
- Porality (0:All Negative 1:ACK=N STB=P 2:ACK=P STB=N 3: All Positive)
入力信号の極性を指定します。
(TTLレベルのみ有効、絶縁レベル時は指定時も無効)
0：STB,ACK 共負論理 1：STB 負論理、ACK 正論理
2：STB 正論理、ACK 負論理 3：STB,ACK 共正論理

2.12.18 PIO bit out [32] デバイスドライバブロック

PIO bit out [32] デバイスドライバは、ビット単位の出力デバイスドライバブロックです。スカラーの入力を受け取り、指定されたポートのビットへ出力します。このポートは32ビット単位のアクセスになります。ボードから出力される信号は、このブロックへの入力レベルが0の時OFF、1の時ONです。ボードの出力初期値は全ビット共OFFです。

図 2.12-17 ADSP324-145 bit out [32]

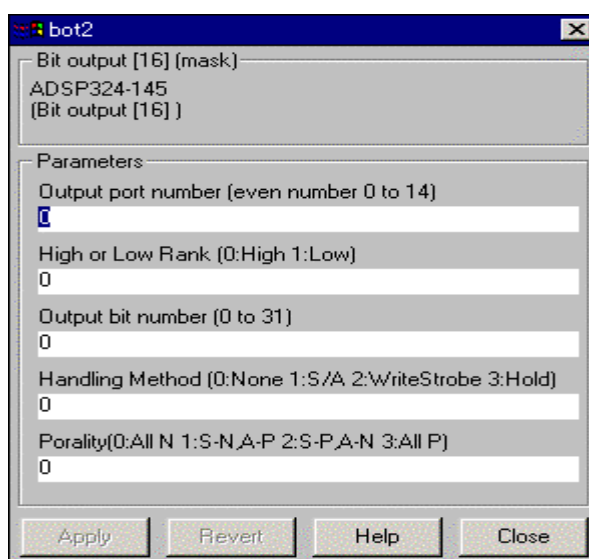


- Output port number (even number in 0 to 14)
出力の対象となるビットが含まれるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表 2.12.4を参照してください。
- Output bit number [0 to 31]
出力の対象となるビット番号を指定します。指定できるビット番号は、0～31です。ビット番号とハードウェアとの対応は、表 2.12.5を参照してください。
- Handling Method [0: None 1: Hold 2: S/A 3: WriteStrobe]
出力のハンドリング方式を指定します。ポート番号毎に決定されます。このブロックを複数使用する場合は、指定ポート番号毎の設定は同一にしてください。ハンドリング方式の詳細は、表 2.12.6を参照してください。
- Porality (0:All Negative 1:ACK=N STB=P 2:ACK=P STB=N 3: All Positive)
入力信号の極性を指定します。
(TTLレベルのみ有効、絶縁レベル時は指定時も無効)
0 : STB, ACK 共負論理 1 : STB 負論理、ACK 正論理
2 : STB 正論理、ACK 負論理 3 : STB, ACK 共正論理

2.12.19 PIO bit out [16] デバイスドライバブロック

PIO bit out [16] デバイスドライバは、ビット単位の出力デバイスドライバブロックです。スカラーの入力を受け取り、指定されたポートの指定ビット位置（上位 / 下位）へ出力します。このポートは16ビット単位のアクセスになります。ボードから出力される信号は、このブロックへの入力レベルが0の時OFF、1の時ONです。ボードの出力初期値は全ビット共OFFです。

図 2.12-18 ADSP324-145 bit out [16]

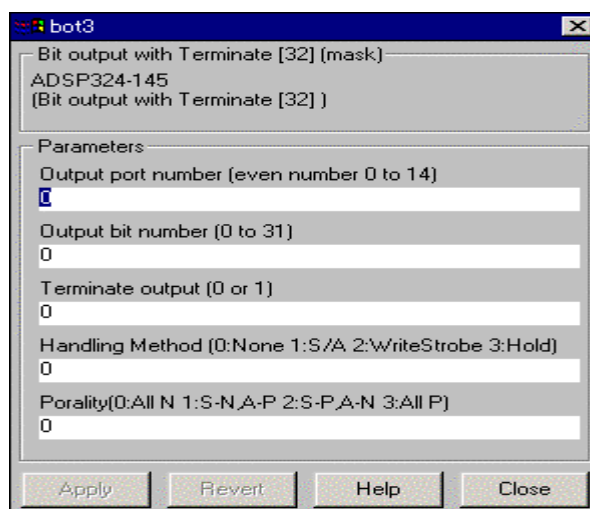


- Output port number (even number in 0 to 14)
出力の対象となるビットが含まれるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表 2.12.4を参照してください。
- Higher and Low Rank(0: Higer 1: Low)
ポートでの16ビットの位置を0：上位又は1：下位で指定してください。
- Output bit number [0 to 15]
出力の対象となるビット番号を指定します。指定できるビット番号は、0～15です。ビット番号とハードウェアとの対応は、表 2.12.5を参照してください。
- Handling Method [0: None 1: Hold 2: S/A 3: WriteStrobe]
出力のハンドリング方式を指定します。ポート番号毎に決定されます。このブロックを複数使用する場合は、指定ポート番号毎の設定は同一にしてください。ハンドリング方式の詳細は、表 2.12.6を参照してください。
- Porality (0:All Negative 1:ACK=N STB=P 2:ACK=P STB=N 3: All Positive)
入力信号の極性を指定します。
(TTLレベルのみ有効、絶縁レベル時は指定時も無効)
0：STB, ACK 共負論理 1：STB 負論理、ACK 正論理
2：STB 正論理、ACK 負論理 3：STB, ACK 共正論理

2.12.20 PIO bit out with Terminate [32] デバイスドライバブロック

PIO bit output Terminate out [32] デバイスドライバーは、ビット単位の出力デバイスドライバブロックです。スカラーの入力を受け取り、指定されたポートのビットへ出力します。このポートは32ビット単位のアクセスになります。ボードから出力される信号は、このブロックへの入力レベルが0の時OFF、1の時ONです。ボードの出力初期値は全ビット共OFFです。このブロックは、実時間モデルが終了する際に予め設定しておいた値を出力することができます。

図 2.12-19 ADSP324-145 bit output Terminate out [32]

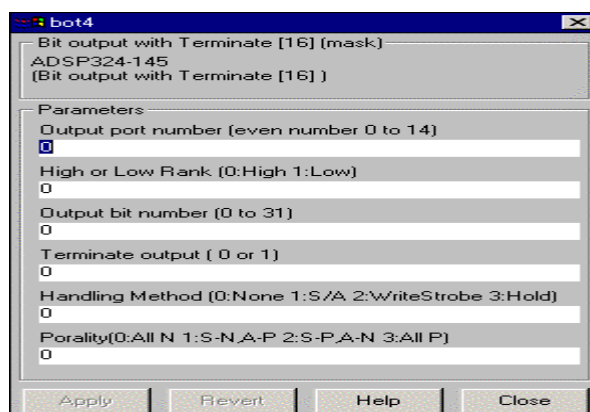


- Output port number (even number in 0 to 14)
出力の対象となるビットが含まれるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表 2.12.4を参照してください。
- Output bit number [0 to 31]
出力の対象となるビット番号を指定します。指定できるビット番号は、0～31です。ビット番号とハードウェアとの対応は、表 2.12.5を参照してください。
- Terminate output [0 or 1]
実時間モデルが終了時する際に出力したい値を設定します。
- Handling Method [0: None 1: Hold 2: S/A 3: WriteStrobe]
出力のハンドリング方式を指定します。ポート番号毎に決定されます。このブロックを複数使用する場合は、指定ポート番号毎の設定は同一にしてください。ハンドリング方式の詳細は、表 2.12.6を参照してください。
- Porality (0:All Negative 1:ACK=N STB=P 2:ACK=P STB=N 3: All Positive)
入力信号の極性を指定します。
(TTLレベルのみ有効、絶縁レベル時は指定時も無効)
0 : STB, ACK 共負論理 1 : STB 負論理、ACK 正論理
2 : STB 正論理、ACK 負論理 3 : STB, ACK 共正論理

2.12.21 PIO bit out with Terminate [16] デバイスドライバブロック

PIO bit output Terminate out [16] デバイスドライバーは、ビット単位の出力デバイスドライバブロックです。スカラーの入力を受け取り、指定されたポートの指定ビット位置（上位 / 下位）へ出力します。このポートは16ビット単位のアクセスになります。ボードから出力される信号は、このブロックへの入力レベルが0の時OFF、1の時ONです。ボードの出力初期値は全ビット共OFFです。このブロックは、実時間モデルが終了する際に予め設定しておいた値を出力することができます。

図 2.12-20 ADSP324-145 bit output Terminate out [16]



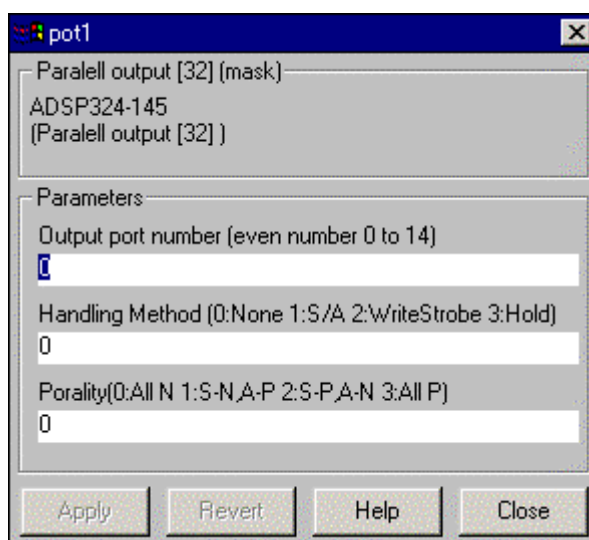
- Output port number (even number in 0 to 14)
出力の対象となるビットが含まれるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表 2.12.4を参照してください。
- Higher and Low Rank(0: Higer 1: Low)
ポートでの16ビットの位置を0：上位又は1：下位で指定してください。
- Output bit number [0 to 31]
出力の対象となるビット番号を指定します。指定できるビット番号は、0～31です。ビット番号とハードウェアとの対応は、表 2.12.5を参照してください。
- Treminate output [0 or 1]
実時間モデルが終了時する際に出力したい値を設定します。
- Handling Method [0: None 1: Hold 2: S/A 3: WriteStrobe]
出力のハンドリング方式を指定します。ポート番号毎に決定されます。このブロックを複数使用する場合は、指定ポート番号毎の設定は同一にしてください。ハンドリング方式の詳細は、表 2.12.6を参照してください。
- Porality (0:All Negative 1:ACK=N STB=P 2:ACK=P STB=N 3: All Positive)
入力信号の極性を指定します。
(TTLレベルのみ有効、絶縁レベル時は指定時も無効)

0 : STB, ACK 共負論理	1 : STB 負論理、ACK 正論理
2 : STB 正論理、ACK 負論理	3 : STB, ACK 共正論理

2.12.22 PIO parallell out [32] デバイスドライバブロック

PIO parallell out [32] デバイスドライバは、複数ビット同時入力デバイスドライバブロックです。指定されたポートの32ビットを入力し、結果をベクトルで出力します。このポートは32ビット単位のアクセスになります。また出力されるベクトルは、幅が32のベクトルで、先頭の要素がビット0、次がビット1の順で、最後の要素がビット31に対応しています。各要素はボードへの入力信号がOFFの時0、ONの時1です。入力信号が接続されていないビットは0になります。ボードが実装されていないポートの全てのビットは不定です。

図 2.12-21 ADSP324-145 parallell out [32]



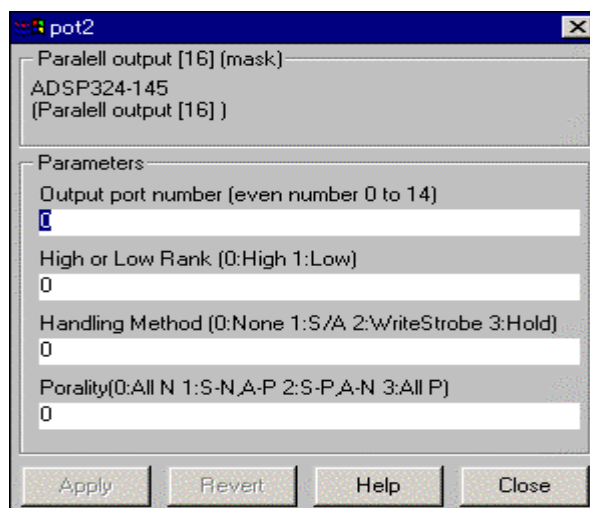
- Output port number (even number in 0 to 14)
出力の対象となるビットが含まれるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表 2.12.4を参照してください。
- Handling Method [0: None 1: Hold 2: S/A 3: WriteStrobe]
出力のハンドリング方式を指定します。ポート番号毎に決定されます。このブロックを複数使用する場合は、指定ポート番号毎の設定は同一にしてください。ハンドリング方式の詳細は、表 2.12.6を参照してください。
- Porality (0:All Negative 1:ACK=N STB=P 2:ACK=P STB=N 3: All Positive)
入力信号の極性を指定します。
(TTLレベルのみ有効、絶縁レベル時は指定時も無効)

0 : STB, ACK 共負論理	1 : STB 負論理、ACK 正論理
2 : STB 正論理、ACK 負論理	3 : STB, ACK 共正論理

2.12.23 PIO parallell out [16] デバイスドライバブロック

PIO parallell out [16] デバイスドライバは、複数ビット同時入力デバイスドライバブロックです。指定されたポートの指定位置（上位/下位）16ビットを入力し、結果をベクトルで出力します。このポートは16ビット単位のアクセスになります。また出力されるベクトルは、幅が16のベクトルで、先頭の要素がビット0、次がビット1の順で、最後の要素がビット15に対応しています。各要素はボードへの入力信号がOFFの時0、ONの時1です。入力信号が接続されていないビットは0になります。ボードが実装されていないポートの全てのビットは不定です。

図 2.12-22 ADSP324-145 parallell out [16]



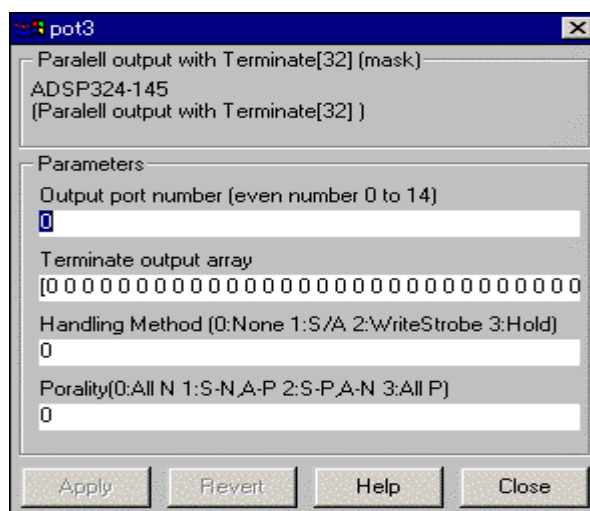
- Output port number (even number in 0 to 14)
出力の対象となるビットが含まれるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表 2.12.4を参照してください。
- Higher and Low Rank(0: Higer 1: Low)
ポートでの16ビットの位置を0：上位又は1：下位で指定してください。
- Handling Method [0: None 1: Hold 2: S/A 3: WriteStrobe]
出力のハンドリング方式を指定します。ポート番号毎に決定されます。このブロックを複数使用する場合は、指定ポート番号毎の設定は同一にしてください。ハンドリング方式の詳細は、表 2.12.6を参照してください。
- Porality (0:All Negative 1:ACK=N STB=P 2:ACK=P STB=N 3: All Positive)
入力信号の極性を指定します。
(TTLレベルのみ有効、絶縁レベル時は指定時も無効)
0：STB, ACK 共負論理 1：STB 負論理、ACK 正論理
2：STB 正論理、ACK 負論理 3：STB, ACK 共正論理

2.12.24 PIO parallel out with Terminate [32] デバイスドライバブロック

PIO parallel Terminate out [32] デバイスドライバは、複数ビット同時入力デバイスドライバブロックです。指定されたポートの32ビットを入力し、結果をベクトルで出力します。このポートは32ビット単位のアクセスになります。また出力されるベクトルは、幅が32のベクトルで、先頭の要素がビット0、次がビット1の順で、最後の要素がビット31に対応しています。各要素はボードへの入力信号がOFFの時0、ONの時1です。入力信号が接続されていないビットは0になります。ボードが実装されていないポートの全てのビットは不定です。

このブロックは、実時間モデルが終了する際に予め設定しておいた値を出力することができます。

図 2.12-23 ADSP324-145 parallel Terminate out [32]



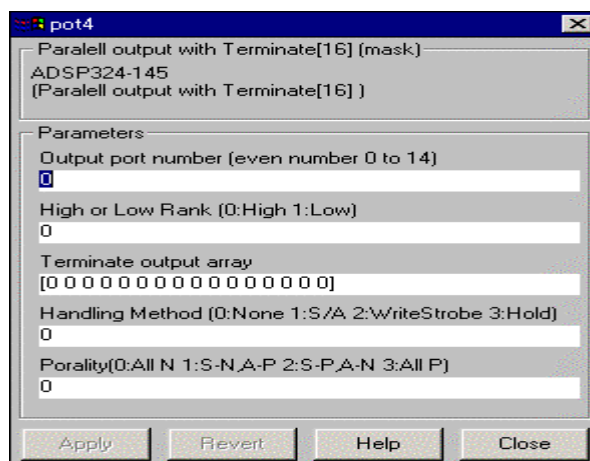
- Output port number (even number in 0 to 14)
出力の対象となるビットが含まれるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表 2.12.4を参照してください。
- Treminate output [0 or 1]
実時間モデルが終了時する際に出力したい値を設定します。
- Handling Method [0: None 1: Hold 2: S/A 3: WriteStrobe]
出力のハンドリング方式を指定します。ポート番号毎に決定されます。このブロックを複数使用する場合は、指定ポート番号毎の設定は同一にしてください。ハンドリング方式の詳細は、表 2.12.6を参照してください。
- Porality (0:All Negative 1:ACK=N STB=P 2:ACK=P STB=N 3: All Positive)
入力信号の極性を指定します。
(TTLレベルのみ有効、絶縁レベル時は指定時も無効)

0 : STB, ACK 共負論理	1 : STB 負論理、ACK 正論理
2 : STB 正論理、ACK 負論理	3 : STB, ACK 共正論理

2.12.25 PIO parallell out with Terminate [16] デバイスドライバブロック

PIO parallell Terminate out [16] デバイスドライバは、複数ビット同時入力デバイスドライバブロックです。指定されたポートの指定位置（上位/下位）16ビットを入力し、結果をベクトルで出力します。このポートは16ビット単位のアクセスになります。また出力されるベクトルは、幅が16のベクトルで、先頭の要素がビット0、次がビット1の順で、最後の要素がビット15に対応しています。各要素はボードへの入力信号がOFFの時0、ONの時1です。入力信号が接続されていないビットは0になります。ボードが実装されていないポートの全てのビットは不定です。このブロックは、実時間モデルが終了する際に予め設定しておいた値を出力することができます。

図 2.12-24 ADSP324-145 parallell Terminate out [16]

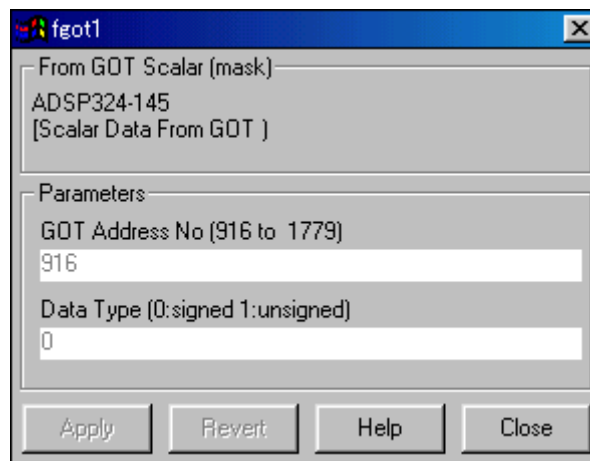


- Output port number (even number in 0 to 14)
出力の対象となるビットが含まれるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表 2.12.4を参照してください。
- Higher and Low Rank(0: Higer 1: Low)
ポートでの16ビットの位置を0：上位又は1：下位で指定してください。
- Treminate output [0 or 1]
実時間モデルが終了時する際に出力したい値を設定します。
- Handling Method [0: None 1: Hold 2: S/A 3: WriteStrobe]
出力のハンドリング方式を指定します。ポート番号毎に決定されます。このブロックを複数使用する場合は、指定ポート番号毎の設定は同一にしてください。ハンドリング方式の詳細は、表 2.12.6を参照してください。
- Porality (0:All Negative 1:ACK=N STB=P 2:ACK=P STB=N 3: All Positive)
入力信号の極性を指定します。
(T T Lレベルのみ有効、絶縁レベル時は指定時も無効)
0：STB, ACK 共負論理 1：STB 負論理、ACK 正論理
2：STB 正論理、ACK 負論理 3：STB, ACK 共正論理

2.12.26 From GOT デバイスドライバブロック

From GOT デバイスドライバは、DSP 操作ターミナル(ADSP-GOT シリーズ)の付属 GOT 用のデバイスドライバブロックです。指定された GOT アドレスのデータをスカラーで取得します。

図 2.12-25 ADSP324-145 From GOT

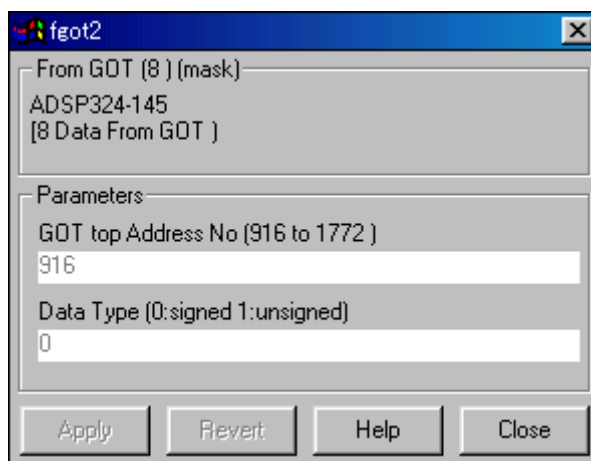


- GOT Address No (916 to 1779)
GOT のアドレス番号を 916～1779 の範囲で指定してください。詳細については “DSP 操作ターミナル・GOT 操作ガイド” を参照してください。
- Data Type (0:signed 1:unsigned)
0 : 符号付データ、1 : 符号なしデータ のいずれかを指定してください。この指定は “DSP 操作ターミナル” の画面オブジェクト基本設定値と同一指定にしてください。

2.12.27 From GOT[8]デバイスドライバブロック

From GOT[8]デバイスドライバは、DSP操作ターミナル(ADSP-GOT シリーズ) の付属 GOT用のデバイスドライバブロックです。指定された GOT アドレスのデータをベクトル幅8で取得します。

図 2.12-26 ADSP324-145 From GOT[8]

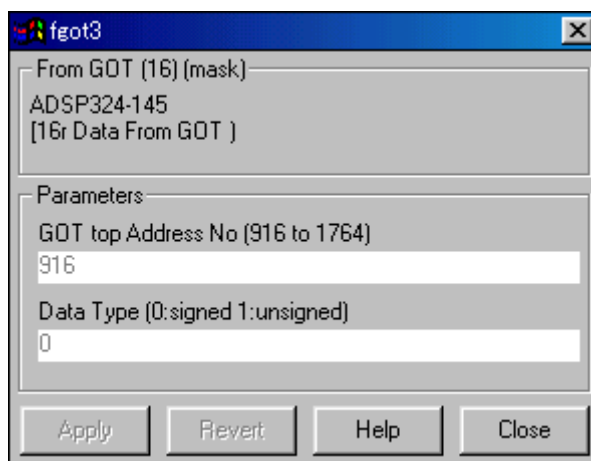


- GOT Address No (916 to 1772)
GOT の先頭アドレス番号を 916～1772 の範囲で指定してください。詳細については
“DSP操作ターミナル・GOT操作ガイド” を参照してください。
- Data Type (0:signed 1:unsigned)
0：符号付データ、1：符号なしデータ のいずれかを指定してください。この
指定は“DSP操作ターミナル”の画面オブジェクト基本設定値と同一指定にし
てください。

2.12.28 From GOT[16]デバイスドライバブロック

From GOT[16]デバイスドライバは、DSP操作ターミナル(ADSP-GOT シリーズ) の付属GOT用のデバイスドライバブロックです。指定された GOT アドレスのデータをベクトル幅16で取得します。

図 2.12-27 ADSP324-145 From GOT[16]

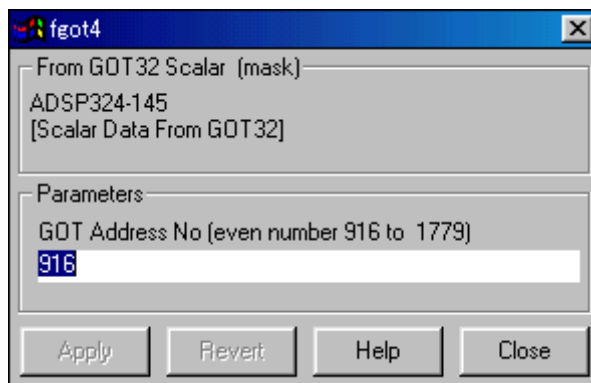


- GOT Address No (916 to 1764)
GOTの先頭アドレス番号を916～1764の範囲で指定してください。詳細については“DSP操作ターミナル・GOT操作ガイド”を参照してください。
- Data Type (0:signed 1:unsigned)
0：符号付データ、1：符号なしデータ のいずれかを指定してください。この指定は“DSP操作ターミナル”の画面オブジェクト基本設定値と同一指定にしてください。

