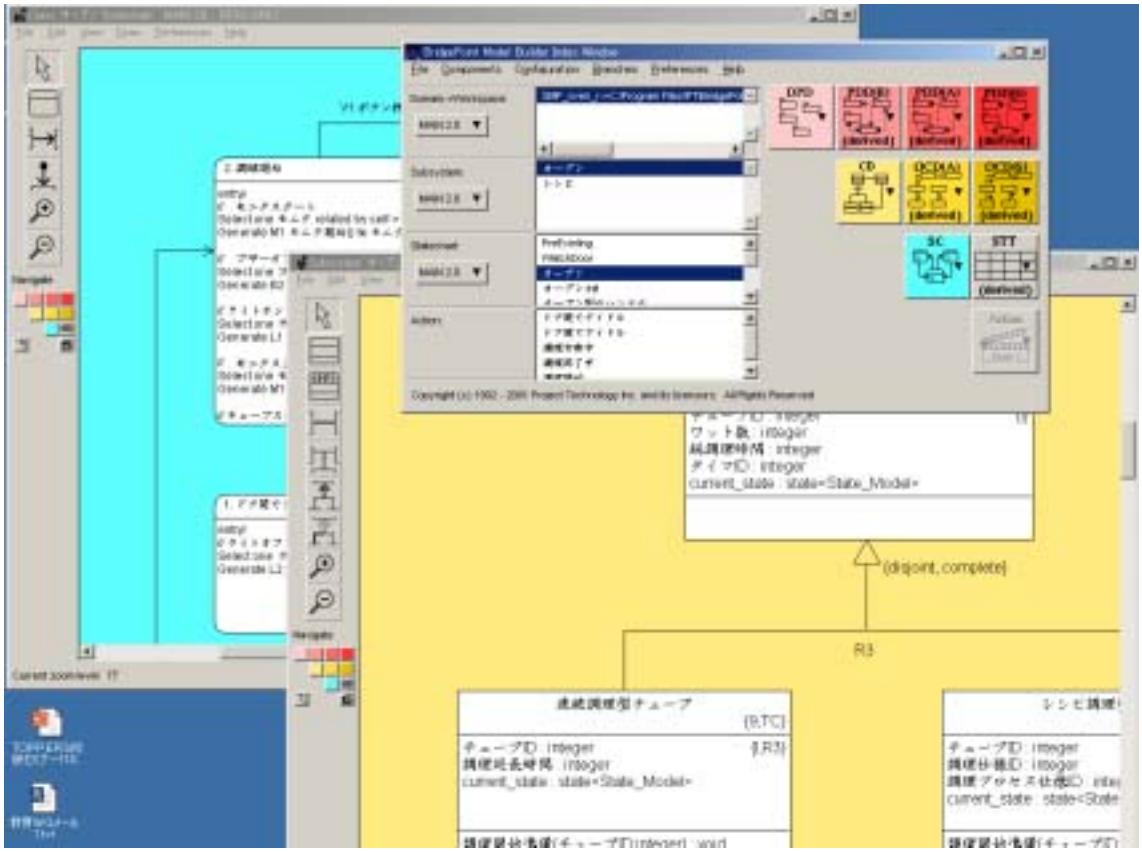
BrigePoint MC3020 の TOPPERS/JSP 対応



2004/02/12　　(第４版)

(株)リコー
画像システム事業本部 プラットフォーム開発センター
ソフトウェア生産技術開発室
竹内　良輔

００.変更履歴

履歴表

版	日付	作成者	変更内容
1	2003/05/16	竹内良輔	初版
2	2003/10/31	竹内良輔	BridgePoint6.1 への対応
3	2003/12/17	竹内良輔	システムの解説を追加
4	2004/02/12	竹内良輔	TOPPERS/JSP1.4 への対応、タイマーブリッジ専用モデルを追加

０.目次

１.概要

この説明文に添付されたソフトウェアは米国 Project Technology 社と（株）東陽テクニカが作成したものを、（株）リコー　画像システム開発事業本部　プラット・フォーム開発センターにて TOPPERS/JSP1.4 に用に改良を行ったものです。また、添付されたソフトウェアは Project Technology 社と（株）東陽テクニカのご好意により公開可能となりました。

1-1.開発環境

このソフトウェアは Windows2000 上で、Microsoft VC++6.0、BridgePoint6.1、MC3020 Version3.1.0 東陽拡張版（2003 年 9 月版）をインストールして開発を行いました。

BridgePoint による開発は、モデルビルダーで作成したモデルをモデルベリファイアまたはモデルデバッカを用いてシミュレーションを行い、その後、モデルコンパイラにてＣまたはＣ＋＋言語を生成し実環境で実行させるという手順をとります。ここではモデルコンパイラとして MC3020 を用いたソースコードを TOPPERS に対応させる手順を主体に説明を行います。MC3020 はモデルビルダーで作成したモデルから、ノンＯＳのＣ言語を生成するモデル・コンパイラです。本書に記述する内容は、上記のモデルから Windows 版 TOPPERS 用のＣ言語ソースを生成する為に必要な MC3020 の改造方法について記述します。MC3020 については Project Technology 社から提供されたものと、(株)東陽テクニカにて拡張を行ったものについて最新のバージョンについて説明を行います。

1-2.組込みＭＤＡについて

BridgePoint は、最も古くから日本に紹介されたＭＤＡツールとして知られています。MDA とは何でしょうか？ＭＤＡとは OMG(Object Management Group)が提唱しているモデル駆動型アーキテクチャー（Model Driven Architecture）という技術です。現在、ＵＭＬを用いてオブジェクト指向により記述されたソフトウェアのモデルは実装を行う為に実装に依存した表現を含んでいます。そのため、ソフトウェアの実装環境が変わってしまうと実装に依存した部分を作り変えるという作業が発生してしまいます。ＭＤＡでは、実装に依存しないモデルを PIM(Platform Independent Model)、実装に依存したモデルを PSM(Platform Specific Model)といふうに分離して考え、PIM から PSM への変換ルールを設けて、必要なときに実装環境に合わせて変換を行うことにより、ソフトウェアのナレッジ化を目指すものです。ＭＤＡ自体は広い分野に対応した技術ですが、組込みに対応しモデルではなく、実行可能なプログラムに変換する技術を組込みＭＤＡと呼んでいます。

今回、サンプルとするモデルの説明は、別紙「Project Technology 社 DesignPoint MC-3020 の紹介」に記載されています。この文中ＯＯＡモデルと記載されているモデルが PIM にあたります。

1-3. Project Technology 社版モデルコンパイラ

本資料は、Windows で動作する現在の最新版 BridgePoint6.1、モデルコンパイラ MC3020 Verision3.10 を対象にして説明を行います。MC3020 版については、Version2.2.6 から Version3.1.0 において大きな変更が行われています。

1）Unix 実行環境として MKS Toolkit を使用していたものを Cygwin に変更しています。その結果として、今まで VC++をメインのコンパイラとしていたが、Cygwin 上の gcc で実行できるように拡張が行われています。

2）静的なオブジェクト生成を PEI という手順でサポートしています。

本説明では、(株)東陽テクニカで作成した「オーブン」のモデルを例にして最終的に TOPPERS の Windows 版上で動作させる為にどのような作業が必要であるかを説明します。そのため、コンパイラを gcc からテンポラリィに VC++に設定しなおす手順を説明する必要があります。

1-4. (株)東陽テクニカ版モデルコンパイラ

このコンパイラは、Project Technology 社版モデルコンパイラをベースに、過去のバージョンと互換を取る為に以下の拡張を行っています。

1）日本語で記述された属性名、イベント名、メソッド名を生成時に C コンパイラが認識可能な名称に変換します。

2）populate.arc を用いた静的なオブジェクト生成を行います。

(株)東陽テクニカ版モデルコンパイラでは、populate.arc を用いた色づけ処理と、Project Technology 社 PEI を用いた手順の両方を選択して使用できるような拡張が行われています。

「オーブン」モデル自体が属性名、イベント名、メソッド名を日本語記述している為、このモデルをそのまま使用するためには、(株)東陽テクニカ版モデルコンパイラに拡張を行う必要があります。Version 3.1.0 に対応した(株)東陽テクニカ版モデルコンパイラでは「オーブン」モデルをサポートしておらず。このバージョンで「オーブン」モデルを生成実行してみる必要があります。

