Magic eDeveloper V10

Magic eDeveloper V10









Enabling Business with Superior Technology

本書および添付サンプル(以下、本製品)の著作権は、マジックソフトウェアジャパン株式会社(MSJ)にあります。MSJ の書面による事前の許可なしでは、いかなる条件下でも、本製品 のいかなる部分も、電子的、機械的、撮影、録音、その他のいかなる手段によっても、コピー、検索システムへの記憶、電送を行うことはできません。

本製品の内容につきましては、万全を期して作成していますが、万一誤りや不正確な記述があったとしても、MSE(Magic Software Enterprises Ltd.)および MSJ はいかなる責任、債務も負いません。本製品を使用した結果、または使用不可能な結果 生じた間接的、偶発的、副次的な損害(営利損失、業務中断、業務情報の損失などの損害も含む)に関し、事前に損害の可能性 が勧告されていた場合であっても、MSE および MSJ、その管理者、役員、従業員、代理人は、いかなる場合にも一切責任を負い ません。MSE および MSJ は、本製品の商業価値や特定の用途に対する適合性の保証を含め、明示的あるいは黙示的な保証は 一切していません。

本製品に記載の内容は、将来予告なしに変更することがあります。

サードパーティ各社商標の引用は、MSE および MSJ の製品に対する互換性に関しての情報提供のみを目的としてなされるものです。一般に、会社名、製品名は各社の商標または登録商標です。

本製品において、説明のためにサンプルとして引用されている会社名、製品名、住所、人物は、特に断り書きのないかぎり、すべて架空のものであり、実在のものについて言及するものではありません。

初版 2007年10月12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	
1.1 レコードメインからイベント指向プログラムへ	5
1.2 本書の目的と範囲	6
1.3 前提条件	7
第2章旧バージョンのレコードメインの動作	
2.1 タスクエディッタの構成	8
2.2 レコードメインの基本的考え方	9
2.3 レコードメインのフローパラメータ	
2.3.1 フローパラメータ	
2.3.2 通常モード	
2.3.3 高速モード	
2.3.4 方向 パラメータ	
2.3.5 コマンド実行順序	
2.4 レコードメインの書き方ABC	
2.5 レコードメインの問題点	14
2.5.1 記述性	14
2.5.2 プログラムの可読性	14
2.5.3 複雑なブラックボックス	
第3章 V10 のイベント指向プログラミング	
3.1 タスクエディッタの構成	
3.2 ハンドラの種別とタイプ	
3.3 ハンドラの書き方	
3.4 ハンドラの作成手順	
第4章コントロールレベルハンドラの基本	
4.1 コントロールレベルのハンドラの動作	
4.2 コントロールレベルハンドラの書き方ABC	
4.3 レコードメインとコントロールレベルハンドラとの違い	
4.3.1 カーソルの移動方向	
4.3.2 カーソル移動の原因	
第5章コントロールレベルハンドラの例	
5.1 例① リンクの成功チェック	
5.1.1 レコードメインでのやりかた	
5.1.2 コントロールレベルハンドラでの書き方	
5.1.3 微妙な動作の違い	
5.2 例② 受注番号の後、条件により次の項目をスキップ	
5.2.1 レコードメインでのやりかた	

5.2.2 コントロールレベルハンドラでのやりかた	
5.2.3 別のやりかた	
5.3 例③ 項目更新による再計算ロジック	
5.3.1 レコードメインでのやりかた	35
5.3.2 コントロールレベルハンドラでのやりかた	
5.3.3 微妙な動作の違い	
二つの項目更新コマンドの実行順序	
実行タイミング	
第6章項目変更ハンドラ	
6.1 項目変更ハンドラとは?	
6.2 再計算ロジックの例	
6.3 項目変更ハンドラの利点	43
第7章プッシュボタン	
7.1 旧バージョンでのやりかた	45
7.2 イベントハンドラでのやりかた	47
7.2.1 ユーザ定義イベントの利用	
7.2.2 プッシュボタンの設定	
7.2.3 イベントハンドラの設定	52
7.3 イベントハンドラ方式の利点	57
第8章ズーム	
8.1 レコードメインでのやりかた	
8.1.1「B=前置」を利用	59
8.1.2「A=後置」を利用	59
8.1.3 ズーム時に複数のコマンドを実行	60
8.2 イベントハンドラを使うと・・・	61
8.2.1 ズーム項目	61
8.2.2 イベントハンドラ (その1)	61
8.2.3 内部イベント「ズーム」を直接ハンドルするときの問題	65
8.2.4 ユーザ定義イベント 「GU_ズーム」	
8.2.5 イベントハンドラ (その2)	
8.2.6 実行時の動作	72
第9章おわりに	