2.12.29 From GOT32 デバイスドライバブロック

From GOT32 デバイスドライバーは、DSP 操作ターミナル (ADSP-GOT シリーズ) の付属 GOT 用のデバイスドライバブロックです。指定された GOT アドレスのデータをスカラーで取得します。

図 2.12-28 ADSP324-145 From GOT32



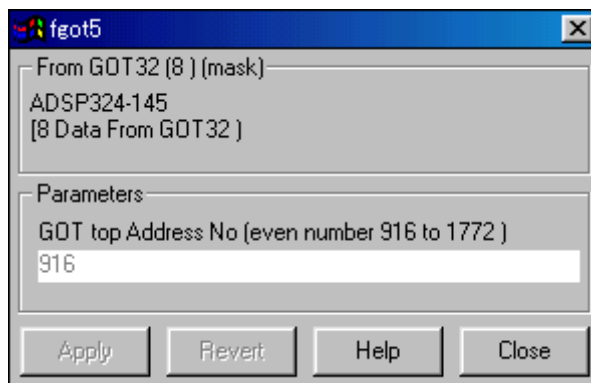
- GOT Address No (916 to 1779)

GOT のアドレス番号を 916～1779 の範囲で指定してください。詳細については “DSP 操作ターミナル・GOT 操作ガイド” を参照してください。

2.12.30 From GOT32 [8]デバイスドライバブロック

From GOT32 [8]デバイスドライバーは、DSP 操作ターミナル(ADSP-GOT シリーズ) の付属GOT用のデバイスドライバブロックです。指定された GOT アドレスのデータをベクトル幅8で取得します。

図 2.12-29 ADSP324-145 From GOT32 [8]



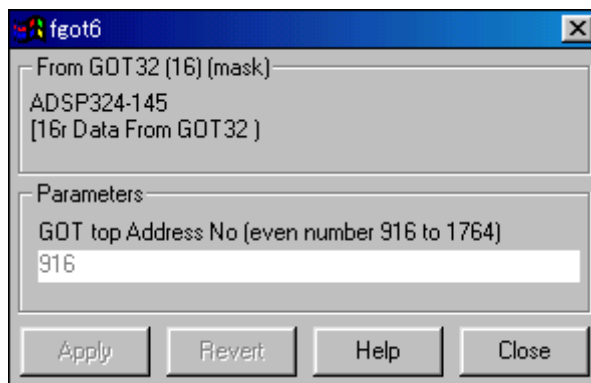
- GOT Address No (916 to 1772)

GOT の先頭アドレス番号を 916～1772 の範囲で指定してください。詳細については“DSP 操作ターミナル・GOT 操作ガイド”を参照してください。

2.12.31 From GOT32 [16]デバイスドライバブロック

From GOT32 [16]デバイスドライバは、DSP操作ターミナル(ADSP-GOT シリーズ) の付属GOT用のデバイスドライバブロックです。指定された GOT アドレスのデータをベクトル幅16で取得します。

図 2.12-30 ADSP324-145 From GOT32 [16]



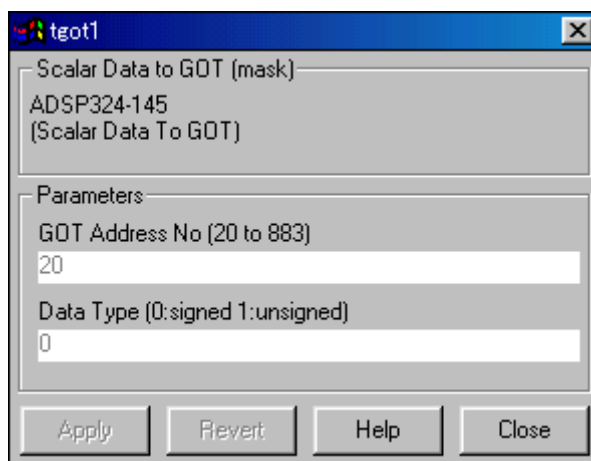
- GOT Address No (916 to 1764)

GOTの先頭アドレス番号を916～1764の範囲で指定してください。詳細については“DSP操作ターミナル・GOT操作ガイド”を参照してください。

2.12.32 To GOT デバイスドライバブロック

To GOT デバイスドライバは、DSP 操作ターミナル(ADSP-GOT シリーズ) の付属 GOT 用のデバイスドライバブロックです。指定された GOT アドレスのデータをスカラーで出力します。

図 2.12-31 ADSP324-145 To GOT

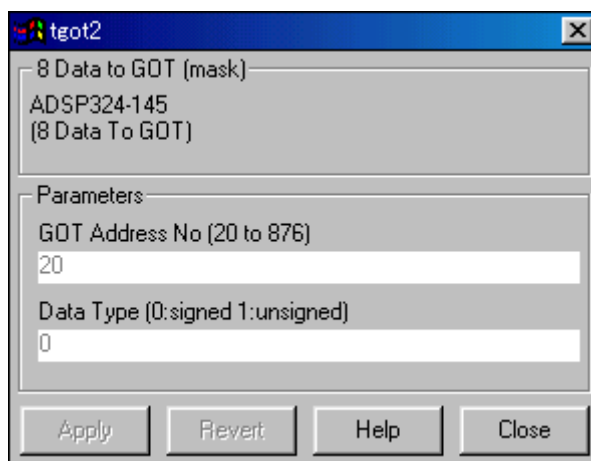


- GOT Address No (20 to 883)
GOT の先頭アドレス番号を 20～883 の範囲で指定してください。詳細については“DSP 操作ターミナル・GOT 操作ガイド”を参照してください。
- Data Type (0:signed 1:unsigned)
0：符号付データ、1：符号なしデータ のいずれかを指定してください。この指定は“DSP 操作ターミナル”の画面オブジェクト基本設定値と同一指定にしてください。

2.12.33 To GOT[8]デバイスドライバブロック

To GOT[8]デバイスドライバは、DSP操作ターミナル(ADSP-GOT シリーズ)の付属GOT用のデバイスドライバブロックです。指定された GOT アドレスのデータをベクトル幅8で出力します。

図 2.12-32 ADSP324-145 To GOT[8]

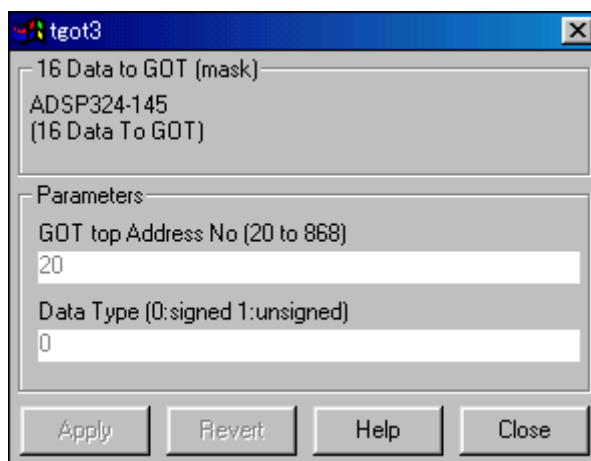


- GOT Address No (20 to 876)
GOTの先頭アドレス番号を20～876の範囲で指定してください。詳細については“DSP操作ターミナル・GOT操作ガイド”を参照してください。
- Data Type (0:signed 1:unsigned)
0：符号付データ、1：符号なしデータ のいずれかを指定してください。この指定は“DSP操作ターミナル”の画面オブジェクト基本設定値と同一指定にしてください。

2.12.34 To GOT[16]デバイスドライバブロック

To GOT[16]デバイスドライバは、DSP操作ターミナル(ADSP-GOT シリーズ)の付属GOT用のデバイスドライバブロックです。指定された GOT アドレスのデータをベクトル幅16で出力します。

図 2.12-33 ADSP324-145 To GOT[16]

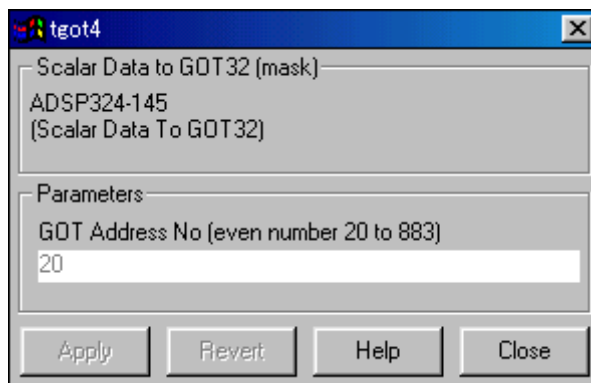


- GOT Address No (20 to 868)
GOTの先頭アドレス番号を20～868の範囲で指定してください。詳細については“DSP操作ターミナル・GOT操作ガイド”を参照してください。
- Data Type (0:signed 1:unsigned)
0：符号付データ、1：符号なしデータ のいずれかを指定してください。この指定は“DSP操作ターミナル”の画面オブジェクト基本設定値と同一指定にしてください。

2.12.35 To GOT32 デバイスドライバブロック

To GOT32 デバイスドライバは、DSP 操作ターミナル(ADSP-GOT シリーズ) の付属 GOT 用のデバイスドライバブロックです。指定された GOT アドレスのデータをスカラーで出力します。

図 2.12-34 ADSP324-145 To GOT32



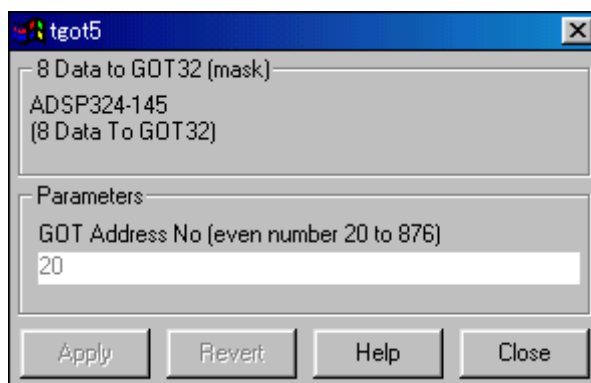
- GOT Address No (20 to 883)

GOT の先頭アドレス番号を 20～883 の範囲で指定してください。詳細については“DSP 操作ターミナル・GOT 操作ガイド”を参照してください。

2. 12. 36 To GOT32 [8]デバイスドライバブロック

To GOT32 [8]デバイスドライバは、D S P 操作ターミナル(ADSP-GOT シリーズ) の付属 G O T 用のデバイスドライバブロックです。指定された GOT アドレスのデータをベクトル幅 8 で出力します。

図 2. 12-35 ADSP324-145 To GOT32 [8]



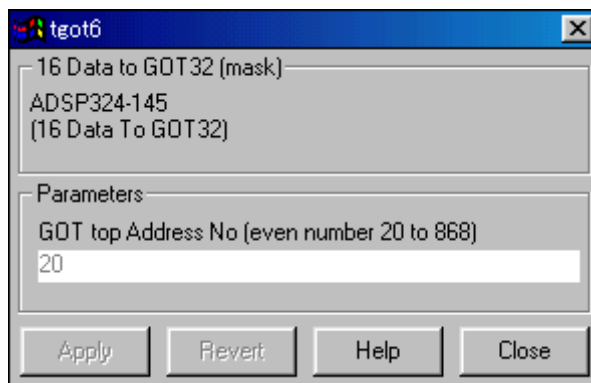
- GOT Address No (20 to 876)

GOT の先頭アドレス番号を 20～876 の範囲で指定してください。詳細については“D S P 操作ターミナル・G O T 操作ガイド”を参照してください。

2.12.37 To GOT32 [16]デバイスドライバブロック

To GOT32 [16]デバイスドライバは、DSP操作ターミナル(ADSP-GOT シリーズ)の付属GOT用のデバイスドライバブロックです。指定された GOT アドレスのデータをベクトル幅16で出力します。

図 2.12-36 ADSP324-145 To GOT32 [16]



- GOT Address No (20 to 868)

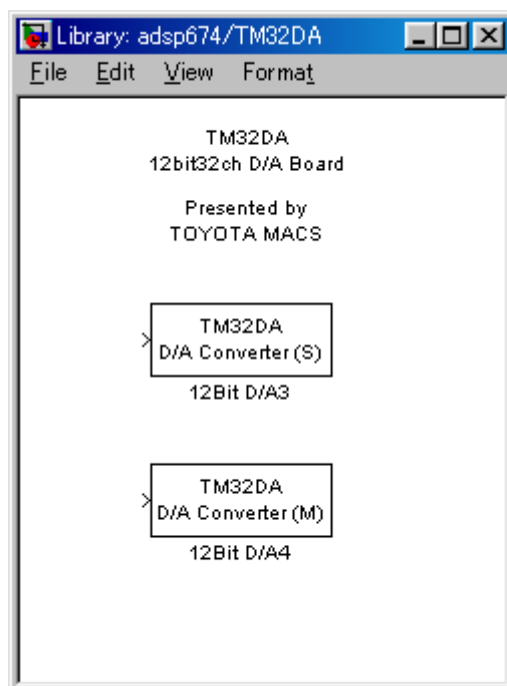
GOTの先頭アドレス番号を20～868の範囲で指定してください。詳細については“DSP操作ターミナル・GOT操作ガイド”を参照してください。

2.13 TM32DA デバイスドライバブロック

2.13.1 概要

図 2.13-1に、TM32DA（12ビット多チャンネルD/Aボード）用デバイスドライバウィンドウを開いた所を示します。

図 2.13-1 TM32DA ドライバウィンドウ



この中にはADSP32X-03/53とのインターフェースを取るための4つのデバイスドライバブロックが登録されています。

- D/A input (S) 1チャンネルのD/A出力デバイスドライバブロックです。
- D/A input (M) 多チャンネルのD/A出力デバイスドライバブロックです。

1チャンネルの入力（出力）デバイスドライバは、任意のチャンネルからデータを入力（出力）できます。特定のチャンネルのみをアクセスしたい場合や、アクセスしたいチャンネルが連続するチャンネルでない場合に有効です。

多チャンネルの入力（出力）デバイスドライバは、チャンネル番号0から連続する任意の数のチャンネルからデータを入力（出力）できます。複数のチャンネルをベクトルとして取り扱う場合や、入力（出力）の為のオーバーヘッドを低減する目的等で連続するチャンネルを一括してアクセスする場合等に利用します。

1チャンネル版と多チャンネル版は一度に取り扱うチャンネルの数やチャンネル番号の指定が異なるのみで、本質的には全く差異はありません。例えば、多チャンネル版で入力した複数チャンネルのA/D入力データをDemuxブロックで個別の信号に分解したものと、1チャンネル版で個々に入力したものととは全く同一として取り扱えます。D/Aの場合も同様です。特に、A/Dでは、同じサンプリング時間を設定した複数のデバイスドライバを使用した場合のサンプリングの同時性は保証されています。1チャンネル版と多チャンネル版のドライバを混在した場合でも同様です。

表 2.13.1 チャンネル番号とTM3 2DAボードとの対応

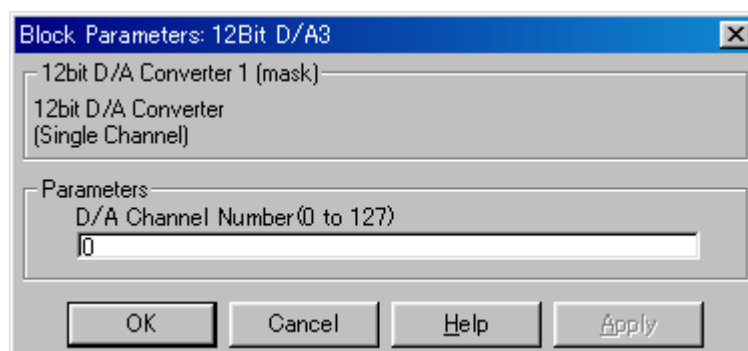
ボード	CH00	CH01	...	CH30	CH31
1 台目	0	1	...	3 0	3 1
2 台目	3 2	3 3	...	6 2	6 3
3 台目	6 4	6 5	...	9 4	9 5
4 台目	9 6	9 7	...	1 2 6	1 2 7

2.13.2 1チャンネルD/A出力デバイスドライバブロック

1チャンネルD/A出力デバイスドライバブロックは、スカラーデータを受取り、指定されたチャンネル番号のD/Aへ出力します。

このブロックへの入力データの単位は電圧値[V]です。物理単位からの変換は Gain ブロック等を使って予め行なってください。

図 2.13-2 TM32DA D/A output (S)



- D/A channel number (0 to 127)

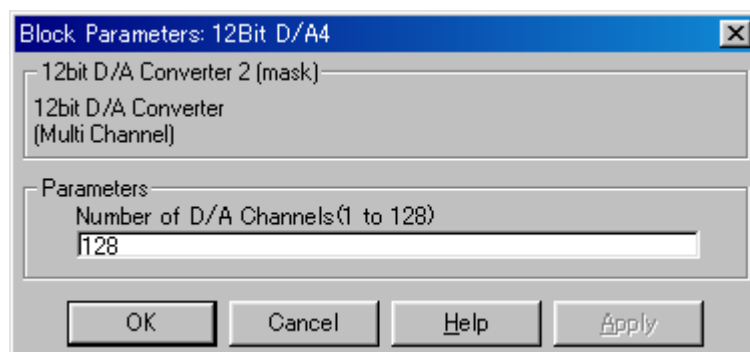
出力するD/Aのチャンネル番号を0～127の整数で指定します。チャンネル番号とハードウェアとの対応は、表 2.13.1 (141ページ) を参照してください。

2.13.3 多チャンネルD/A出力デバイスドライバブロック

多チャンネルD/A出力デバイスドライバブロックは、ベクトルデータを受取り、複数チャンネルのD/Aへ一括して出力します。出力の対象となるD/Aチャンネルは、チャンネル番号0から連続するチャンネルで、先頭チャンネル番号は0で固定となっています。このブロックへの入力ベクトルの幅と、このブロックに指定したチャンネル数とは一致していなければいけません。このブロックへ入力されるベクトルの、最初の要素がチャンネル番号0のD/Aへ出力されます。

このブロックへの入力データの単位は電圧値[V]です。物理単位からの変換は Gain ブロック等を使って予め行なってください。

図 2.13-3 TM32DA D/A output (M)



- Number of D/A channels (1 to 128)

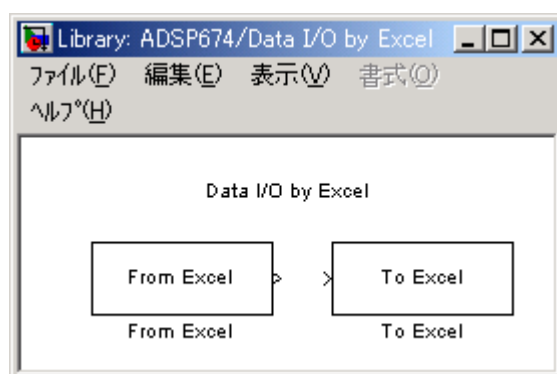
出力するD/Aのチャンネル数を1～128の整数で指定します。チャンネル番号とハードウェアとの対応は、表 2.13.1 (141ページ) を参照してください。

2.14 Data I/O By Excel ブロック

2.14.1 概要

図 2.14-1 に、Data I/O By Excel 用ウィンドウを開いた所を示します。

図 2.14-1 Data I/O By Excel ウィンドウ



この中には自動テスト機能で使用可能な From Excel/To Excel ブロックが登録されています。

- From Excel Microsoft Excel にて作成したデータを実行時動的にブロックへ取り込みます。
- To Excel Microsoft Excel ブックへ実行時に動的にデータを出力します。

To Excel ブロックは、自動テストにて指定項目を設定し、実行した場合のみ有効となります。

これらのブロックは、データ用の Excel ブックが必要となります。

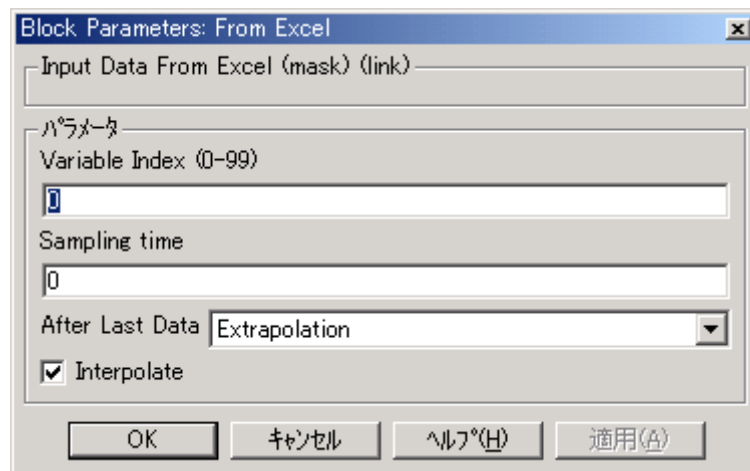
1) 自動テスト時は、第7章自動テストを参照してください。

2) From Excel を RTMON で使用する場合も Excel ブックが必要となります。Excel ブックの作成方法については第7章自動テストを参照してください。RTMON を実行時は“リアルタイムオプション”ダイアログにて Excel ブックの指定と FrmExcel ブロック使用にチェックをしてからモデルを実行してください。

2.14.2 From Excel ブロック

From Excel ブロックは、Microsoft 社製 Excel にて作成した、時間付データを実行時に動的にアクセスしモデルへデータを取り込みます。このブロックは自動テストにてモデルを実行した場合と R T M O N にて R T オプションで“Frmxls ブロック使用”を指定し実行した場合のみ有効となります。又実行前にデータ領域を D S P へ確保します。D S P 上に必要なメモリが確保できない場合は使用できません。

図 2.14-1 From Excel



- Variable Index(0-99)
自動テスト用の Excel ブックで指定する変数 Index を 0 ～ 9 9 の範囲で指定します。ToExcel ブロックと共通ですので、両ブロック間で重複しないように設定してください。
- Sampling time
データを取り込むサンプリングタイムを指定します。0 指定時はモデルのステップサイズに依存します。
- After Last Data
指定した時間以降のデータ実装方法を指定します。
Extrapolation：直前のデータ内容よりデータを外挿します。
Setting To Zero：ゼロデータを出力します。
Holding FinalValue：指定最終データを出力します。
- Interpolate
データの指定範囲内でモデルのステップタイムに該当するデータがない場合の実装方法を指定します。
チェックあり：データは内挿されます。
チェックなし：取り込んだデータの最近のデータ値を出力します。

<データ取り込み例>

1) 最終データ後の出力データ: Extrapolation、Interpolate : チェックあり 指定時

① Excel データ)

```
time 1 2 3 5 8
data 0 1 5 50 100
```

↓

ステップサイズ 1 Sec での取り込みデータ)

```
time 0 1 2 3 4 5 6 7 8 9 10 11
data 補間 1 内挿 50 6 3.33 100 1 3.33
150 外挿
```

② Excel データ)

```
time 0 1 2 2 3
data 0 0 1 5 10
```

↓

ステップサイズ 1 Sec での取り込みデータ)

```
time 0 1 2 3 4
data 0 0 5 10 15
```

*Excel データが同じタイムステップで複数存在する時は、最後のデータが使用されます。

*最終データが同じタイムステップで複数存在する時は Extrapolation が保障されませんので、注意してください。

2) 最終データ後の出力データ: Setting To Zero、Interpolate : チェックなし 指定時

① Excel データ)

```
time 1 2 3 5 8
data 0 1 5 50 100
```

↓

ステップサイズ 1 Sec での取り込みデータ)

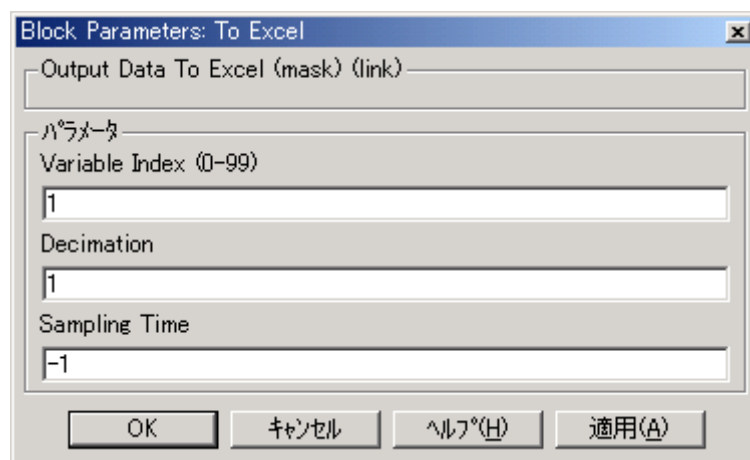
```
time 0 1 2 3 4 5 6 7 8 9 10 11
data 0 0 1 5 5 50 50 50 100 0 0 0
```

Setting To Zero

2.14.3 To Excel ブロック

To Excel ブロックは、指定された Microsoft 社製 Excel ブックのシートへ、時間と入力データを書き込みます。このブロックは自動テストにてモデルを実行した時のみ有効となります。このブロックは実行前にデータ領域をDSP上に確保します。DSP上に必要なメモリが確保できない場合は使用できません。

図 2.14-2 To Excel



- Variable Index(0-99)
自動テスト用のExcelブックで指定する変数 Index を 0 ～ 9 9 の範囲で指定します。FrmExcel ブロックと共通ですので、両ブロック間で重複しないように設定してください。
- Decimation
書き込みデータの間引きを指定します。1 指定時は間引き無し、2 指定時は、2 データに対し 1 データ書き込みとなります。
- Sampling Time
データを書き込みサンプリングタイムを指定します。

3 他のDSPとの互換性

3.1 他のDSPとの互換性

他のDSP用の環境「ADSP324-41B」「ADSP404-241B」（共にバージョン4.x）で作成したブロック線図は、一部のブロックを除きコードを再生成する事で「ADSP674-341」での動作が可能です。

図 3.1-1 異種DSP用環境間のI/Oデバイスの互換性

ブロックライブラリ	互換性
ADSP32X-00/50 ブロック	無し
ADSP404-00 ブロック	無し
ADSP32X-03/53 ブロック	有り
ADSP32X-06/56 ブロック	有り
ADSP32X-11/61 ブロック	有り
ADSP32X-13/63 ブロック	有り
ADSP324-141 ブロック	有り
ADSP324-143 ブロック	有り
TM32DA ブロック	有り