2.「オーブン」モデルの生成と実行

最新バージョンの BridgePoint は、これまで例題としてきた「オーブン」モデルに関して例題からはずしています。最新バージョンの MC3020 モデルコンパイラを用いて、過去のバージョンと同等にソースコード生成を行う手順を説明したのち、TOPPERS 環境への対応について説明を行います。「オーブン」モデルは以下の２つのドメインから構成されます。

１）オーブンドメイン(SMP_oven_r)

２）タイマードメイン(SMP_ooa_timer)

タイマードメインはタイマー機能を実現するドメインであり、BridgePoint が本来サポートする Timer Bridge を使用すれば、ドメインとしなくても同等な機能が実現されます。（オリジナルの MC3020 ではリカーリングタイマーをサポートしておらず、これをサポートする必要があります。）

ソースコードの生成を行うには、旧バージョンのバックファイルから２つのドメインをワークスペースとして取り込み、モデルコンパイラが参照できるようにコンフィグレーションを作成する必要があります。もちろん、モデルコンパイラを(株)東陽テクニカ拡張版を用います。

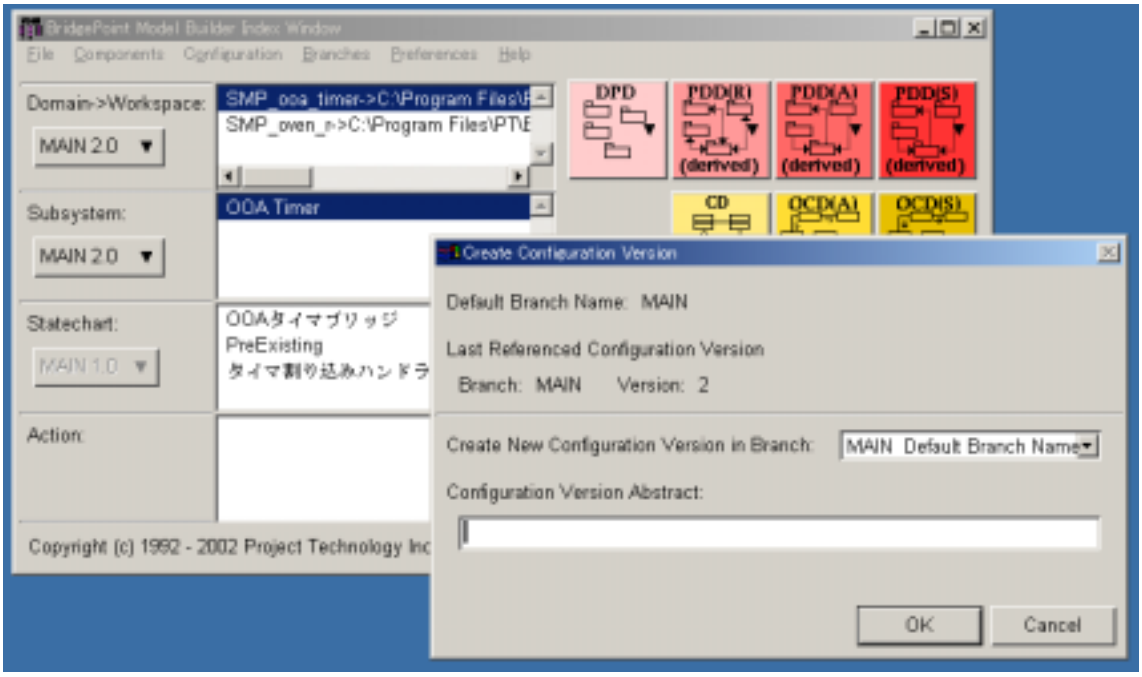


図 2-1.ワークスペースの取り込みとコンフィグレーションの作成

2-1.生成手順

MC3020 を用いて、C 言語で記述されたコードを生成する手順について説明を行います。TOPPERS が VC++で動作する為、このでは Cygwin 用の gcc ではなく、VC++を用いた

コード生成の手順についての説明を行います。加えて、東陽パレットを用いず、Cygwin のコマンドによる生成の説明を行います。説明を明確にする為、モデルコンパイラとワークスペースを以下のデレクトリィに作成します。読者の設定に合わせてこのデレクトリィを変更して実行してみてください。

・モデルコンパイラ

C:/usr/mc3020

・ワークスペース

C:/usr/BP/Oven

１）ワークスペースの作成

Cygwin を開く、モデルコンパイラの bin ファイルに移動し、rox_init_node コマンドにてワークスペースを作成します。このコマンドにて必要なデレクトリィを作成する為、前もってワークスペースようのデレクトリィを作成する必要はありません。
以下の青字の部分が入力を行ったコマンドです。結果として c:/usr/BP/Oven 以下にワークスペースが作成されます。

```
$ cd c:/usr/mc3020/bin
$ ./rox_init_node
c:/usr/mc3020/bin
Enter the pathname to directory in which you wish to tanslate:
c:/usr/BP/Oven
Creating application node: c:/usr/BP/Oven

Application translation node successfully installed!

(1) Change directory to: /cygdrive/c/usr/BP/Oven
    and enter ‘make help’ for a list of available targets.
(2) Use ‘make dom_node’ target to add a domain to the system.
    ‘make dom_node’ without further parameters will provide examples.
(3) Edit Makefile.user to provide specific commands and options
    related to your C complier, assembler, linker, etc.

$
```

２）ドメインの作成

続けて、ワークスペースに移動して、オーブンドメインとタイマードメインを作成しま

す。

```
$ cd ../../BP/Oven
$ make dom_node domain=SMP_oven_r
make[1]: Entering directory '/cygdrive/c/usr/BP/Oven'
rox_get_dom_sql:  INFO:  'c:/usr/BP/Oven/SMP_oven_r/schema/sql/SMP_oven_r.sql'
CREATED.
'rox_get_dom_sql: INFO: Domain 'SMP_oven_r' using version '2
 on branch 'MAIN'
rox_dom_init: 'c:/usr/BP/Oven/SMP_oven_r/Makefile' CREATED.

Application domain node successfully installed!

(1)  Before continuing with translation. you must register domain 'SMP_oven_r'
     in system coloring files: system/color/registry.clr.
(2) Domain coloring files have been placed in: SMP_oven_r/color.
    You may wish to edit these at this point before continuing with translation.

make[1]: Leaving directory 'cygdrive/c/usr/BP/Oven'

$ make dom_node domain=SMP_ooa_timer
make[1]: Entering directory '/cygdrive/usr/BP/Oven'
                    :
make[1]: Laving directory 'cygdrive/usr/BP/Oven'

$
```

以下に、生成されたワークスペースを示します。
Oven の下の SMP_oven_r デレクトリィと SMP_ooa_timer デレクトリィがドメインの生成に必要なファイルを格納するデレクトリィです。ドメインのデレクトリィ下の説明を行います。color デレクトリィがドメインのカラーファイルを格納する領域です。カラーファイルとはソースコードの生成時にいろいろな機能をソースコードの形で付加する為のパラメータファイルです。gen デレクトリィの下にインクルードファイルやソースファイルが生成されます。user デレクトリィの下にユーザ定義のインクルードファイルやソースファイルを格納します。skel デレクトリィにはユーザのインクルードファイルやソースファイルを作成する為のスケルトンファイルが生成されます。system デレクトリィはシ

ステム部分の生成に必要な部分を格納するデレクトリィです。構成はドメインのものとほぼ同等です。

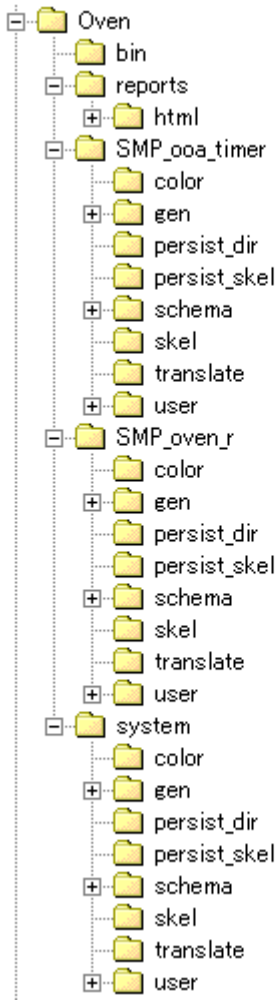


図 2-2.生成されたワークスペース

3）カラーリング

まず、2つのドメインとシステム用のカラーファイルの修正を、エデッタを用いて修正します。その後、system/user 中の手書きソースファイルをスケルトンに従って作成します。また、VC++用のコンパイル、リンクを行う為に Oven の下の Makefile.user を書き換える必要があります。

