Magic eDeveloper V10
レコードメインからイベントへの変換規則

Magic Software Japan K.K.

Enabling Business with Superior Technology

初版　2007 年 11 月 19 日

マジックソフトウェア・ジャパン株式会社

目次

第 1 章はじめに 5
1.1 本書の目的 5
1.2 前提条件 5
第 2 章セグメントの決定 6
2.1 ロジック 6
2.2 コントロール項目 6
2.3 セグメント 7
第 3 章基本変換規則 8
3.1 セグメント内のロジックの変換 8
3.1.1 レコードメインの動作 8
3.1.2 変換規則 8
3.2 最初のコントロールに先行する処理 10
3.3 パーク制御 10
第 4 章エラーコマンド 11
4.1 レコードメインでの動き 11
4.2 変換規則 11
第 5 章前置および後置 13
5.1 概要 13
5.2 ユーザ定義イベントの利用 13
5.3 前置の場合 13
5.3.1 前置ズームセグメントの決定 13
5.3.2 前ブロックのフロー方向特性 14
5.3.3 前置ズームセグメントの変換規則 14
5.3.4 動作の違い 15
5.4 後置の場合 15
5.5 前置/後置ズームセグメント内の[エラー]処理コマンド 15
5.5.1 未解決のズーム 16
第 6 章ブロックコマンド 17
6.1 ケース1 17
6.1.1 変換規則 17
6.1.2 逆の順序 18
6.1.3 エラー処理コマンド 18
6.1.4 ブロック Else 18
6.2 ケース2 18
6.2.1 変換規則 19
6.3 ケース3 19
6.3.1 変換規則 20
6.3.2 条件の拡散 21
6.3.3 グローバル変数の利用 22
6.3.4 ブロックの条件式の変換 23
6.3.5 コントロールのパーク条件の設定 23
6.4 ケース4 23
6.4.1 レコードメインでの動作 23
6.4.2 グローバル変数の利用 24

6.4.3 変換規則 24
6.4.4 前置と後置ブロックに条件が付いている場合 25
6.4.5 前置と後置の違い 25
第 7 章その他の話題 26
7.1LEVEL 関数 26
7.2 コントロール名の欠落 26
7.3 コメント行 26
7.4 複数フォーム 26
7.5 既存のコントロールレベルのハンドラ 26
7.6 バッチとブラウザタスクのレコードメイン処理 26
7.7 アクセス不能な処理コマンドの扱い 26

第1章　はじめに

1.1　本書の目的

Magic eDeveloper V10 では、レコードメイン（RM）のロジックは非推奨となりました。

旧バージョンから V10 に移行した時、レコードメイン中のロジックは[RM 互換]というハンドラに移行されます。このロジックは、イベントハンドラ（ロジックユニット）を使って、イベント志向のプログラミング方法に書き直すことが推奨されています。

レコードメインからイベントへの変換は、ドキュメント[Magic eDeveloper V10 レコードメインからイベントへ]で基本的なことが解説されていますが、このドキュメントは簡単のために、詳細な変換規則や複雑なケースについての説明を省略していました。

本書では、レコードメインのロジックをなるべく正確にイベント指向プログラミングで再現させるために、より複雑なケースも含めて、詳細な変換規則について説明します。

本書では、レコードメインで記述していたものを、V10 でもできるだけ同じ動作をすることを目的としています。そのため、変換規則はあらゆる可能性をできるだけ網羅することを主眼としているので、詳細かつ機械的な規則となってしまいます。

実際にレコードメインからイベント指向で書きなおしをする場合には、本書の規則を機械的に適用すると、意図していた仕様には不要なロジックが数多く出てくる可能性があります。

従って、本書の内容は書き直し作業の参考とするに止め、[元来何をしたいのか？]に立ち帰って、イベント指向的な発想に立って設計しなおす方が、より直感的で分かりやすいプログラムを作ることができる場合が多いと思います。

プログラムの組み方によっては、本書でカバーしきれなかったケースも出てくる可能性があります。本書の規則が 100%のケースをカバーしているとは限らないことも予めご承知願います。

1.2　前提条件

読者は次の知識を基本的に理解していることを前提としています。

- 旧バージョンでのレコードメインの動作
- V10 のタスクエディタの操作法とイベントハンドラの動作

第2章　セグメントの決定

まず最初になすべきことは、レコードメインのロジックを[セグメント]に分割することです。

セグメントについて説明する前に、本書でよく使う言葉についていくつか定義します。

2.1　ロジック

本書では[ロジック]という言葉を[手続き型の Magic の処理コマンド]という意味で使います。例えば、項目更新、エラー、アクション、コール、外部コール、フォーム（V9Plus でのデータ入力/データ出力）などのコマンドがこれに該当します。これに対し、データ定義を行う処理コマンド（セレクトコマンド、リンクコマンド、リンク終了コマンド）はロジックとは呼びません。

2.2　コントロール項目

[コントロール]とは、フォーム上に配置された表示用のオブジェクト（エディットコントロール、スタティックコントロール、コンボボックス等）を指します。

本書では[コントロール項目]という言葉を、次のような条件をすべて満たす[セレクト]処理コマンドという意味で使います。

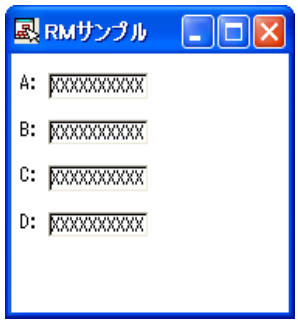
- レコードメインの中で使われている。
- フォーム上のコントロールにデータを表示するよう設定されている。
- パークする可能性があるもの(条件付きでもよい)

例えば、次のようなレコードメインがあるとします。

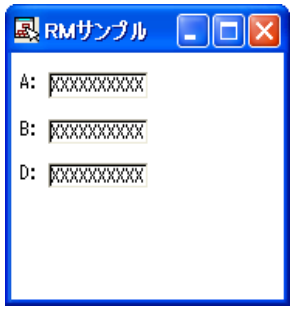
処理テーブル: レコード メイン

#	処理コマンド			内容			範囲		位置付		フロー		条件
1	ｾﾚｸﾄ	V=変数 :	1	A	代入:	0	0	0	0	0	S	C	Yes
2	ｾﾚｸﾄ	V=変数 :	2	B	代入:	0	0	0	0	0	S	C	Yes
3	ｴﾗｰ		0	Err1	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			C	C	Yes
4	ｾﾚｸﾄ	V=変数 :	3	C	代入:	0	0	0	0	0	S	C	Yes
5	ｴﾗｰ		0	Err2	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			C	C	Yes
6	ｴﾗｰ		0	Err3	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			C	C	Yes
7	ｾﾚｸﾄ	V=変数 :	4	D	代入:	0	0	0	0	0	S	C	Yes

このとき、右図のように、変数 A、B、C、D いずれもが画面に表示されていれば、これらの[セレクト]コマンドはコントロール項目です。



もし、右図のように、変数 C が画面に表示されていなければ、この[セレクト]コマンド(4 行目）はコントロール項目ではありません。



また、変数 C が画面に表示されていても、もし[セレクト]コマンドの[条件]が No になっていたら、パークしない項目なので、これもコントロール項目ではありません。

処理テーブル: レコード メイン

#	処理コマンド			内容			範囲		位置付		フロー		条件
1	ｾﾚｸﾄ	V=変数 :	1	A	代入:	0	0	0	0	0	S	C	Yes
2	ｾﾚｸﾄ	V=変数 :	2	B	代入:	0	0	0	0	0	S	C	Yes
3	ｴﾗｰ		0	Err1	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			C	C	Yes
4	ｾﾚｸﾄ	V=変数 :	3	C	代入:	0	0	0	0	0	S	C	No
5	ｴﾗｰ		0	Err2	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			C	C	Yes
6	ｴﾗｰ		0	Err3	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			C	C	Yes
7	ｾﾚｸﾄ	V=変数 :	4	D	代入:	0	0	0	0	0	S	C	Yes

本書の主題である[レコードメインのロジックをイベント指向で書きなおす]という点から見ると、画面に表示されないデータ項目やパークしないデータ項目は考慮の対象外となります。従って、そういうデータ項目を定義している[セレクト]コマンドは無視することができます。

これ以降、特に断らない限り、例に出てくる[セレクト] コマンドは、すべてコントロール項目であると考えてください。

また、コントロールレベルのハンドラを定義するときには、フォーム上のコントロール名を参照します。本書では簡単のために、コントロール名は変数名と同一に設定されているものとします。

2.3 セグメント

[セグメント]とは、レコードメイン中で、あるコントロール項目から次のコントロール項目の間に定義された一連のロジック(処理コマンド)を言います。

例えば、次の図の例では、次の二つのセグメントがあります。

- コントロール B と C にはさまれたセグメント (3 行目のエラーコマンド)
- コントロール C と D にはさまれたセグメント (5～6 行目の二つのエラーコマンド)