ADSP 674-341C
ML6.x 用 I/O ライブラリ

第 1 版 2001.03.20

第 6 版 2006.02.01

中部電機株式会社 CS 事業部

TEL 0532-61-9566

FAX 0532-63-1081

ユーザー定義ブロック

(ユーザー定義 S-Function)

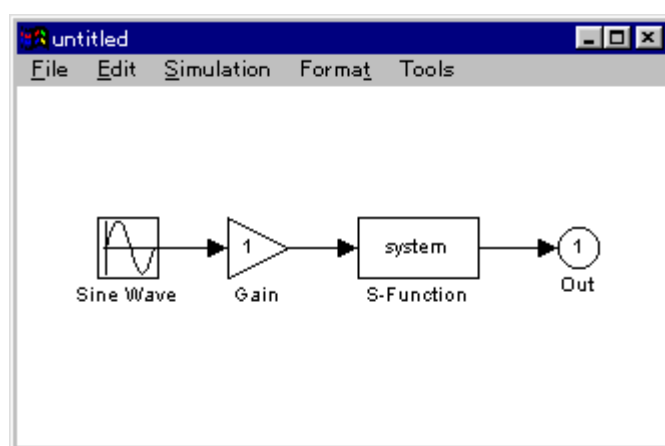
<< 目次 >>

1	概要	2
2	利用出来る S - F u n c t i o n	2
3	ユーザ定義ブロックの作成.....	3
3.1	概略手順.....	3
3.2	ソースファイルの作成.....	3
3.3	D L L の作成.....	5
3.4	S I M U L I N K での動作確認	5
3.5	ライブラリの作成.....	6
3.6	ライブラリ作成時のテンポラリフォルダー	8
3.7	RTW でのコード生成.....	9
3.8	RTW での動作確認.....	9
3.9	CCS での動作確認.....	9

1 概要

ここでは、S I M U L I N K の拡張機能である S-Function を用いて作成されているブロックを用いたブロック線図の実時間コード化について説明します。

S-Function は S I M U L I N K の拡張機能の一つで、ブロック線図で使用出来る、カスタマイズされたブロックを作成する機能です。出来たブロックは下のブロック線図の様に標準のブロックと接続して、ブロック線図の一部として使用します。



2 利用出来る S-Function

S I M U L I N K では、色々な言語による S-Function が利用出来ましたが、Real-Time Workshop を用いて D S P 用の R T モデルを作成する場合には C 言語による S-function しか利用出来ません。

<S-Function で利用可能な言語>

記述言語	S I M U L I N K	Real-Time Workshop
MATLAB 言語(M ファイル)	可	不可
C 言語	可	可
F O R T R A N 言語	可	不可

これは専ら、D S P 用のコンパイラが、C コンパイラしか無い事に由来しています。従って、将来 Real-Time Workshop を用いて D S P 用実時間コードを作成する予定がある場合、S-Function は C 言語で作成する必要があります。

3 ユーザ定義ブロックの作成

3.1 概略手順

DSPを用いた実時間モデルで使える S-Function によるユーザ定義ブロックの作成手順は以下の通りとなります。

- 1、ソースファイルの作成
- 2、DLLの作成
- 3、SIMULINKでの動作確認
- 4、ライブラリの作成
- 5、RTWでのコード生成
- 6、RTモデルでの動作確認

3.2 ソースファイルの作成

ソースファイルはMATLAB エディター(又は汎用のテキストエディタ)にて作成します。SIMULINKには、S-Function のサンプルとして CSFUNC が用意されています。本製品ではこれに倣ったサンプル、CSFUNC.C が以下のフォルダーに用意されています。

MATLAB\RTW\CYADSP32X\SAMPLE\SFUNCTION

これを参考にすると良いでしょう。同じフォルダーにある、Csfunctest.mdl はこの S-Function を使用するブロック線図の例です。

この関数及びソースファイルの詳しい内容等、及び S-Function の詳しい説明はSIMULINKのユーザズガイドをご参照ください。

Using SIMULINK Version 2 の8章 「S-Function」が参考になります。

作成する S-Function のソースファイル名には次に示す制約があります。

- 名称の先頭は、英字で始めてください。
良い例： a3bc.c
悪い例： 3abc.c (理由=数字で始まっている)
- 名称の文字数は、8文字以内としてください。
良い例： goodname.c
悪い例： nameislonger.c
- 名称で使う文字は英数字と _ 記号のみとしてください。
良い例： a32_abc.c
悪い例： a(#1).mdl (理由=かっこ、#等の記号を使っている)
- 英字の大文字と小文字は同じものとして取り扱われます。

作成する S-Function はSIMULINK用のDLLと、DSP用のライブラリの2つが必要となりますが、場合によっては、例えば希望とする処理が数値計算等メモリー内のデー

タ操作だけで終わってしまう様な場合、両者は、同一のソースファイルを使う事ができます。

しかしそうでなく、例えばDSPのI/Oを扱う様な場合は、DLLは（PC上で動作し、DSPのI/Oは直接扱えませんから）ダミーとなり、DSP用ライブラリになった時点で初めて処理実体が必要となります。この様な場合は、同一ソースファイルでありながら、部分部分でDLL用にコンパイルしている時とDSP用にコンパイルしている時と異なる記述が必要となります。そういった場合には、次の条件付きコンパイルを用いると同一ソースファイルとする事が可能です。

```
#ifdef  MATLAB_MEX_FILE
    /* SIMULINK 用 DLL の時に有効となります。 */
#else
    /* DSP用ライブラリの時に有効となります。 */
#endif
```

3.3 D L Lの作成

MATLAB 6.xでは、ユーザ定義 S-Function を含むブロック線図から Real-Time Workshop で実時間コードを作成する場合には、必ず、同 S-Function のD L L版を用意する必要があります。S I M U L I N K上ではMファイル版の S-Function でも動作しますが、Real-Time Workshop によるコード生成は、D L Lが無いと正しく行えません。

D L Lによる S-Function は mex コマンドで行います。Mex コマンドについてはS I M U L I N Kのユーザズマニュアルを参照してください。以下が参考になります。

MATLAB Application Program Interface Guide の2章 「Creating C Language MEX-Files」
MATLAB Command window から help mex 又は mex -help を実行

3.4 D L L作成用コンパイラの動作保証範囲

mex コマンドで使用するコンパイラについて、本製品出荷時の動作保証範囲は、MATLAB のコンパイラ Lcc のみとなっています。悪しからずご了承願います。

3.5 S I M U L I N Kでの動作確認

D L Lによる S-Function が出来たらS I M U L I N K上で正しく動作するか確認します。作成したD L Lはカレントディレクトリ又は MATLAB の PATH の通ったフォルダーに格納します。デバッグの段階ではブロック線図と共にカレントディレクトリに置いておき、正常動作が確認できた段階でP A T Hの通ったフォルダーに移動する方が便利です。

ここではD S Pの I / Oに関する確認は行えません。専らブロック線図との I / F 部分と、計算処理部分についてのみ動作確認します。

3.6 ライブラリの作成

SIMULINK上で正しく動作する事が確認出来たら、DSP用ライブラリを作成します。

RTモデル用ライブラリ化は、以下の手順で行います。

- (1) ライブラリ生成用ディレクトリを作成し、ここにMATLABカレントディレクトリを移動してください。(MATLABのCDコマンド参照)
- (2) カレントディレクトリに、**S-Function** のソースファイルをコピーします。カレントディレクトリには、この **S-Function** のソース以外のCソース (拡張子 **.c**) を格納しないでください。誤ってライブラリ化されてしまいます。
- (3) ライブラリの作成は、ライブラリ作成コマンド **genlib674** で行います。

標準的な使い方は、**genlib674 -r -q** です。これによりCソースがコンパイルされ、**opt_lib.lib** という名称のライブラリが作成され、所定のディレクトリにコピーされます。

- (4) ライブラリを複数作成する場合は、**genlib674 -r -q mylib** 等とすると、任意の名称のライブラリ **mylib.lib** が作成されます。但し、**a674_rt** 及び **libsrc67** は予約となっていてユーザは利用出来ません。**-r** はライブラリを所定のディレクトリに自動的にコピーする指示です。この指示をしない場合、カレントディレクトリにライブラリが作成されていますので、後でコピーしてください。**-q** は **-r** の指示により自動コピーする際に、以前に作成した同名ライブラリが有っても強制的に上書きします。**-q** が無いと上書きするか否か問い合わせてきます。
- (5) **genlib674 -help** と入力するとオンラインヘルプが参照できます。併せ参照してください。出来たライブラリをコピーする先のディレクトリ名もこれで確認できます。
- (6) **mex** で **S-Function** の **DLL** コード生成が正しく出来ても、**genlib674** によるコンパイルでエラーが出る場合があります。この理由の殆どは、Cコンパイラのエラーチェック仕様の相違によるものです。DSP用Cコンパイラの方が (必要以上に) 型チェックが厳しい為、**MEX** の **DLL** で正常にコンパイル出来て、正常に動作しているにも拘わらず、エラーが出る場合があります。

その最大のものが、**real_T *** と **double *** の相違です。**real_T** は`#define`により **double** として定義されていますが、DSP用コンパイラは別物として扱ってしまいます。例えば **simstruc.h** で、関数プロトタイプ宣言の引数が **real_T *** と定義されていて、実際の関数の引数が **double *** となっていると、エラーとなってしまう場合があります。

コンパイラが出力するエラーメッセージはMATLAB Command Windowに表示されますが、ファイル化されテンポラリフォルダーにも格納されています。ファイルの位置については3.7

ライブラリ作成時のテンポラリフォルダーを参照してください。

- (7) ADSP32X シリーズと、ADSP674 とは同じソースファイルを用いてライブラリ化できますが、ライブラリそのものは共用出来ません。同じフォルダーで **genlib32** と **genlib674** をそれぞれ行い、個別にライブラリの作成を行ってください。この時ライブラリ名称を同じとすると、カレントディレクトリ上でライブラリが上書きされますので、生成する都度、それぞれ所定のフォルダーにコピーしてください。

注意) 上記内容は **Small** モデルを基準に説明しています。**Large** モデルで作成時は **genlib674b** にて作成してください。又モデルによりライブラリが格納されるディレクトリが以下のように異なります。

Small モデル	MATLAB フォルダ ¥rtw¥c¥adsp32x¥lib674
Large モデル	MATLAB フォルダ ¥rtw¥c¥adsp32x¥lib674¥big

3.7 ライブラリ作成時のテンポラリフォルダー

ライブラリ作成時に作業用フォルダーの下に下記フォルダーが作成されます。

UserSFunLib_gen_Work

この中には以下のファイルが格納されています。

オブジェクトファイル	*.obj	S-Funtcion のソースファイル名の拡張子を .OBJ に変えたファイルです。
エラーログファイル	err	コンパイル中の作業進行状況とその途上で発生したコンパイルエラーが記録されています。コンパイラが出力したメッセージそのものをファイル化したもので、テキストエディターで閲覧が可能です。エラー訂正の為の情報となります。閲覧後に再度ライブラリを作成する場合はエディターを一旦閉じないと最新のエラー情報がファイルに反映されない場合がありますので注意が必要です。
ライブラリファイル	*.lib	ユーザが指定した名称の（指定を省略した場合はデフォルトの名称の）ライブラリファイルです。コンパイルエラーが無い場合のみ作成されます。一度正しく作成された後 S-Function のソースファイルを変更し、再度ライブラリ作成をしようとしたときにコンパイルエラーが発生した場合はライブラリは古いままに残っています。手動でコピーする場合は日付を良く確認してください。

これらのファイルは、ライブラリが正常に作成され、かつ、ライブラリを所定のフォルダーにコピーした後は不用となります。フォルダごと削除しても構いません。

3.8 RTW でのコード生成

ライブラリ化が終わり、作成されたライブラリが所定のフォルダに格納された後、再度 Real-Time Workshop 実時間コードを生成します。これにより先程作成されたライブラリは自動的にリンクされます。

3.9 RTW での動作確認

Real-Time Workshop で生成した実時間モデルは、リアルタイムモニターにて実行してみ、所要所の波形を見る（或いはロギングして見る）のが唯一の方法です。

DSP 用ライブラリになってからの、S-Function 単体のデバッグは一般には困難です。もし S-Function（又はその内部）のデバッグを行う必要があるなら、ソースファイルにパッチを当てる事で動作結果をメモリー（大域変数）に残す様に変更し、リアルタイムモニターにて動作後、弊社付属ソフトウェア（ADSP 674 ユーティリティ）にてメモリーを表示する方法が使えます。メモリーアドレスは、モデルと同じディレクトリにマップファイル *model.MAP* が生成されていますので、これを参照してください。

3.10 CCS での動作確認

もし CCS と XDS をお持ちであれば、これを用いて DSP ボード上で S-Function のデバッグが可能です。具体的方法はユーザ定義関数と同じですのでそちらを参照してください。

ユーザ定義ブロック
(ユーザ定義 S-Function)

第 4 版 2003.06.11

中部電機株式会社 C S 事業部

TEL 0532-61-9566

FAX 0532-63-1081

ユーザ一定義関数

(ユーザ記述による C 言語関数)

<< 目次 >>

1	概要	3
2	実行形態	4
2.1	処理順序	4
3	関数詳細説明	7
3.1	UserInitializeSizes()	7
3.2	UserInitialize()	8
3.3	UserOutput()	8
3.4	UserUpdate()	9
3.5	UserBackground()	9
3.6	UserTerminate()	9
4	データ交換	11
4.1	データ交換規則	11
4.2	入出力点数の自動整合	13
4.3	入出力受け渡しバッファ	14
5	ユーザ定義関数からのリアルタイムモデル停止	15
5.1	方法 1 ～ Stop Simulation	15
5.2	方法 2 ～ シミュレーション停止フラグの操作	15
6	使用方法	16
6.1	USERFUNC.C のインストール	16
6.2	USERFUNC.C のコンパイル	16
6.2.1	自動によるコンパイル	16
6.2.2	手動によるコンパイル	17
6.3	手動による USERFUNC.C のコンパイルとリンク	18
6.4	USERFUNC.C のデバッグ	19
6.4.1	ダミーmain によるデバッグ	19
6.4.2	ブロック線図ぐるみのデバッグ	19
6.5	個別モデル専用のユーザ定義関数	20
6.6	ML 5.x とML 6.x の共用	20
7	制限事項	21
7.1	動作上の機能制限	21

7.2	使用出来ないソースファイル名	21
7.3	長時間に渡るDSPの専有	23
7.4	長時間に渡る割り込み禁止	24

1 概要

本章は、MATLAB 対応ライブラリ「ADSP324-41C」「ADSP404-241C」「ADSP674-341C」(Rev 4.x 版以後) 於いて、リアルタイムモデルにユーザ定義関数を組み込む方法について説明しています。

この機能は SIMULINK では記述しづらい手続きや、より高速化／高機能化等を実現したい場合、ユーザの C 言語によるソフトウェア資産を活用したい場合等に、ユーザが記述した C 言語の関数を SIMULINK ブロック線図で記述した処理に組み込む事により、両者を交互に実行する事により、擬似的に並行して実行する為ものです。予め定められた関数名と大域変数を用いる事によりブロック線図とのインターフェースを取り、データを交換する事もできます。

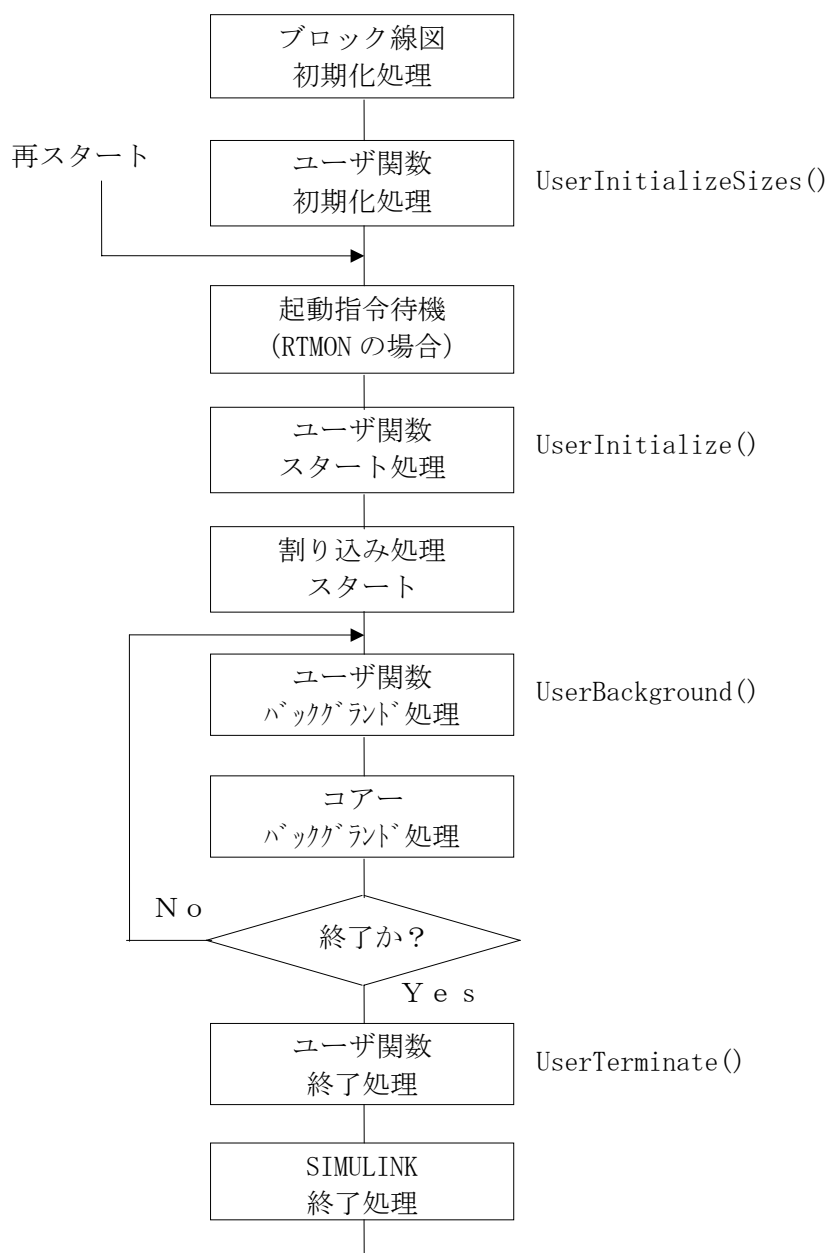
2 実行形態

ここでは、SIMULINK ブロック線図の処理とユーザ定義関数の処理との順序関係、データ交換の方法等を説明します。

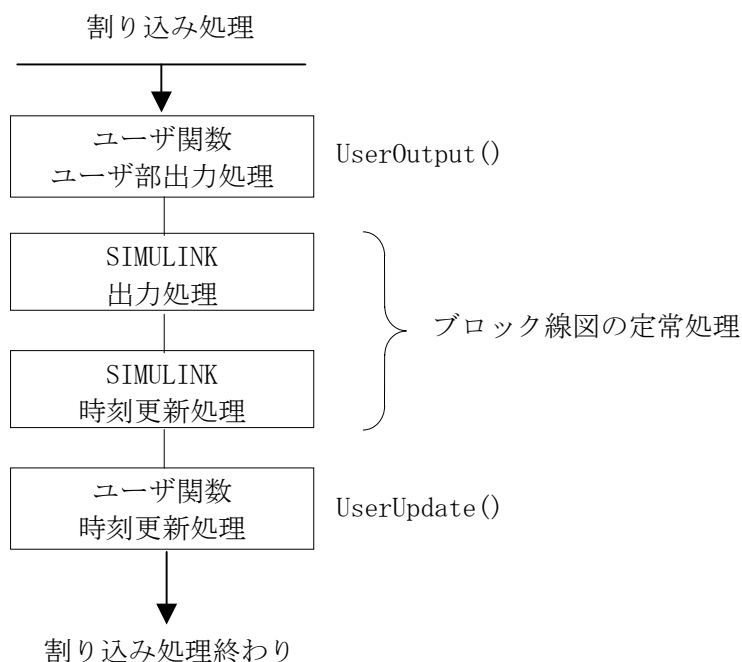
2.1 処理順序

処理の順番は以下のフローチャートにより行なわれます。

- メインルーチン



● 割り込み処理ルーチン



メインルーチンは、初期化・起動司令待機後、割り込みを起動したらバックグラウンド処理を行います。ここでは通常、

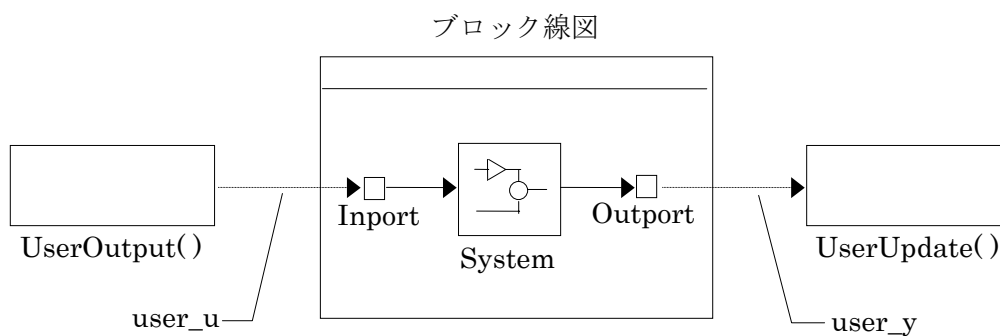
- 1、リアルタイムモデルが終了時刻に達したか確認を行う。
- 2、RTMON、又は、UILからのランタイムアクセスを処理する。
- 3、それ以外のバックグラウンド処理を行う。

以上の作業を行っています。

一方、割り込み処理は、実行形式によりタイマー又は外部割り込みにより駆動され、リアルタイムモデルの1ステップ分の計算を行い、メインルーチンへ戻ります。

フローチャートに示す通り、ユーザ定義関数では、これらの処理の要所々々でユーザの定義する関数を呼び出すことが出来ます。

別の見方として、SIMULINK ブロック線図的な表記では下記の様な接続関係となります。



中央が SIMULINK のブロック線図で、両横の がユーザが定義した処理です。
処理順序は図の向かって左側から右側へ順に処理されます。破線矢印（——▶）は大域変数を使用したデータを転送を表しています。ブロック線図には現れません。このように、ユーザ処理と SIMULINK ブロック線図との間のデータ交換は SIMULINK のトップレベルの Inport、Outport を通じて行います。
user_u, user_y と Inport, Outport との対応の詳細については“データ交換”の章を参照してください。

3 関数詳細説明

ユーザ定義の関数は予め予約された関数名が6つ用意されています。

UserInitializeSizes()	ユーザ固有の初期化処理を行います。また、ユーザ関数とSIMULINKとのデータ交換に使う領域もここで確保します。
UserInitialize()	システムが機動・停止を繰り返す必要がある場合、再起動に必要な初期化処理を行います。
UserOutput()	ユーザブロックからSIMULINKへの出力を計算します。
UserUpdate()	SIMULINKの出力を受け取り処理すると共に時刻更新の処理を行います。
UserBackground()	ユーザのバックグラウンド処理を行います。
UserTerminate()	システム停止の処理を行います。

上記関数はテンプレートファイル USERFUNC.C に予め定義されており、最低限必要な処理が予め記述されています。ユーザはこれに必要な手続きを書き足します。関数名が定められている事を除けば、通常の間数と同じです。

3.1 UserInitializeSizes()

関数定義 int UserInitializeSizes(int in_num, int out_num);

意味 この関数はモデル起動直後に1度だけ呼び出されます。システムの初期化、作業領域の確保等の初期化作業のうち最初に1度だけ行えばよい処理をここで行なってください。

引数 int in_num;

SIMULINK ブロック線図への入力ポート (Inport) の数が与えられます。データ受け渡しに使用する大域変数配列 user_u は、最低限これ以上の長さが必用です。この変数は同時に、他のユーザ定義関数から参照可能とする為、u_num 等の大域変数に保存する事をお勧めします。これにより user_u を、確保した長さ以上にアクセスしないよう管理可能です。

int out_num;

SIMULINK ブロック線図の出力ポート (Outport) の数が与えられます。データ受け渡しに使用する大域変数配列 user_y は、最低限これ以上の長さが必用です。user_y は、ユーザの使用の有無に関わらず、out_num の点数分書き込みが行われます。この変数は同時に、他のユーザ定義関数から参照可能とする為、y_num 等の大域変数に保存する事をお勧めします。これにより user_y を、確保した長さ以上にアクセスしないよう管理可能です。

戻り値 エラーの有無を返します。正常終了で 0 を、それ以外は 0 以外を返します。エラーを返すとメインルーチンは処理を中断します。リアルタイムモニターでは、このエラーをヒープ領域不足として扱い、ステータス表示します。ここでユーザがエラーを返す可能性がある場合、表示の意味に注意してください。

3.2 UserInitialize()

関数定義 `void UserInitialize( void );`

意味 システムが起動停止を繰り返すような場合、起動ごとに必用となる初期化処理をここに記述します。

引数 ありません。

戻り値 ありません。

3.3 UserOutput()

関数定義 `void UserOutput( double t );`

意味 定常状態でステップサイズ毎に、最初に呼び出される処理です。ユーザ処理部分の出力、(即ち SIMULINK ブロック線図への入力) を決定します。

引数

`double t;`

現在の時刻が与えられます。システム起動後最初の呼び出しには、リアルタイムモニター、又は SIMULINK の Real-Time Options で指示した開始時刻 (Start time) が与えられます。通常は 0 です。その後、呼び出し毎にステップサイズだけ加算されます。

戻り値 ありません。

3.4 UserUpdate()

関数定義	<code>void UserUpdate(void);</code>
意味	定常状態でステップサイズ毎に、SIMULINK ブロック線図の計算が終了した直後に呼び出される処理です。必用に応じて SIMULINK ブロック線図の出力を受け取り、処理をします。また、時刻更新の為の処理もここで行います。
引数	ありません。
戻り値	ありません。