Makefile.user の 162 行目から 166 行目を以下のように変更

```
ROX_MC_TARGET_COMPILER    = “GNU”
CMD_COMPILE               = ${GNU_CC_CMD}
CMD_ASSEMBLE              =
CMD_PREPROCESS            =
```

```
CMD_LINK                    = ${GNU_LD_CMD}
                              ↓
ROX_MC_TARGET_COMPILER      = “MVC”
CMD_COMPILER                = ${MVC_CC_CMD}
CMD_ASSEMBLE                =
CMD_PREPROCESS              =
CMD_LINK                    = ${MVC_LD_CMD}
```

ここでは添付ファイル上の対応のデレクトリィに対応のファイルを入れ替えてください。
下表に載っていないものは、変更の必要のないものです。
また、カラーファイルの旧バージョンとの差異については、次節で説明を行います。

添付ファイル上のデレクトリィ	変更となるファイル	ワークスペース上のデレクトリィ
BP_TOPPERS/Oven	Makefile.user	C:/usr/BP/Oven
BP_TOPPERS/Oven/SMP_oven_r/color	domain.clr	C:/usr/BP/Oven/SMP_oven_r/color
	object.clr	
	populate.arc	
BP_TOPPERS/Oven/SMP_ooa_timer/color	domain.clr	C:/usr/BP/Oven/SMP_ooa_timer/color
	object.clr	
	populate.arc	
BP_TOPPERS/Oven/system/color	bridge.clr	C:/usr/BP/Oven/system/color
	registry.clr	
	sys_functions.arc	
	system.clr	

表 2-1.カラーファイルの修正

４）スケルトンの生成

カラーファイルの修正後、スケルトンの生成を行います。

```
$ make bridge_skel domain=SMP_oven_r
make[1]: Entering directory ‘cygdrive/c/usr/BP/Oven’
make[2]: Entering directory ‘cygdrive/c/usr/BP/Oven/SMP_oven_r’
make[3]: Entering directory ‘cygdrive/c/usr/BP/Oven/SMP_oven_r’
Regenerating database schma/gen/SMP_oven_r.gen for domain ‘SMP_oven_r’...
                                :
make[2]: Leaving directory ‘cygdrive/c/usr/BP/Oven/SMP_oven_r’
```

```
make[1]: Leaving directory ‘cygdrive/c/usr/BP/Oven’

$ make bridge_skel domain=SMP_ooa_timer
make[1]: Entering directory ‘cygdrive/c/usr/BP/Oven’
make[2]: Entering directory ‘cygdrive/c/usr/BP/Oven/SMP_ooa_timer’
make[3]: Entering directory ‘cygdrive/c/usr/BP/Oven/SMP_ooa_timer’
Regenerating database schma/gen/SMP_oven_r.gen for domain ‘SMP_ooa_timer’...
                                        :
make[2]: Leaving directory ‘cygdrive/c/usr/BP/Oven/SMP_ooa_timer’
make[1]: Leaving directory ‘cygdrive/c/usr/BP/Oven’

$
```

モデルの外部の実装依存部は、モデルでは表せない為、この部分を手書きする必要があります。手書きの部分はソースファイルの枠の部分のみ生成されます。実装する場合、枠内の部分を自分で記述する必要があります。枠のみのファイルをスケルトンファイルと呼びます。
このモデルでは、システム部のみ手書きインクルードファイルとソースファイルが必要となる為 C:/usr/BP/Oven/system/user/skel の下にスケルトンファイルが生成されました。
以下のインクルードファイルは修正がない為、C:/usr/BP/Oven/system/user/include にそのまま移し変えてください。

4-1) COM_bridge.h

4-2) TIM_I_bridge.h

手書き必要な部分は添付のファイルとして用意しました。ここでは下表に従って、添付ファイルから入れ替えを行ってください。

添付ファイル上のデレクトリィ	変更となるファイル	ワークスペース上のデレクトリィ
BP_TOPPERS/Oven/system/user/include	rox_trance.h	C:/usr/BP/Oven/system/user/include
	PIO_bridge.h	
	sys_user_co.h	
BP_TOPPERS/Oven/system/user/source	COM_bridge.c	C:/usr/BP/Oven/system/user/source
	PIO_bridge.c	
	TIM_I_bridge.c	
	sys_user_co.c	

５）生成とコンパイルリンク

all コマンドで生成、コンパイル、リンクを行ってください。

```
$ make all
                    :
c:/usr/BP/Oven/SMP_ooa_timer/gen/object/T_dom_init.o
Your executable has been placed in c:/usr/BP/Oven/bin
SYSTEM BUILD COMPLETED – enjoy…
Make[2]: Leaving directory '/cygdrive/c/usr/BP/Oven'
Make[1]: Leaving directory '/cygdrive/c/usr/BP/Oven'
```

コンパイル・エラー等が発生した場合、手書きファイル等を修正してください。

1)PIO_JP_XX なし等のエラーが発生した場合、PIO_bridge.c 中のブリッジメソッドの日本語からの変換時、送付ファイルと異なった名前に変換された可能性があります。スケルトンの PIO_bridge.h を参考にして PIO_bridge.h と PIO_bridge.c を変更してください。

2)T_INT_T_J_XX なし等のエラーが発生した場合、T_INT_T_bridge のインターリーブブリッジの日本語名が、添付ファイルと異なった変換が行われた可能があります。sys_user_co.c 中のインターリーブブリッジの呼び出し部分を変更してください。

その他、当方で気づいたモデルコンパイラの注意事項を 2-3 節に記載します。

コンパイルとリンクのみ行いたい場合は、all_no_gen コマンドを行ってください。

```
$ make all_no_gen
```

６）実行

ＤＯＳ窓を開いて c:¥usr¥BP¥Oven¥bin にて rox.exe を実行してください

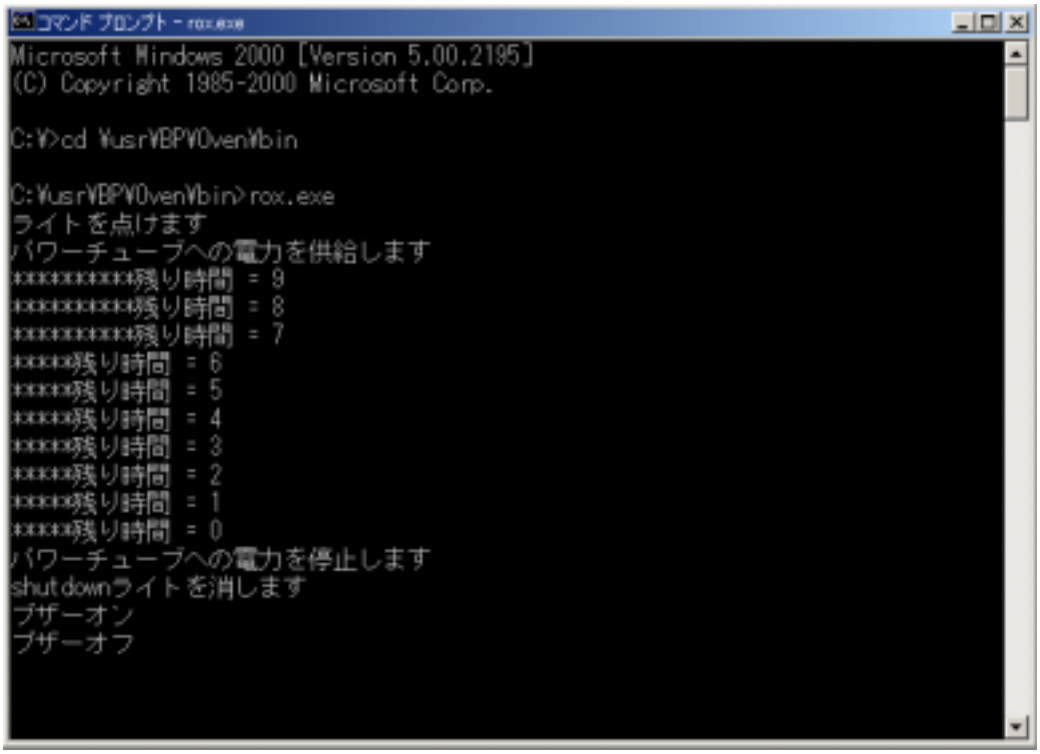


図 2-3.「オーブン」モデルの実行

2-2.カラーリング

モデルコンパイラ MC3020 の３つのバージョンを比較します。

東陽拡張版 V3.1.0 の TagOldStatePopulation は静的インスタンスの生成方法を東陽拡張版の仕様で行うかどうかの設定で、この例では定義しなければならない。object.clr の TagStaticPopulation は Project Technology 版では TagStaticInstancePopulation と TagReadOnly の２つに分けられている。TagTransformerGenerartion の指定がなくなり、クラスのメソッドは自動生成となりました。