処理テーブル: レコード メイン

#	処理コマンド			内容			範囲		位置付		フロー		条件
1	ｾﾚｸﾄ	V=変数 :	1	A	代入:	0	0	0	0	0	S	C	Yes
2	ｾﾚｸﾄ	V=変数 :	2	B	代入:	0	0	0	0	0	S	C	Yes
3	ｴﾗｰ		0	Err1	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			C	C	Yes
4	ｾﾚｸﾄ	V=変数 :	3	C	代入:	0	0	0	0	0	S	C	Yes
5	ｴﾗｰ		0	Err2	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			C	C	Yes
6	ｴﾗｰ		0	Err3	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			C	C	Yes
7	ｾﾚｸﾄ	V=変数 :	4	D	代入:	0	0	0	0	0	S	C	Yes

もし、4 行目の[セレクト] コマンドの[条件] 欄が No であると、これはコントロール項目ではないので、コントロール項目 B と D とにはさまれたブロック (3 行目、および 5～6 行目の 3 つのエラーコマンド)が一つだけある、ということになります。

処理テーブル: レコード メイン

#	処理コマンド			内容			範囲		位置付		フロー		条件
1	ｾﾚｸﾄ	V=変数 :	1	A	代入:	0	0	0	0	0	S	C	Yes
2	ｾﾚｸﾄ	V=変数 :	2	B	代入:	0	0	0	0	0	S	C	Yes
3	ｴﾗｰ		0	Err1	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			C	C	Yes
4	ｾﾚｸﾄ	V=変数 :	3	C	代入:	0	0	0	0	0	S	C	No
5	ｴﾗｰ		0	Err2	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			C	C	Yes
6	ｴﾗｰ		0	Err3	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			C	C	Yes
7	ｾﾚｸﾄ	V=変数 :	4	D	代入:	0	0	0	0	0	S	C	Yes

第3章　基本変換規則

まず、最も単純なケースについての基本変換規則を説明します。

より複雑なケースについては、次章以下で説明します。

3.1　セグメント内のロジックの変換

3.1.1　レコードメインの動作

レコードメイン中のロジックのセグメントは、カーソル移動時に、カーソル移動の方向に合わせて順次実行されていきます。

例えば、V9Plus でのレコードメインが次のようになっていたとします。

処理テーブル: レコード メイン

#	処理コマンド			内容				範囲		位置付		フロー		条件
1	ｾﾚｸﾄ	V=変数 :	1	a	代入:	0	0	0	0	0	S	C	Yes
2	ｾﾚｸﾄ	V=変数 :	2	b	代入:	0	0	0	0	0	S	C	Yes
3	ｾﾚｸﾄ	V=変数 :	3	c	代入:	0	0	0	0	0	S	C	Yes
4	ｴﾗｰ		0	1_SC	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			S	C	Yes
5	ｴﾗｰ		0	2_FC	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			F	C	Yes
6	ｴﾗｰ		0	3_CC	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			C	C	Yes
7													
8	ｴﾗｰ		0	4_SF	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			S	F	Yes
9	ｴﾗｰ		0	5_FF	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			F	F	Yes
10	ｴﾗｰ		0	6_CF	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			C	F	Yes
11													
12	ｴﾗｰ		0	7_SB	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			S	B	Yes
13	ｴﾗｰ		0	8_FB	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			F	B	Yes
14	ｴﾗｰ		0	9_CB	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			C	B	Yes
15	ｾﾚｸﾄ	V=変数 :	4	d	代入:	0	0	0	0	0	S	C	Yes
16	ｾﾚｸﾄ	V=変数 :	5	e	代入:	0	0	0	0	0	S	C	Yes

ここには、コントロール項目 c と d にはさまれたセグメントが一つあり、9つのエラーコマンドが記述されています。この場合、カーソル移動時の動作は、次のようになります。

- 項目 c から 項目 d へカーソルが移動する場合には、エラーコマンドが上から下へ順に実行されていきます。

  ただし、コマンドの実行は、フローパラメータの設定により次のような制限を受けます。

  - カーソルは順方向に進むので、フローの[方向]が B (B=後方向)になっているもの(12～14 行目) は実行されません。
  - Tab/Enter キーによりカーソルが移動する場合には、通常モードとなるので、[F=高速]に設定されているコマンド(5、9 行目)は実行されません。
- 項目 d から項目 c にカーソルが移動する場合には、逆方向の移動になるので、セグメントの中のコマンドが、下から上に向けて逆順に実行されます。

  この場合にも、フローのパラメータの設定により実行が制限されます。

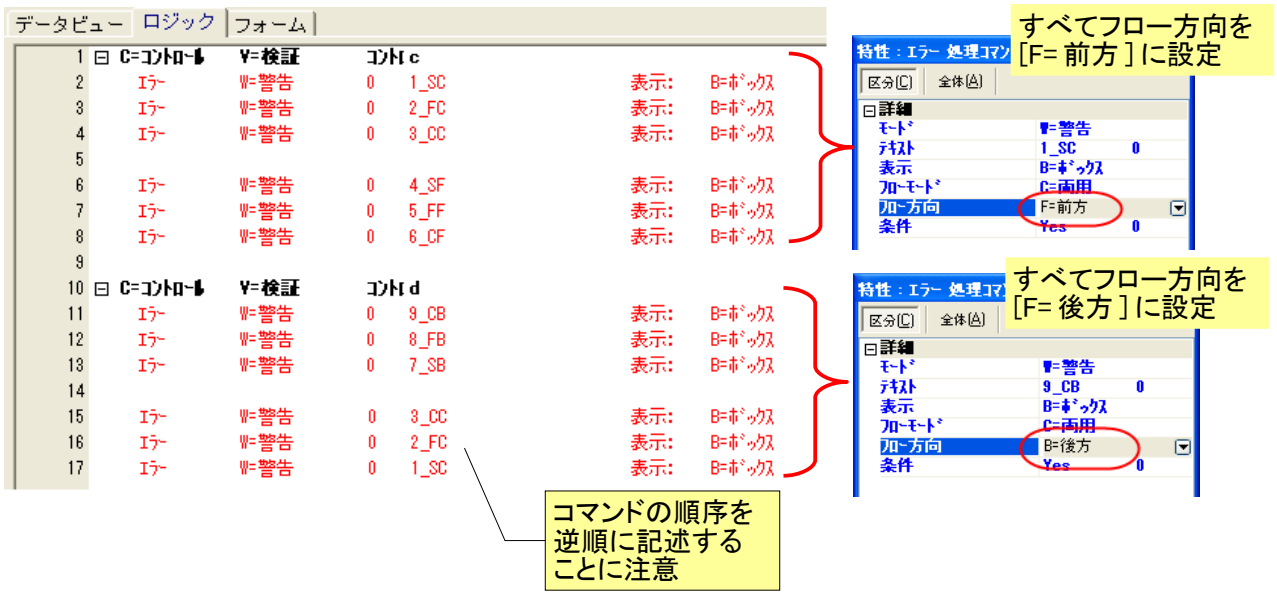
  - カーソルが逆方向に進んでいるので、[方向]が[F=前方向] になっているもの (8～10 行目)は実行されません。
  - Shift+Tab キーによりカーソルが移動する場合には、通常モードで実行されるので、[フローモード] が [F=高速] に設定されているコマンド(5、13 行目)も実行されません。

3.1.2　変換規則

コントロール項目 A とコントロール項目 B の間のセグメントにあるすべてのロジックは、以下のように変換します。

1. [方向]欄が[F=前方]または[C=両方向]に設定されたすべての処理コマンドは、コントロール A の[コントロール検証]ハンドラに定義します。
   - [方向]特性が[C=両方向]になっている全ての設定は、[F=前方]に設定します。
   - この基準に合う処理コマンドが存在しない場合、[コントロール検証]ハンドラを作成する必要がありません。
2. [方向]欄が[B=後方]または、[C=両方向]に設定されたすべての処理コマンドは、**逆の順序で**コントロール B の[コントロール検証]ハンドラに定義します。
   - [方向]欄が[C=両方向]の全ての設定は、[B=後方]に設定します。
   - この基準に合う処理コマンドが存在しない場合、[コントロール検証]ハンドラは作成する必要がありません。

この規則に従って、前出のレコードメインのセグメントを V10 のコントロールレベルハンドラで書きなおすと、次の図のようになります。コントロール c に定義された[コントロール検証]ハンドラは、c から d へ移動する場合のもので、コントロール d に定義された[コントロール検証]ハンドラは、d から c へ移動する場合のものです。