3.5 UserBackground()

関数定義	<code>void UserBackground (void);</code>
意味	ユーザのバックグラウンド処理を記述します。通常メインルーチンでのバックグラウンド処理は、割り込みによるリアルタイムモデルの計算を待ち受ける傍ら、RTMON, UIL 等によるランタイムアクセス（実行時のパラメータ変更）をソフト的なハンドリングで行っていますので、これらを妨げないように、時分割で処理する必要があります。長時間かかりそうな処理は、何回かに分け、この関数が呼び出される毎に順番に一部分ずつ実行するようにしないといけません。詳しくは、7.3 長時間に渡るDSPの専有 を参照してください。
引数	ありません。
戻り値	ありません。

3.6 UserTerminate()

関数定義	<code>void UserTerminate(void);</code>
意味	システムの終了処理を記述します。リアルタイムモデルが停止する直前に1度だけ呼び出されます。リアルタイムモデルは以下の条件で停止します。 1、 現在時刻が終了時刻（Stop time）に到達した場合。 2、 ブロック線図の Stop Simulation ブロックの入力が0以外になった場合。 3、 リアルタイムモニターの停止ボタンが押された場合。 4、 ユーザプログラムから、UIL 経由でリアルタイムモデルの停止機能が呼び出された場合。 5、 ユーザ定義関数で、シミュレーション停止フラグを操作した場合。

D S P ボード付属コマンドの RESET324、HOLD324 等でD S Pを停止した場合は、この関数は呼び出されません。

引数 ありません。

戻り値 ありません。

4 データ交換

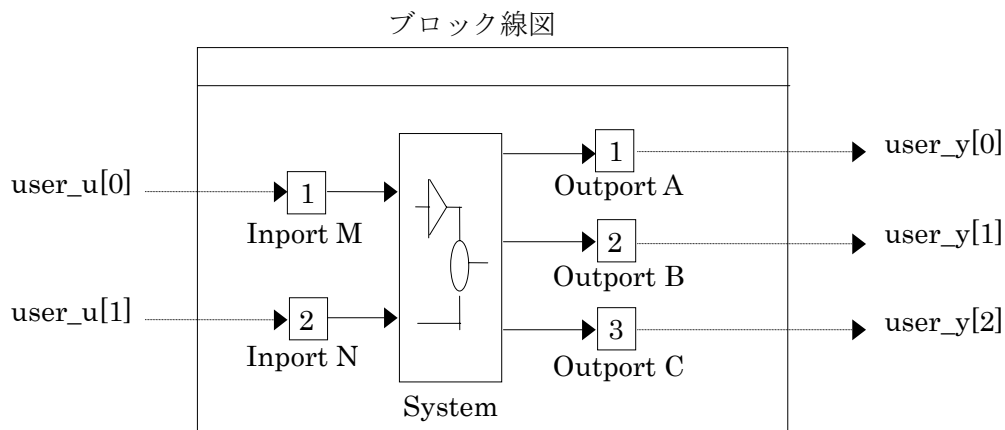
4.1 データ交換規則

ユーザ定義処理と SIMULINK ブロック線図との間は大域変数を経由してデータ交換をします。データ領域は USERFUNC.C に大域変数として定義されており、実体は初期化関数 UserInitializeSizes() 内で malloc() にて割り付けます。

double	*user_u;	SIMULNI ブロック線図への入力データを格納します。
double	*user_y;	SIMULNI ブロック線図からの出力データが格納されます。
int	u_num;	user_u の長さを記憶するのに使用します。
int	y_num;	user_y の長さを記憶するのに使用します。

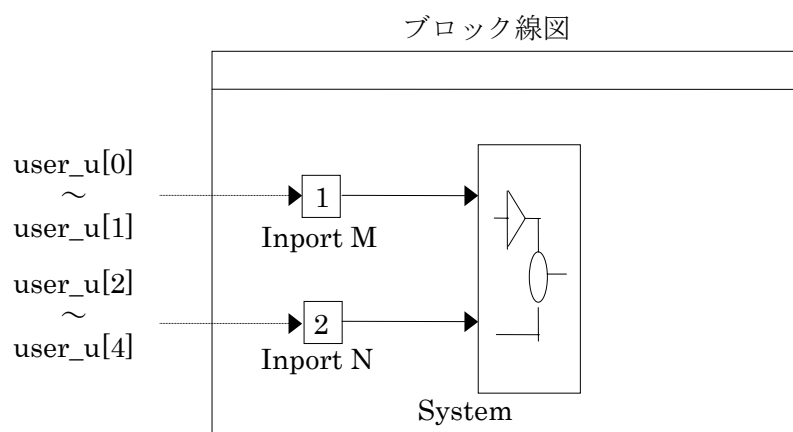
user_u、user_y の長さはそれぞれ u_num、y_num で管理し、確保した長さ以上にアクセスしないように注意してください。user_u[] (user_y[]) の添え字の最大は、u_num-1 (y_num-1) となります。

user_u、user_y と SIMULINK ブロック線図の Inport、Outport との対応は下記に示す通りです。この場合は、Inport、Outport はいずれもスカラーとします。



この例のように、SIMULINK ブロック線図のポート番号（□の中の番号）- 1 が配列の添え字となります。この場合、u_num == 2、y_num == 3 となります。

Inport、Outport がベクトルの場合の例を次に示します。



1 本のベクトルは配列の中の連続する変数に割り当てられ、ベクトルの要素が若い方が配列の添え字の若い方に相当します。次の Inport はそのすぐ後の変数に相当します。

例えば、このブロック線図で、Inport M が幅 2、Inport N が幅 3 のベクトルであるとする
と、以下の様に対応します。

user_u[0]	Input M (1)
user_u[1]	Input M (2)
user_u[2]	Input N (1)
user_u[3]	Input N (2)
user_u[4]	Input N (3)

4.2 入出力点数の自動整合

ユーザ定義関数でブロック線図との間でデータ交換をする、入出力ブロックの点数 (user_u[] の添え字) は、ブロック線図とユーザ定義関数とで整合していないといけません。特に、ブロック線図で定義された以上の長さについてアクセスをすると、場合によってはDSPが暴走する等の不都合が生じる場合があります。

MATLAB 6.x では、この点数を表す定数がヘッダに定義されており、これを参照する事により、コンパイル時点でエラーを発生させてユーザに警告する事が可能です。

例えばユーザ定義関数側が出力 (ユーザ出力→ブロック線図) として 2つのポートを、入力 (ブロック線図→ユーザ時刻更新) として 3つのポートを、それぞれ想定してコードを作成した場合、*MODEL.UFN* コードの適当なカ所に、

```
#if ML51
#if NINPUTS != 2
    #error : Not expected number of InputPort of Model.
#endif
#if NOUTPUTS != 3
    #error : Not expected number of OutputPort of Model.
#endif
#endif
```

と記述します。

NINPUTS がブロック線図への入力、NOUTPUTS がブロック線図からの出力の点数を表しています。点数とはブロックの数でなく、各ブロックのベクトルの、幅の合計ですので注意が必要です。入力ブロックが1個でも、ここへ接続されている信号が幅3のベクトルの場合 NINPUTS は3となります。

もし、ブロック線図に想定外の点数の入出力ポートが定義されると、**Not expected number ...** が MATLAB Command window に表示され、コード生成作業が中断されます。

また、この定数を参照する事により、自動的に入出力点数を整合させる事も可能です。コンパイル時点で点数が判明していますので、作業用変数領域の確保が

```
double work[NINPUTS];
```

等としての確に確保する事ができます。

4.3 入出力受け渡しバッファ

MATLAB 5.1 の場合は、MATLAB 4.2c 版の場合と同じく、`user_u`、`user_y` でも受け渡しができますが、入出力ポートを定義した大域変数が用意されており、これを直接参照する事もできます。入力ブロック、出力ブロックを、それぞれ構造体で定義しています。構造体定義は *MODEL.H* にあり、実体は *MODEL.PRM* に記述されています。

<例 : *MODEL.H* (抜粋) >

```
/*
 * External Inputs Structure
 *
 * Note: Individual field names are derived from the block name.
 */

typedef struct ExternalInputs_tag {
    real_T In[2];
} ExternalInputs;

/*
 * External Outputs Structure
 *
 * Note: Individual field names are derived from the output block names
 */

typedef struct ExternalOutputs_tag {
    real_T Out[3];
} ExternalOutputs;
```

<例 : *MODEL.PRM* (抜粋) >

```
/* External Inputs Structure */
ExternalInputs rtU;

/* External Outputs Structure */
ExternalOutputs rtY;
```

5 ユーザ定義関数からのリアルタイムモデル停止

ユーザ定義関数からリアルタイムモデルを安全に強制停止させるには、以下の方法を用いてください。何れの方法も、停止要求が受理された後、ブロック線図の停止処理、ユーザ定義関数の停止処理等が呼び出され、R Tモデルが停止します。

これ以外の方法（例えばタイマーを強制的に止める等）で停止させた場合、本来行うべき終了処理（Terminate）が呼びだされません。

5.1 方法1 ～ Stop Simulation

ユーザ定義関数の出力 `user_u` を使って、停止制御用信号をブロック線図に渡します。ブロック線図ではこれを **Stop Simulation** ブロックに入力します。

常時は、0 を出力しておき、停止させたい場合は1 を渡します。

この方法は、停止する直前にブロック線図でもこれを参照して、何がしかの処理が行えますので、ブロック線図内で停止直前に行うべき処理がある場合に利用できます。

5.2 方法2 ～ シミュレーション停止フラグの操作

メイン関数に対し、シミュレーションを停止させる要求を渡すフラグが用意されています。これを直接操作します。

参照変数	<code>extern SimStruct *const rtS;</code>
停止指示	<code>ssSetStopRequested(rtS,1);</code>

この方法はブロック線図に頼らず強制的に停止が可能です。

6 使用方法

6.1 USERFUNC.C のインストール

本製品のインストール直後の状態では、ユーザ定義関数は使用しない状態になっています。これを使用出来るようにするには、以下の作業を行います。

MATLAB をインストールしたフォルダ（%MATLAB_ROOT%）以下の、次に示すフォルダに USERFUNC.C が格納されています。これを MATLAB のカレントディレクトリにコピーします。

%MATLAB_ROOT%\Rtw\c\Adsp32x\Rt\	(ADSP324-41C の場合)
%MATLAB_ROOT%\Rtw\c\Adsp32x\Rt404\	(ADSP404-241C の場合)
%MATLAB_ROOT%\Rtw\c\Adsp32x\Rt674\	(ADSP674-341C の場合)

その後、MATLAB のカレントディレクトリの USERFUNC.C を必要に応じて変更してください。

6.2 USERFUNC.C のコンパイル

6.2.1 自動によるコンパイル

ユーザ定義関数のファイル “USERFUNC.C” は、上記変更により、Build ボタンから自動的にコンパイルされます。記述したユーザ定義関数にエラーが無ければこの方法で充分で、特にユーザが個別にコンパイルする必要はありません。

しかし一般には、プログラマーが記述したソースファイルは色々な意味でエラーを含んでおり、書きっぱなしでそのまま正常に動作するのは稀なのが通常です。当然、1 つでもエラーがあると実行ファイルは生成されません。その度に USERFUNC.C を修正し再度 Build をするのは、SIMULINK モデルのコード生成とコンパイルをも毎行なう事になり、その分、無駄な時間を費やす事になります。

そこで、最低限タイプミスや文法上の誤りを USERFUNC.C から予め取り除いておく事をお勧めします。

6.2.2 手動によるコンパイル

最低限タイプミスや文法上の誤りを USERFUNC.C から予め取り除いておく等の目的で、ユーザ定義関数のみを手動でコンパイルする事が可能です。

手動によるコンパイルの方法は以下の通りです。

1. MS-DOSプロンプトを起動します。
2. DOSのカレントディレクトリをSIMULINKブロック線図が格納されているディレクトリ (Matlab Command Window のカレントディレクトリ) に移動します。
3. 以下のコマンドでコンパイルします。

<code>cl30 userfunc.c</code>	(ADSP324-41C の場合)
<code>cl30 -v44 userfunc.c</code>	(ADSP404-241C の場合)
<code>cl6x -mv6700 userfunc.c</code>	(ADSP674-341C の場合)

不幸にもエラーがあまりに多く、エラーメッセージが画面からスクロールアウトしてしまう場合は、

```
cl30 userfunc.c > error
```

等として、エラーメッセージをファイルに残すとよいでしょう。この様にすると画面に出力されていたエラーメッセージが、ファイル `error` に保存されます。

C言語ではちょっとしたタイプミス1文字でもそれが文法的なエラーだと、それを契機に連鎖的に多数のエラーが出る場合があります。これはコンパイラが最初のエラーにより後続の正しい部分をエラーと誤解してしまう為です。真のエラーの数とエラーメッセージの数は必ずしも一致しませんから、エラーメッセージが多い事自体は全く悲観する必要はありません。このような場合、えてして、最初に表示されたエラー1つを解決すれば（文法的には）全て解決します。従って、先頭に近い側のエラーから順に確認して行き、原因が判らない場合はもう一度コンパイルしてみます。

肝心の最初に表示されるエラー行をスクロールアウトで見失わない為には、エラーメッセージをファイル化するのは有効な方法です。

6.3 手動による USERFUNC.C のコンパイルとリンク

コンパイル段階でエラーが無くなってもリンクの段階でエラーが出たり、又は、ブロック線図は無変更でユーザ定義関数のみ変更した場合等で、ユーザ定義関数のコンパイルとリンクのみ行いたい場合があります。その場合はブロック線図のコード化とコンパイルは不要ですのでその時間が無駄となります。

その場合は以下の方法で手動によるユーザ定義関数のコンパイルとリンクが行えます。

- (1) ユーザ定義関数ソース（エラーがあっても構わない）がカレントディレクトリにある状態で、RTWでコード生成を行います。
- (2) この時の、MATLAB Command Window に表示されたメッセージに注目します。例えば、test.mdl をコード生成した場合のメッセージは以下となります。

```
### Starting RTW build procedure for model: test
### Invoking Target Language Compiler on test.rtw
### Loading TMW TLC function libraries
### Initial pass through model to cache user defined code
### Creating source file test.c
### Creating part 1 of registration file test.reg
### Creating parameter file test.prm
### Creating model header file test.h
### Creating part 2 of registration file test.reg
### TLC code generation complete.
### test.mk which is generated from D:\MATLAB\511\rtw\c\adsp32x\32x.tmf is
up to date
### Building test: nmake /nologo -f test.mk STEP=0.01 UFUNC=un
### Compiling test.c
### Compiling rt5sim.c
(以下省略)
```

- (3) 上記 _____ 部分を元にして、バッチファイル MK.BAT をカレントディレクトリに作ります。この場合 MK.BAT の内容は以下となります。

```
nmake /nologo -f test.mk STEP=0.01 UFUNC=un
```

内容はメッセージの _____ 部分そのままですので、コピー&ペーストを使うと便利です。

- (4) コンパイル作業は、MATLAB Command Window で !mk とします。

6.4 USERFUNC.C のデバッグ

一般にデバッグ作業は、弊社デバッガ (ADSP324 のみ)、MS C アダプタ等の開発ツールの使用か又は CCS 等の開発環境の利用が基本となります。これらのツールを使ってユーザ定義関数のデバッグが可能です。

6.4.1 ダミーmainによるデバッグ

SIMULINK ブロック線図ぐるみのデバッグはかなり大変で、お勧めできるものではありません。そこで、テスト用のダミーの SIMULINK ブロック線図に代わる部分を作成しておき、これを相手にしてユーザ関数部分のみをデバッグしておき、最後に SIMULINK に組み込んで実行する方法が可能です。その場合に使用するように、テストの為に main() 関数とダミーの SIMULINK ブロック線図処理関数のテンプレートが USERFUNC.C に組み込まれています。同ファイルの後半 #if TEST_MAIN 以下最後までがその部分に該当します。この部分はファイル先頭のタイトルコメント直後の、#define TEST_MAIN 0 により、通常は無効となっています。これを #define TEST_MAIN 1 と変更することにより、main() が有効となるようにしてご利用ください。

6.4.2 ブロック線図ぐるみのデバッグ

CCS のデバッグ環境をお持ちの場合にはブロック線図ぐるみでのデバッグが可能です。以下の手順で行ってください。

- (1) コード生成は通常通り Real-Time Workshop で行います。
- (2) 作成された実行ファイルの有るフォルダーに下記のファイルをコピーしてください。
%MATLAB_ROOT%\Rtw\c\Adsp32x\Rt\の全て (ADSP324-41C の場合)
%MATLAB_ROOT%\Rtw\c\Adsp32x\Rt\404\の全て (ADSP404-241C の場合)
%MATLAB_ROOT%\Rtw\c\Adsp32x\Rt\674\の全て (ADSP674-341C の場合)
更に、作成した USERFUNC.C も同様にコピーしてください。
- (3) CCS でプロジェクトを作成してください。プロジェクトに (2) でコピーしたソースを全て追加します。
- (4) RTMON を起動し実行ファイルをロードします。
- (5) CCS で実行ファイルを再度ロードします。
- (6) 必要に応じてブレイクポイントを設定した後 DSP プログラムを CCS から実行開始してください。
- (7) RTMON で実行開始します。これにより DSP のブロック線図の実行が開始し、上記 (6) で設定したブレイクポイントに到達すると CCS でのデバッグ操作が可能となります。

6.5 個別モデル専用のユーザ定義関数

ユーザ定義関数ソースファイル `userfunc.c` は、同じディレクトリ内にある全てのモデルで共用となり、このディレクトリ内で作成される全ての実時間モデルに、同じ関数がリンクされます。

もし、モデル毎に異なるユーザ定義関数を用意したい場合は、ソースファイルの名称を標準の `userfunc.c` でなく、`model.ufn` とします。（`model` はブロック線図の名称と同じ）ファイルの内容は同じで、単にファイル名称を変更するだけです。同じディレクトリに `userfunc.c` と `model.ufn` とが共に存在する場合は以下の優先順位となります。

＜リンクされるユーザ定義関数＞

順位	条件（ソースファイルの存在）	採用されるソースファイル
1	<code>model.ufn</code> と <code>userfunc.c</code> が共にある場合。	<code>model.ufn</code>
2	<code>model.ufn</code> が無く <code>userfunc.c</code> が有る場合	<code>userfunc.c</code>
3	<code>model.ufn</code> 、 <code>userfunc.c</code> 共に無い場合	無し

6.6 ML5.x と ML6.x の共用

ユーザ定義関数を、MATLAB の 5.x と 6.x の両方の版について、ソースファイルを共用させたい場合で、版の相違により、どうしても異なる記述が要求される場合があります。

この場合、DSP用開発環境で定義されている定数を使って条件付きコンパイルをすることができます。

参照する定数は ML51 です。

この定数は、MATLAB 6.x の場合には未定義、MATLAB 5.x の場合には 値が 1 と定義されています。従って下記の様な使い方ができます。

```
#if      ML51
        /* MATLAB 5.1 用コードを記述 */
#else   /* of ML51 */
        /* MATLAB 6.x 用コードを記述 */
#endif  /* of ML51 */
```

7 制限事項

7.1 動作上の機能制限

MATLAB 対応ライブラリは、I/Oブロックが、実時間実行ではないものの、SIMULINK 画面から利用できるのが、ひとつの特徴です (MATLAB 4.2c のみ)。
しかし、ユーザ関数組み込み機能については、SIMULINK 画面からは利用できません。
SIMULINK 画面でシミュレーションを実行した場合、弊社製品の有無にかかわらず、トップレベルのブロック線図の Inport については常に 0 が入力され、又、Outport については出力は無視されます。

ユーザ関数とのインターフェースを、SIMULINK 画面で確認するには、Inport の代わりにダミーのサブブロックを、Outport の代わりに Scope や To-Workspace 等を使い、模擬的に行なってください。

7.2 使用出来ないソースファイル名

ユーザ定義関数では、利用出来るソースファイルの名称に制限があります。本製品で使っているソースファイル名称と重複すると、不都合が生じますので注意してください。
本製品では、以下のソースファイルを使用しています。尚、版により若干変更がある場合がありますので、念の為、ディレクトリを直接参照し、確認する事をお勧めします。

● ディレクトリ : MATLAB\RTW\CYADSP32X\RT

● ヘッダファイル

ADSP32X.H
ADSPTYPE.H
CTRL.H
EXTRNFCN.H
HEAPEX.H
LOGOPT.H
LOGOPT0.H
MINIMON.H
RT5COMP.H
RT5CPU.H
RT5SIM.H
RT_ERROR.H
RT_LOG.H
RT_TRIG.H

● ソースファイル

ODE0.C
ODE1.C
ODE2.C
ODE3.C

ODE4.C
ODE5.C
RT5CPU.C
RT5EVENT.C
RT5GETT.C
RT5MAIN.C
RT5ODE.C
RT5SIM.C
RT5TIMER.C
RT_ACCES.C
RT_BKGND.C
RT_LOG.C
RT_MON.C
RT_TRIG.C
USERFUNC.C
及び *MODEL*.UFN

7.3 長時間に渡るDSPの専有

バックグラウンド処理はアイドルループを回りながら、RTMONからのアクセスに応答する傍ら実行されます。RTMONからのアクセスとは、例えばゲイン値変更とか、変数表示等をさしています。

従って、極端な例では、ユーザ定義のバックグラウンド処理で無限ループを構成すると、RTMONからのアクセスに応答が出来ず、RTMONからはDSPが止まったか、に見えてしまいます。

無限ループを構成しないまでも、出来るだけ短い時間で、一旦関数から戻る様な記述をしてください。

例として、ユーザ定義関数のバックグラウンド処理で、ある条件が整うのを待機する様な処理を記述したい場合、以下の様に、条件が整っていなければ一旦関数から戻り、次回呼び出された時再度継続する様な記述としてください。

バックグラウンド処理記述の例

```
static int  flg=0;
void UserBackground()
{
    switch( flg ){
        case 0:  if( ! ConditionOk() )    /* 条件が整わなければ */
                                                         /* 一旦戻る */
                                                         /* 整ったらフラグをセットし */
                                                         /* 一旦戻る */
                    return;
                    else    flg = 1;
                    return;
        case 1:  NextProc();               /* 条件が整った後の処理 */
                    flg = 2;               /* フラグセットし */
                    return;               /* 戻る */
        default: return;                  /* 以後何もしない */
    }
}
```

この例の想定は、関数 ConditionOK() は条件が整っているか確認する関数で、条件は長時間の渡り整わない可能性があるものとします。この関数は、条件が整ったら TRUE を、それ以外は FALSE を返すものとします。

7.4 長時間に渡る割り込み禁止

R Tモデルでのブロック線図の計算は、タイマー（又は外部からの）割り込みにて計算を駆動しています。従って、ST レジスタの GIE の操作により、割り込みを禁止するとブロック線図の計算が停止してしまいます。

出来れば、IE レジスタの停止したい割り込みに相当するビットの操作にて割り込みを抑止してください。

どうしても ST レジスタの GIE の操作により割り込みを禁止する必要がある場合は、出来るだけ短時間に抑えてください。禁止して良い時間の目安は、ステップサイズ実行時間（ブロック線図計算に要する時間）以内です。

ユーザ定義関数
(ユーザ記述による C 言語関数)

第 1 版 2001.03.20

第 2 版 2001.06.15

中部電機株式会社 C S 事業部

TEL 0532-61-9566

FAX 0532-63-1081

MATLAB対応ライブラリ（マルチ）
ADSP324-41C／ADSP404-241C／ADSP674-341C
リアルタイムモニター・チュートリアル

中部電機株式会社

<< 目 次 >>

1	概要.....	1
2	用語・略語の説明	1
3	モデルの作成	2
3.1	ユーザプログラミングによるモデル	2
3.2	S I M U L I N Kによるモデル	2
3.2.1	作業用フォルダーの作成.....	2
3.2.2	カレントディレクトリの設定	2
3.2.3	モデルの作成.....	4
3.2.4	モデルの評価.....	10
3.2.5	コード化	12
4	R T M O Nの起動	15
4.1	R T M O Nの起動	15
5	モデルペアーの作成.....	16
5.1	モデルペアーの概念.....	16
5.2	モデルペアーの新規作成	16
5.3	モデルペアーの編集.....	21
6	モデルのロード.....	22
6.1	シングルD S Pモデルのロード	22
6.2	マルチD S Pモデルのロード	23
7	実行パラメータ（R Tオプション）変更	24
8	実行開始.....	27
9	変数モニター	28
9.1	変数の選択	28
9.2	ベクトル変数の選択.....	29
9.3	表示点数の設定	30
9.4	スコープの自動配置.....	31
9.5	変数の数値読み取り	32
9.6	スコープのレンジ変更	32

9.7	Y軸オフセットを付ける	33
9.8	Y軸オフセットの除去	34
9.9	変数の削除	34
9.10	スコープ状態の保存・選択	34
10	計算所要時間の測定	35
11	データロギング	37
11.1	ロギング項目	37
11.2	ロギング項目選択	38
11.3	データ間引き	38
11.4	データ点数	39
11.5	ファイル形式・ファイル名	39
11.6	マニュアルトリガー	39
12	モデルの終了	40
13	ロギングデータの保存	41
13.1	シングルモデルの場合	41
13.1.1	上書き保存	41
13.1.2	新規保存	41
13.2	マルチモデルの場合	41
14	ロギングデータのMLでの利用	43
15	自動テスト用ブック作成	44
15.1	基本ブック指定	44
15.2	作成ブック指定	45

1 概要

本書は、MATLAB、SIMULINK（共に米国 The Math Works Inc 社製品）及びMATLAB対応ライブラリマルチ「ADSP324-41C」「ADSP404-241C」「ADSP674-341C」（中部電機（株）製品）を用いて、実時間アプリケーションを作成する具体的手順を、実例を元に説明しています。