アーキファイル	東陽拡張版 3.0 Rev.A	Project Technology V3.1.0	東陽拡張版 V3.1.0
domain.clr	なし	TagFunctionTranslationOff	TagFunctionTranslationOff
	なし	なし	TagUseOldStatePopulation
object.clr	TagStaticPopulation	なし	TagStaticPopulation
	なし	TagClassOprationTranslationOff	TagClassOprationTranslationOff
	なし	TagPEIsDefinedInData	TagPEIsDefinedInData
	なし	TagStaticInstancePopulation	TagStaticInstancePopulation
	なし	TagReadOnly	TagReadOnly
	TagTransformerGeneration	なし	なし

populate.arc	StaticInstancePopulationBody	アーキファイルなし	StaticInstancePopulationBody の定義に変更あり
system.clr	なし	TagCollectionFlavor	TagCollectionFlavor
	なし	TagModelDebuggingOn	TagModelDebuggingOn
	なし		(TagModifySystemPrefix)
	なし		(TagModifySystemMethodPrefix)
	なし		(TagModifySystemTypePrefix)
	なし		(TagModifySystemDefinePrefix)
	なし		(TagModifySystemFilePrefix)

表 2-2.バージョンによるアーキファイルの差異

2-3.生成時の注意事項

生成時問題となった項目をまとめておく、モデルコンパイラのリリース状況によっては解消されている可能性もあります。

1）生成時、libiconv.a というライブラリィがリンクされる。

C:/usr/mc3020/make 中の Makefile.root の 421 行目を削除。

```
420    mv $${target_file}.new $${target_file} ; ¥
421    echo "${CYGLIB}/libconv.a" >> $${target_file}
                          ↓
420    mv $${target_file}.new $${target_file} ;
```

2）カラーリングは設定したが、属性やメソッドに日本語を含むソースコードが生成される。

Windows の環境変数に以下の変数を追加してください。

ROX_MC_XLATE_STRINGS=TRUE

３.TOPPERS への対応

3-1.システム部の拡張方針

MC3020 はノンＯＳ用のモデルコンパイラである為、未処理の場合でも状態遷移用のパーサーは動作状態となります。そのため、TOPPERS のような RTOS 上で動作させる場合、パーサーに割り当てられたタスクが常に動作中の状態となり、実装上の問題となります。この対応では、パーサー用のタスクとタイマー用のタスク(タイマーブリッジを使用しない場合は不要)が動作し、２つのタスクとも状態遷移がない場合は、スリープ状態になるような拡張を行います。パーサー用のタスクをドメイン毎に割り当てる等の拡張を行えばより効率的な対応ができるが、ここではそこまでの拡張は行いません。

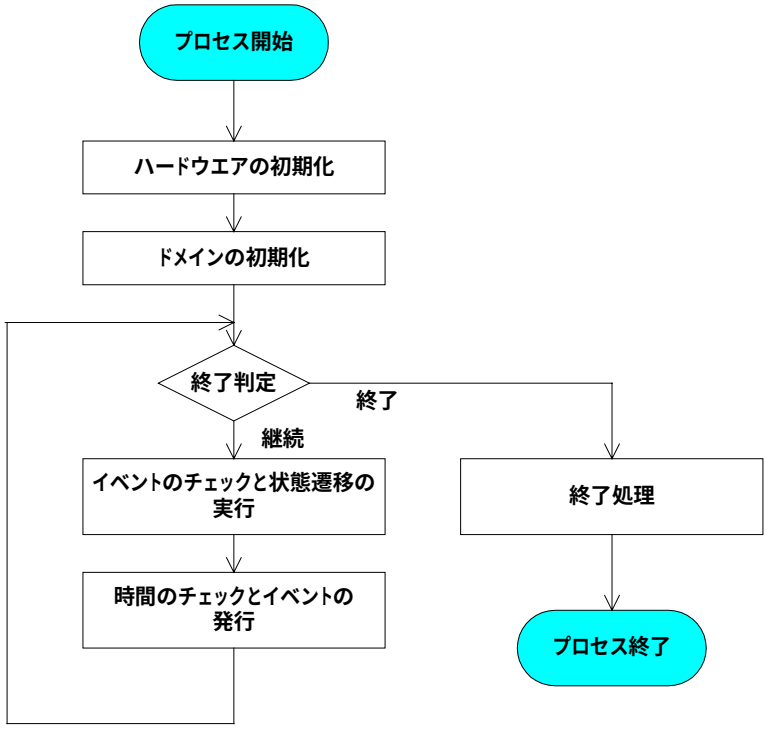


図 3-1.ノンＯＳの場合の実行形態

図 3-1 では終了判定から時間のチェックとイベント発行までの経路について実行中は常に動作状態となります。

図 3-2 が拡張を行ったシステム部の実行状態です。「パーサー開始」と「タイマー開始」におのおのタスクが割り当てられます。状態遷移によって、タイマーのイベントが必要となった場合は、時間イベント要求（タイマーブリッジ）によって、タイマータスクに時間イベントの設定を行います。設定時間の経過時、タイマータスクはイベントをパーサータスクに送りオブジェクトの状態遷移を促します。「オーブン」モデルは、タイマーブリッジを使用していないため、タイマータスクは不要ですが。TOPPERS 用のタイマーブリッジ（時間イベント要求とタイマータスク）は添付ファイルに置いておきます。

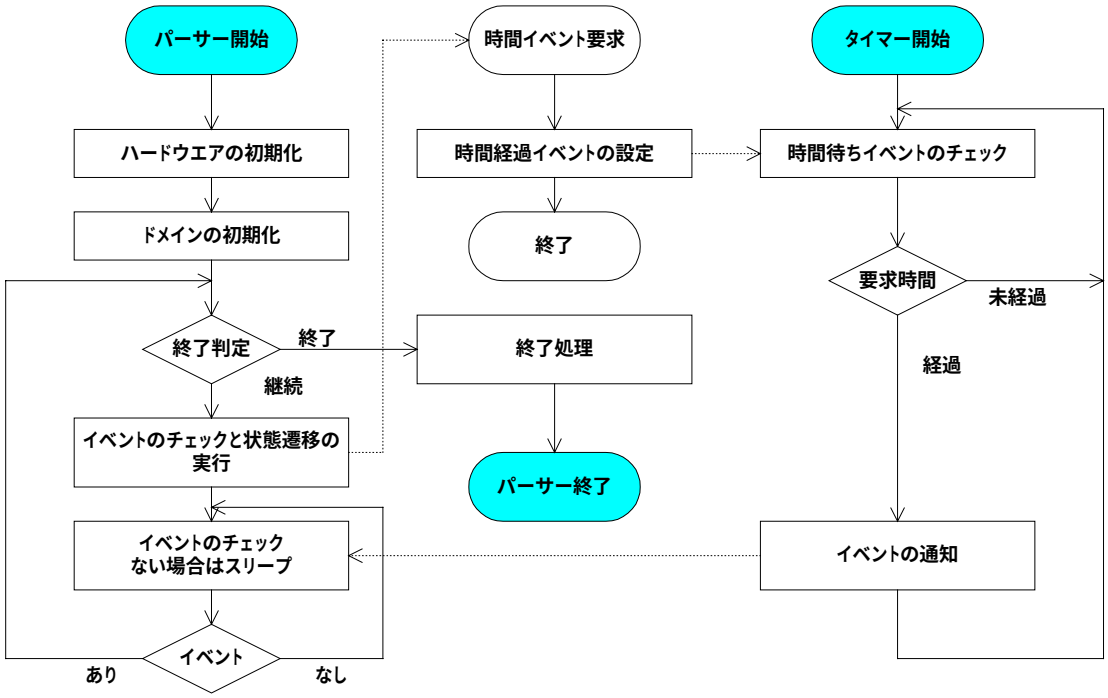


図 3-2.TOPPERS 対応のシステム部の実行状態

BridgePoint を図 3-2 の構成をとるようなシステム部のソースを生成する為には、アーキファイルとカラーリングの拡張が必要となります。

3-2.MC3020 の構成

MC3020 モデルコンパイラの構成を示すドキュメントは発行されていない。(少なくとも、本書の作成にあたり、そのような資料を参照していない) そのため、現状の構成から類推した部分もあり、正確な構成を記述していない部分があるかもしれないが、ご容赦いただきたい。