動作の違い

高速モードで移動する時には、[コントロール検証]ハンドラで書く場合と、レコードメインでロジックを書く場合で、次のように動作が異なります。

- レコードメイン中のロジックは、データが修正された場合のみ、実行されます。
- [コントロール検証]ハンドラ中のロジックは、データ修正の有無に関わらず、実行されます。

このため、[コントロール検証]ハンドラ内での変更内容を確認する必要があります。同様な調整作業がキーボード操作でレコード間を移動する場合にも必要になります。

高速モードは、次の場合に実行されるモードです。

- マウス操作によって、別のコントロール、あるいは別のレコードに移動する場合。
- 別のレコードに移動する場合。(ラインモードの画面で、[↑]や[↓]キーなどで移動する、あるいは、スクリーンモードの画面で、[PgUp] あるいは [PgDn] キーで移動するなど)
- ESC キーでタスクを終了する場合。

3.2 最初のコントロールに先行する処理

レコードメインの最初のコントロール項目より前に定義されている処理コマンドは、以下のように変更します。

1. [フローモード]欄が[S=標準]あるいは[C=両方]で、[方向]欄が[F=前方]あるいは[C=両方向]に設定された全ての処理コマンドは、
   - 最初のコントロールの[コントロール前]に定義します。
   - [方向]特性を[F=前方]に設定します
2. [フローモード]欄が[S=標準]あるいは[C=両方]で、[方向]欄が[B=後方]あるいは[C=両方向]に設定された全ての処理コマンドは、
   - 最初のコントロールの[コントロール後]に定義します。
   - [方向]特性を[B=後方]に設定します。
3. [フローモード]欄が[F=高速]あるいは[C=両方]で、[方向]欄が[F=前方]あるいは[C=両方向]に設定された全ての処理コマンドは、
   - [レコード前]の最後に定義します。

3.3 パーク制御

パーク制御の動作

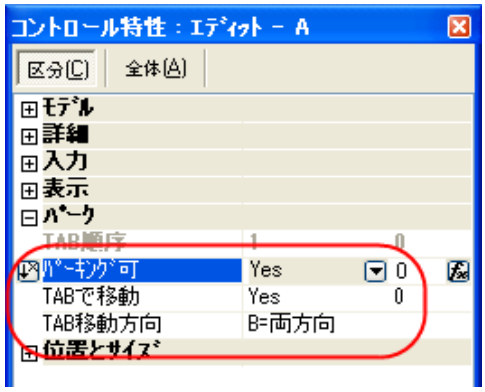
旧バージョンにおいて、カーソルが項目にパークするかどうかは、レコードメインの[セレクト]コマンドの[条件]により、以下のように制御されます。

- [条件]式が True と評価されるか、あるいは Yes と設定されている場合には、パークします。
- [条件]式が False と評価されるか、あるいは No と設定されている場合には、パークしません。

また、[フローモード]パラメータの設定によってもパークの有無が制御されます。例えば、[方向] パラメータが[B=後方向]と設定されている場合には、カーソルが逆方向から移動してきた場合にのみパークします。

V10 では、カーソルのパークは、フォーム上のコントロール特性によって制御します。[パーキング可] 特性によってパークの有無が制御されます。ここには、Yes/No の他、式を指定することもできます。

また、カーソルの移動方向によってパークの有無を制御したい場合には、[TAB 移動方向] に設定します。



変換規則

カーソル制御については、次の規則により変換します。

- レコードメインの[セレクト]コマンドの[条件]部分は、そのまま、対応するコントロールのコントロール特性の[パーキング可]に設定します。
- [方向]パラメータは、そのまま、[TAB 移動方向] パラメータに設定します。
- [フローモード] パラメータについては、V10 ではそのまま対応するコントロール特性がありません。Flow() 関数を使って、[パーキング可]の条件式に設定することになります。

第4章　エラーコマンド

レコードメイン中で[モード]が[E=エラー]の[エラー]処理コマンドがある場合には、あたかも処理の流れが反射していくような動きになります。ここでは、このような[エラー]処理コマンドがある場合について説明します。

セグメントに、[エラー]処理コマンドだけがある場合には、ここで説明するような変換は行う必要はなく、通常の基本変換規則に従って変換します。

4.1　レコードメインでの動き

レコードメイン中で[モード]が[E=エラー]の[エラー]処理コマンドが実行されると、そこで実行が中断され、今来た道を逆に戻っていく要領で、先ほどパークしていたコントロール項目まで戻ります。その際、[エラー]コマンドとコントロール項目との間にあるコマンドは、逆方向に再度実行されます。

例えば、次のようなプログラムがあるとします。

処理テーブル: レコード メイン

#	処理コマンド			内容				範囲		位置付		フロー		条件
1	ｾﾚｸﾄ	V=変数 :	1	A	代入:	0		0	0	0	0	S	C	Yes
2	ｴﾗｰ		0	Warning 1 F	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ				C	F	Yes
3	ｴﾗｰ		0	Warning 1 B	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ				C	B	Yes
4	ｴﾗｰ		0	Warning 1 C	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ				C	C	Yes
5	ｴﾗｰ		0	Error	ﾓｰﾄﾞ:	E=ｴﾗｰ	表示:	B=ﾎﾞｯｸ				C	C	1
6	ｴﾗｰ		0	Warning 2 C	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ				C	C	Yes
7	ｴﾗｰ		0	Warning 2 B	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ				C	B	Yes
8	ｴﾗｰ		0	Warning 2 F	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ				C	F	Yes
9	ｾﾚｸﾄ	V=変数 :	2	B	代入:	0		0	0	0	0	S	C	Yes
10														

条件 :　　ｴﾗｰ発生ﾌﾗｸﾞ

この場合、項目 A から B に移動しようとした場合、条件式 1 が True だった場合には、次のようになります。

1. 項目 A から、通常モード、順方向（上から下）で処理が始まります。
2. 2 行目の [Warning 1 F]が実行されます。
3. 3 行目の [Warning 1 B]は、[方向] が [B=後方向] に設定されているので、実行されません。
4. 4 行名の [Warning 1 C]が実行されます。
5. 5 行目の[エラー]コマンドは、[モード]が[E=エラー]なので、ダイアログボックス[Error]を出した後、実行が中断され、先にパークしていた項目 A に向けて、逆方向に（通常モード、逆方向で）処理が行われるようになります。
6. 4 行目の [Warning 1 C]が実行されます。
7. 3 行目の [Warning 1 B]は、[方向] が [B=後方向] に設定されていますが、今度は逆方向のモードなので、実行されます。
8. 2 行目の [Warning 1 F]は、[方向] が [F=前方向] に設定されているので、今度は実行されません。
9. 1 行目の[セレクト]コマンドに戻り、カーソルがパークして、ユーザの入力待ちになります。

4.2　変換規則

このようなレコードメインの動作と同等の動作を V10 で行わせるためには、次のようにします。

ここで扱う[エラー]コマンドは、モードが[E=エラー] であるものに限ります。モードが[W=警告]の場合には、実行が中断されないので、他の処理コマンドと同様に扱います。

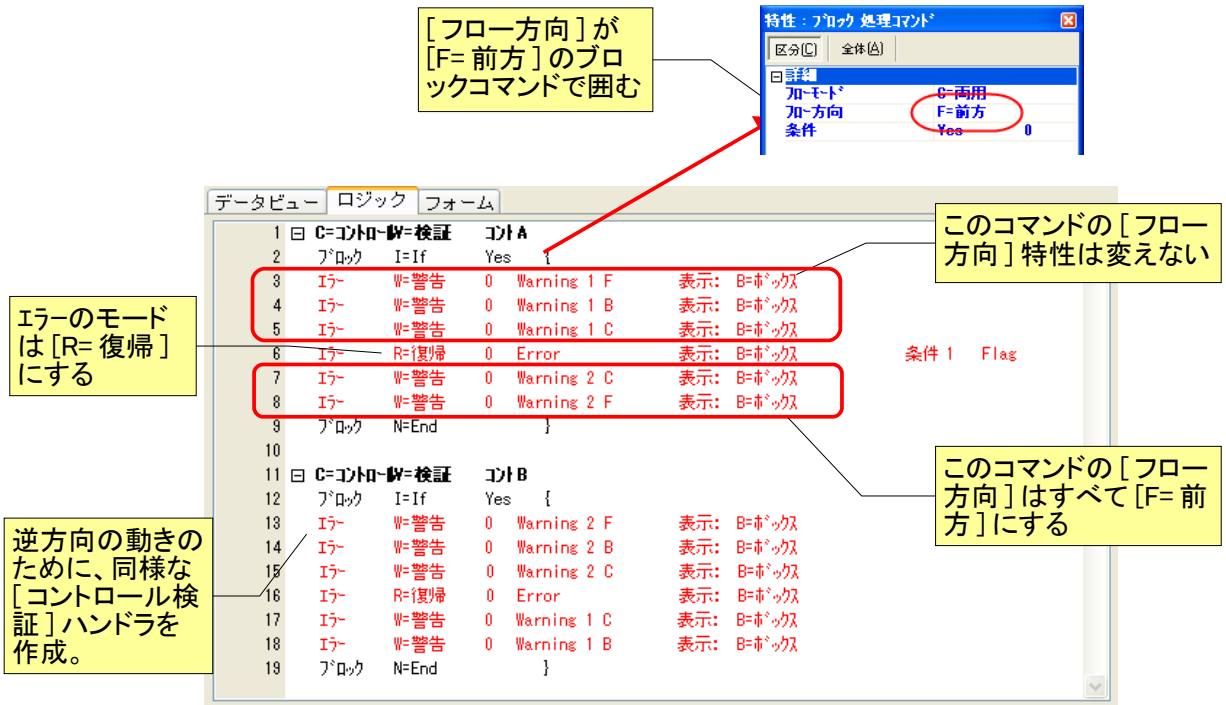
1. まず、セグメントの最後から、[方向]欄が[F=前方]あるいは[C=両方向]である[エラー]処理コマンドを探します。
2. コントロール項目 A からこの[エラー]処理コマンドまでの全ての処理コマンドを、コントロール A の[コントロール検証]に置きます。