第1章 はじめに

1.1 レコードメインからイベント指向プログラムへ

Magic eDeveloper V10 では、イベント指向でプログラム作成を行うようになっており、旧来のレコードメインは、 旧バージョンからの移行のために互換レベルを残してあるだけで、新規タスクでは推奨しないようになりました。

この動きは、実は V9 から始まっており、V9 ですでにイベントハンドラによるイベント指向プログラム作成の基本 機能が実装されていました。しかし、旧バージョンとの互換性を重視したため、レコードメインも正式サポート機 能としてそのまま残されており、V9 および V9Plus で作成されたアプリケーションは、開発者の選択により、レコー ドメインによるもの、イベント指向によるもの、あるいはその混合という形になっているのが実情でした。

V10 では、この中途半端な状態から一歩進み、レコードメインを使わずイベント指向でプログラムを作成するよう、明確に方向が示されました。

この流れの目指すところは、次のようなものがあります。

- 書きやすく読みやすいプログラムを作るため。本書で具体的に見ていきますが、レコードメインのプロ グラムより、イベント指向のプログラムの方が、書くときにも読むときにもすっきりと見通しのよいものに することができます。
- V10機能を最大限生かすため。V10はイベントをベースとした機能が多数あります。その機能を最大限 生かすために、イベント指向の考え方を十分に理解することが必要になります。
- 将来の Rich Client をにらみ。本書執筆の時点(2007 年 10 月)で、V10 の大きな新機能として Rich Client (コードネーム) が計画されており、開発作業も進んでいます。Rich Client においては、レコード メインはサポートされておらず、イベント指向プログラミングのみがサポートされています。

このような理由から、V10に移行してゆく今が、レコードメインからイベントへの考え方の切り替えのよい機会であろうと思われます。

本書の説明では、V8 から V10 への移行を前提として例をとりあげましたが、V9/V9Plus ではレコードメイン、イベントプログラム両方がサポートされているので、本書の考え方は、

- V8 → V9/V9Plus イベント指向プログラム
- V9/V9Plus レコードメイン \rightarrow V10

という移行においてもほぼそのまま当てはめることができます。



1.2 本書の目的と範囲

本書は、レコードメインによる Magic プログラミングに慣れている Magic 開発者が、イベント指向のプログラミン グ方法を理解し、移行することができるよう、概念と機能およびプログラム方法について具体例をあげて説明し ます。

V10 でのイベント関連機能については非常に幅の広い話題がありますが、本書ではレコードメインからの移行 に重点を置き、オンラインプログラムの画面制御に関連する話題だけにフォーカスします。具体的には、

- コントロールレベルのハンドラ(コントロール前処理、コントロール後処理、コントロール検証 ハンドラ)
- 項目変更ハンドラ
- プッシュボタンにおけるイベントの利用
- ズームの実現方法

などが含まれます。

逆に、重要なイベント関連機能でありながら、本書に含まれないか、あるいはごく簡単にしか触れない話題には、次のようなものがあります。

- タスク、レコード、グループレベルでのイベントの扱い。
- 「イベント実行」コマンド。
- イベントタイプ(システム、内部、ユーザ定義、タイマー、式、エラー、ActiveX)。イベントとしては、主に ユーザ定義イベントのみを扱っています。
- GUIモデルにおけるイベント関連パラメータ。プッシュボタンでの「実行イベント」のみを扱っています。
- ユーザ定義イベントでの「強制終了」パラメータ(N=なし、E=編集、C=コントロール、R=レコード更新前、 P=レコード更新後)。ズームの章で、「E=編集」の動作について説明している他は、説明を省略しています。



1.3 前提条件

本書の読者は、次のような知識と経験があることを前提としています。

- dbMAGIC V8 レベルの Magic の知識を持っている。
- 旧バージョン (V9Plus 以下)でレコードメインを使ったプログラムを作ったことがあり、レコードメインの動作仕様について理解している。