本書で説明している各製品のバージョンは以下の通りです。

MATLAB等	6.0、6.1 版
MATLAB対応ライブラリ（マルチ）	4.03.02 版
DSP用コンパイラ（C3x）	5.0、5.1 版
DSP用コンパイラ（C6x）	1.1 版

注）マニュアル記載の MATLAB 図は、Ver6.0 を使用しています。

2 用語・略語の説明

本書では説明の簡略化の為、次の用語（略語）を使用する場合があります。

ディレクトリ	フォルダーと同じ意味
DLG	ダイアログボックス
ML	MATLAB
ML-LIB	MATLAB 対応ライブラリ
RTW	Real-Time Workshop
RTMON	リアルタイムモニター（本製品に含まれる）
RTモデル	リアルタイムモデル
SL	SIMULINK
TI	Texas Instruments（DSP チップ製造元）
TMW	The Math Works Inc.（MATLAB 製造元）

3 モデルの作成

本製品で取り扱える実時間モデルの作成には、2つの方法があります。

3.1 ユーザプログラミングによるモデル

1 つめは、汎用エディターでC言語又はアセンブラを用いてプログラミングし、弊社ソースレベルデバッガ／MSCアダプタ等の製品又は、T I 社製エミュレータの何れかを用いてデバッグを行い、実行可能形式にしたものです。本製品では、この形式の実時間モデルについては、ロード・実行はできますが、メモリー内部の値や計算結果のモニター等を行う事は出来ません。デバッガ等を用いて完全にデバッグを行ったうえ、ロードすればただちに実行できる形式にしてください。

尚、本書ではこの方法については説明を行っていません。

3.2 S I M U L I N Kによるモデル

2 つめは、S Lにより作成したブロック線図を元にR T Wを用いてC言語ソースを経て実行形式プログラムを自動生成する方法です。以下本書ではこれをR Tモデルと呼びます。

例として、D S Pで正弦波を生成し、これをD A変換器に出力すると共に、A D変換器からデータを取り込むモデルを作ってみます。

3.2.1 作業用フォルダーの作成

MLにはカレントディレクトリという概念があり、作成されたブロック線図や、R T Wで作成するR Tモデルは全て、このカレントディレクトリに保存されます。この作業用フォルダは、エクスプローラ等のファイル管理ツールで作成します。

3.2.2 カレントディレクトリの設定

次に、MLを起動します。

ML起動後、最初に行う作業は、カレントディレクトリを、希望とするフォルダーに移動する事です。

MLインストール後、通常に起動すると、カレントディレクトリは C:\¥MatlabR12¥Work になっています。MATLAB Command Window に c d と入力すると現在のカレントディレクトリが表示されます。

<例>

》 cd

C:\¥MatlabR12¥Work

これを例えば、D:\¥USER¥SomeOne¥MatlabWork に移動するには、

》 cd D:\¥USER¥SomeOne¥MatlabWork

と入力します。

もし、毎回入力せず、自動的にカレントディレクトリを移動させたいのであれば、例えばデスクトップにショートカットを作成し、このプロパティのショートカットの作業フォルダー(S)にカレントディレクトリを入力しておきます。



<ショートカットの設定例>

ML起動時のカレントディレクトリを自動的に移動する方法は他にもありますので、どの方法を用いても構いませんが、要点は、保存したブロック線図やデータファイル等を見失わないで済む為、これからMLやSLで作業をしようとする際には、必ず「カレントディレクトリが何処になっているか？」を確認する事です。

3.2.3 モデルの作成

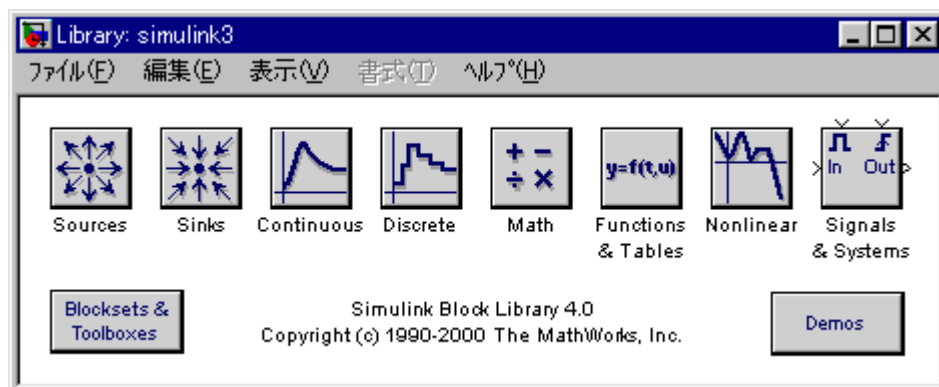
新規モデルの作成は、MATLAB のメニューから、ファイル(F) → 新規作成(N) → モデル (M) を選択します。作成されたブロック線図は内容が空で MATLAB により暫定的な名前 (例えば Untitled) が付けられています。名称を決定するには、ブロック線図メニューから、File(F) ~ Save As(A)で名称を付けて (例えば test 等) 保存します。拡張子は省略すれば、自動的に .mdl が追加されます。この時、フォルダーはMLカレントディレクトリになっていますので、変更しないようにします。

ファイル名称は、R T Wで使用する場合、制約がありますので次の制約を守って命名してください。

- (1) 数字で始まってはいけません。__ 又は アルファベットで始めます。
(例: _model.mdl、model.mdl=OK、1model.mdl=NG)
- (2) かな、漢字は使用してはいけません。
(例: abc_123.mdl=OK、ｶﾅ.mdl、漢字.mdl=NG)
- (3) __以外の記号は使ってはいけません。
(例: abc_123.mdl=OK、abc\$.mdl、abc%&.mdl=NG)
- (4) 大文字小文字は、同じと見なされます。

こうして保存したブロック線図は、次回ML起動時に、MATLAB Command Windw にブロック線図の名称を入力する事により、開く事ができます。

続いて作成したブロック線図に適宜ブロックを貼り付け、線で繋いでシステムを構築していきます。SLの標準ブロックは、MATLAB Launch Pad (MATLAB 5.x と同様の方法) による利用方法と、MATLAB4.x と同様の方法の両方が利用可能です。後者は MATLAB Command window に simulink と入力して Simulink ライブラリブラウザを開き、Simulink を選択してマウス右クリックから Simulink ライブラリを開くを選択します。次の要な、SLの標準ブロックライブラリが開きます。



< S L 標準ブロックライブラリ >

更に、**Sources** 等のブロックアイコンをダブルクリックすると個々のブロックを収録したライブラリブロック線図が開きます。

LaunchPad からの利用と、ブロック線図形式の表示からの利用は、本質的には同じです。

尚、本書の説明では後者の方法でブロック線図を作成する事とします。

次に、ML-LIBのライブラリを開く為、MATLAB Command Window に対して、コマンドを入力します。

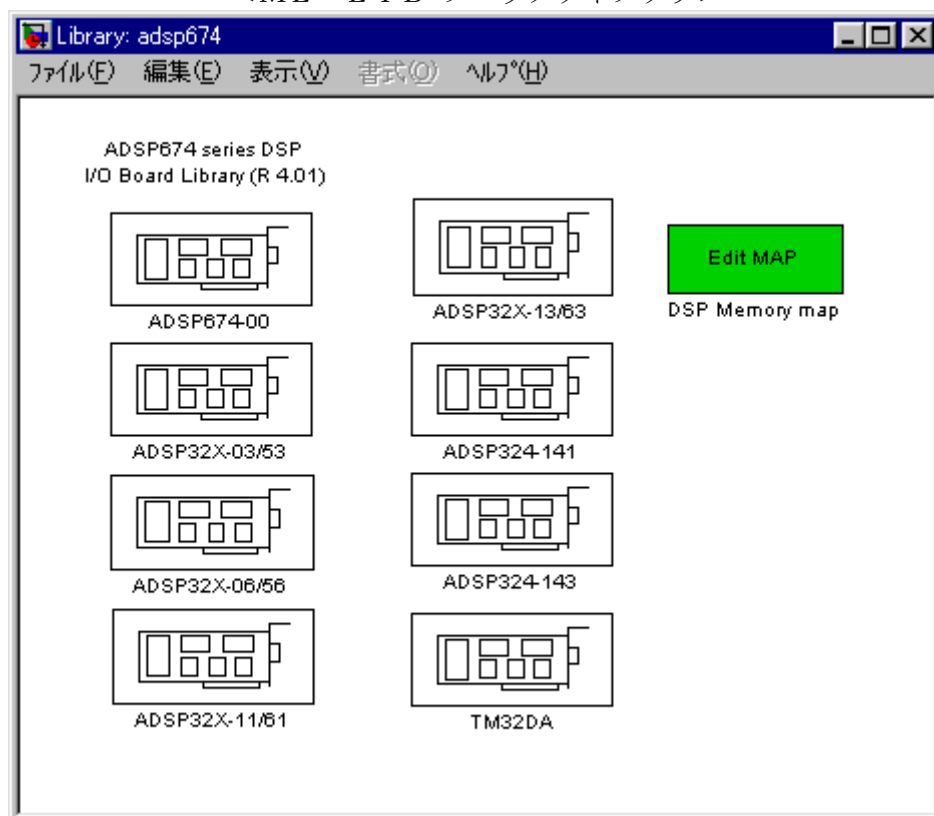
ADPS324 の場合 adsp32x

ADSP404 の場合 adsp404

ADSP674 の場合 adsp674

これにより、次の様なライブラリウィンドウが開きます。（下の例は adsp674 の場合です）

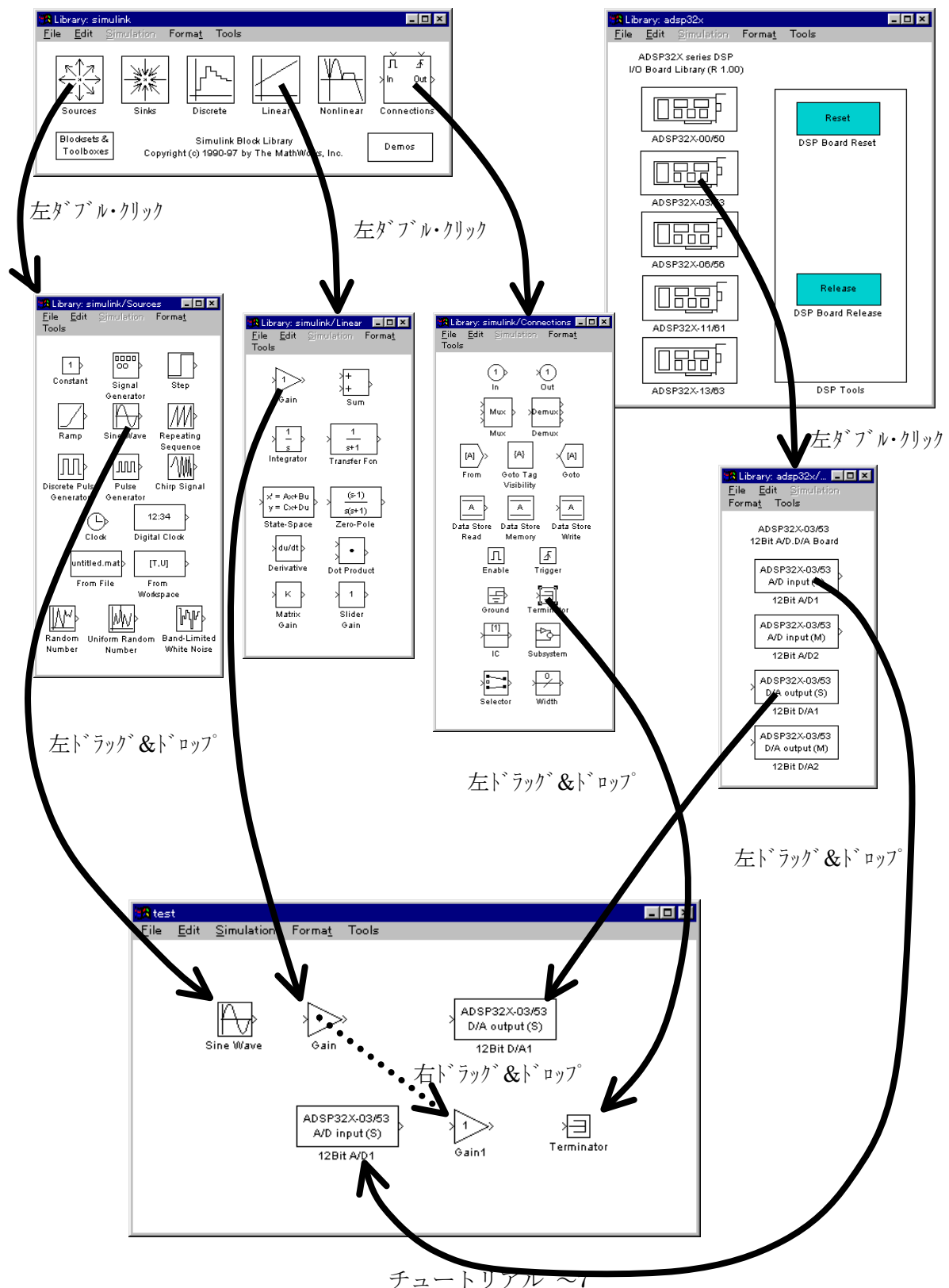
<ML-LIB ブロックライブラリ>



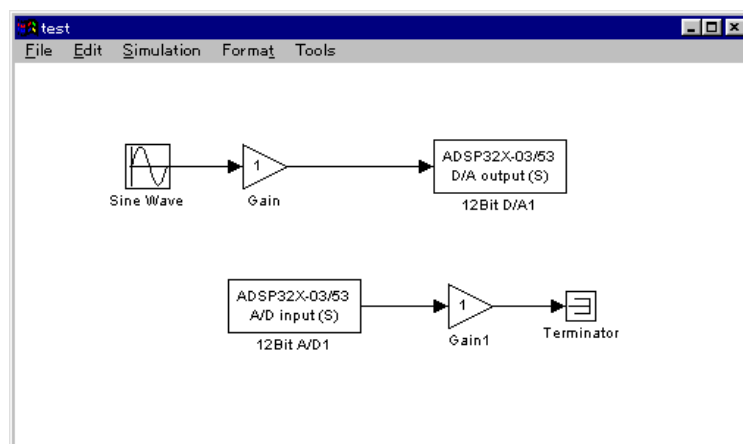
開いたライブラリにはライブラリをボード毎に分類して収録しれたブロックアイコンがありますので、これをダブルクリックし、開いたブロックライブラリの中から必要なライブラリをドラッグし、ブロック線図にドロップします。

ここでは例として、正弦波を DA 変換機に出力すると共に、AD 変換器からデータを入力するブロック線図を作成してみます。

正弦波は、S Lブロックライブラリの中の Sources に入っていますので、まず Sources をマウス左ボタンでダブルクリックし、開いたライブラリの中から Sine Wave をマウス左ボタンでドラッグし、ブロック線図にドロップします。以下同様に次ページに示すブロックをドラッグ&ドロップしてください。



次に、ブロック線図に貼り付けたブロックの出力から次のブロックへの入力へと、マウス右ボタンでドラッグし結線します。

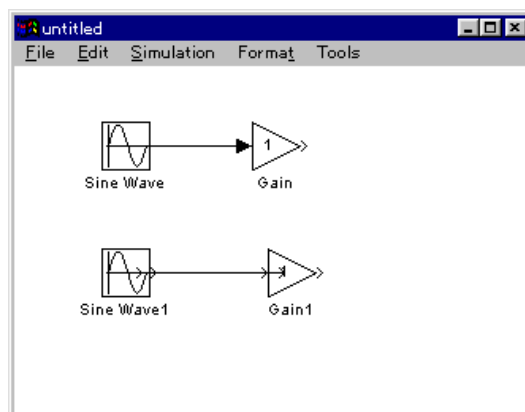


<結線が終わったブロック線図の例>

結線が正しく接続出来ている線は $\longrightarrow$ の形になっています。

次の例は、上側のブロック（Sine Wave と Gain の間）は入力出力共に正しく結線出来ていますが、下側（Sine Wave1 と Gain1 との間）は入力出力共に正しく結線出来ていません。ブロックと線の端が重なっているのに、一見つながっているかに見えますが、線の形でつながっていない事が判ります。

<結線不良（下側）の例>



この場合は線を一旦削除して引き直すか、又はブロックアイコンをマウス左でドラッグして一旦位置を移動し、正しく結線出来た後元の位置に戻します。

結線が出来たら、パラメータを設定します。ここでは正弦波の周波数と A/D ブロックのサンプルタイムを変更します。

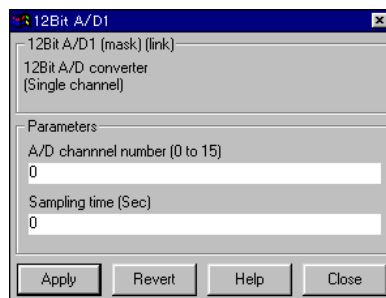
Sine Wave ブロックアイコンをダブルクリックすると下の初期設定ダイアログが開きますので、周波数〔単位はラジアン／秒〕を設定します。例えば 10 Hz であれば、 $\pi * 2 * 10$ を設定し、Close ボタンを押します。



<Sine Wave 初期設定 D L G の例>

A/D変換器の初期設定ダイアログも同様に開き、下記のようにサンプルタイムを0に設定します。サンプルタイム0は連続時間系を意味し、ステップサイズ毎にADからデータを取る事を意味します。

< A D 変換器の初期設定 >



尚、ADとDAはこの例では共にCH0を使います。これ以外のCHの場合や、複数のCHを使う（複数のブロックを使う）場合は適宜CH番号の設定が必要です。

設定が出来たら、ブロック線図のメニューから、File ～ Save でブロック線図を保存します。

3.2.4 モデルの評価

モデルが希望通り動作するか否かは、S L上で評価出来る場合と出来ない場合があります。例えば外部機器（制御対象等）と接続しないといけない等、S Lだけでは評価出来ないのが一般的ですが、出来れば擬似データ等を用いて予め、ある程度の目処を付けておく事をお勧めします。

その際に注意が必要なのは、評価の最終段階では、積分アルゴリズムを「固定ステップサイズ」にしても大丈夫か確認する事です。

一般に数値シミュレーションでは、計算精度と計算時間の短縮の為、可変時間刻みのアルゴリズムが良く用いられます。これは、状態変数の導関数を計算し、傾きが急な場合計算精度を上げるため時間刻みを短くし、逆に傾きが緩やかな場合は時間刻みを長くしても計算精度は低下しないので、計算回数を減らして計算に費やす時間を節約する（計算回数を減らす事は同時に誤差の累積を減らす意味をも含む）為、時間刻みを長くします。S Lでもこれら、可変時間刻みのアルゴリズムが多数組み込まれています。

しかし、最終目的が「実時間モデルを作る事」の場合、実時間モデルでは可変時間刻みが実現出来ない為、S Lで評価の段階で、固定時間刻みで評価をしておく必要があります。固定時間刻みの場合、必然的に時間刻みの幅（ステップサイズ）により計算結果が違って来る場合がありますので、どの程度のステップサイズで計算すれば良いかも確認しておく必要があります。

固定時間刻みにするには、ブロック線図のメニューから Simulation ~ Parameters..を選択し次の Simulation parameters D L Gを開きます。この D L Gの Solver タブを選択し、Solver options グループの Type コンボボックスから Fixed- step を選択します。続いてすぐ下の Fixed step size:に希望するステップサイズを入力します。auto のままでは R T Wでコード化出来ません。最終的には必ず、ある具体的なステップサイズ値を入力してください。

又、モデルに連続系の状態変数が無いのに、discrete 以外の積分アルゴリズムを選択した場合は、警告が出ます。コード生成の時も同様に警告されますので、この時点で discrete に変更してください。

次の例は、固定時間刻みで積分アルゴリズム無し（今回使う例 test.mdl では積分アルゴリズムが不要）に設定した例です。



シミュレーションパラメータ設定例＞

- ①ここで固定／可変時間刻みを選択します。**Fixed-step** が固定時間刻みです。
 - ②ステップサイズを設定します。単位は秒です。具体的な数値を入力してください。
 - ③積分アルゴリズムを選択します。
 - ④シングルタスクを選択します。
- まず最初に、①で固定／可変を選択します。これによりダイアログの表示項目が変わりますので、続いて②～④を設定します。

3.2.5 コード化

評価が終わったモデルは次の手順でコード化します。

ブロック線図のメニューから、Tools ～ RTW Options.. を選択し、Simulation parametersDLGを開きます。

Solver タブを選択し、Solver options グループの Type が Fixed-step である事、その右の積分アルゴリズムが希望するアルゴリズムである事、その下の Fixed step size:も同様に希望するステップサイズ値になっている事を確認します。

この値は、DSPで実行する際の設定とは直接には関係しません（RTMONではブロック線図の設定とは別に、自由に設定できます）が、RTMONでRTモデルをロードした際にデフォルト設定として呼び出す事ができます。

次に RTW タブを選択し、以下を設定してください。

①システムターゲットファイル: chubu.tlc

②テンプレート Make ファイル:

ADSP324 の場合 a32x.tmf

ADSP404 の場合 a404.tmf

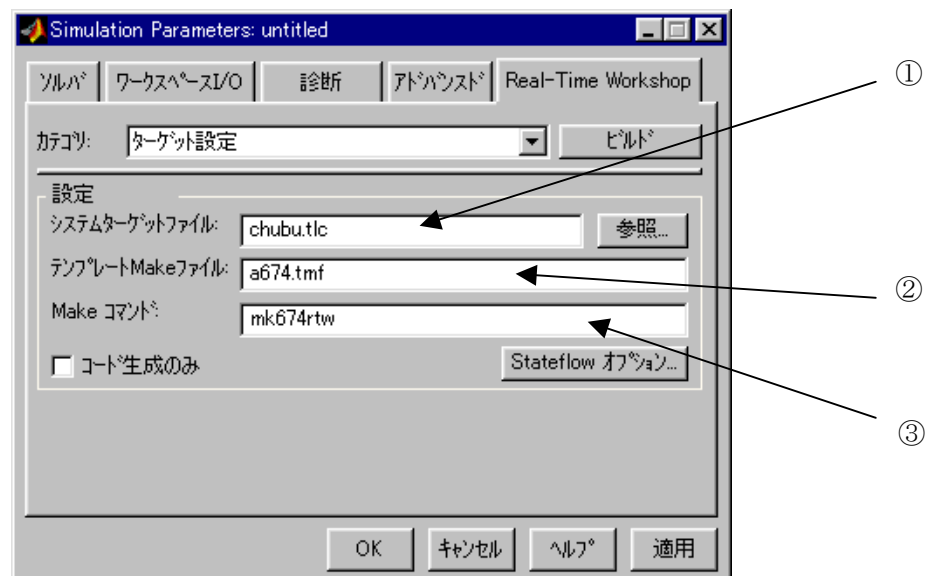
ADSP674 の場合 a674.tmf (メモリモデル Large 時 a674b.tmf)

③Make コマンド:

ADSP324 の場合 mk32xrtw

ADSP404 の場合 mk404rtw

ADSP674 の場合 mk674rtw (メモリモデル Large 時 mk674rtwb)



< RTW設定DLG設定例 >

この状態で一旦ブロック線図を保存します。そうすれば、万が一PCがハングした場合でも、行った設定はディスクに残っており、次回ML起動時にブロック線図を開けば設定が復元されます。

☆ADSP674 の場合メモリモデル **Large** とは、“EditMap”ブロックにより各セクションを異なる領域に配置した場合、ポインタが 32 bit 表現必要時に選択します。**Large** モデルに設定しない場合、コンパイル時にエラーとなります。又独自に作成したライブラリをリンク使用時はライブラリも **Large** モデルで作成してください。(genlibb)

注) 新規以外で作成されたモデルをコード化する場合、弊社ホームページの Q&A「MATLAB 対応ライブラリでよくある Q & A」を参照してください。

コード生成の実施は、このDLGのRTWタブを選択し、Build ボタン（図の⑥）を押します。

MATLAB command window に以下の表示がなされ、コード生成が完了します。

本製品の版番号や、生成するRTモデルの実行形式、及び、初めてコード生成するか2度め以後かにより、表示される行数や内容が異なります。

良否の判断は、全ての行が#### で始まっており（コード生成時のエラーメッセージは#### で始まらない）、かつ、最後の2行が以下と同様（但し test の部分は個々のブロック線図の名称）であれば、良好と判断してください。

```
#### Starting RTW build procedure for model: test
#### Invoking Target Language Compiler on test.rtw
#### Loading TMW TLC function libraries
#### Initial pass through model to cache user defined code
#### Creating source file test.c
#### Creating part 1 of registration file test.reg
#### Creating parameter file test.prm
#### Creating model header file test.h
#### Creating part 2 of registration file test.reg
#### TLC code generation complete.
#### test.mk which is generated from D:\MATLAB\511\rtw\c\adsp32x\32x.tmf is up to date
#### Building test: nmake /nologo -f test.mk STEP=0.001
#### Compiling test.c
#### Compiling rt5sim.c
#### Compiling rt5main.c
#### Compiling rt_log.c
#### Compiling rt_trig.c
#### Compiling rt_acces.c
#### Compiling rt_mon.c
#### Compiling rt_bkgnd.c
#### Compiling rt5gett.c
#### Compiling rt5cpu.c
#### Compiling rt5timer.c
#### Compiling rt5ode.c
#### Compiling ode0.c
#### Compiling ode1.c
#### Compiling ode2.c
#### Compiling ode3.c
#### Compiling ode4.c
#### Compiling ode5.c
#### Linking test.out
#### Created
#### Successful completion of RTW build procedure for model: test
```

生成されたRTモデルの実行ファイルは拡張子.OUT（例：*model.out*）で、MLのカレントディレクトリのすぐ下にブロック線図の名称に因んだフォルダーが自動的に作成され、その中に格納されます。例えばカレントディレクトリが C:\MatlabR12\Work で、ブロック線図が Test の場合は