   ここで、第 3 章 基本変換規則 とは異なり、注意すべき点は以下のことです。

   - [方向]の設定に関わらず、すべての処理コマンドを対象とすること。(基本変換規則では、[方向]が[C=両方向]あるいは[F=前方向]の処理コマンドのみが対象でした)。
   - 処理コマンドの[方向]特性の値はそのまま維持すること。(基本変換規則では、[フロー方向]を[F=前方]に設定していました。

   これは、[エラー]コマンドで中断した後に、逆方向に実行が進むことがあるからです。
3. [エラー]コマンドからコントロール項目 B までの間に定義された処理コマンドを、コントロール A の[コントロール検証]ハンドラに置きます。
   - ただし、[方向]欄が[F=前方]あるいは[C=両方向]と定義されている処理コマンドのみを対象とします。
   - これらの処理コマンドのセグメント全体を、[ブロック]処理コマンドで囲みます。
   - ブロックコマンドは、[フローモード]特性が[C=両用]、[方向]特性が[F=前方]と設定します。
4. [エラー]処理コマンドのモードを[R=復帰]に設定します。
5. 同様の変換処理を、コントロール B の[コントロール検証]ハンドラ内に配置された処理コマンドに対しても行います。

前出の例は、次のように変換されます。



第5章　前置および後置

5.1　概要

レコードメインにおける 前置（[B=前置]）と後置([A=後置])のフローモードは、ズームキー（F5）またはダブルクリックした場合の処理を記述するために用います。

ロジックの実行後、前置モードではカーソルはもとの項目に止まり、後置モードでは次のコントロールに移動します。

前置/後置モードで実行される処理コマンドは、常に[フローモード]特性が[S=通常]で[方向]特性が[F=前方]として動作します。

イベント指向で書くと、この処理はイベントハンドラを使って書きなおすことになります。

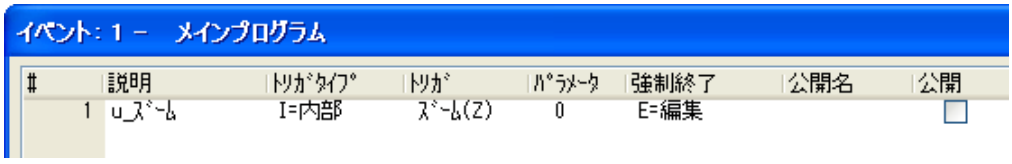
5.2　ユーザ定義イベントの利用

イベントハンドラを使ってズームの処理を行う場合には、内部イベントの[ズーム]を直接イベントハンドラでハンドリングすると、次の理由により正しい動作となりません。

- レコードメインの前置/後置モードの場合には、ロジックが実行される前に、編集モードが終了します。
- イベントハンドラで内部イベントの[ズーム]をハンドリングした場合は、編集モードが終了しないまま、ロジックは編集モード内で実行されます。
- この結果、入力途中の値が反映されないとか、項目の値をプログラムで変更した後に値がすぐに表示されない、などの問題が出ます。

このため、イベントハンドラでハンドリングする前に、編集モードを終了させることが必要になります。これは、次のような、[強制終了] が[E=編集]のユーザ定義イベントを介在させて処理することにより達成されます。

設定	値
名前	[u_ズーム]（任意でよいが、わかり易い名前にする）
トリガ	[内部イベント] の[ズーム]
強制終了	E=編集
公開名	<空白>
公開	未チェック



すべての変換処理において、このようなユーザ定義イベントを、メインプログラムの[ユーザイベント]テーブルに追加します。

5.3　前置の場合

5.3.1　前置ズームセグメントの決定

[前置ズームセグメント]とは、あるコントロール項目上でユーザがズームした場合に、実行すべき処理コマンドのことです。このとき、このコントロール項目を[ズーム項目]と呼びます。

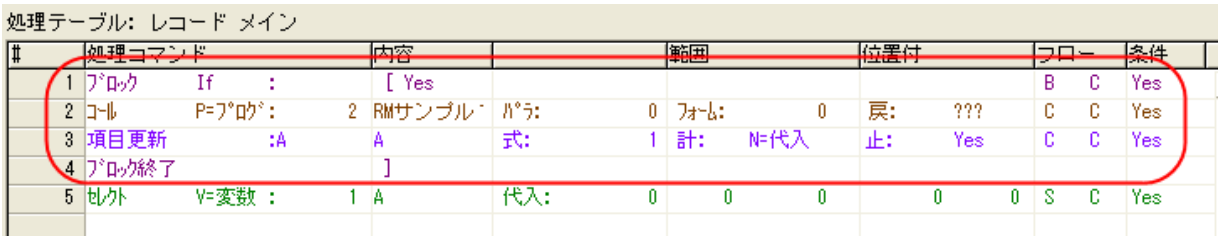
コントロール項目の直前に、[フローモード] パラメータが[B=前置] の処理コマンドがあれば、その処理コマ

ンドは前置ズームセグメントとなります。

例えば、次の図では、1 行目の[コール]コマンドが前置ズームセグメントで、2 行目の[セレクト]コマンドがズーム項目です。

また、コントロール項目の直前の処理コマンドが[ブロック終了]コマンドであった場合には、対応する[ブロック]コマンドの[フローモード]パラメータが調べられ、それが[B=前置]であれば、そのブロックコマンドで囲まれる処理全体が前置ズームセグメントとなります。

例えば、次の図では、1 行目のブロックコマンドから、4 行目のブロックコマンドまでが、前置ズームセグメントとなります。



前置ズームセグメントを識別する際に、[セレクト]コマンドと処理コマンドの間にコメント行があっても無視します。本書で「直前」あるいは「直後」と言う場合には、コメント行のことは考慮に入れません。

前置ズームセグメントが 1 つの処理コマンドからなる場合、そのコマンドの[方向]特性は無視されます。

同様に、前置ズームセグメントが[ブロック]コマンドで囲まれている場合、その[ブロック]コマンドの[方向]特性は無視されます。

5.3.2 前ブロックのフロー方向特性

[ブロック]コマンドのが、V10 の場合も同様に無視されます。

5.3.3 前置ズームセグメントの変換規則

前置ズームセグメントは、次のようにイベントハンドラに変換します。

1. ハンドラの作成
   - ユーザイベント [u_ズーム] を使用したイベントハンドラを、ズーム項目のコントロールに対して作成します。
   - ハンドラは、処理されるコントロールに固有のものですので、コントロール名を指定します。
2. 処理コマンドのコピー
   - 前置ズームセグメント内の処理コマンドは、[方向]と[フローモード]の内容を維持したままでイベントハンドラ内にコピーします。
   - ただし、[フローモード]欄が[S=通常]または[C=両方]で、かつ、[方向]欄が[F=前方]または[C=両方向]に設定されたものだけを対象とします。それ以外はコピーしません。
3. [ブロック]処理コマンド自身
   - 前置ズームセグメントが[ブロック]コマンドで囲まれている場合、[ブロック]処理コマンド自身も、イベントハンドラ内にコピーします。