この他、Magic Studio V10のタスクエディッタの基本(画面構成、データビューやロジック、フォームなどの作成・ 修正のための操作方法など)を理解していればベターですが、必須ではありません。本書では V10のイベント 指向プログラミングになれていない読者も念頭において、V10での操作方法も簡単に説明を加えています。 V10の操作にすでに慣れている読者は、その部分は読み飛ばしても構いません。

そして、次のような問題意識を持っている開発者を対象として書かれています。

- レコードメインで書いてきたものを、イベント指向プログラムでどのように書いていけばよいのかよくわからない。
- イベント指向プログラムで作ってみたことはあるが、まだ体系的に理解していないので、不安が残る。

本書がイベント指向という新しいパラダイムに移行していく一助となれば幸いです。



本書で出てくるアプリケーションは、V8 で作られた実際のアプリケーションを V10 に移行したもので す。現実の大きなアプリケーションの一部なので、サンプルとしてご提供できませんが、ご了承くだ さい。説明に出てくるプログラムは、主に受注入力を行うタスクです。

第2章 旧バージョンのレコードメインの動作

V10のイベント指向プログラミングについて説明する前に、本章でまずは旧バージョンでのレコードメインの動作について説明します。この内容は、レコードメインで Magic プログラムを作成されてきた開発者にとっては既知の内容と思いますが、おさらいと頭の整理の意味で一章を割いて解説します。

2.1 タスクエディッタの構成

最初に、タスクエディッタの構成について復習しましょう。

下図は V8 のタスクエディッタの画面です。

V8 のタスクエディッタ

		. 1	「ブレイク」処理レ^	°ŀ		前処理	ж) [後処理	トランザ・クジョン) 「17	r-	
レコート、ダスク、	107300		1 レコー	ř.		0	91	8	L=おu-り	A=	7ボート	×
ブレイカレベルたど	<u> </u>		2 タスク			0		7	Yes	A=	78*-1	`
JUN JUN MAC												
		処理テー	ブル・レコード メ	イン								
	ί μ	処理コ	マンド	「内容	ſ		範囲		[位置付	ר [D~	条件
処理テ―ブル	1	빈카	¥=変数 :	V.商品DLRC	代入	0	0	0	0	0 8	С	Yes
	2	也外	V=変数 : :	Y.物流拠点省略值DLRC	代入	0	0	0	0	0 8	C	Yes
レコードメインでは	3	11/21	Y=変数 : :	▼.物流拠点(開始)DLRC	代入	0	0	0	0	0 8	C	Yes
	4	비사	V=泼锻 : ·	↓ V.初流拠点(終了)DLRC	- 代入	0	- 0	0	0	0 8	0	Yes
● ナーダ正義	0	2005	V-3636X : :	V. 39710#DERG		U		U		0 5	U	Tes
● オンライン画面		երնե	- R=実行-ね・	受注#	(ቶ አ	29	29	29	0	0 8	C	Yee
	8	アクション		KBPUT (')次項目'ACT)		20		20	展?	? S	F	7
上での操作	9	也小	R=実データ : :	受注明解释	代入	24	. 0	0	28	0 8	C	Yes
	10	ブロック		[P.\$Z\$+* = 'C' OR P			_			C	C	7
を正義	11	3-11	P=プログラム: 56	SO選択 商品一覧	٨°۶	3	77-6	0	Doク N	o B	С	Yes
	12	빈카	R=実デ-タ: :	商品#	代入	0	0	0	0	0 8	С	Yes
	13	电外	R=実データ:	は 商品Sub#	代入	0	0	0	0	0 8	С	No
	14	909	Q=照会 : 1	商品	1)7"	oDX 1	<u>NR</u>	A=昇順	良し	W		Yes
	15	11/21		商品#	- <u>15</u>	0		0	39	39 2	U	No
	16	10/7F	N-夫/ ジビン Ra宇託-h・	PBBBSUD# 相校刑册	- 15/	0	- 0	0	40	40 5	C C	No
	19	10/1	R=実デーク :	商品名	100	0	- 0	0	0	0 8	0	No
	19	セレクト	R=実データ: 1	×7°47	代入	0	0	Ŭ	0	0 5	C	No
	20	セレクト	R=実データ: 1	単価	代入	0	0	0	0	0 8	C	No
	21	もレクト	R=実データ: 1	; 原価_CDROM_輸入	代入	0	0	0	0	0 S	С	No
	22	11/21	R=実データ: 22	原価	代入	0	0	0	0	0 S	С	No
	23	もりか	R=実データ: 23	商品如7°	代入	0	0	0	0	0 S	С	No
	24	빈카	R=実データ: 2:	□ 商品タイプ2	代入	0	0	0	0	0 8	С	No