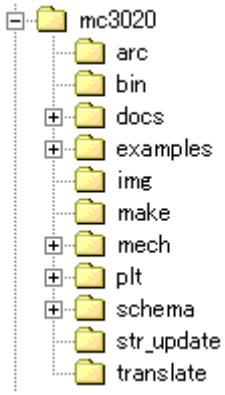


図 3-3.mc3020 のデレクトリィ構成

第１層	第２層	第３層	内容
mc3020	arc		アーキファイルの格納領域
	bin		実行形式ファイルの格納領域
	docs	faq	FAQ のドキュメント
		rn	コンパイラのリリースノート
		ug	ユーザーズ・ガイド
	examples	ae	ae のデモのサンプル・ファイル
		aslegos	aslogos のサンプル・ファイル
		autosample	
		pei	PEI のサンプル・ファイル
	img		
	make		Make ファイルのメイン部の格納領域
	mech	indep	マシン非依存部のインクルードファイル、ソースファイル
		windows	Cygwin 用の ksh の構築環境
	plt	dtd	
		shellscript	
	schema	colors	オリジナルのカラーファイルの格納領域
		keydir	BridgePoint の実行キーの領域
		sql	
	str_update		ユーティリティデレクトリィ
	translate		拡張キャラクタの変換用デレクトリィ

表 3-1.mc3020 のデレクトリィ構成

TOPPERS 用の変更を行うソースファイルは、C:/usr/BP/Oven/system/gen 中の sys_init.c と sys_evpool.c です。sys_init.c パーサー部であり、sys_evpool.c はイベントの管理を行います。しかし、これらのソースファイルはモデルコンパイラの生成物であり、これらはソースコードの生成時、アーキファイルの記述に従って生成されます。従って、以下の２点の拡張が必要となります。

１）RTOS の対応の必要がない場合、ノンＯＳ用の生成ができ、TOPPERS 以外の RTOS でも対応ができるような設定が必要となる。これを解決する為にカラーファイルである sys_functions.arc を拡張し、カラーリングの設定に従って適切なソースコードが生成されるように拡張を行う。

２）カラーリングの設定が生成に反映されるように、mc3020/arc 中のアーキファイルの拡張を行う。アーキファイルとソースファイルの対応は以下の通りとなる。

・system.arc　→　sys_init.c

・sys_evpool.arc　→　sys_evpool.c

3-3.カラーリングの拡張

sys_functions.arc の拡張項目を以下に示す。このカラーファイルを mc3020/schema/colors に入れておくことによって、rox_init_node を用いたワークスペースの生成時、このカラーファイルがワークスペースにセットされます。カラーリングによって RTOS 依存のコードが生成される場合、RTOS 用のインクルードファイルの参照が必要となります。RTOS 用のインクルードの追加は sys_user_co.h にて行うようにしておく、sys_user_co.h は対象となる２つのソースファイルは参照を行っている。

	カラー設定	内容
1	GetMainTaskSleepRequest	イベントがなければパーサーをスリープ状態に移行する。
2	EnterEventMutex	イベント処理の排他制御のロックを行う
3	LeaveEventMutex	イベント処理の排他制御のロック解除を行う
4	WakeUpMainDispatcher	スリープ中のパーサーを起床する。
5	WakeUpMainDispatcherInterleave	WakeUpMainDispatcher と同等であるが、インターリーブブリッジ中で発行されるので、割り込み状態を考慮しなければならない。

表 3-2.TOPPERS 用の追加カラー設定

２章で使用した sys_finctions.arc から TOPPERS 用の変更は以下の通りとなります。

カラー設定	デフォルト設定	TOPPERS 用設定
GetMainTaskEntryDeclaration	Int main(int argc, char * argv[])	void entry_task(VP_INT exinf)

GetMainTaskSleepRequest	/* no RTOS function */	if(wai_sem(SEM_MAIN) < E_OK){
		syslog_0(LOG_NOTICE, “Main Dispatcher Semaphore WaitError!”);
		}

GetMainTaskEntryReturn	return 0;	/* not return value */
EnterEventMutex	.// .assign attr_result = “/*entry event mutex */”	.// .assign attr_result = “wai_sem(SEM_EVENT); /* entry event mutex */”
LeaveEventMutex	.// .assign attr_result = “/* leave event mutex */”	.// .assign attr_result = “sig_sem(SEM_EVENT); /* leave event mutex */”
WakeUpMainDispatcher	.//	/* wake up main process */

	.assign attr_result = "/* wakeup main event dispatcher */"	{
		ER ercd;
		ercd = sig_sem(SEM_MAIN);
		if(ercd != E_OK && ercd != E_QOVER){
		syslog_1(LOG_NOTICE, "Wakeup Semaphore Error! Error id = %d", ercd);
		}
		}

WakeUpMainDispatcherInterleave	.//
	.assign attr_result = "/* wakeup main event dispatcher */"
		/* wake up main process */
		{
		ER ercd;
		ercd = sig_sem(SEM_MAIN);
		if(ercd != E_OK && ercd != E_QOVER){
		syslog_1(LOG_NOTICE, "Wakeup Semaphore Error! Error id = %d", ercd);
		}
		}

表 3-3.sys_functions.arc の設定比較

3-4.アーキファイルの拡張

システム部のソースファイルを生成するアーキファイルは以下の２つを変更しました。

１）system.arc

system.arc はシステム・ソース sys_init.c を生成するアーキファイルです。sys_init.c は、main process を持ち、ドメインのオブジェクトの状態遷移とインターリーブブリッジを行う。system.arc は以下の部分を改造しました。

1-1)カラーリング項目:GetMainTaskSleepRequest に対応して main process 中でタスク・スリープ状態に移行できるように修正しました。

1-2)カラーリング項目:WakeUpMainDispatcher に対応して、パーサー中のイベント付け替え処理でも実行状態に移行できるように起床機能に対応しました。

1-3)カラーリング項目:WakeUpMainDispatcherInterleave の対応して、インターリーブブリッジ処理後の main process の起床処理に対応しました。

２）sys_evpool.arc

sys_evpool.arcはイベント処理郡のソース sys_evpool.cを生成するアーキファイルである。このファイルの変更は以下の２つです。

2-1)カラーリング項目:EntryEventMutex と LeaveEventMutex に対応して、各イベントファンクションがマルチタスクで呼ばれても誤動作しないように Mutex によるロックを行う。Mutex には Entry と Leave があり、Entry はロック状態への移行、Leave はロックの解除を実行する。

2-2)カラーリング項目:WakeUpMainDispatcher にて対応して、Self と NonSelf のイベント送信時に main process の起床を行う。

詳細は添付ファイルを参照して頂きたい。

3-5.タイマーブリッジの変更

「オーブン」モデルではタイマーブリッジを使用していないが、タイマーブリッジについてもタスク処理となるように修正しました。通常タイマーブリッジを使用する場合、ノンＯＳ版では sys_user_co.c 中の UserInitializationCallOut()にて TIM_init()関数を使用してタイマーブリッジの初期化を行います。UserBackgroundProcessingCallOut()にて TIM_tick()関数を使用してタイマーイベントのチェックを行っています。ＲＴＯＳ版では TIM_init()関数にてタスクの起動を行います。TIM_tick()は未処理となる。作成したタイマーブリッジは以下の機能をサポートしています。