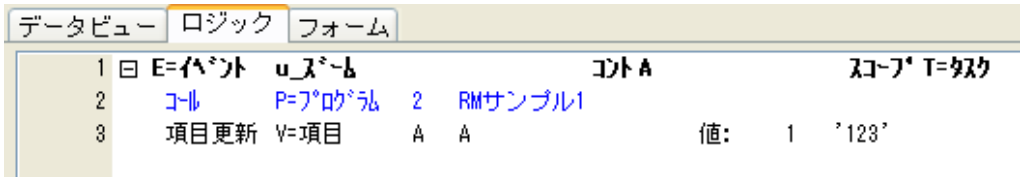
- ○ [ブロック]処理コマンドに[条件]が設定されている場合、この条件はイベントハンドラの[有効]特性として設定します。
- ○ [ブロック]処理コマンドに[条件]が設定されていない場合には、[ブロック]処理コマンドと対応する[ブロック終了]処理コマンドは、イベントハンドラ内にコピーする必要はありません。

通常、次のいずれかにあてはまる処理コマンドは、ズーム時にも実行されないので、イベントハンドラ内にコピーしません。

- [フローモード] 欄が[F=高速] のもの
- [方向] 欄が[B=後方向] のもの

ただし、ズームセグメント中に [E=エラー]モードの[エラー]コマンドのある場合には、[方向]欄が[B=後方向] のものもコピーすることがあります。詳しくは 5.5 前置/後置ズームセグメント内の[エラー]処理コマンドを参照してください。

上記の規則によって書き直したズームハンドラは、次の図のようになります。



5.3.4 動作の違い

ステータスバーの[ズーム]表示に違いがあります。

- レコードメインでは、前置ズームセグメントの[条件]が False の場合、ステータスバーの[ズーム]が表示されません。
- イベントハンドラでは、[有効]条件が False の場合でも[ズーム]が表示されます。

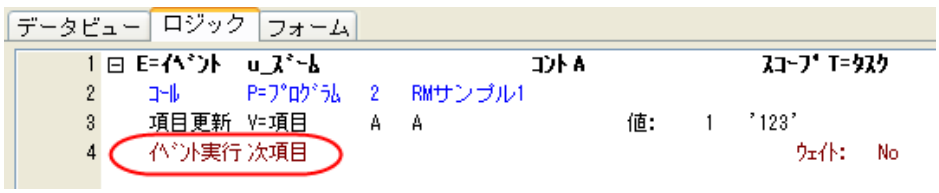
5.4 後置の場合

フローモードが [A=後置] は、以下の 2 点を除いて、[B=前置]の場合と同じです。

- [後置ズームセグメント]は、ズーム項目の直後に置きます。
- ズーム処理の完了後に、カーソルは次のコントロール項目に移動します。

イベントハンドラでは、処理の完了後に次のコントロール項目に移動させるため、[次項目]イベントを発行するようにします。

1. [イベント実行]処理コマンドを、イベントハンドラ内の最後の処理コマンドとして追加します。
2. [イベント実行]処理コマンドには、内部イベントの[次項目]イベントを設定します。



5.5 前置/後置ズームセグメント内の[エラー]処理コマンド

前置/後置ズームセグメント内に、[モード]が[E=エラー]である[エラー]処理コマンドがあった場合には、[エ

ラー]コマンドで処理の流れが反射して返ってくることがあるので、若干変更規則が変わります。

5.3.3 の前置ズームセグメントの変換規則 では、次のいずれかの処理コマンドはコピーしませんでした。

- ［方向］特性が[B=後方]
- [フローモード]が[F=高速]

しかし、[エラー]コマンドのある場合には、事情が変わります。

- [エラー]コマンドより前にあるコマンドが逆方向で実行されることがあるので、［方向］特性が[B=後方]である処理コマンドもコピーしておく必要があります。
- [エラー]コマンドより後にある［方向］特性が[B=後方]である処理コマンドは、実行されることがないので、コピーしません。

[フローモード]が[F=高速]である処理コマンドは、いずれにしても実行されないので、コピーする必要はありません。

このことを考慮した変換規則は、次のようになります。ここでは前置ズームセグメントについて説明しますが、後置ズームセグメントでも同様です。

1. まず、ズームセグメントの最後から、次の条件をすべて満たす［エラー］コマンドを探します。
   - ［モード］が[E=エラー]
   - [フローモード]が[S=通常]あるいは[C=両方向]
   - ［方向］欄が[F=前方]あるいは[C=両方向]
2. このような［エラー］コマンドが見つかった場合、
   - 前置ズームセグメントの先頭から、この[エラー]コマンドの間の処理コマンドを、イベントハンドラにコピーします。
   - ［方向］特性が[B=後方]である処理コマンドをイベントハンドラにコピーする際には、このコマンドの［方向］特性は[B=後方]のままにします。
   - [フローモード]が[F=高速]であるものは除外します。
3. ［エラー］コマンドの[モード]は、[E=エラー]から[R=復帰]モードに変更します。
4. この［エラー］コマンド以降に定義されている処理コマンドも、イベントハンドラにコピーします。ただし、次のいずれかの条件を満たす処理コマンドは、ここで実行されないため削除します。
   - ［フローモード]が[F＝高速]
   - ［方向］が[B=後方]

5.5.1 未解決のズーム

処理コマンドの位置によっては、[フローモード]が[B=前置]あるいは[A=後置] であっても、前置/後置ズームセグメントとならない場合があります。これらのケースは通常、処理コマンドがコントロール項目と隣接していない場合です。

コントロールに隣接していない場合、通常のフローの中でも、ズーム処理としても、実行される可能性がないので、削除して構いません。

第6章　ブロックコマンド

[ブロック]処理コマンドは、ロジック（処理コマンド）を実行させる場合だけでなく、[セレクト]処理コマンドに対しても有効です。[セレクト]コマンドに対しては、[ブロック]コマンドは、パークの有無を制御するものとして作用します。

[ブロック]処理コマンドの扱いは以下のケースに分けることができます。各ケースごとに異なる変換処理を行う必要があります。

以下、[フローモード]が[C=両方]であり、かつ、[方向]が[C=両方] であることを、略して[CC] と書きます。

- ケース1：条件が設定された CC の[ブロック]コマンドで、中にコントロール項目がない場合
- ケース2：CC でない[ブロック]コマンドで、中にコントロール項目がない場合
- ケース3：[ブロック]コマンドの中に、コントロール項目がある場合
- ケース4：前置/後置の[ブロック]コマンドで、中にコントロール項目がある場合

6.1　ケース1

ケース1は、条件が設定された CC の[ブロック]コマンドで、中にコントロール項目がない場合であり、最も簡単なケースです。
例えば、V9Plus の RM が次のようになっていたとします。

処理テーブル: レコード メイン

#	処理コマンド			内容			範囲		位置付		フロー		条件
1	ｾﾚｸﾄ	V=変数:	1	a	代入:	0	0	0	0	0	S	C	Yes
2	ｾﾚｸﾄ	V=変数:	2	b	代入:	0	0	0	0	0	S	C	Yes
3	ｾﾚｸﾄ	V=変数:	3	c	代入:	0	0	0	0	0	S	C	Yes
4	ｴﾗｰ		0	1_SC	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			S	C	Yes
5	ｴﾗｰ		0	2_FC	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			F	C	Yes
6	ﾌﾞﾛｯｸ	If	:	[Toggle							C	C	1
7	ｴﾗｰ		0	3_CC	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			C	C	Yes
8													
9	ｴﾗｰ		0	4_SF	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			S	F	Yes
10	ｴﾗｰ		0	5_FF	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			F	F	Yes
11	ｴﾗｰ		0	6_CF	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			C	F	Yes
12													
13	ｴﾗｰ		0	7_SB	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			S	B	Yes
14	ﾌﾞﾛｯｸ終了			]									
15	ｴﾗｰ		0	8_FB	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			F	B	Yes
16	ｴﾗｰ		0	9_CB	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			C	B	Yes
17	ｾﾚｸﾄ	V=変数:	4	d	代入:	0	0	0	0	0	S	C	Yes
18	ｾﾚｸﾄ	V=変数:	5	e	代入:	0	0	0	0	0	S	C	Yes

ここでは、6 行目～14 行目が、[ブロック]～[ブロック終了]コマンドで囲まれています。この間にある処理コマンドの実行は、[ブロック]コマンドの[条件]により制御されます。

6.1.1　変換規則

- [コントロール検証]ハンドラ内の処理コマンドを[ブロック]処理コマンドで囲んでいる場合、[コントロール検証]ハンドラ内に[ブロック]処理コマンドを含めて処理コマンドを作成します。[ブロック]処理コマンドには、元の実行条件を設定します。

- [ブロック]処理コマンドは、他の処理コマンドと同じように、基本変換規則（第 3 章 基本変換規則 8 ページを参照）に従って変換します。
- [ブロック]処理コマンドが、セグメント中のすべての処理コマンドを囲んでいる場合、[ブロック]処理コマンドの条件は[コントロール検証]ハンドラの[有効]条件として設定することもできます。