ここで見るように、画面は大きく分けて、上下の二つのテーブルからなります。

上のテーブルは レベルテーブル と呼ばれ、レコード、タスク、ブレイクなどのレベルからなります。各レベルに はそれぞれ前処理と後処理とがあり、トランザクションやエラーの設定を行えます。レコードレベルだけは特別 に レコードメイン のレベルがあります。

下のテーブルは、**処理テーブル**と呼ばれ、各レベルにおいて実行すべきコマンドが記述されます。本書ではレ コードメインが話題ですので、レコードメインの処理テーブルを図では表示しています。

レコードメインでは、セレクトコマンド、およびリンクコマンドによって、このタスクで利用するデータ項目を定義しており、その間に、オンライン画面上での操作を、コールコマンド、項目更新コマンド、その他の処理コマンドにより記述しています。

2.2 レコードメインの基本的考え方

レコードメインでオンライン画面操作を記述する場合の基本的な考え方は、

カーソルの動きに応じて、セレクトコマンド間に処理コマンドを記述する

というものです。ここで、セレクトコマンドは画面上に表示されているデータ項目に対応したものであり、処理コ マンドは、項目から項目にカーソルが移動するときに、処理する内容を記述します。

具体的な例を挙げてみると、下図では、フォームとそれに対応するレコードメインを示しています。実行時、カー ソルが「受注数量」の項目 (72 行目のセレクトコマンド)にあるとき、ユーザが Enter/Tab キーを押して、その次 の「備考」項目 (75 行目のセレクトコマンド)にカーソルが移動するとします。そのときには、この二つのセレクト コマンドの間にある二つの項目更新コマンド (73 行目と 74 行目)とが実行されます。



2.3 レコードメインのフローパラメータ

次に、レコードメインでのコマンドについて、もっと細かく見ていきましょう。

2.3.1 フローパラメータ

フローパラメータは、レコードメインの各コマンドで、「フロー」と書かれている欄に設定することのできる、二つの パラメータです。このパラメータで、コマンドが有効になる(実行される)条件を指定することができます。



最初(左側)のパラメータは、「フローモード」と呼びます。これには、

- S=通常
- F=高速
- C=両用
- B=前置
- A=後置

があります。

このうち、「S=通常」と「F=高速」については、次節以下で説明します。

「C=両用」は「S=通常」と「F=高速」とを兼ねるもので、「常に有効」という意味になります。

また、「B=前置」と「A=後置」は、ズームを実現するために設定するモードであり、第8章「ズーム」で説明します。

2番目(右側)のパラメータは「方向」パラメータであり、これには、

- F=前方
- B=後方
- C=両方向

があります。

これはカーソル移動方向によりコマンド実行の有効/無効を決めるものです。

「C=両方向」というのは「F=前方」と「B=後方」を兼ねるもので、やはり「常に有効」の意味となります。

2.3.2 通常モード

フローモードの「S=通常モード」というのは、Enter/Tab キー(「次項目」アクション)で動く場合のモードです。



dbMAGIC V5 以来、「次項目」アクションの標準キー割り当ては Tab キーですが、ユーザの利用の 便のために Enter キーを「次項目」アクションに割り当てているアプリケーションが多いので、本書 では「Enter または Tab キー」あるいは「Enter/Tab キー」としています。

例えば、下図では 定価、販価、原価、仕切、数量、・・・ などの項目がありますが、カーソルがこれらの項目間 をEnter/Tab キーによって一つづつ動いていく場合には、通常モードとなります。

S=通常モード(Enter/Tabキーで動く場合のモード)								
	##. ######							
	定価/販価/原価 仕切/数量 売上/原価/粗利 沙개生成依頼	備考						
		XXXXXXXXXXXXXX ▲						