C:\¥MatlabR12¥Work¥test_grt_rtw¥

DSP 用実行形式ファイル test.out として作成されます。

4 RTMONの起動

4.1 RTMONの起動

RTMONは、Windowsのスタートメニューから スタート ～ プログラム(P) ～ リアルタイムモニタ ～ リアルタイムモニター (マルチ) を選択して起動します。

5 モデルペアーの作成

5.1 モデルペアーの概念

これからロードしようとするR TモデルがマルチD S Pの場合、各D S PにR Tモデルを一括してロード出来る様にする為に必要な情報である、何台のD S Pを使うか？、どのR Tモデルを何番のD S Pにロードするか？等を定義し、これらの情報をR Tモデルとは別に一括して保存・管理します。これを「モデルペアー」と呼びます。D S Pを1台しか使わない場合は必要ありません。

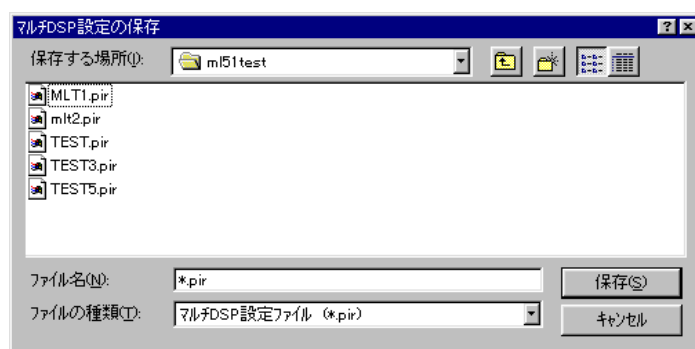
マルチD S Pの場合、各D S Pにはそれぞれ1つずつR Tモデルを割り当てます。R Tモデルはブロック線図によるものでも、C言語やアセンブラで記述したものでも構いません。又、同じR Tモデルを複数のD S Pに割り当てても構いません。

5.2 モデルペアーの新規作成

ここでは、0、1、2番の計3台のD S Pを使い、各D S Pにそれぞれ、dsp0.out、dsp1.out、dsp2.out をロードする場合を想定します。

まず、R TMONのメニューから、 ファイル(F) ～ 新規マルチ DSP 設定(N)を選択し、マルチD S P設定の保存D L Gを開きます。既存のファイル名称と重複しないよう気を付けてファイル名称を決定し、保存ボタンを押してください。

<マルチD S P設定の保存D L G>



こうして保存されたモデルペアーは拡張子 .PIR ファイルとして保存されます。

ファイル名称を決定しますと、マルチDSP設定DLGが開きます。

<マルチDSP設定DLG>



開いた直後のDLGは全て空になっています。DSPボード番号の表示が0になっており0番のDSPへロードするモデルが未だ設定されていない事を示しています。

最初に0番DSPの設定をします。メインモジュールの右の参照ボタン（2個あるうちの上側）を押し、実行ファイル選択DLGを開きます。このときDLG左下にある「相対パス指定」にチェックを入れておく事により、モデルペアーファイルを基準としたRTモデルの相対位置関係を崩さない限り他のディレクトリに移動させてもモデルのロードが可能になります。



<実行ファイル選択DLG>

ロードしたいRTモデル、この場合 dsp0.out を選択します（上の図）。RTモデルはMLのカレントディレクトリに作成されていますので、このDLGでは“ファイルの場所”を使ってフォルダーを間違えない様にしてください。MLとRTMONは直結していませんので、MLでカレントディレクトリをセットしてあっても、その設定はRTMONには直接反映していません。

選択したら、開く(O)ボタンを押します。すると、マルチDSP設定DLGに今選択したRTモデルのディレクトリ名とファイル名の拡張子を除いた部分が表示されます。(下図)
これで0番DSPへロードするRTモデルの設定は完了です。補助モジュールの欄は、通常は空白のままで構いません。

<マルチDSP設定DLG>



次に、1番DSPにロードするRTモデルを設定する為、DSPボード番号を切り替えます。右上のスライダー①をマウス左ボタンで少し右にドラッグするか、又はスライダーの右端にあるボタン②を1回、マウス左ボタンで押します。DLGの表示は次の様になります。



ボード番号の表示が1に変わり、1番DSPの設定が未だ完了していない事を表しています。先の0番DSPの場合同様に参照(R)ボタンによりRTモデルを選択します。

同様に0～2番DSPまで全て設定したら、一覧（L）ボタンを押してRTモデルの一覧を表示し、希望通り設定されているか確認してください。

<モジュール一覧表示DLG>

番号	モジュール名
0	M dsp0
1	S dsp1
2	S dsp2
3	S -----
4	S -----
5	S -----
6	S -----
7	S -----
8	S -----
9	S -----
10	S -----
11	S -----
12	S -----
13	S -----
14	S -----
15	S -----

もしロードの不要なDSP番号に誤って設定してしまったら、スライダーでその番号のDSPを選択し、削除ボタンで削除します。

次に、モデルをロードする順番を指定する為、順番（O）ボタンを押します。

<マルチDSPロード順番DLG>

(未設定)

(設定完了)

未設定の状態では上図左の様に0番DSPしか設定されていません。

DSP番号で、0→1→2の順にロードしたい場合は、Clear ボタン→0 ボタン→1 ボタン→2 ボタンの順に押します。全てのRTモデルがブロック線図による通常のRTMON

用モデルの場合は、若い番号順で良いでしょう。

番号を途中で間違えたら 0 番から正しい順で再度押して下さい。

前頁右図、設定完了の様に正しく設定出来たら、マルチDSPロード順番DLGのOKボタンを押します。

5.3 モデルペ어의編集

既に作成してあるモデルペ어의ファイルの内容を変更する場合は、RTMONのメニューから ファイル(F) ～ 既存マルチDSP編集(E) を選択します。

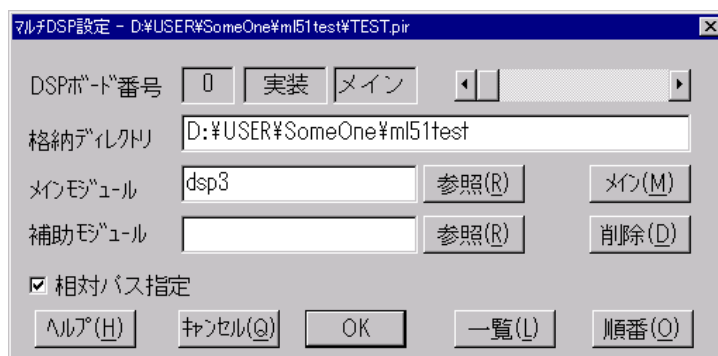
マルチDSP設定の編集DLGが開きますので、ここで編集したいモデルペアのファイルを選択します。



<マルチDSP設定の編集DLG>

モデルペアのファイルを選択すると、マルチDSP設定DLGが開き、前回保存した際の設定が表示されます。必要に応じて変更を加え、OKボタンで保存してください。

<マルチDSP設定DLG>



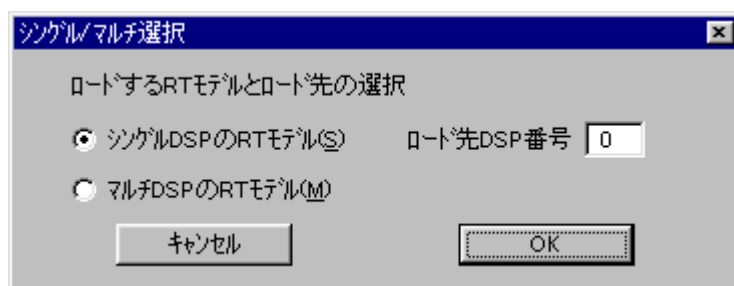
6 モデルのロード

6.1 シングルDSPモデルのロード

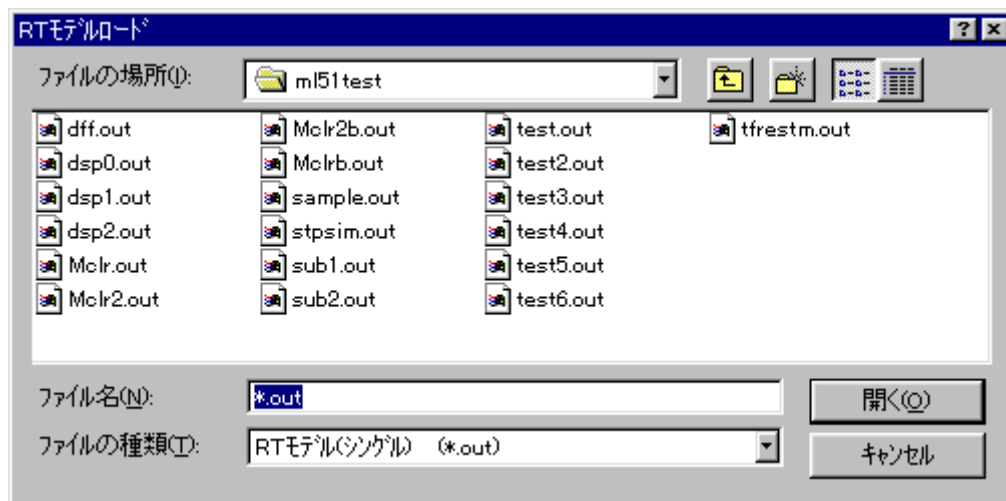
RTモデルのロードはRTMONのメニューから、ファイル(F) ～ RTモデルロード(O) を選択します。

シングル/マルチ選択DLGが開きますので、シングル DSP の R モデルを選択し、更にロードしたい DSP ボード番号を設定（エディットボックスをマウス左ボタンでクリックし、キーボードで DEL キー、BS キー、数字キー等を使ってボード番号を直接設定）してください。（下図）続いてOKボタンを押してください。

シングル/マルチ選択DLGはロードの度に開きますが、設定は前回の設定が再現されます。変更が無ければOKボタンだけで結構です。



RTモデルロードDLGが開きますので、ロードしたいRTモデルのファイル名を選択し、開くボタンを押してください。



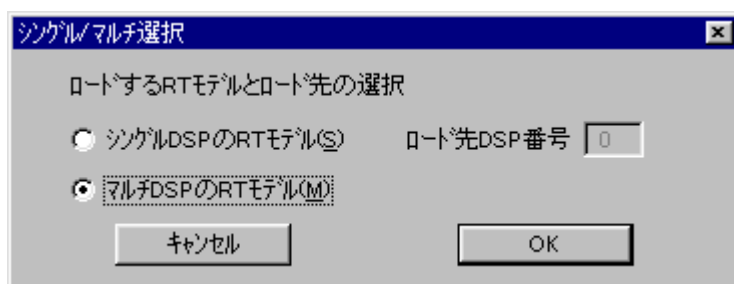
これによりRTモデルがロードされ、モデルウィンドウが開きます。

6.2 マルチDSPモデルのロード

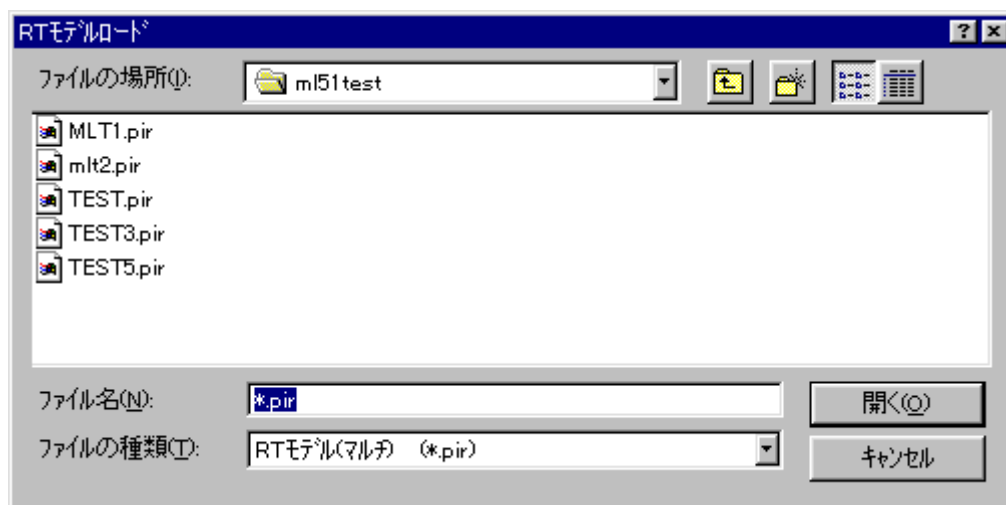
RTモデルのロードはRTMONのメニューから、ファイル(F) ～ RT モデルロード(O) を選択します。

シングル/マルチ選択DLGが開きますので、マルチDSPのRモデルを選択します。（下図）続いてOKボタンを押してください。

シングル/マルチ選択DLGはロードの度に開きますが、設定は前回の設定が再現されます。変更が無ければOKボタンだけで結構です。



RTモデルロードDLGが開きますので、ロードしたいモデルペアーのファイル名を選択し、開くボタンを押してください。

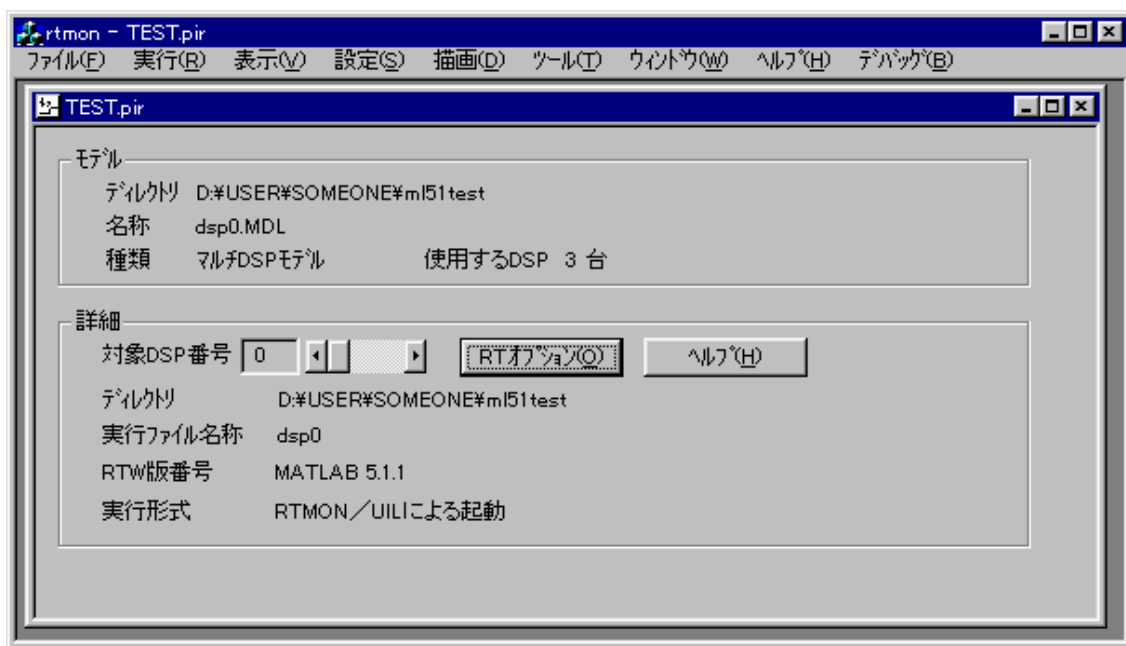


これによりRTモデルがロードされ、モデルウィンドウが開きます。

7 実行パラメータ（RTオプション）変更

積分アルゴリズム、ステップサイズ等、RTモデル計算の実行パラメータは、実行に先立ちRTMONから決定します。RTMONでの設定はブロック線図の設定を反映させる事もできますが、それとは別にRTMON上で一時的に異なるパラメータで実行する事もできます。それには、元のブロック線図を変更する必要はありません。

変更は、モデルウィンドウ（下図）にあるRTオプション(O)ボタンでリアルタイムオプションDLGを開きます。



注：この図のアプリケーションの中のチャイルドウィンドウがモデルウィンドウと呼ばれるウィンドウです。この図では、モデルウィンドウを見易くする為、それ以外の表示を隠蔽したうえ、大きさを調整してありますので、通常RTモデルをロードした直後とは表示が異なります。

シングルDSPモデルの場合は使っているボード番号がモデルウィンドウに表示されていますので、そのままRTオプション(O)ボタンを押します。

マルチDSPモデルの場合、設定は個々のRTモデル毎にありますので、モデルウィンドウで対象DSP番号を選択してうえ、RTオプション(O)ボタンを押します。対象DSP番号はその右にあるスライダーで選択します。

1個のRTモデルには1個のRTオプションしか存在しませんので、シングルDSPモデルとマルチDSPモデルと同じRTモデルを使う場合、例えばシングルDSPモデルとしてロードした際の設定を変更すると、マルチDSPモデルの同じRTモデルに、その変更が反映されます。

リアルタイムオプションの設定画面です。積分オプションを初めモデル実行に際してのオプションを設定します。

アルゴリズムと、ステップサイズは、その右にある RTW 設定ボタンを押すと RTW でコード生成した時点のブロック線図の設定が呼び出されます。呼び出されるのは、RTW でコード生成した時点の設定である事に注意してください。ブロック線図の **Simulation parameters** の設定を変更しても、RTW でコード生成しないとその変更内容は RT モデルには反映されません。

ブロック線図の設定と異なる設定で動作させたい場合、アルゴリズムはコンボボックスのリストの中から使いたいアルゴリズムを選択します。ステップサイズはその時間値 [単位：秒] を直接入力します。指数形式 (1.5e-003) または、実数形式 (0.0015) いずれでも構いません。

開始時刻は、通常は 0 のままとしてください。

終了時刻は、一定時間 RT モデルを動作させたら自動的に停止させたい場合、その時刻値 [単位：秒] を、それ以外は十分に大きな値を設定します。

時間倍率は通常は 1 としてください。

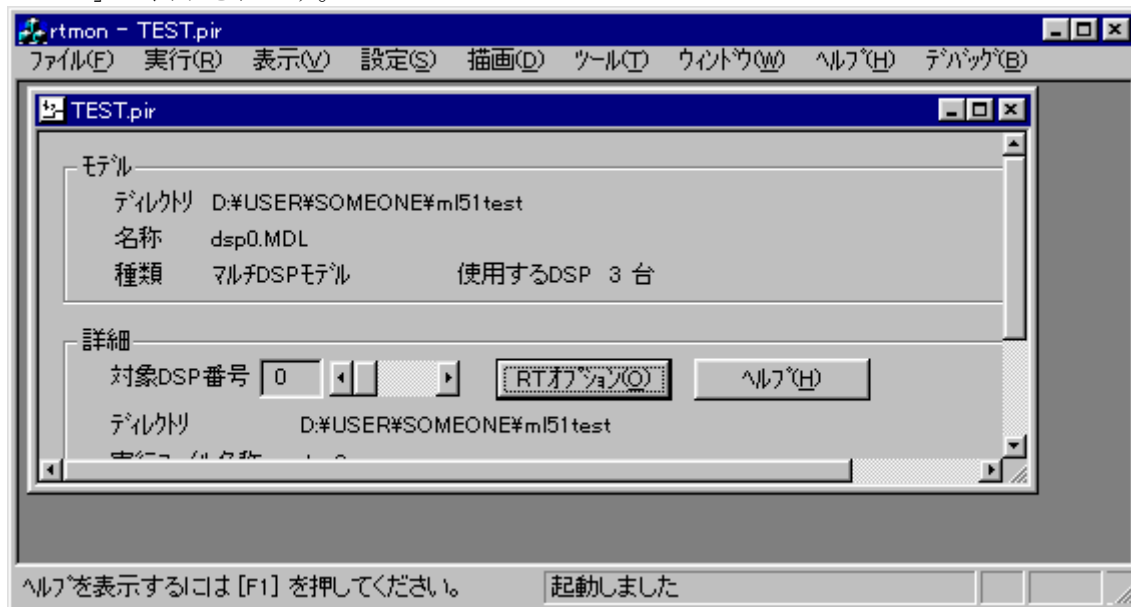
また、モデルにて **frmxls** ブロックを使用した場合必ず“**Frmxls ブロック使用**”へチェックし **Excel** ブックの指定をしてください。ここで指定すると実行前に **Excel** より **DSP** へデータ転送が実行されます。ブロックを使用していてこの指定をしない場合モデルの動作は保証できません。

設定が完了したら、OKボタンを押してください。変更内容はディスクファイルに保存され次回ロード時には同じ設定が自動的に再現されます。尚、ここで設定した値は **DSP** 上での **R T** モデルの実行に採用されますが、ブロック線図の **Simulation parameters** には反映されません。

8 実行開始

実行開始は、メニューから 実行 (R) ～ 実行開始(B) (又はツールバーの実行開始ボタン)で行います。

正常に実行が開始された場合は、RTMONのステータスバー (場面最下段) に「起動しました」と表示されます。



9 変数モニター

変数とは、ブロック線図に配置されたブロックから出力された信号を意味します。ブロック線図の表示でいうとブロックとブロックの間の結線に相当します。DSPのRTモデルではDSP内部のメモリーに割り当てられ、これを表示／ロギングする事ができます。変数にはそれを出力するブロックの名称が割り当てられます。

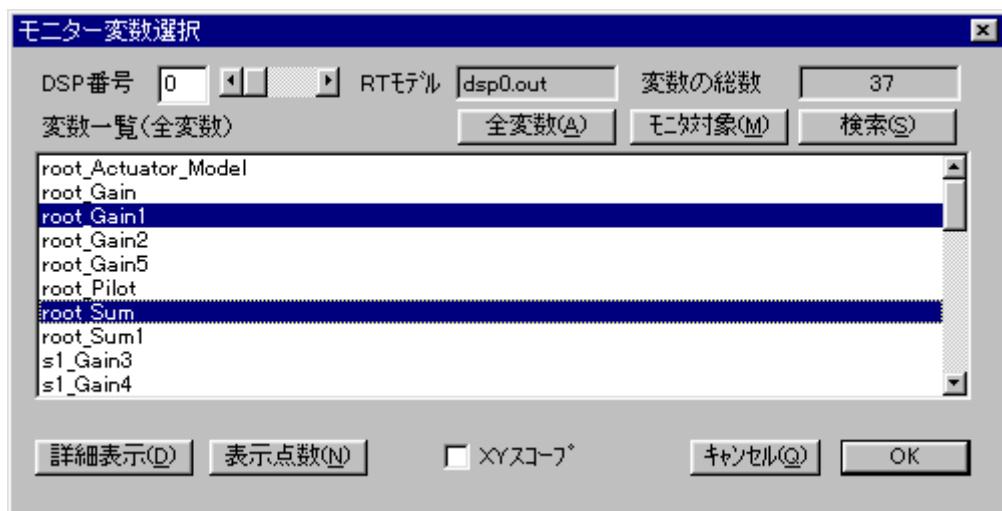
9.1 変数の選択

変数モニターはRTMONのメニューから、表示(V) ～ 変数グラフ表示(V) でモニター変数選択DLG（下図）を開き、ここでモニターしたい変数を選択します。



モニター変数選択DLGには、ロードされたRTモデルにある参照可能な変数の一覧が表示されます。例えばブロック線図のトップレベルに Gain ブロックが有る場合、この名称を元にした変数名称として root_Gain と表示されます。ブロック線図上の名称と変数名称との関係については、サブシステム・ビューアーのヘルプを参照してください。

RTモデルを作成後、初めてロードした状態では何も選択されない状態となっていますので、このDLGのリストの中から観測したい変数名をマウス左ボタンでクリックし選択状態にします。例えば、トップレベルの Gain1 と Sum の各ブロックの出力を観測したい場合、root_Gain1 と root_Sum を選択します。（下図）



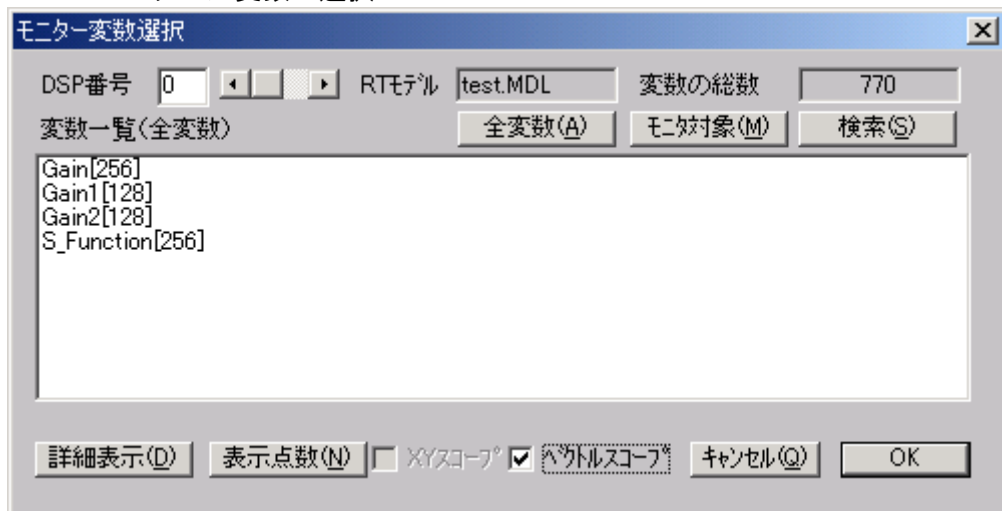
マルチDSPの場合はDSP番号の右のスライダー又はエディットボックスで目的とするDSPボード番号を選択したうえ、そのボードの変数を選択します。