	関数	引数	処理内容	対応
1	TIM_timer_start	1:ee_event_inst 2:ee_microsecond	microsecond 後に、ee_event_inst が発行されるようにワンショットタイマーを起動する。戻り値はタイマーの管理テーブルへのポインタ(Escher_Timer_t *)	○
2	TIM_timer_start_recurring	1.ee_event_inst 2.ee_microsecons	microsecond 後に、ee_event_inst が発行されるように周期タイマーを起動する。戻り値はタイマーの管理テーブルへのポインタ(Esher_Timer_t *)	○
3	TIM_timer_remaining_time	1.ee_timer_inst_ref	タイマーの管理テーブルへのポインタを引数にして対応のタイマーの到達時間を microsecond で返す。	○
4	TIM_timer_reset_time	1.ee_microsecond 2.ee_timer_inst_ref	設定タイマーの到達時間を変更する。成功すれば true、失敗すれば false を返す。	○
5	TIM_timer_add_time	1.ee_microsecond 2.ee_timer_inst_ref	設定タイマーの到達時間を延長する。成功すれば true、失敗すれば false を返す。	○
6	TIM_timer_cancel	1.ee_timer_inst_ref	設定タイマーをキャンセルする。成功すれば true、失敗すれば false を返す。	○
7	TIM_current_date	void	現在の日付を Escher_Date_t タイプで返す。	×
8	TIM_create_date	1.ee_day 2:ee_hour 3:ee_minute	引数の日付でカレンダー時間を修正する。戻り値は Esher_Date_t の設定値。	×

		4:ee_month 5:ee_second 6:ee_year		
9	TIM_get_second	1:ee_date	ee_date(Esher_Date_t)から秒を取り出す。	×
10	TIM_get_minute	1:ee_date	ee_date(Esher_Date_t)から分を取り出す。	×
11	TIM_get_hour	1:ee_date	ee_date(Esher_Date_t)から時を取り出す。	×
12	TIM_get_day	1:ee_date	ee_date(Esher_Date_t)から日を取り出す。	×
13	TIM_get_month	1:ee_date	ee_date(Esher_Date_t)から月を取り出す。	×
14	TIM_get_year	1:ee_date	ee_date(Esher_Date_t)から年を取り出す。	×
15	TIM_current_clock	Void	現在クロックを取り出す。Millisecond 値	○
16	TIM_timer_start_stub	1.param 2.(*func)(int) 3.ee_microsecond	シミュレーション用の拡張 microsecond 後に、param を引数とした func()を実行する。戻り値はタイマーの管理テーブルへのポインタ (Esher_Timer_t *)	○

表 3-4.タイマーブリッジの関数

１６はシミュレーションを行う為に、外部のイベントを発行する為の拡張機能で、本来のタイマーブリッジには含まれません。

3-6.TRACE 用標準出力の変更

カラーリング設定でトレース等のログ出力を行う場合、現状の C:/usr/BP/Oven/system/user/include 中の rox_trace.h の設定では printf()文を用いた標準出力に出力されます。ITRON4.0 では標準出力はサポートしていないので、対応として TOPPERS で定義している関数 syslog_printf()または、syslog_n()（syslog_n の n は数字です）に対応してログアウトするように修正してください。この修正は rox_trace.h に対して行います。

４.TOPPERS 環境の対応

4-1.TOPPERS 対応ソースコードの生成

前章で生成した「オーブン」モデルを、TOPPERS の Windows 版で動作できるように、再度生成しなおしてみます。このとき、make all で生成したのでは、TOPPERS のカーネル部を組み込めませんし、VC++のデバック機能を使用して実行形式のデバックもできません。そこで、make gen_all でソースコードのみ生成し、TOPPERS の Windows 版のワークスペースから、生成されたソースコードを取り込んで取り込んで実行を行うようにコンパイルとリンクを行います。

TOPPERS 側でも BridgePoint で生成されたソースコードを取り込む為に、コンフィギュファイルやソースファイルを作成しなければなりませんが、その説明は次節に置き生成手順を説明します。

１）アーキファイルとソースファイルの設定

まず、TOPPERS の Windows 版 jsp を「オーブン」のワークスペースにコピーで移し変えます。

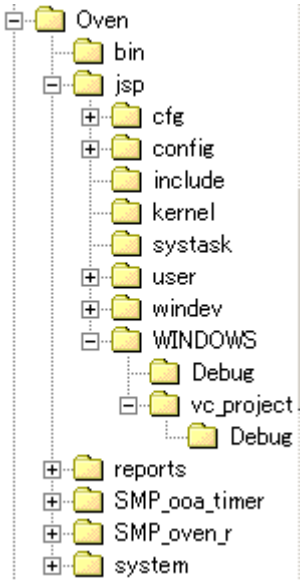


図 4-1.「オーブン」ワークスペースへ TOPPERS ワークスペースの取り込み

添付ファイルから「オーブン」のワークスペースと mc3020 へ変更ファイルをコピーにて移し変えます。MC3020 の置き換えファイルについては、元に戻せるように名称を変えて保存しておきましょう。（例：system.arc→system.arc.org）以下に移し変えとなるファイルの一覧を示します。

添付ファイル上のデレクトリィ	変更となるファイル	対象のデレクトリィ
BP_TOPPERS/mc3020/arc	sys_evpool.arc	C:/usr/mc3020/arc
	system.arc	

BP_TOPPERS/mc3020/mech	itron（デレクトリィごとコピーしてください）	C:/usr/mc3020/mech
BP_TOPPERS/mc3020/schema/colors	sys_functions.arc	C:/usr/mc3020/schema/colors
BP_TOPPERS/Oven/system/color	sys_functions.arc.toppers を sys_functions.arc に書き換えてコピー	C:/usr/BP/Oven/system/color

表 4-1.TOPPERS の為の書き換え

C:/usr/mc3020/schma/colors 中のカラーファイルはワークスペースの作成時にシステムのカラー領域に反映されます。sys_functions.arc.toppers と sys_user_co.h.toppers には TOPPERS 用の設定が行われています。

２）Cygwin の窓から、前に生成したシステムとドメインのソースコードをクリアします。その後、SMP_oven_r と SMP_ooa_timer のブリッジのスケルトンを生成してください。

```
$ make clean_sys
make[1]: Entering directory '/cugdrive/c/usr/BP/Oven'
make[2]: Entering directory '/cugdrive/c/usr/BP/Oven/system'
make[2]: Leaving directory '/cugdrive/c/usr/BP/Oven/system'
make[1]: Leaving directory '/cugdrive/c/usr/BP/Oven'

$ make clean_all_dom
make[1]: Entering directory '/cugdrive/c/usr/BP/Oven'
make[2]: Entering directory '/cugdrive/c/usr/BP/Oven'
make[3]: Entering directory '/cugdrive/c/usr/BP/Oven/SMP_oven_r'
make[4]: Entering directory '/cugdrive/c/usr/BP/Oven/SMP_oven_r'
make[4]: Leaving directory '/cugdrive/c/usr/BP/Oven/SMP_oven_r'
make[3]: Leaving directory '/cugdrive/c/usr/BP/Oven/SMP_oven_r'
make[2]: Leaving directory '/cugdrive/c/usr/BP/Oven'
make[2]: Entering directory '/cugdrive/c/usr/BP/Oven'
make[3]: Entering directory '/cugdrive/c/usr/BP/Oven/SMP_ooa_timer'
make[4]: Entering directory '/cugdrive/c/usr/BP/Oven/SMP_ooa_timer'
make[4]: Leaving directory '/cugdrive/c/usr/BP/Oven/SMP_ooa_timer'
make[3]: Leaving directory '/cugdrive/c/usr/BP/Oven/SMP_ooa_timer'
make[2]: Leaving directory '/cugdrive/c/usr/BP/Oven'
make[1]: Leaving directory '/cugdrive/c/usr/BP/Oven'
```

```
$ make bridge_skel domain=SMP_oven_r
make[1]: Entering directory ‘cygdrive/c/usr/BP/Oven’
make[2]: Entering directory ‘cygdrive/c/usr/BP/Oven/SMP_oven_r’
make[3]: Entering directory ‘cygdrive/c/usr/BP/Oven/SMP_oven_r’
Regenerating database schma/gen/SMP_oven_r.gen for domain ‘SMP_oven_r’...
                                        :
make[2]: Leaving directory ‘cygdrive/c/usr/BP/Oven/SMP_oven_r’
make[1]: Leaving directory ‘cygdrive/c/usr/BP/Oven’

$ make bridge_skel domain=SMP_ooa_timer
make[1]: Entering directory ‘cygdrive/c/usr/BP/Oven’
make[2]: Entering directory ‘cygdrive/c/usr/BP/Oven/SMP_ooa_timer’
make[3]: Entering directory ‘cygdrive/c/usr/BP/Oven/SMP_ooa_timer’
Regenerating database schma/gen/SMP_oven_r.gen for domain ‘SMP_ooa_timer’...
                                        :
make[2]: Leaving directory ‘cygdrive/c/usr/BP/Oven/SMP_ooa_timer’
make[1]: Leaving directory ‘cygdrive/c/usr/BP/Oven’

$
```