なお、Magic には「次項目」アクションに対応する「前項目」アクションもあり、標準キー割り当てでは Shift+Tab キーなどに割り当てられていますが、「前項目」アクションにより、逆方向に(後ろから前に)カーソルが動く場合 にも、通常モードとなります。

2.3.3 高速モード

高速モードというのは、「次項目」「前項目」アクション以外の原因によってカーソルが動く場合のモードです。これには、

- マウスクリック
- 「↑」「↓」キーによるレコードの移動
- ESC キーによるタスク終了

などでカーソルが動く場合が該当します。

なお、マウスクリックの場合には、たとえ隣接する項目に移動する場合であっても、高速モードとなります。



2.3.4 方向 パラメータ

「方向」パラメータは、カーソルの動く方向によって、コマンドの有効/無効を制御するパラメータです。

- 「F=前方」に設定されている場合には、順方向にカーソルが動く場合にのみ、コマンドが実行されます。
- 「B=後方」に設定されている場合には、逆方向にカーソルが動く場合にのみ、コマンドが実行されます。
- 「C=両方向」に設定されている場合には、いずれの方向にカーソルが動いてもコマンドが実行されます。

2.3.5 コマンド実行順序

「方向」パラメータに関連して注意しておかなければならないのは、処理コマンドが二つ以上ある場合には、カーソルの移動方向によって、処理コマンドの実行順序が反対になる、ということです。即ち、

- 順方向(上から下)にカーソルが動く場合には、コマンドも上から下に実行される。
- 逆方向(下から上)にカーソルが動く場合には、コマンドも下から上に実行される。

ということです。

例えば、下図のようなプログラムでは、カーソルが「受注数量」と「備考」の間を移動するときに、二つの項目更 新コマンドが実行されるように設定されています。このとき、

- カーソルが「受注数量」から「備考」に向けて、順方向に移動する場合には、受注金額の項目更新(73 行目)がまず実行されてから粗利金額の項目更新(74 行目)が実行されます。
- 逆に、カーソルが「備考」から「受注数量」に向けて、逆方向に動く場合には、下にある粗利金額(74 行 目)が先に実行され、つぎに上にある受注金額(73 行目)が実行されます。



通常のプログラミング言語では、実行は常に上から下に向けて行われるものですが、Magic のレコードメインの ように、カーソルの動きによって上から実行されたり下から実行されたりするのは、Magic に独特な実行方式で す。これは Magic 初心者が戸惑う Magic ルールの一つです。

また、アプリケーション開発者にとっては、順方向に実行された場合と逆方向に実行された場合とで結果が異 なるようなことになりかねないものであり、カーソルが順方向に動く場合と逆方向に動く場合とで矛盾なく動作す るように考慮して設計する必要が出てきます。

2.4 レコードメインの書き方ABC

以上の説明から、レコードメインを使ったプログラミングの方法について、

- A) どこで?
- B) いつ?
- C) 何を?

という観点で整理してみると、次のようになります。

レコードメインの書き方

- A) どこで?
 - → ある項目(セレクトコマンド)と別の項目(セレクトコマンド)の間で
- B) いつ?
 - → カーソルの動くとき
 - Enter/Tab で動く場合のみ実行させたいなら、「S=通常」モード
 - ↓ ↑ キー、ESC、マウス等で動く場合のみ実行させたいなら、「F=高速」モード
 - 常に実行させるなら、「C=両用」モード

※特定の方向だけに限定したければ、「方向」に「F=前方」または「B=後方」を設定する。

C) 何を? → セレクトコマンドの間に、実行するコマンドを記述。



これは簡略化したもので、実際のレコードメインの動作にはもっと複雑なルールがあります。詳細な 説明は省略しますが、本章で説明した以外に、次のような要因がレコードメインの処理の流れに影 響を与えます。

- データビューが修正されているかされていないか。
- エラーコマンド (エラーオプション) が実行されたか。
- ブロックコマンド、セレクトコマンド、処理コマンドなどの位置関係と、フローモードや条件式などの組み 合わせ。

これらを正確に理解しようとすると、レコードメインフローモードは結構難しいブラックボックスです。Magic 初心 者にとっては、学習に時間のかかる壁になり、開発者にとっては、設計やデバッグに時間がかかる原因となり ます。