誤って選択してしまった変数は、再度マウス左ボタンでその変数をクリックすれば非選択状態に戻ります。

選択可能な変数の最大は合計30個です。

注) 観測したい変数名が変数一覧に表示されない場合、弊社ホームページのQ&A「MATLAB対応ライブラリでよくあるQ&A」を参照してください。

9.2 ベクトル変数の選択



ベクトル情報をもつ変数を観測したい場合は、ベクトルスコープをチェックします。

変数一覧にベクトル情報を持つ変数が表示されるので選択後OKボタンを押してください。

9.3 表示点数の設定

選択が完了したら、表示点数ボタンを押し、変数表示点数D L Gを開きます。

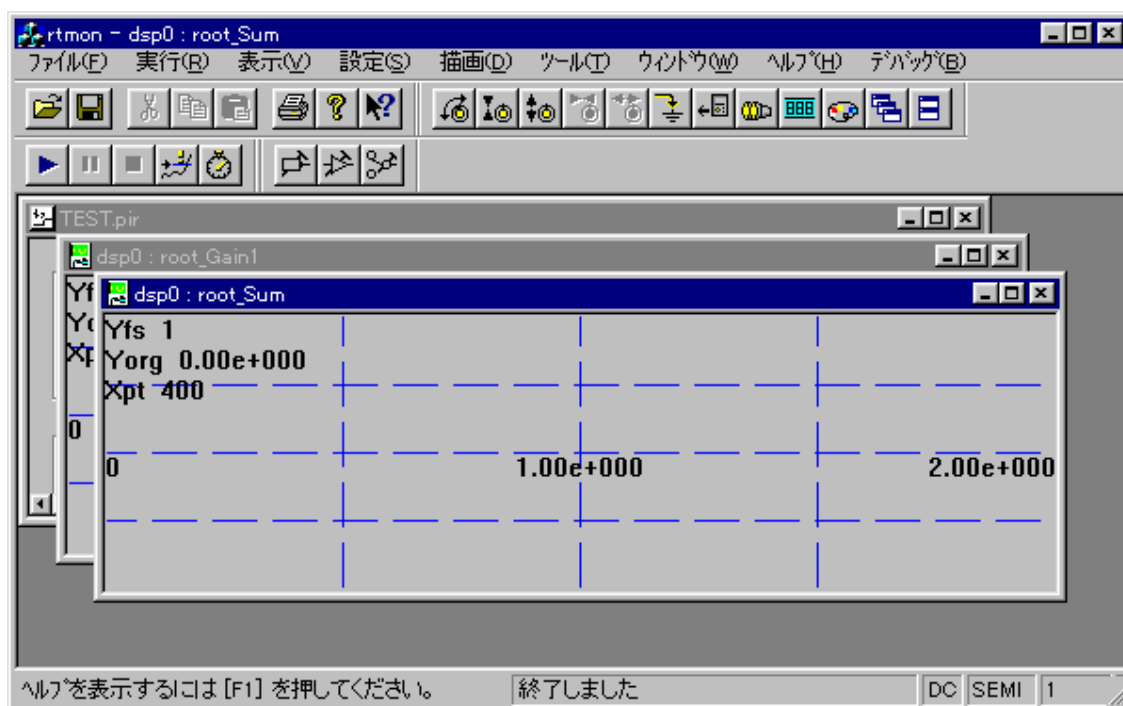
バッファのデータ点数	128	点	1.28	秒	
表示の全データ点数	128	点	1.28	秒	
表示の実施点数	128	点	1.28	秒	
ステップサイズ	0.01	秒			
間引き	1		回に1回リサンプリングする		
ベクトル用の間引き(時間)	1		回に1回リサンプリングする		
ベクトル用の間引き(データ)	1		点に1回リサンプリングする		

ヘルプ(H) キャンセル(Q) OK

バッファのデータ点数を変更するとその右に表示されている時間が変わります。この時間は、グラフ表示した場合の横軸フルスケール値です。上の例では 1.28 秒となっていますので、グラフ表示の左端から右端までの時間が 1.28 秒となります。これを参照しながらデータ点数を決定します。ただし、間引き設定時は表示時間×間引き数が実際の時間となります。

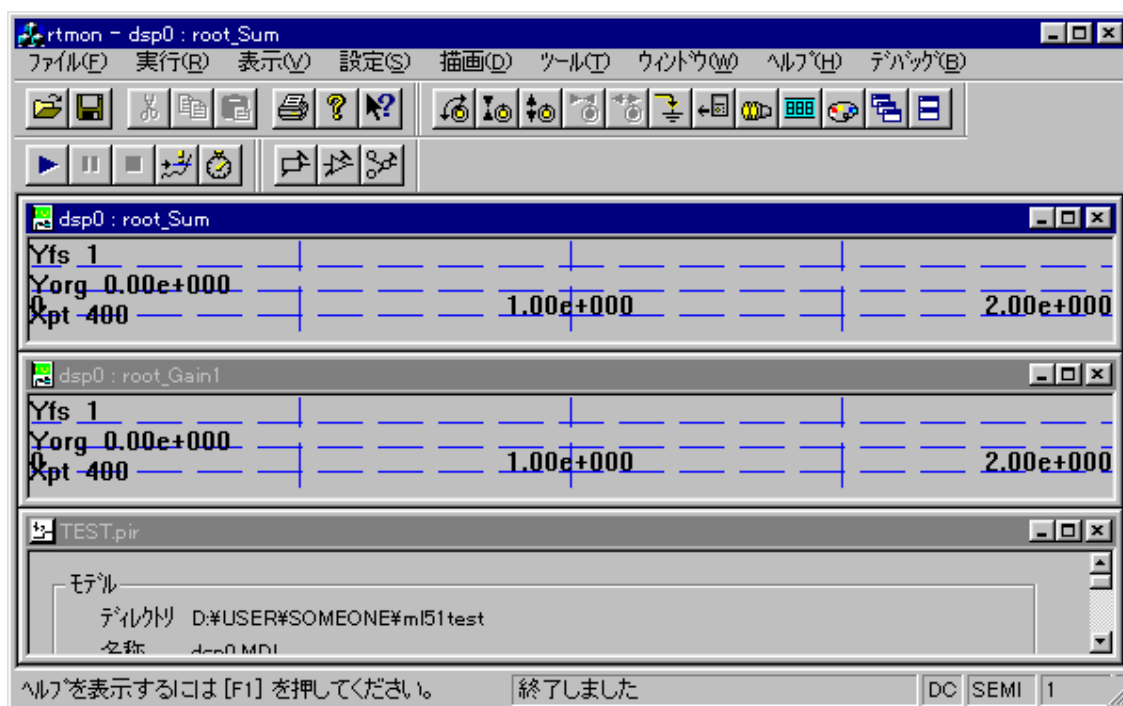
バッファのデータ点数、表示の全データ点数、表示の実施点数は、ここでは共に同じ値に設定します。設定が完了したらOKボタンを押してください。

更に、モニター変数選択D L GのOKボタンを押してください。変数選択が完了し、画面には今選択した変数を表示する為のスコープウィンドウが表示されます。



9.4 スコープの自動配置

開いたばかりのスコープウィンドウは上の図の様に、順次重ねて（カスケード）表示されます。マウスで位置・大きさが適宜変更できますが、綺麗に並べて表示するのが目的なら、メニューから、ウィンドウ(W) ～ 並べて表示(T)（重ねて表示(C)）又は、ツールバーの並べて表示（重ねて表示）ボタンを選択すると、並べて（重ねて）表示するよう自動的に配置されます。また、当面不要な変数はアイコン化したうえで、同操作をすると開いているウィンドウについて再配置されます。下図は“並べて表示”を実施した例です。



9.5 変数の数値読み取り

変数をモニターしているスコープウィンドウ内部をマウス左ボタンで左右にドラッグすると、縦方向の線カーソルが表示され、その位置の、画面左端からのデータ点数値と、変数の数値がステータスバーに表示されます。表示位置はステータスバーの右端ですので、RTMONのメインウィンドウを小さくしている場合は隠れてしまっていて見えない場合があります。RTMONを最大化してもう一度試みてください。

変数がまだスコープに表示されていない場合、ステータスバーの数値を表示する場所には (no data) と表示されます。これによりスコープにデータが無いのか、それとも値が大きすぎて画面からはみ出ているのかが判定できます。

9.6 スコープのレンジ変更

変数の値が大きすぎてスコープ画面からはみ出ている場合、スコープの設定レンジを変更します。

まず、レンジ変更したいスコープウィンドウをマウスでアクティブにします。アクティブか否かは、タイトルバーの色で判断できます。

次に、メニューから、表示(V) ~ 縦軸レンジ拡大(U) (縦軸レンジ縮小(D)) または、ツールバーの Y 軸拡大 (Y 軸縮小) ボタンを押します。1 回押す度に縦軸レンジが約 2 倍 (1 / 2) に変更されます。縦軸レンジは、1 → 2 → 5 → 10 の様に変更できます。

現在の縦軸のフルスケールは、スコープウィンドウの左端付近に Yfs 1 等として、（及び、ステータスバーにもアクティブなスコープのレンジ値が）表示されています。

1 と表示されている場合、縦軸方向の真ん中が 0 で上端が +1、下端が -1 です。

概ねのレンジ値が予め判っている場合、拡大／縮小ボタンを幾度も押さず、直接レンジ値の設定が可能です。目的とするスコープウィンドウをアクティブにし、メニューから、表示(V) ～ スコープ設定 (P) を選択し、スコープ設定 D L G を開きます。



縦軸レンジは“Y軸レンジ設定(FS)”のコンボボックスから選択します。

9.7 Y軸オフセットを付ける

もし、観測しようとする変数に直流分が乗っている場合、「拡大しようとする」と画面からはみ出してしまう。かといって、縮小すると波形の細部が見えない」、といった事態になります。その場合は、Y軸方向にずらす事ができます。

その方法は3通りあります。

1つ目は、Y軸方向にずらす量（DCオフセット値）を直接設定する事です。設定はY軸レンジ設定と同じ、スコープ設定 D L G の“DCオフセット”で行います。例えば、信号の平均値が約 0.5 の場合、DCオフセットに 0.5 と入力します。これによりY軸方向の真ん中が 0.5 になります。信号の平均値は、スコープウィンドウをマウス左ボタンでダブルクリックして開いた“スコープ読み取り D L G”で読み取れます。

2つ目は、マウスを使って視覚的に行う方法です。まず目的の信号が画面に入るまで縮小します。この時、余り小さくし過ぎず、波形がかろうじて画面からはみださない程度とします。この状態でスコープウィンドウの内部をマウス右ボタンで上下にドラッグします。

ドラッグしている間は横方向の線カーソルが表示され、ステータスバーにカーソルの位置の信号値が表示されます。このままドラッグし、波形のY軸方向の中心と思しき位置でマウス右ボタンを放します。これにより、ボタンを放した時点のカーソルの位置がY軸方向の真ん中に来る様、Y軸方向にグラフが移動されます。これは、ボタンを放した時点のY軸値を読み取り、先のDCオフセット値に設定したのと同じ意味となります。

3つ目は、自動的に行う方法です。スコープ設定DLGを開き、入力結合をACに変更します。測定機のおシロスコープのAC/DC結合のイメージ的には似ていますが、違う点は、実際にハード的にAC結合にするのではなく、データ処理によりAC結合の様な動作をしている点です。この設定の場合スコープは、以下の手順でDCオフセットを自動調整します。(1) 1回目の表示は現在のDCオフセット値でグラフ表示します。(2) 表示したデータの平均値を求めます。(3) この値をDCオフセット値に設定します。従って2回以上表示を反復する場合に効果が現れ、前回表示したデータの平均値が、画面真ん中に来るよう今回のデータの表示が行われます。平均値が安定している、比較的周期性のある信号の場合に効果が発揮されます。逆に、突発的な現象や、信号変化が緩やかで1周期が横軸の長さより長い場合等は逆効果かも知れません。

9.8 Y軸オフセットの除去

Y軸方向に移動したグラフを元に戻し、Y軸方向真ん中を数値0にするには、スコープ設定DLGでDCオフセット値を0に設定するか、又はメニューから、表示(V) ~ DCオフセット除去(Z)を選択するか、又は、ツールバーの表示オフセット除去ボタンを押します。

9.9 変数の削除

不要になった変数モニターを削除するには2通りの方法があります。

1つ目の方法は、不要なスコープウィンドウ右上のXボタンを押す事です。

もう1つの方法は、モニター変数選択DLGを開き、不要となった変数をマウスで非選択状態にしたうえ、OKボタンを押す事です。

9.10 スコープ状態の保存・選択

現在のスコープ状態が保存できます。通常、初めてモデルロード時は何も表示されません。2回目以降のロード時は前回終了時のスコープ状態がモデル名.scf というファイル名で保存されているのでこのファイルより状態を復帰します。このファイルとは別に現在のスコープ状態を保存・選択できます。

保存・・・ 表示→スコープ保存 ファイル名をつけて保存してください

選択・・・ 表示→スコープ選択 保存したファイル名にて呼び出してください。

10 計算所要時間の測定

D S P 内部でブロック線図の計算にどれだけ時間が消費されているかを知ることができます。原理的にステップサイズはこの時間より短くすることは出来ませんから、この時間を知る事は、ステップサイズがどこまで短くできるかを知る上で重要な事です。又、時間余裕を知る意味でも重要な事です。

まず、正確な計算所要時間を把握するには、ステップサイズを充分大きくしたうえで、R T モデルを動作させる必要があります。計算所要時間がステップより長いと異常と見なしてR T モデルが自動的に停止する仕組みになっている為です。

R T オプション設定のステップサイズを、例えば0.1秒とか1秒等、充分大きい値に設定し、モデルを起動します。

次に、メニューから 表示(V) ~ 計算所要時間表示(N) (又はツールバーの計算所要時間ボタン) を選択し、計算所要時間D L Gを表示します。

尚、ここで計算所要時間が表示できるモデルはR T Wで生成したモデルのみです。

番号	所要時間 [μSec]	ピーク	番号	所要時間 [μSec]	ピーク
0	36.89	☒	8	(未実装)	☐
1	45.80	☐	9	(未実装)	☐
2	(未実装)	☐	10	(未実装)	☐
3	(未実装)	☐	11	(未実装)	☐
4	(未実装)	☐	12	(未実装)	☐
5	(未実装)	☐	13	(未実装)	☐
6	(未実装)	☐	14	(未実装)	☐
7	(未実装)	☐	15	(未実装)	☐

この表示の更新の間隔は約1秒ですが、ステップサイズがこれより長いと(例えば10秒)、正しい所要時間値が表示されるのは、R T モデル起動、又はこのD L Gを開いてからステップサイズ(10秒)経過後となります。

通常、表示値は一定ではありません。ばらつきます。一般のR T モデルの場合、ばらつきが数μ秒以内であれば正常と見なして構いません。

しかし、R T モデルがユーザ定義関数等を含んでいて、この関数の実行時間が回を追う毎に変化する場合は、それがこのD L Gに反映され、表示値がばらつきます。

また、積分アルゴリズム、データロギングの有無、変数モニタの有無、イネーブルドサブシステムやトリガードサブシステムが有る場合はその実行の有無により、所要時間は大きく変わります。これらの条件を色々変え、最長の値を見極める必要があります。

R Tモデルを安全に動作させるには、数値を何回か測定し、その最高値を所要時間と見なし、ステップサイズはこれより長い時間を設定しないといけません。

上記のような所要時間にばらつきがでるモデルについては、“ピーク” 欄へチェックをして計測してください。実行時からの最長の時間が表示されます。

11 データロギング

データロギングは、DSP内部メモリーにRTモデルの動作中のデータを貯え、RTモデルの動作終了後これを取り出しファイル化するメカニズムです。

ロギングデータは、1ステップの計算を行う都度1組みのデータが生成されます。従ってステップサイズTでN秒間動作を継続するとデータは N/T 組み生成されます。一方、DSPに保存できるデータの量にはメモリー容量の都合で上限があります。これを超える事は出来ません。保存しきれないデータは保存領域がいっぱいになった段階で、古い順に捨てられます。

11.1 ロギング項目

ロギングできる項目は以下の通りです。

○現在時刻値	Time(T)	
○状態変数	States(X)	
○出力変数	Outputs(Y)	
○変数	Variable	
○Scope ブロック	Scope	(ML 4.2c のみ)
○To Workspace ブロック	To Workspace	(ML 4.2c のみ)

現在時刻値は、ブロック線図に Clock ブロックを付けた場合、このブロックの出力値と同じです。後にMLで時刻データが必要になる場合、わざわざブロック線図に Clock ブロックを取り付けなくても、現在時刻値が保存できます。

状態変数は、モデルが状態変数を持っている場合、計算ステップ毎の状態変数値が保存できます。但し、ここで保存できるのは、メジャータイムステップ（ステップサイズ間隔）のみです。

出力変数はブロック線図のトップレベルに取り付けた出力ポートに出力されるデータです。これは、ブロック線図から大域変数を経由してユーザ定義関数へ渡されるデータでもあります。

変数は全変数のうち、変数モニターで選択した変数についてロギングされます。

Scope ブロック／To Workspace ブロックは、ML 4.2c の場合にのみ有効です。各ブロックへ出力されるデータがロギングされます。

11.2 ロギング項目選択

上記データロギング項目のどれを取り込むかは、リアルタイムオプションDLGの下半分、データロギンググループのチェックボックスで設定します。

リアルタイムオプション(dsp0.MDL)

積分オプション

アルゴリズム: ode1(Euler) RTW設定(A)

ステップサイズ: 0.005 RTW設定(S)

開始時刻: 0

終了時刻: 99999

時間倍率: 1

データロギング

☒ Time(T) ☒ States(X) ☐ Scope

☒ Variable ☐ Output(Y) ☐ To Workspace

間引き: 1 回に1回ロギングする

バッファサイズ: 2 点

ファイル形式: Binary

ファイル名: dsp0.MAT 参照(R)

☐ フォルダ変更

OK

キャンセル(Q)

ヘルプ(H)

デフォルト

登録(S)

呼出(L)

上の例では、現在時刻、状態変数、変数、が選択されています。

11.3 データ間引き

長時間に渡りデータを保存したいが、全データを保存するにはメモリー容量が不足する場合、データの間引きを使います。

間引き設定は、1を設定した場合、毎回データが保存されます。2では2回に1回、3では3回に1回のデータが保存され、それ以外のデータは破棄されます。

アンチエイリアシング等の処理は行われていません。必要なら、ブロック線図上でフィルターリングを行う必要があります。

11.4 データ点数

データ点数は、DSP内部に確保するバッファの長さと考えられます。点数の単位は2種類から選択できます。上の図でバッファサイズの右の、点と表示されているコンボボックスを開きます。選択肢は点とK点(×1024)です。点の場合は、その左に表示されている数値そのものがバッファの点数=保存できるデータ点数となります。K点(×1024)の場合は、その左の数字×1024点がバッファの点数となります。例えば、2K点と2048点とは同じ意味を持ちます。

このデータ点数は、ロギング項目毎、変数毎にこの点数分用意されます。

11.5 ファイル形式・ファイル名

ロギングデータをファイルの保存する場合の形式と名称を設定します。

形式は通常はBinalyとしてください。この形式がMLのデータファイルと互換があります。ASCIIはML以外のプログラムとデータ交換する場合用に使用します。MLでは特殊な場合を除きロード出来ません。

ファイル名の拡張子は、Binalyの場合は.MAT、ASCIIの場合は.DATとしてください。

また、“フォルダ変更”にチェック無し時はモデル名_grt_rtwフォルダへ、チェック有り時はモデルのあるフォルダへ保存されます。(MATLAB Version 6.0以降)

11.6 マニュアルトリガー

ロギングデータは、モデル実行時に任意の時刻に取得することができます。



トリガーON  をクリックするとその時点でデータロギングが停止します。ロギングを再スタートするにはトリガー有効  をクリックしてください。

12 モデルの終了

R Tモデルは次の条件で終了します。

- 終了時刻に達した時。
- R Tモデルで **Stop simulation** ブロックが使われていて、停止条件が成立した時。
- R TMONで停止操作をした時。
- トリガー機能が有効でかつ自動停止が有効となっており、トリガー条件が成立した時。

R TMONでは、以下の何れかの操作を行うとR Tモデルの動作を停止させる事ができます。

- 1、メニューから、実行(R) ～ 実行終了(E)を選択する。
- 2、ツールバーの実行終了ボタンを押す。
- 3、モデルウィンドウのXボタンを押す。
- 4、別のR Tモデルをロードする。

できるだけ1又は2の方法を使用してください。

13 ログイングデータの保存

R Tモデルでログイングしたデータは、R Tモデルの停止後ファイルに保存します。

13.1 シングルモデルの場合

シングルモデルの場合、上書き保存（予め決めておいた名称で保存）と新規保存（保存の時点で名称を決める）ができます。

13.1.1 上書き保存

予め名称を決めておくには、リアルタイムオプションD L Gのファイル名エディットボックスで決定します。

上書き保存の実行は、メニューから、ファイル(F) ～ データ上書保存(S)、又は、で保存します。保存されるデータは、R Tモデルが格納されているフォルダーに保存されます。

13.1.2 新規保存

上書き保存の実行は、メニューから、ファイル(F) ～ データ新規保存(A)を選択します。



ログイングデータファイル新規保存D L Gが開きますので、ここで既存のファイル名称を確認のうえ、ファイル名(N) に保存したいファイル名称を入力し、保存(S)ボタンを押します。新規保存ですが、既存ファイル名を指定した場合は警告を表示したうえ、上書き保存されます。

13.2 マルチモデルの場合

マルチモデルの場合、上書き保存（予め決めておいた名称での保存）しか行えません。

データファイルはD S P毎に1個ずつ個別に保存されます。これは、複数のR Tモデルで同じ名称の変数（＝別々のブロック線図に同じ名称のブロック）が存在する可能性がある為、複数のログイングデータを1つにまとめられないことによります。

個々のデータファイルの名称は、リアルタイムオプションDLGで予め設定しておきます。

上書き保存の実行は、メニューから、ファイル(F) ～ データ上書き保存(S)、又は、で保存します。保存されるデータは、RTモデルが格納されているフォルダーに保存されます。

14 ロギングデータのMLでの利用

R TMONで保存したデータをMLで利用するには、MATLAB Command Window で load コマンドでロードします。これで初めてDSPで採ったデータがMLワークスペースへロードされます。

マルチモデルのロギングデータの場合、各DSPのデータは個別のファイルになっていますので、load コマンドを反復してロードします。この時注意が必要なのは、各データで同一名称が存在する場合、警告されず、単に後でロードされたデータによって置き換えられる事です。

ロードされたデータの変数名は、以下の通りです。

現在時刻	T
状態変数	X
出力変数	Y
変数	変数選択DLGに表示された名称と同じ。
Scope	Scope の名称
To Workspace	To Workspace ブロックに設定した名称

Tはバッファサイズ×1の縦長ベクトルです。

Xはバッファサイズ×状態変数の数のマトリクスです。列ベクトルが1つの状態変数の時系列ベクトルです。行方向にどんな状態変数が格納されているかは、現状では簡単に表示する機構が用意されていません。RTWが出力したソースファイル *model.h* の構造体宣言

```
typedef struct States_tag {  
.....
```

```
} States ;
```

が参考となります。データはこの構造体のメンバーの順に並んでいます。メンバーが配列の場合はその添え字の0から順に並んでいます。

Xのなかの特定の状態変数例えば2行目の状態変数の時系列を取り出し描画するには、MATLAB Copmmmand Windowd で

```
X2 = X(:,2); plot(T,X2);
```

と実行します。

Yはバッファサイズ×出力点数のマトリクスです。列ベクトルが1つの出力の時系列ベクトルです。X同様に操作できます。

それ以外のデータも全て列ベクトルです。

15 自動テスト用ブック作成

メニューの設定→自動テスト用ブック作成 にて作成できます。このブックは Microsoft 社製 Excel で作成するブックのことです。予め PC に Excel がインストールされていないと使用できません。動作可能 Excel のバージョンについては、“自動テストユーザーズ・マニュアル”を参照してください。

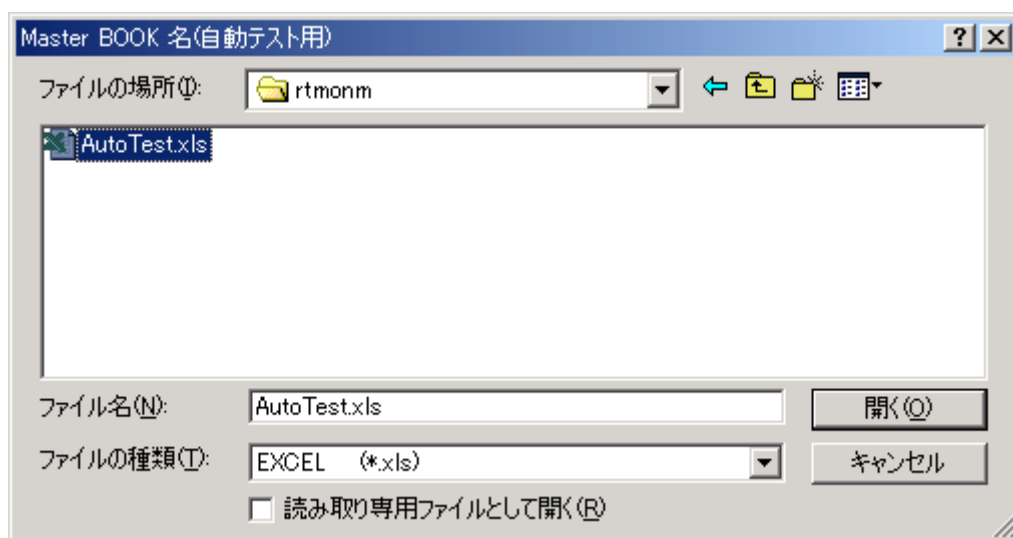
このブックは、自動テスト実行時における実行条件と、Frmxls/Toxls ブロックで使用するデータの場所を示すものです。自動テスト実行時又は RTMON にて Frmxls ブロック付のモデル実行時には必ず作成してください。

尚、Frmxls/Toxls ブロック実行時に、Excel で指定した内容とブロックで指定した各パラメータの内容の整合性チェックはしませんので、ブック作成時に確認してください。もし整合が取れていない場合は、モデルの動作は保証できませんので、予めご了承ください。

整合性とは・・・List シートにて指定する Index とブロックで指定した Index 値の一致、List シートで指定するブロックの種類などです。詳細は“I/O デバイスライブラリ”又は“自動テストユーザーズ・マニュアル”を参照してください。