6.1.2 逆の順序

逆方向にカーソルが移動した場合の処理のために、コントロール B の[コントロール検証]ハンドラを作成しますが、このハンドラ内の処理コマンドは次のようにします。

- [ブロック]処理コマンド以外のコマンドは、すべて、順序を反転してコピーします。
- [ブロック]処理コマンド、および[ブロック終了]処理コマンドは、その位置を保持します。（順序を反転しません）

前出の例を V10 の[コントロール検証]で書きなおすと、次のようになります。

ここで、

- コントロール c の[コントロール検証]ハンドラ中のすべての処理コマンド（[ブロック]コマンドを含む）の[方向]特性は、[F=前方]です。
- コントロール d の[コントロール検証]ハンドラ中のすべての処理コマンド（[ブロック]コマンドを含む）の[方向]特性は、[B=後方]です。

6.1.3 エラー処理コマンド

[エラー]処理コマンドの扱い方は、第 4 章 エラーコマンド (11ページ) で説明した内容と同じです。

6.1.4 ブロック Else

[ブロック Else] 処理コマンドは、そのまま変換します。

逆移動のためにロジックを反転する場合、[ブロック Else]では反転しません。[ブロック]処理コマンド行の間の処理コマンドのみ反転します。

6.2 ケース2

ケース2は、CC でない[ブロック]コマンドで、中にコントロール項目がない場合です。

例えば、次のようなケースです。

処理テーブル: レコード メイン

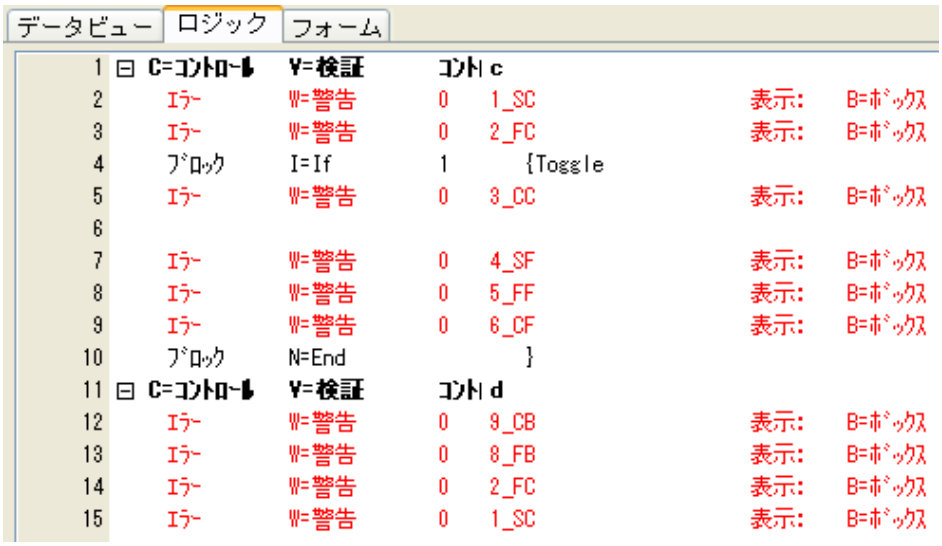
#	処理コマンド			内容			範囲		位置付		フロー		条件
1	ｾﾚｸﾄ	V=変数 :	1	a	代入:	0	0	0	0	0	S	C	Yes
2	ｾﾚｸﾄ	V=変数 :	2	b	代入:	0	0	0	0	0	S	C	Yes
3	ｾﾚｸﾄ	V=変数 :	3	c	代入:	0	0	0	0	0	S	C	Yes
4	ｴﾗｰ		0	1_SC	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			S	C	Yes
5	ｴﾗｰ		0	2_FC	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			F	C	Yes
6	ﾌﾞﾛｯｸ	If :		[Toggle							C	F	1
7	ｴﾗｰ		0	3_CC	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			C	C	Yes
8													
9	ｴﾗｰ		0	4_SF	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			S	F	Yes
10	ｴﾗｰ		0	5_FF	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			F	F	Yes
11	ｴﾗｰ		0	6_CF	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			C	F	Yes
12													
13	ｴﾗｰ		0	7_SB	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			S	B	Yes
14	ﾌﾞﾛｯｸ終了			]									
15	ｴﾗｰ		0	8_FB	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			F	B	Yes
16	ｴﾗｰ		0	9_CB	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			C	B	Yes
17	ｾﾚｸﾄ	V=変数 :	4	d	代入:	0	0	0	0	0	S	C	Yes
18	ｾﾚｸﾄ	V=変数 :	5	e	代入:	0	0	0	0	0	S	C	Yes

ここでケース1の例と異なる点は、6 行目の[ブロック]コマンドの[方向]が[F=前方]になっていることです。

6.2.1 変換規則

- [ブロック]処理コマンドの[方向]欄が[F=前方]の場合、[ブロック]コマンド内の処理コマンドは、
  - コントロール A の[コントロール検証]ハンドラには、処理コマンドをコピーします。
  - コピーする処理コマンドの[フローモード]と[方向]の特性はそのままの値にしておきます。
  - コントロール B の[コントロール検証]ハンドラには、処理コマンドをコピーしません。
- [方向]が[B=後方]の場合はその逆で、コントロール B の[コントロール検証]ハンドラにだけコピーし、コントロール A の[コントロール検証]ハンドラにはコピーしません。

従って、前述の例では、次のようになります。



コントロール d の[コントロール検証] ハンドラで、[ブロック] コマンドの中身がコピーされていないことに注意してください。

6.3 ケース3

ケース3は、[ブロック]コマンドの中に、コントロール項目がある場合です。

6.3.1 変換規則

[ブロック]処理コマンド内にコントロール項目がある場合、ブロック内に複数のセグメントがあることになります。

1. それぞれのセグメントについて、各セグメントが[ブロック]処理コマンドによって囲まれているものとして、上述のケース1とケース2の規則に基づき、セグメントを変換します。
2. コントロール項目に対応する、フォーム上のコントロールの[パーク可]特性に、[ブロック]処理コマンドに設定された実行条件を設定します。

例えば: V9Plus の RM ロジックが次のようになっていたとします。

処理テーブル: レコード メイン

#	処理コマンド			内容			範囲		位置付		フロー		条件
1	ｾﾚｸﾄ	V=変数 :	1	a	代入:	0	0	0	0	0	S	C	Yes
2	ﾌﾞﾛｯｸ	If :		[Toggle							C	C	1
3	ｾﾚｸﾄ	V=変数 :	2	b	代入:	0	0	0	0	0	S	C	Yes
4	ｴﾗｰ		0	1_SC	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			S	C	Yes
5	ｴﾗｰ		0	2_FC	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			F	C	Yes
6	ｴﾗｰ		0	3_CC	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			C	C	Yes
7													
8	ｴﾗｰ		0	4_SF	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			S	F	Yes
9	ｴﾗｰ		0	5_FF	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			F	F	Yes
10	ｴﾗｰ		0	6_CF	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			C	F	Yes
11													
12	ｴﾗｰ		0	7_SB	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			S	B	Yes
13	ｴﾗｰ		0	8_FB	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			F	B	Yes
14	ｴﾗｰ		0	9_CB	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			C	B	Yes
15	ｾﾚｸﾄ	V=変数 :	3	c	代入:	0	0	0	0	0	S	C	Yes
16	ｴﾗｰ		0	11_SC	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			S	C	Yes
17	ｴﾗｰ		0	12_FC	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			F	C	Yes
18	ｴﾗｰ		0	13_CC	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			C	C	Yes
19													
20	ｴﾗｰ		0	14_SF	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			S	F	Yes
21	ｴﾗｰ		0	15_FF	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			F	F	Yes
22	ｴﾗｰ		0	16_CF	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			C	F	Yes
23													
24	ｴﾗｰ		0	17_SB	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			S	B	Yes
25	ｴﾗｰ		0	18_FB	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			F	B	Yes
26	ｴﾗｰ		0	19_CB	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			C	B	Yes
27	ｾﾚｸﾄ	V=変数 :	4	d	代入:	0	0	0	0	0	S	C	Yes
28	ﾌﾞﾛｯｸ終了			]									
29	ｾﾚｸﾄ	V=変数 :	5	e	代入:	0	0	0	0	0	S	C	Yes

これを V10 で書きなおすと、次のような3つの[コントロール検証]ハンドラとなります。

① コントロール b の[コントロール検証]ハンドラ:

データビュー | ロジック | フォーム

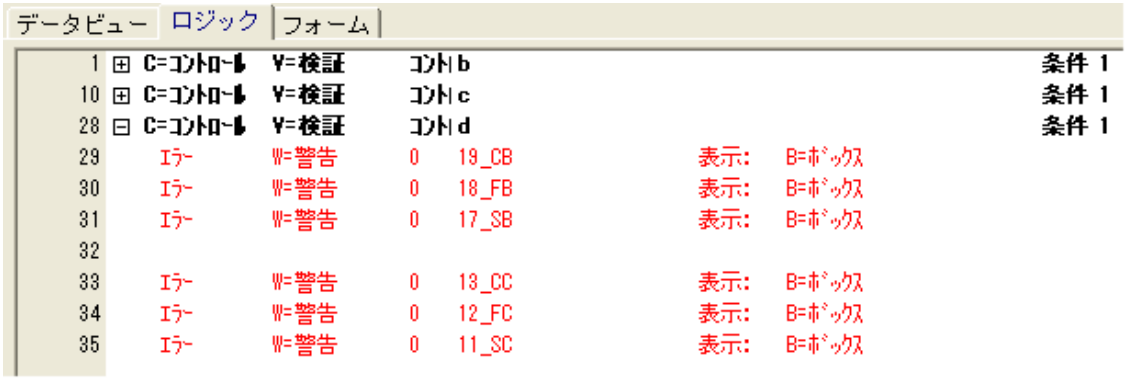
1	⊟ C=ｺﾝﾄﾛｰﾙ	V=検証	ｺﾝﾄ b					条件 1
2		ｴﾗｰ	W=警告	0	1_SC	表示:	B=ﾎﾞｯｸｽ	
3		ｴﾗｰ	W=警告	0	2_FC	表示:	B=ﾎﾞｯｸｽ	
4		ｴﾗｰ	R=復帰	0	3_CC	表示:	B=ﾎﾞｯｸｽ	
5								
6		ｴﾗｰ	W=警告	0	4_SF	表示:	B=ﾎﾞｯｸｽ	
7		ｴﾗｰ	W=警告	0	5_FF	表示:	B=ﾎﾞｯｸｽ	
8		ｴﾗｰ	W=警告	0	6_CF	表示:	B=ﾎﾞｯｸｽ	
9								
10	⊞ C=ｺﾝﾄﾛｰﾙ	V=検証	ｺﾝﾄ c					条件 1
28	⊞ C=ｺﾝﾄﾛｰﾙ	V=検証	ｺﾝﾄ d					条件 1

ここで、行 2～8 の[方向]特性は、[F=前方]です。

② コントロールcの[コントロール検証]ハンドラ:

ここで、行 11～17 の[方向]特性は、[B=後方]であり、行 20～26 の[方向]特性は、[F=前方]です。

③ コントロールcの[コントロール検証]ハンドラ:



ここで、行 29～35 の[方向]特性は、[B=後方]です。

また、コントロール b、c、d の[パーク可] 特性も、式 1 で条件付けします。

6.3.2 条件の拡散

以上述べたような変換を行うと、[ブロック]コマンドに設定されていた[条件]式が、何箇所かで参照されることになります。こうなると、実行タイミングの違いによって、同じ式であっても評価結果が異なる可能性が出てきます。これはすなわち、実行結果が異なってくる、ということになります。この現象を、ここでは[条件の拡散]と呼びます。

条件の拡散による動作の違いを回避するため、次のような工夫が必要になります。

- [条件]式は、適当なタイミングで 1 度だけ評価します。
- その結果は、何らかの手段で記録しておきます。

- 後に条件を参照するときには、記録しておいた結果を参照します。条件式を再度評価することはしません。

条件式の評価結果を記録しておくためには、次のいずれかの方法が考えられます。

- 変数項目を新たに設けて使う。
- グローバル変数を使う。

変数項目を使う場合には、次のような点に注意をする必要があります。

- データビューに、変数が追加される。
- 変数項目の更新処理を実行すると、今まで発生しなかったレコードロックを引き起こす可能性がある。

一方、グローバル変数を使う場合には、このような問題は出ませんが、次の欠点があります。

- プログラム中での扱いが煩わしい。
- デバッガで値を見ることができない。

以下では、変数を使わずに、グローバル変数を使った方法で説明しますが、変数を使う方法でもやりかた自体は変わりありません。実際には、変数を使った方が、プログラムは作りやすいでしょう。

6.3.3 グローバル変数の利用

グローバル変数名

条件の評価結果を記憶しておくグローバル変数は、ここでは以下のようなフォーマットで命名します。こうすれば、他との重複、混同を避けることができます。

```
Prg[タスク番号]CND[オリジナルの式番号に 0 を追加]
```

例えば、タスク 11.1.3 の条件式 12 ならば、[Prg11.1.3CND0012]といった名前になります。

グローバル変数への値の設定

ローバル変数に値を設定するためには、[アクション]処理コマンドで SetParam() 関数を実行します。

```
SetParam([グローバル変数名], [条件式])
```

ここで、[条件式]は、[ブロック]処理コマンドの元の条件式です。

グローバル変数の値の参照

グローバル変数の値を参照するには、GetParam() 関数を使います。

```
GetParam([グローバル変数名])
```

まだ SetParam で値が設定されていないグローバル変数を参照して GetParam 関数を実行した場合、Null が返ります。条件の判定では、Null 値は False と同様のものと評価されますので、通常は問題ありません。

グローバル変数のクリア

グローバル変数は、いくらかのメインメモリを消費します。もう使われないグローバル変数がゴミとなってしまう問題を回避するため、グローバル変数を使う各タスクの[タスク後]でこの値を Null にリセットします。Null に設定すると、グローバル変数は利用しているメモリを解放します。

```
SetParam([グローバル変数名], Null())
```

6.3.4　ブロックの条件式の変換

[ブロック]処理コマンドで囲まれる最初のセグメントに対応する[コントロール検証]ハンドラ内で、条件式を評価して、結果をグローバル変数に設定します。このときには、

- [アクション]処理コマンドと SetParam 式を使います。
- [アクション]処理コマンドの[フローモード]と[方向]は、もとの[ブロック]処理コマンドの[フローモード]と[方向]の設定をもとに、基本規則に従って作成します。

逆方向からのカーソルの動きに対応するため、[ブロック]処理コマンドで囲まれる最後のセグメントに対応する[コントロール検証]ハンドラ内でも、同じ[アクション]処理コマンドを作成します。

各セグメントの実行条件は、設定したグローバル変数を GetParam 式で取得した値にします。

6.3.5　コントロールのパーク条件の設定

ロジックの実行条件は、フォーム上のコントロールの[パーク可]特性の条件としても設定します。コントロールの[パーク可]特性にすでに条件が設定されている場合、元の条件と GetParam 関数が AND で連結されたものを新しい条件とします。

例えば、論理条件が、

```
GetParam('Prg11CND0012')
```

で、[セレクト]処理コマンドの[条件]が

```
B = 12
```

だった場合、[パーク可]特性の新しい条件は

```
B=12 AND GetParam('Prg11CND0012')
```

となります。

6.4　ケース4

ケース4は、前置/後置の[ブロック]コマンドで、中にコントロール項目がある場合です。

6.4.1　レコードメインでの動作

この場合には、次のような動作になります。

- ブロック内のコントロール項目は、通常はパーク不可です。
- ズーム項目でズームした場合にのみ、ブロック内のコントロール項目がパーク可能になります。
- ズームすると、最初のコントロール項目までの処理コマンドが実行され、その後、そのコントロール項目にパークします。
- ズーム処理をいったん抜けると(つまり、[ブロック]コマンドの外に出ると)、そのコントロール項目は再びパーク不可となります。

例えば、次のようなレコードメインがあったとします。

処理テーブル: レコード メイン

#	処理コマンド			内容			範囲		位置付		フロー		条件
1	ﾌﾞﾛｯｸ	If	:	[Yes							B	C	Yes
2	ｴﾗｰ		0	Zoom 1	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			C	C	Yes
3	ｾﾚｸﾄ	V=変数 :	1	A	代入:	0	0	0	0	0	S	C	Yes
4	ｴﾗｰ		0	Zoom 2	ﾓｰﾄﾞ:	W=警告	表示:	B=ﾎﾞｯｸ			C	C	Yes
5	ﾌﾞﾛｯｸ終了			]									
6	ｾﾚｸﾄ	V=変数 :	2	B	代入:	0	0	0	0	0	S	C	Yes
7	ｾﾚｸﾄ	V=変数 :	3	C	代入:	0	0	0	0	0	S	C	Yes

変数項目 B がズーム項目であり、1～5 行目のブロックで囲まれた部分が前置ズームセグメントとなっています。

通常は、項目 A はパーク不可で、項目 B と C がパーク可能ですが、項目 B でズームすると、次のような動作になります。

1. 最初にエラーコマンド[Zoom 1]が実行される。
2. 項目 A にパークする。
3. ユーザが Tab キーを押すと、エラーコマンド[Zoom 2]が実行される。
4. 項目 B にパークし、項目 A はパーク不可になる。