2.5 レコードメインの問題点

ここで、レコードメインによるプログラミングの問題点について考えてみたいと思います。

2.5.1 記述性

キーボードのみで操作していた DOS の時代には、カーソルの動きを処理の流れと合わせて記述するレコードメインの記述方法は自然で直感的であり、また、一つの「レコードメイン」という処理テーブルの中に多くの意味を 多重的に指定することのできる、という意味で、記述性の高い方法でした。

しかし、キーボードに加えてマウスが導入され、WindowsのGUI部品が多用されるようになり、更にはタイマー や式、エラーなど、ユーザ入力以外の要因で割り込み的な処理の記述をしたくなるアプリケーションが増えるに 従って、レコードメインでの記述では制御が複雑になって実装が困難な場面が出てくるようになりました。

2.5.2 プログラムの可読性

プログラムが複雑になるにつれ、プログラムの見通しの悪さも目立つようになってきました。例えば、下図はレ コードメインの例です。

	I	# [7	ドレイク	如那	‼∧°I	և	[前処理「ソ	わ	後処理	トランサドクシ	a)	I7-				レコードメイン
		1		L=	1 K	· ·		0	91	8	L=カンロック		A=71	1°-1		-	ドニズいつけがち
		2		タス	ク			0		7	Yes		A=Pi	i'-			ここでいう何か起
																- I	るのか 目通しが
	処理テ	ーブル	: レコ	۲	メイ	ン ――						/				_	
	(如理	<u>יבר</u>	12			内容			節田		「位置付		[]n-	. [冬件		し <u>、</u>
	1 1/2	<u></u>	- 5教	:	1	Y.商品DLRC	代入	0	4000	0		0	S	C	Yes		- 0
	2 11/21	V=3	変数	:	2	V.物流拠点省略値DLRC	代入	0	0	0/	0	0	S	C	Yes		
	3 tl/7ト	V=3	変数	:	3	V.物流拠点(開始)DLRC	代入	0	0	0	0	0	S	C	Yes		
	4 セルクト	V=3	変数	:	4	V.物流拠点(終了)DLRC	代入	0	0	0	0	0	S	С	Yes		
	5 tl/h	V=3	変数	:	5	Y.ÿU7₩#DLRC	代入	0	0	0	0	0	S	С	Yes		
	6																
	7 빈/가	R=3	レデータ	:	1	受注	代入	29	29	29	0	0	S	С	No		
	8 アクショ	2		:	1	KBPUT ('次項目'ACT)					戻	??	S	F	7		
	9 tl/h	R=3	転ぶり	:	2	受注明解衅	代入	24	0	0	28	0	S	С	Yes		
1	0 7°Dy	5				[P. タスクモード = 'C' OR P.							C	C	7		
1	1 346	P=7	[•] •ロク*ラ	s:	566	SO選択 商品一覧	N°5	3	78-6	0	D92	No	В	С	Yes		
1	2 빈/가	R=3	レデータ	:	3	商品#	代入	0	0	0	0	0	S	С	Yes		
1	3 tl/h	R=3	レデータ	:	4	商品Sub#	代入	0	0	0	0	0	S	С	No		
1	4 900	Q=Ħ	<u> </u>	1	16	商品	わディ	わえ 1	順	A=昇順	戻	LW			Yes		
1	5 빈/가	R=∋	モデータ	:	1	商品#	代入	0	0	0	39	39	S	С	No		
1	6 tl/7h	R=3	モディータ	:	2	商品Sub#	代入	0	0	0	40	40	S	C	No		
1	7 11/271	R=9	見ディータ	:	9	規格_型番	代人	0	0	0	0	0	S	C	No		
1	8 21/21	K=3	モナシータ	:	<u></u>	商品名		0	0	0	0	0	8	U	No		
1	9 20/21	K= 3	モナシージ ローン ト	:	11	37°43 2017	15	0	0	0	0	U	2	U	No		
2	0 20/21	K= 3	モディージ ローシート	:	13	単値	15A	0	0	0	0	0	2	U O	No		
2	1 20/21	K= 5	モデュージ ロージート		16	原1曲_CURUM_輸入	15/	0	0	0	0	0	2	0	NO		
2	2 20/21	K=3	モナシージ ローシート	:	22	京日につい	15A	0	0	0	0	0	2	U O	No		
2	3 20/21	K=3	モデュージ ロージート	•	28	商品外グ	15A	0	0	0	0	0	2	0	NO		
- 24	4 セレクト	K= ja	モナシータ		29	商品94/22	1ta	U	U U	U	U U	U	2	υļ	No	-	