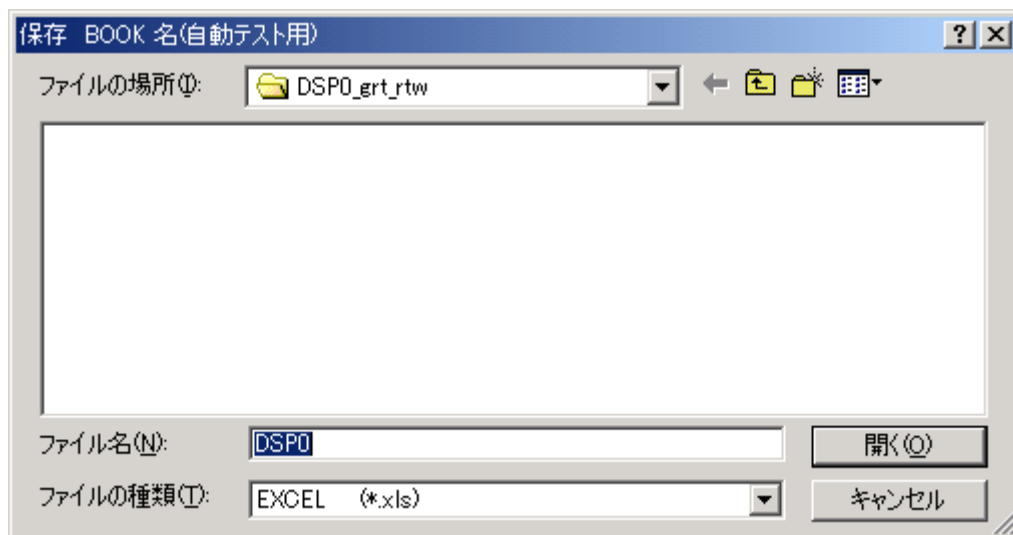
15.1 基本ブック指定

モデルをロードします。次に自動テスト用ブック作成を選択し、以下のような基本となるブック “MATLAB インストールフォルダ¥rtw¥c¥adsp32x¥rt¥Autotest.xls”を指定します。

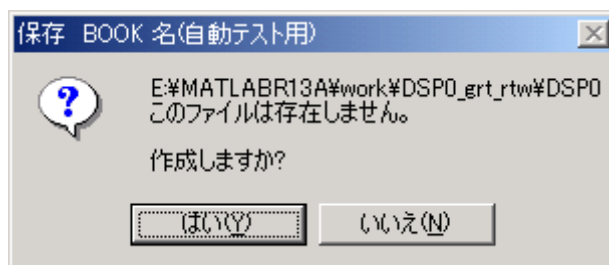


15.2 作成ブック指定

作成するブックの保存場所、名称を指定します。



ブックを新規作成時は以下のダイアログが表示されますので“はい”を選択してブックを作成します。



以上で現在の RT オプションの状態で作成されますので、“自動テストユーザーズ・マニュアル”を参照の上実行条件、データを指定しブックを完成し、自動テストを起動してください。

<< メ モ >>

MATLAB対応ライブラリ (マルチ)

ADSP 324-41C

ADSP 404-241C

ADSP 674-341C

リアルタイムモニター・チュートリアル

第1版 2001.05.04

第9版 2006.02.08

中部電機株式会社 CS事業部

TEL 0532-63-5311

FAX 0532-61-9566

自動テスト ユーザーズガイド

中部電機株式会社

目 次

I.	概要	1
II.	動作環境	1
III.	実行手順	2
1.	モデル作成	2
2.	パラメータファイル作成	3
3.	EXCELBOOK作成	4
4.	自動テスト起動	5
IV.	実行サンプル	7
1.	モデル作成 → モデルビルド	7
2.	パラメータファイル作成 → RTMONにて作成	7
3.	EXCELBOOK作成 → テンプレートブックをコピーして作成	8
4.	起動 → MATLAB COMMAND WINDOWから起動	9
V.	エラーコード表	10

I. 概要

自動テスト機能は、MATLAB Command Windowと弊社製リアルタイムモニタ（以下RTMON）より起動できる、リアルタイムテストを自動化するソフトウェアです。実行時に、ステップサイズ／終了時間等の動作条件や、Gain／Constant／Switch等のパラメータをダイナミックに変更できます。

また、新たに自動テスト用I/Oブロックとして `frmxls/toxls` が追加されたことにより、入力データを予め他の計測器等でExcelファイルとして準備し、実行時にダイナミックに入力できたり、出力データをExcelファイルとして保存も可能となりました。

このことにより、MATLABスクリプトにて起動コマンドを記述すれば、テスト毎に条件・パラメータを最適化しながら連続テストも可能です。

II. 動作環境

- MATLAB対応ライブラリ（Ver 5.1.0以上）が動作する環境
（動作環境は別紙MATLAB対応ライブラリユーザズマニュアルを参照してください。）
 - Microsoft Excel 2000 for Windows
- * Microsoft は、米国 Microsoft Corporation の米国およびその他の国における登録商標です。
- * MATLABは、米国 The Math Works Inc.の登録商標です。

III. 実行手順

1. モデル作成

MATLABにてモデルを作成し、ビルドします。

自動テスト用 I/O ブロックとして、frmxls / toxls ブロックが使用できます。

(frmxls / toxls ブロックは自動テスト時のみ有効のため、それ以外で実行時は動作が保証されません。)

frmxls…Excel 形式の時刻付データを実行時にダイナミックに入力するブロック。

<Parameter>

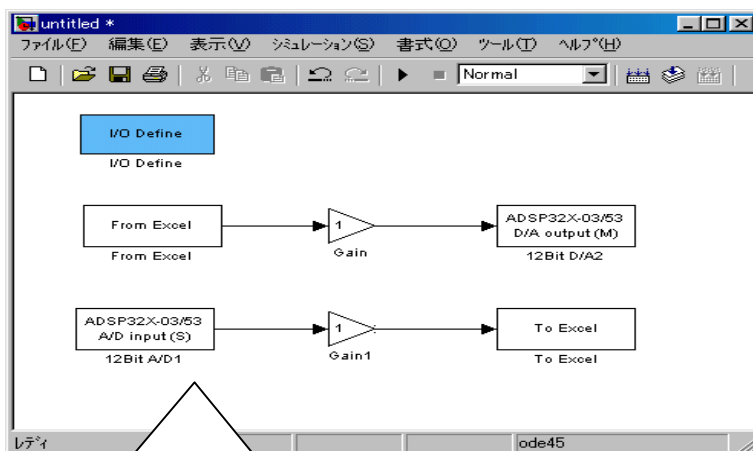
- VariableIndex … データ ID として 0 ～ 99 までの整数。
toxls の VariableIndex と重複しないように設定。

- SampleTime … 入力サンプルタイム。

toxls …実行結果を Excel 形式の時刻付データとして出力するブロック。

<Parameter>

- VariableIndex … データ ID として 0 ～ 99 までの整数。
frmxls の VariableIndex と重複しないように設定。
- Decimation … 間引き数指定。(ex. 2 指定時サンプルタイム
の 2 回に 1 回出力)
- SampleTime … 入力サンプルタイム。

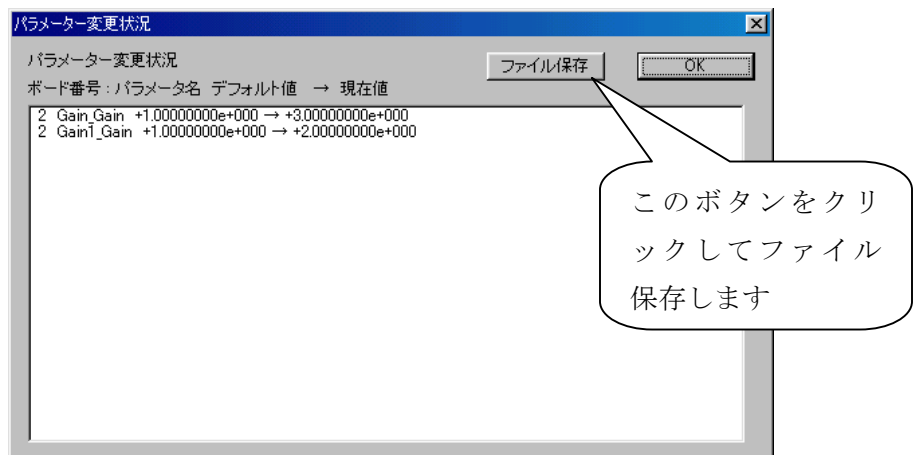


toxls と frmxls ブロックの VariableIndex は、重複しないように設定してください！

2. パラメータファイル作成

- ① R T MONを起動し、メニュー → R Tモデルロード を選択モデルをロードしてください。
- ② A) メニュー → 設定 より設定したい項目（ゲイン・定数・閾値・その他のパラメータ）を選択し値を設定してください。
B) 弊社製モデルコントロール（別売オプション）の、ボリューム・スライダ・数値入力コントロールからもブロックパラメータの設定が可能です。
- ③②にてすべて設定が終了したら、メニュー → 設定 → 変更状況の表示 を選択し“ファイル保存“にてTEXTファイルに保存します。

*その他のパラメータについては、一部実行時に変更できないものがあります。



3. Excel Book 作成

下記ディレクトリにテンプレートExcel Book “Autotest.xls” が格納されていますので、このBookを利用し動作条件を設定してください。このBookは、“condition” “list” の2種類のシートから構成されています。

また、Bookの作成はRTMONからも可能です。(メニュー→設定→自動テスト用Book作成)

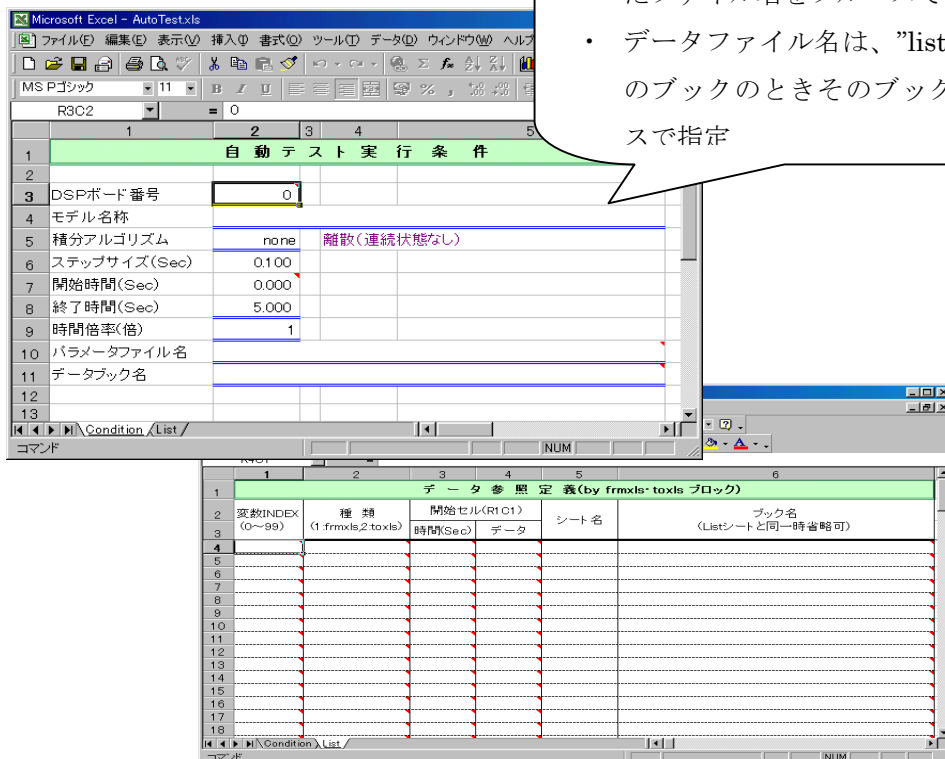
“condition” …動作条件設定用

“list” …frmxls / toxls ブロックのデータ設定用。frmxls / toxls ブロック未使用時もこのシートは必要です。

<テンプレートExcel Bookディレクトリ>

MATLAB インストールディレクトリ\rtw\c\adsp32x\rtmonm (adsp324 時)

MATLAB インストールディレクトリ\rtw\c\adsp32x\rtmonm67 (adsp674 時)



4. 自動テスト起動

1) MATLAB Command Window 利用 (MATLAB Ver5.2.1 以上対応)

<コマンド>

- `setoptn` …Condition シート：動作条件設定
- `setprm` …Condition シート：パラメータファイル名設定
(RTMONにて作成したファイルをフルパス指定)
- `setdata` …Condition シート：データブック名設定
(“list” シートのブック名が”Condition” シートのブックと異なるときのみフルパス指定)
- `autotest` …自動テスト起動
- `sample` …自動テスト連続起動

上記コマンドを利用して実行します。各コマンドとも終了時には終了メッセージがエラー時には、エラーメッセージとエラーコードが表示されます。エラーの詳細についてはエラーコード表を参照してください。

また、コマンドの引数詳細は、ヘルプを参照してください。

例 `help setoptn`

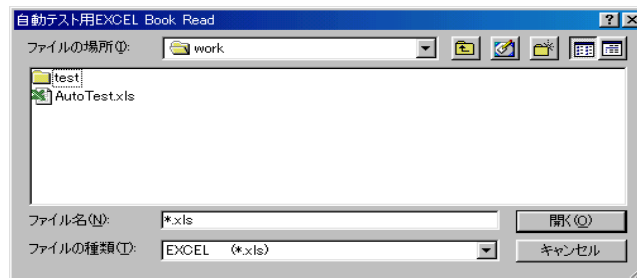
◎コマンド使用例

- ①>>`setoptn( 'autotest.xls' , 0.01)` ← **step size 0.01Sec 設定**
- ②>>`autotest( 'autotest.xls' )` ← **自動テスト起動**
- `AutoTest start!<C:¥MATLABR132¥work¥test10_grt_rtw¥autotest.xls>`
- `--> Finish!`
- ↑ **起動、終了メッセージ表示**
- ④>>`setoptn( 'autotest.xls' )` ← **現在の設定値表示**
- ⑤>>`setprm( 'autotest.xls' , '' )` ← **パラメータファイル名消去**
- ⑥>>`sample( 'autotest.xls' , ' autotest1.xls' )`
- ↑ **自動テスト連続実行**

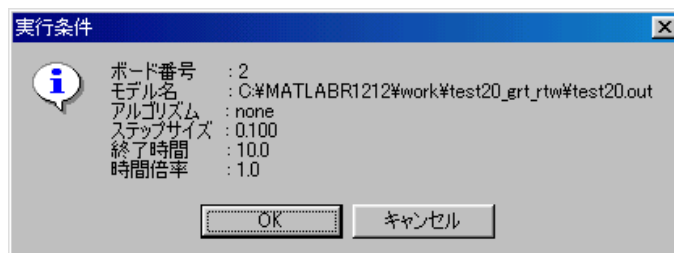
2) RTMON 利用

- ① RTMONを起動し、メニュー → 実行 → 自動テスト を選択します。
(モデル未ロード時のみ有効)

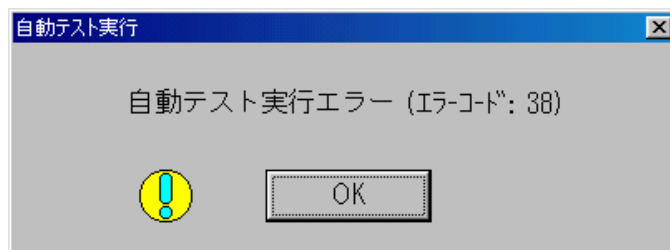
- ② 下記ダイアログにて計画用Excel Bookを選択します。



- ③ 「開く」を選択すると実行条件確認メッセージが表示されます。内容を確認し応答してください。「OK」を選択するとテストが開始されます。
実行中は、実行中ダイアログが表示され、終了するとダイアログが消えます。
実行中に関係するExcel Bookを操作（開く・編集・消去など）するとエラーとなります。

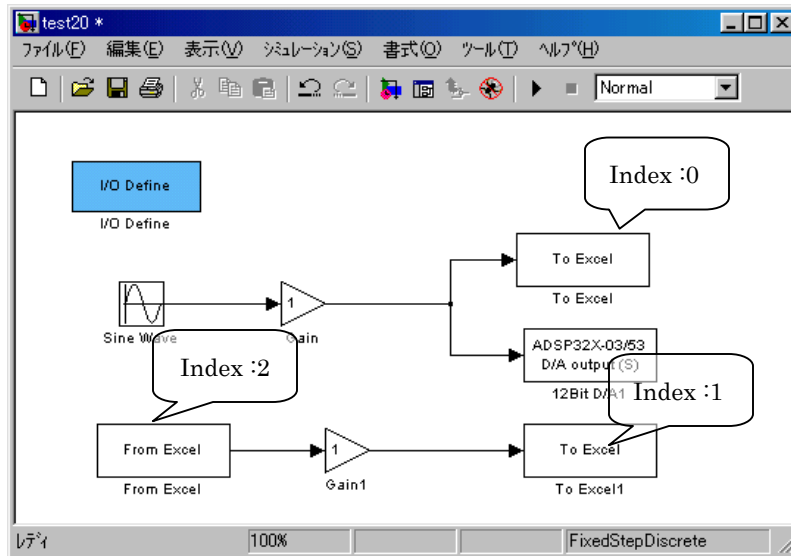


- ④ テスト実行中にエラーが発生すると下記ダイアログメッセージが表示されます。詳細については、エラーコード表を参照してください。

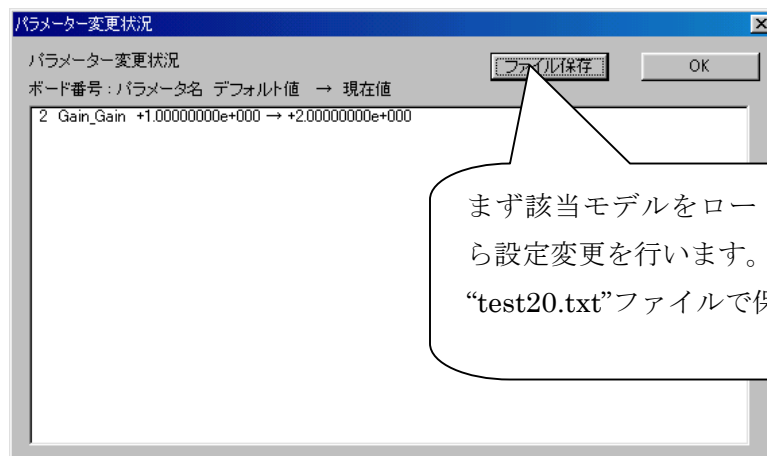


IV. 実行サンプル

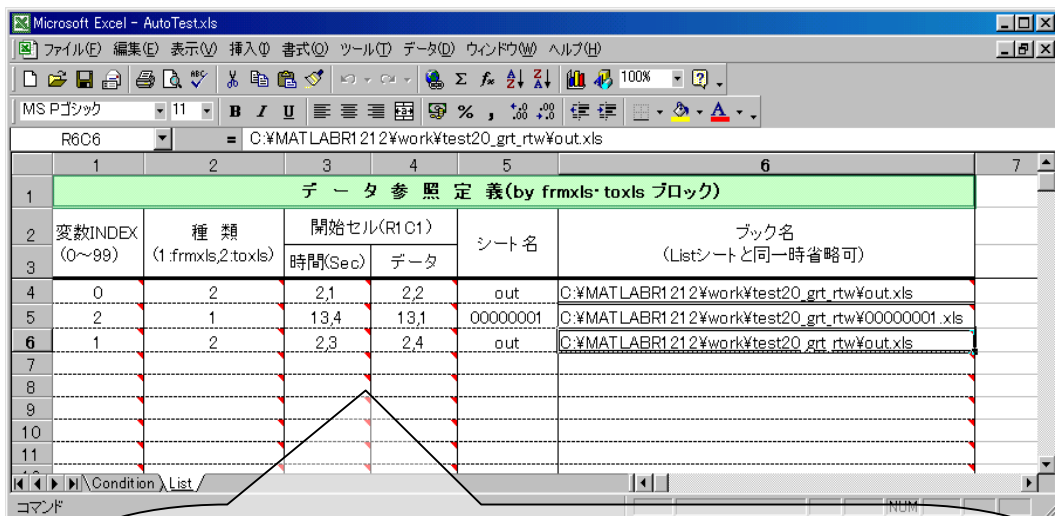
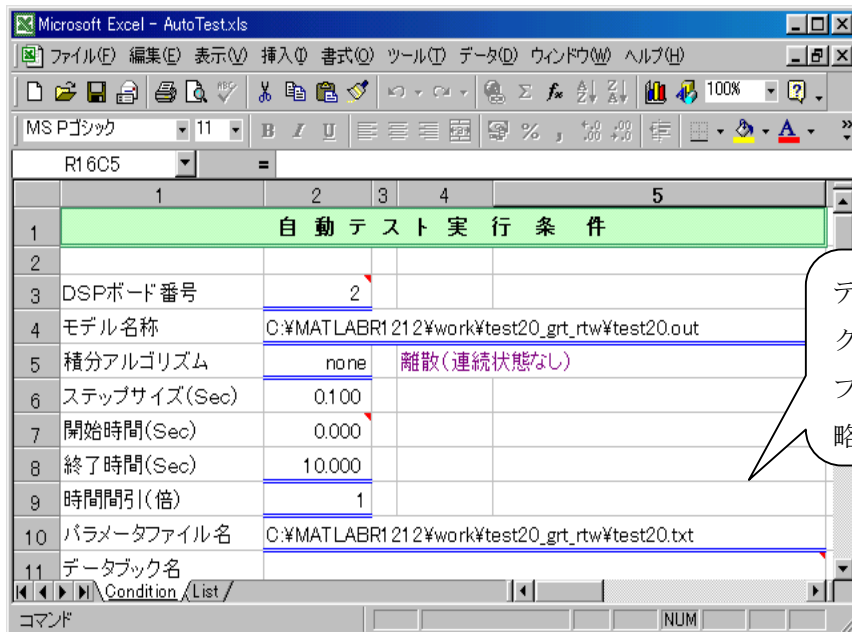
1. モデル作成 → モデルビルド



2. パラメータファイル作成 → RTMONにて作成



3. Excel Book作成 → テンプレートブックをコピーして作成

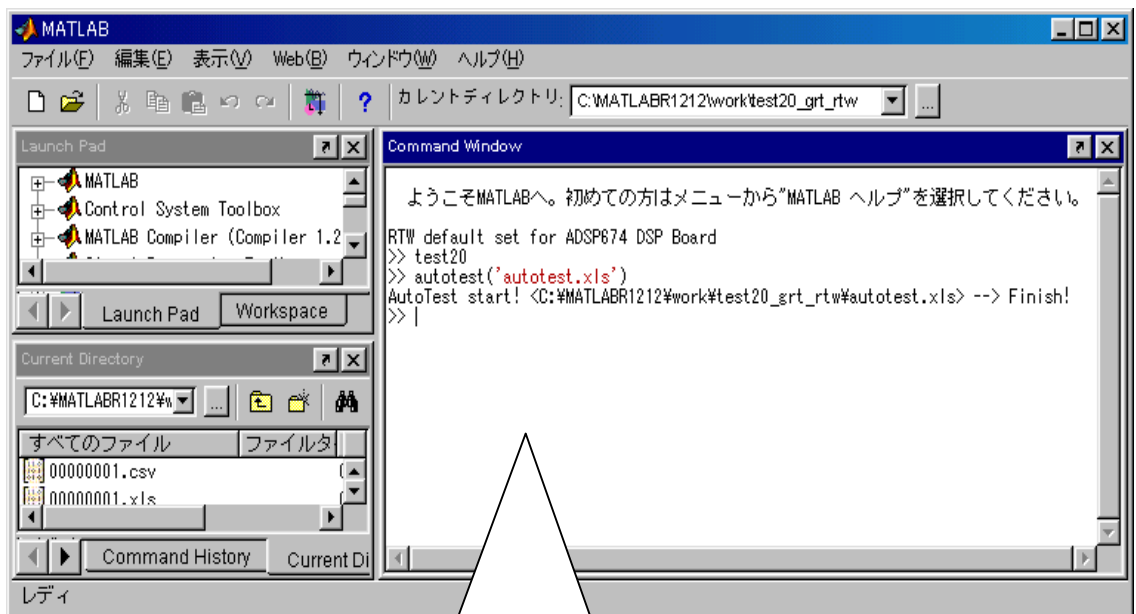


セル番地は、R1C1 形式にてコンマで区切り入力。

シート・ブックは、toxls ブロックに限り存在しなければ新規で作成されます。

ブック名は、フルパス名で設定します。但し”list”シートと同一のブックの場合は省略可。

4. 起動 → MATLAB Command Windowから起動



>>autotest('autotest.xls')

と入力し起動。

カレントフォルダにExcel Bookが存在する場合、フォルダ名は省略可。

V. エラーコード表

エラー番号	内 容
1 0	AutoTest への引数エラー
2 0	Excel アプリケーション起動エラー
3 0	IndicationBook Open エラー
3 1	IndicationBook ボード番号エラー
3 2	IndicationBook モデルエラー
3 3	IndicationBook アルゴリズムエラー
3 4	IndicationBook ステップサイズエラー
3 5	IndicationBook 終了時間エラー
3 6	IndicationBook 時間倍率エラー
3 7	IndicationBook データ Book エラー
3 8	D S P 初期化エラー
3 9	データ Book Open エラー
4 0	データ Book Block なし
4 1	パラメータファイルエラー
4 2	データ転送エラー(frmxls ブロック)
4 3	D S P エラー
4 4	D S P タイムエラー
4 5	D S P その他エラー

本マニュアルの内容は製品の改良のため予告なしに
変更されることがありますので、ご了承ください。

自動テスト for MATLAB 対応 Lib

第 1 版 2003.4.20

中部電機株式会社 CS 事業部

TEL 0532-61-9566

E-Mail csg@chubu-el.co.jp

<http://www.chubu-el.co.jp>