6.4.2 グローバル変数の利用

グローバル変数を利用します。このグローバル変数は NULL、True、False の3ステートの変数として使い、各ステートに次のような意味を持たせます。

- NULL: ズームセグメント内のコントロール項目のパークは不可。処理コマンドも実行不可。
- False: ズームセグメント内のコントロール項目のパークは可。処理コマンドの実行は不可。
- True: ズームセグメント内のコントロール項目のパークは可。処理コマンドの実行も可。

このグローバル変数は、以下のフォーマットで命名するようにします。

Prg[タスク番号]Zoom[このタスク内の条件の設定されていないズームブロックの連番]

すべての[ブロック]処理コマンドに対して、グローバル変数を使用します。

ブロックに条件が設定されている場合、フォーマットは、6.3.2 に説明されているように定義します。

6.4.3 変換規則

このグローバル変数を使って、以下のように変換します。

ズームハンドラ

ズーム項目に設定された[u_ズーム]のユーザイベントを使用したイベントハンドラを作成します。

[u_ズーム]イベントハンドラでは、次のようにします。

1. グローバル変数を False に設定します。これにより、ブロック内のコントロール項目がパーク可になります。
2. [ブロック]処理コマンドの先頭から、最初のコントロール項目までに定義されている処理コマンドをコピーします。

   ここでコピーするのは次の処理コマンドのみです。
   - フローモードが[S=通常]あるいは[C=両方向]で、[方向]が[F=前方]あるいは[C=両方向]に設定されているもの。
   - それ以外のコマンドは実行されないので、コピーしません。
3. ブロック内の最初のコントロール項目に位置付けるために、[アクション]コマンドで CtrlGoto 関数を呼び出します。

   ここで「最初のコントロール項目」というのは、次のような条件を満たす[セレクト]コマンドです。
   - フロー特性が[SC]、[SF]または[CC]に設定されている。
   - 項目がフォーム上のコントロールに割り当てられている。
   - [セレクト]処理コマンドが複数ある場合には、ブロック内で最初に現れるもの。

コントロール前処理ハンドラ

最初のコントロールに対する[コントロール前]ハンドラを作成します。

このハンドラでは、グローバル変数を True に設定する[アクション]処理コマンドを定義します。

ズームセグメント内のロジックの変換

基本規則(第 3 章 基本変換規則 を参照)に従って、このブロック内のロジックを内部のコントロール間で分配します。

各[コントロール検証]ハンドラには、グローバル変数の値を参照して条件を設定します。

条件付けは、通常、[ブロック]コマンドを設けて行います。
ただし、[コントロール検証]ハンドラ内に定義された[ブロック]処理コマンド内のロジックが、そのハンドラのロジックの全てである場合には、条件をハンドラの[有効]特性として設定することもできます。

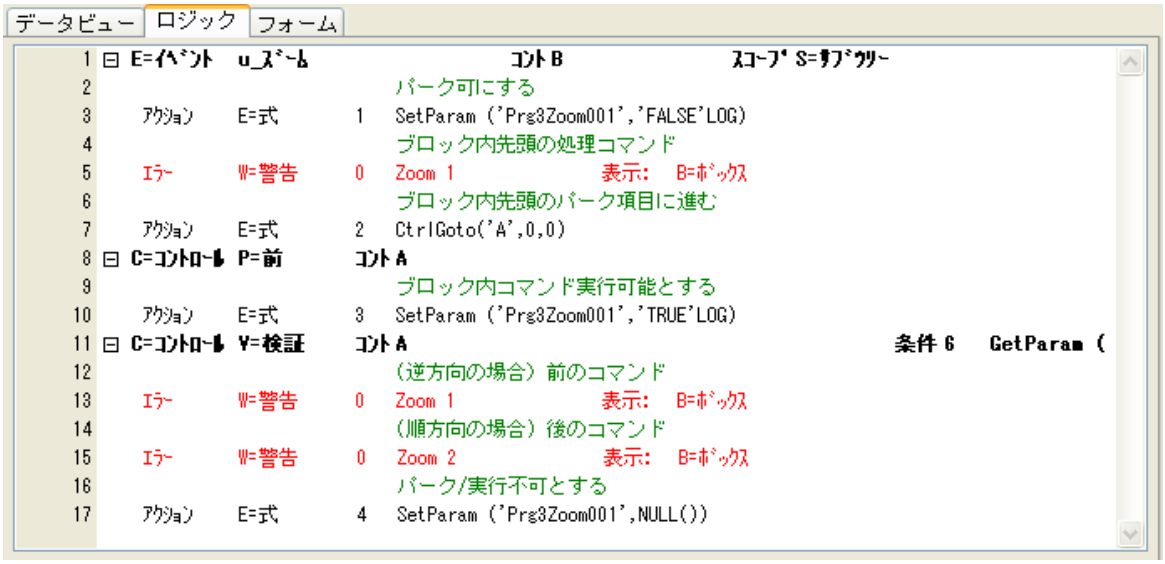
グローバル変数のリセット

ブロック内の最初のコントロールと最後のコントロールの[コントロール検証]ハンドラでは、ズームセグメント内の処理をすべて無効とするようにグローバル変数を NULL に設定します。これにより、コントロール項目はパーク不可となり、処理コマンドの実行も不可となります。

より正確には、次のようにします。

1. ズームセグメント内の最初のコントロールに対する[コントロール検証]ハンドラに、次のような[アクション]処理コマンドを定義します。
   - [フローモード]は[C=両方向]で、[方向]を[B=後方]に設定します。
   - SetParam でグローバル変数を NULL に設定します。
2. ズームセグメント内の最後のコントロールに対する[コントロール検証]ハンドラに、次のような[アクション]処理コマンドを定義します。
   - [フローモード]は[C=両方向]で、[方向]は[F=前方]に設定します。
   - SetParam でグローバル変数を NULL に設定します。

前述の例は、V10 では次のようなハンドラとなります。



6.4.4 前置と後置ブロックに条件が付いている場合

[フローモード]モードが[B=前置]または[A=後置]の[ブロック]処理コマンドに条件が設定されている場合には、[u_ズーム]のイベントハンドラに条件を設定します。

6.4.5 前置と後置の違い

ケース4の場合には、ズームセグメント内のコントロール項目でパークするため、前置と後置の動作との違いがありません。このため、最後に[次項目]の[イベント実行]処理コマンドを定義する必要はありません。

第7章 その他の話題

7.1 Level 関数

タスクがレコードメインにあるかどうかを確認する Level 関数の動作が変わります。コントロールレベルのハンドラを含めたハンドラから、この関数を実行すると、関数は異なる値を返します。

Level 関数が 'RM' と比較される式では、新しい MainLevel 関数に置き換えます。

	V9Plus	V10
例1	Level(1)='RP'	Level(1)='RP'
例 2	Level(1)='RM'	MainLevel(1)='RM'

V10 でも、過去の名残として[RM](レコードメインの略)という言葉を使いますが、V10 では[レコードメイン]というレベルはないので、[ユーザとの対話モード]という意味と考えてください。

7.2 コントロール名の欠落

V10 では、ハンドラが作成されるコントロールに対しては、コントロールの名前を設定する必要があります。しかし、V8 およびそれ以前から移行してきた多くのアプリケーションでは、フォーム上のコントロールにコントロール名が設定されていません。そのため、処理が必要なすべてのコントロールに名前を設定してください。

コントロールの名前には、ユニークな名前を設定する必要があります。同じ名前が別のコントロールに定義されている場合、ユニークな名前になるようにしてください。

7.3 コメント行

コメント行が続くセクションは、セクションに続く処理コマンド行と一緒に移動します。

7.4 複数フォーム

もしタスクに複数の GUI 表示フォームが定義され、タスクの[メインフォーム]特性に式が設定されている場合、すべての GUI 表示フォームに対して変換処理を行います。コントロールハンドラも、各フォームについて、順番に作成します。

7.5 既存のコントロールレベルのハンドラ

[コントロール検証]ハンドラが既に定義されている場合、本書で書いた変換規則により作成される処理コマンドは、既存の処理コマンドの後に追加します。

7.6 バッチとブラウザタスクのレコードメイン処理

バッチタスクとブラウザタスクのレコードメインに定義された(項目定義以外の)すべての処理コマンドは削除します。

7.7 アクセス不能な処理コマンドの扱い

一般的に、V9Plus のレコードメインで[アクセス不能な処理コマンド]とは、レコードメインで実行される機会が全くない処理コマンドを示します。そのような処理コマンドは、V10 に移行する場合に削除してください。

Magic eDeveloper V10

Magic eDeveloper V10
レコードメインからイベントへの変換規則

第 1 版　2007 年 11 月 19 日

発行　〒151-0053 東京都渋谷区代々木三丁目二十五番地三号
あいおい損保新宿ビル 14 階
マジック ソフトウェア・ジャパン (株)
http://www.magicsoftware.co.jp/