これをぱっと見ただけで、どの項目からカーソルが動くときに、何が実行されるのかわかりにくいものがあります。 ユーザの実行時の目から見ると、ある項目の直後に行うべき処理コマンド(例えば、データリンクのエラーチェッ クなど)が、レコードメイン上では、途中に多くの非パーク項目に挟まれていて、かなり離れたところに設定され ているということもよくあります。どの処理コマンドがどの項目で実行されるかは、セレクトコマンドの「条件」欄を チェックして、前後の文脈を見ていかないとわからないことになります。

また、オンライン画面での処理に関連して、イベントテーブルの可読性の悪さも問題になります。次の図は、同

タスクイベントテーブル

47	ታፈላጊት: 718	3- 80入力 受注情	鍜1 (597)							
t [ホットキー	アクション	[経過時間	五	スコープ	コール	Prog/Task	N°5X-9	0-97	タスクイベントテーブル
1		ユーザアクション1	00:00:00	0	T=922	P=プログラム	626	1	No	ドード マロビッジン
2		ユーザアクション2	00:00:00	0	T=927	P=7°07°56	620	2	No No	とのホタンで何か呼び
4		ユーザアクション4	00:00:00	0	T=922	P=7°D2°56	618	2	No	出されるかわからない。
5		ユーザアクション5	00:00:00	0	T=922	P=プログラム	616	3	No	
6		ユーザアクション6	00:00:00	0	T=922	P=プログラム	621	2	No	
- 7		ユーザアクション7	00:00:00	0	T=922	P=プログラム	561	1	No	

この図の例を見てわかるように、タスクイベントでは「ユーザアクション n」と、プログラム番号とが見えるだけで、 「どのボタンを押したときに、どのような処理が行われるのか」という肝心な内容が、これを見ただけではほとん どわかりません。どのボタンにどのユーザアクションが割り当てられており、何番のプログラムが何をするのか、 という記録と対照しない限り、プッシュボタンと処理の関連は一見してはまずわからないといえましょう。

2.5.3 複雑なブラックボックス

2.4 「レコードメインの書き方ABC」で簡単に触れましたが、レコードメインの動作は詳細を見ると意外と複雑な ブラックボックスです。このため、慣れた開発者であっても、期待通りの動きにならないときに、どうしてそうなる のかすぐにわからないため、修正に時間がかかってしまうことがあります。

<u>第3章 V10 のイベント指向プログラミング</u>

V10 では、レコードメインを基本的に廃止し、全面的にイベント指向でプログラムを作るようになりました。イベント指向のプログラミング方法は、複雑なブラックボックスとなってしまったレコードメインではなく、よりシンプルで 直感的にわかりやすい方法でプログラミングを行えるようになることを目指しています。

本章では、V10でのイベント指向プログラミングの基礎について簡単に説明し、次章移行で個々のケースについて例を挙げながら説明していきます。

3.1 タスクエディッタの構成

Magic のプログラムの単位であるタスクの作成はタスクエディッタで行いますが、V10になってタスクエディッタのユーザインターフェースが大きく変更になりました。

まず、タスクエディッタは、

- データビュー
- ロジック
- フォーム

という三つのタブで大分類されています。



データビューというのは、このタスクで利用するデータの定義を行うものです。旧バージョンでは、レコードメインで、セレクトコマンドおよびリンクコマンドで行っていたものです。V10のデータビューでは、データの定義だけを行い、レコードメインで行っていたようなロジックの記述は行いません。

ロジックというのは、文字通りタスクのロジックを記述するものです。

これには、タスクレベル、レコードレベル、ブレイクレベル(バッチタスクのみ)、コントロールレベル(オンラインタ スクのみ)のそれぞれ前処理と後処理、コントロール検証、項目変更、およびイベントハンドラを定義します。 これは、旧バージョンのタスクエディッタと比べ、タスク、レコード、ブレイクレベルは同じで、コントロールレベル (前処理、後処理、検証)はレコードメインの中に記述していたオンライン画面での動作に相当します。

フォームは、旧バージョンでのフォームテーブルと同じですので、ここでは説明を省略します。 以上の説明をまとめ、機能別に旧バージョンと V10 との対応を図示すると、下図のようになります。