ソースコードの生成のみを行います。

```
$ make gen_all
make[1]: Entering directory ‘cygdrive/c/usr/BP/Oven’
                                        :
c:/usr/mc3020/arc/sys_make.arc: 214: INFO: File ‘../system/gen/source/Makefile’
CREATE.
Make[4]: Leaving directory ‘cygdrive/c/usr/BP/Oven/system’
Make[3]: Leaving directory ‘cygdrive/c/usr/BP/Oven/system’
Make[2]: Leaving directory ‘cygdrive/c/usr/BP/Oven’
Make[1]: Leaving directory ‘cygdrive/c/usr/BP/Oven’
```

これで TOPPERS 用のソースコードが生成されました。

4-2.TOPPERS 側での対応

ここでは、JSP-1.4 を対象として記述を行います。MC3020 により生成されたソースコー

ドと手書きのコード、TOPPERS のカーネル、加えて、この２種類のソースコードを結合する為の RTOS リソースの設定が必要です。この設定が sample.cfg に設定が行われています。また、sample.h にタスクの優先度やスタックサイズの定義を行っています。sample.c はハンドラー等の定義であり、TOPPERS に依存したものではありません。この３つのファイルは BP_TOPPERS/Oven/jsp/tools/WINDOWS の下にありますので、c:/usr/BP/Oven/jsp/tools/WINDOWS にコピーしてください。sample.cfg で定義する RTOS のリソースについて下表に示します。他の定義は BridgePoint のシステムには使用しません。

	ラベル	種類	内容
1	ENTRY_TASK	タスク	パーサー用のタスク
2	TIMER_TASK	タスク	タイマー用のタスク、このモデルではタイマーブリッジではない。
3	SEM_MAIN	セマフォ	パーサースリープ用のセマフォ
4	SEM_EVENT	セマフォ	イベント処理の MUTEX 用に使用するセマフォ
5	SEM_TIME	セマフォ	タイマー処理の MUTEX 用に使用するセマフォ

表 4-2.BridgePoint 実行用の RTOS リソース

4-3.jsp でのコンパイルとリンク

C:/usr/BP/Oven/jsp/WINDOWS/vc_project 中の TOPPERS のワークスペース(toppers.dsw)を立ち上げます。sample1 の設定を「オーブン」モデルの設定に変更しなければなりません。以下に TOPPERS ワークスペースに対する置き換えまたは追加の表を示します。

	対応	元のソース	対応のソース	格納場所
1	置き換え	sample1.cfg	sample.cfg	C:/usr/BP/Oven/jsp/WINDOWS
2		sample1.h	sample.h	
3		sample1.c	sample.c	
4	追加		sys_init.c	C:/usr/BP/Oven/system/gen/source
5			sys_sets.c	
6			sys_evpool.c	
7			init_seq.c	
8			COM_bridge.c	C:/usr/BP/Oven/system/user/source
9			PIO_bridge.c	
10			TIM_I_bridge.c	

11			sys_user_co.c	
12			T_dom_init.c	C:/usr/BP/Oven/SMP_ooa_timer/gen/source
13			T_BR_bridge.c	
14			T_INT_T_bridge.c	
15			T_OTIM_object.c	
16			T_sync_services.c	
17			T_TIC_object.c	
18			O_B_action.c	C:/usr/BP/Oven/SMP_oven_r/gen/source
19			O_B_event.c	
20			O_B_object.c	
21			O_INT_T_action.c	
22			O_INT_T_object.c	
23			O_L_action.c	
24			O_L_event.c	
25			O_L_object.c	
26			O_M_action.c	
27			O_M_event.c	
28			O_M_object.c	
29			O_R_SP_object.c	
30			O_RP_SP_object.c	
31			O_ST_object.c	
32			O_T_action.c	
33			O_T_event.c	
34			O_T_object.c	
35			O_TC_action.c	
36			O_TC_event.c	
37			O_TC_object.c	
38			O_TC_xformGen.c	
39			O_TR_action.c	
40			O_TR_event.c	
41			O_TR_object.c	
42			O_TR_xformGen.c	
43			O_V_action.c	
44			O_V_event.c	
45			O_V_object.c	

46			O_W_action.c	
47			O_W_event.c	
48			O_W_object.c	
49			O_dom_init	

表 4-3.TOPPERS ワークスペースへの対応

次に、インクルードの参照設定を行ってください。インクルードパスは VC++のメニューからプロジェクト→設定(S)→C/C++を選択後、カテゴリィ(Y)をプリプロセッサとしてインクルードファイルパス(N)にパスを追加します。追加先は以下に示します。

	インクルードパス	内容
1	../../../../system/user/include	ユーザ記述システムインクルードファイル
2	../../../../system/gen/include	自動生成システムインクルードファイル
3	../../../../SMP_oven_r/gen/include	自動生成オーブンドメインインクルードファイル
4	../../../../SMP_ooa_timer/gen/include	自動生成タイマードメインインクルードファイル

表 4-4.インクルードパスの追加

プリプロセッサの定義に TOPPERS を追加してください。

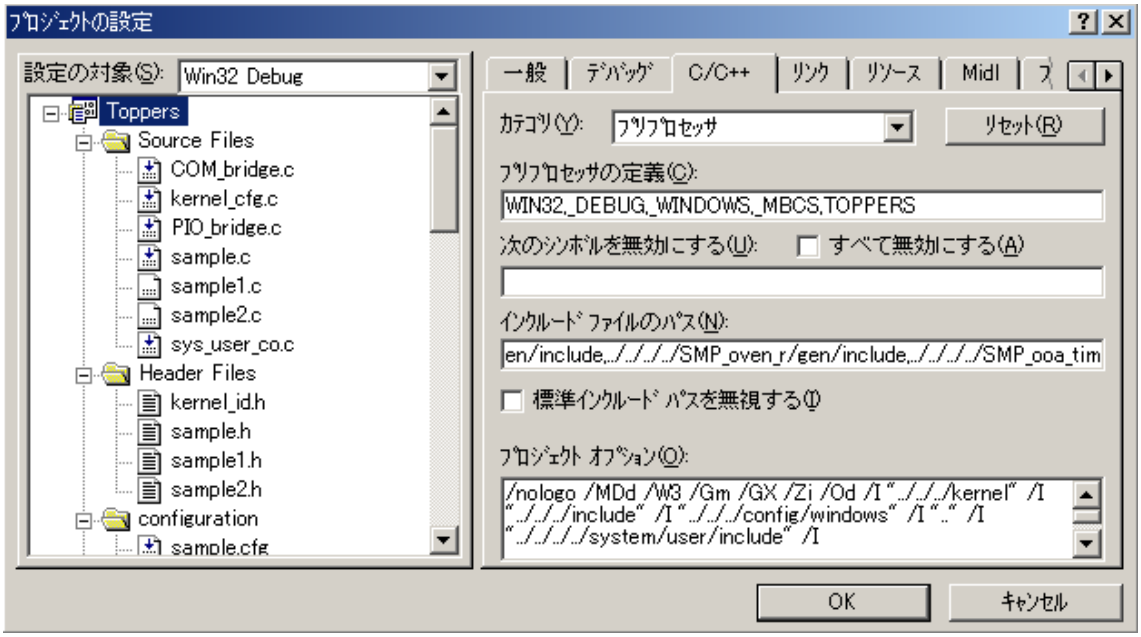


図 4-2.インクルードパスと定義の追加

ここまで終われば、リビルド（R）にて、コンパイル、リンクしてください。

4-4.実行結果

リビルド後、デバックまたは実行を行うと、下図の通り TOPPERS 上でモデルが実行されます。

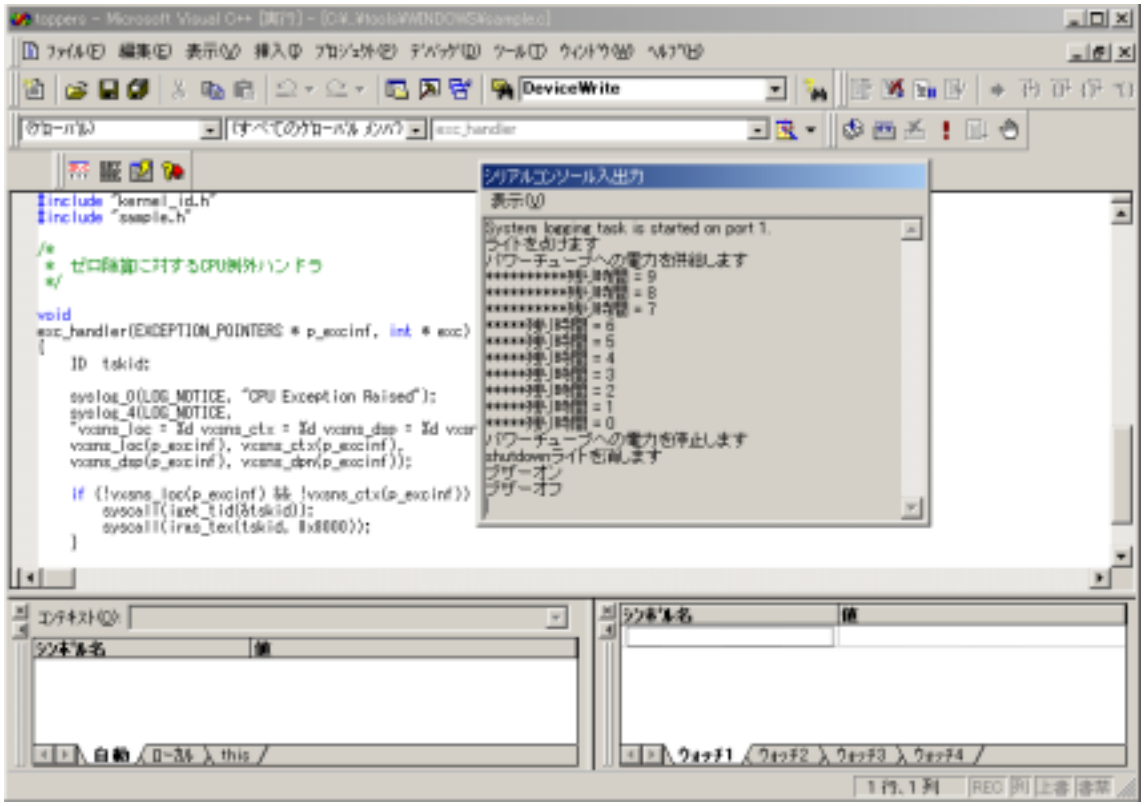


図 4-3.TOPPERS/JSP-1.4 上のモデルの実行

５.タイマーブリッジの対応

この実装では、せっかく作成したタイマーブリッジ(TIM_bridge.c)を使用していません。SMP_ooa_timer ドメインをタイマーブリッジに置き換えればなお、簡単にタイマー機能を実現できます。Oven2 内に SMP_ooa_timer を使用しないオーブン SMP_oven_r2 をバックファイルとビルドに必要なソースをのせました、こちらの方もチャレンジしてみてください。

下図のように、Oven2 では SMP_ooa_timer ドメインを使用しません。そのため、ワークスペースにドメインとして取り込む必要はありません。SMP_ooa_timer ドメインの機能は TIM_bridge.c にて全て対応します。

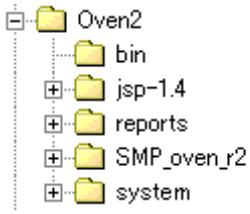


図 5-1.Oven2 のワークスペース

６.添付ファイルのレイアウト

添付ファイルのレイアウトは以下の通りです。mc3020 以下のファイルは(株)東陽テクニカ版 MC3020 Version3.1.0（９月版）をベースに変更を行ったファイル郡です。Oven/jsp 以下は TOPPERS より BridgePoint のシステム部と連結するためのファイル郡です。Oven の下に「オーブン」モデルの２つのバックファイルを入れてあります。これは BridgePoint5.1 で(株)東陽テクニカが配布したものと同等のものです。
Oven2 の jsp 以下は Oven と同等なので添付しません。

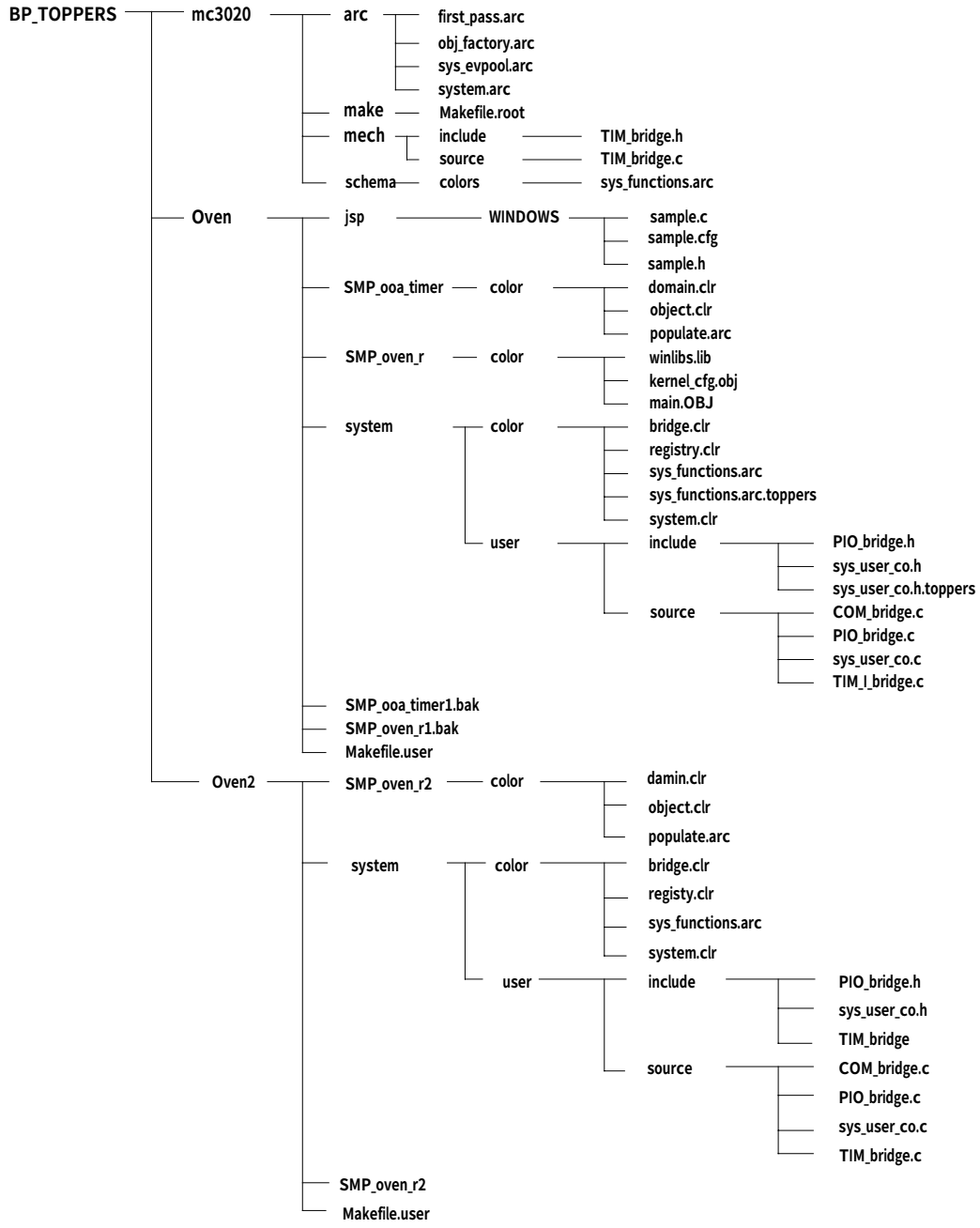


図 6-1.添付ファイルのレイアウト

７.参考文献

1) TOPPERS

http:/www.ertl.jp/TOPPERS

2)BridgePoint

ソフトウェア・アーキテクチャ MC-3020.TOYO ユーザーズガイド
バージョン 2.0 用
発行　株式会社東陽テクニカ　ソストウェア・ソリューション

MC-3020 Model Compiler User's Guide
発行　Rox Software, inc.

７．お礼

この資料の公開にあたり、以下の方にご協力を頂きました。この場を借りてお礼を申し上げたいと思います。

Special Thanks
PROJECT TECNOLOGY inc. President & CEO
John R. Wolfe

（株）東陽テクニカ
二上貴夫さん

この開発にあたり、以下の方々にご指導と助言を頂きました。この場を借りてお礼を申し上げたいと思います。

（株）東陽テクニカ　ソフトウェア・ソリューション
鈴木俊安さん
富士ソフトＡＢＣ（株）システム事業本部
濱田茂雄さん