ロジックタブに記述するハンドラの種類については、次節以下により詳しく説明していきます。

3.2 ハンドラの種別とタイプ

ロジック 画面では、ハンドラ を単位として記述します。V10 でのハンドラには次のような種類のものがあります。



タスク/グループ/レコードレベルハンドラ については、以前のバージョンと変わりありませんので、本書では説明を省略します。

コントロールレベルハンドラはオンラインタスクにのみ作成することのできるハンドラであり、前処理、後処理、 検証というタイプがあります。これはオンライン画面上で、カーソルが移動したときに実行されるもので、第4章 コントロールレベルハンドラで詳しく見ていきます。

イベントハンドラは、イベント指向プログラミングの一番重要となるハンドラであり、これには内部イベント、ユー ザ定義イベント、システムイベント、その他のタイプがあります。イベントハンドラは V9 ですでに導入されていま したが、V10 でもほぼそのままの形で利用できるようになっています。V8 ではアプリケーションイベントテーブル およびタスクイベントテーブルで定義されていたものですが、V8 のものにくらべて使いやすい形になっています。

イベントハンドラについては、第7章 プッシュボタン、および第8章ズームで、ユーザ定義イベントを使った例についてだけ説明します。

項目変更 ハンドラというのは、V10の新機能であり、項目の値に変更があった場合に実行されます。V9Plus では、「コントロール変更」ハンドラというものがありましたが、項目変更ハンドラはこれを更に広く利用できるよう拡張したものです。項目変更ハンドラの詳細は、第6章 項目変更ハンドラで説明します。

3.3 ハンドラの書き方

V10 でのハンドラは、どの種類のハンドラであっても、すべてロジック タブに定義します。

ーつのタスクには一般には複数のハンドラがありますが、必要がなければ一つも書かずに、空のままでもかまいません。例えば、APGで作成したオンライン照会プログラムでは、データビューとフォームの定義だけからない、ロジックが必要ないので、ロジックは空です。

一つのハンドルは、ヘッダ行と詳細行とからなります。

ヘッダ行はハンドラの最初の行であり、ハンドラごとに種別とタイプ他を定義するもので、ハンドラレベル、ハンドラタイプ、コントロール名、その他 (ハンドラタイプによる)を指定します。

ヘッダ行には、コントロール名を指定することができます。ここには、フォーム上のコントロールに付けられたコントロール名を指定し、カーソルがそのコントロールにある場合にだけ、そのハンドラが有効となります。

コントロール名の指定は、コントロールレベルハンドラ(コントロール前処理、後処理、検証)では必須であり、イ ベントハンドラでは任意です。イベントハンドラでコントロール名が指定されなかった場合には、そのイベントハ ンドラは、カーソルの位置に関わらずいつも有効となります。

また、ヘッダ行には **条件** を式で設定することもできます。条件が設定されていた場合、このハンドラはその条件が真になる場合にだけ有効になります。

詳細行は、このハンドラによって処理される処理内容を、項目更新、コール、アクションなどの Magic の処理コ マンドで記述します。詳細行は、一つのハンドラに対して複数行記述することができます。下図で見るように、詳 細行は、ヘッダ行よりも数文字分インデントされて表示され、ヘッダ行と詳細行とが見てするわかるようになって います。また、色定義ファイルを使うことによって、ヘッダ行を色分けして表示することもできるようになります。



3.4 ハンドラの作成手順

ここでは、上に例に示した「受注数量」のコントロール検証ハンドラを作成してみます。コントロール検証ハンドラの使い方は、次章に解説します。



ハンドラの作成手順

- 1. タスクエディッタで、ロジック タブをクリックして、 ロジック画面を表示させます。
- 新しいハンドラを作成するには、まずヘッダ行 を作成します。 ヘッダ行を作成するには、ポップアップメニュー の「ヘッダ行作成」(Ctrl+H)を選びます。 → 新しくヘッダ行が作成されます。
 データビュー ロジック フォーム 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 37
 3
- 3. ヘッダ行の先頭で、ハンドラの種類として、コ ンボボックスから「C=コントロール」を選びます。
- データビュー
 ロシック
 フォーム

 37
 スペム(2)

 第7
 スペム(2)

 第7
 マックド行作成(1)

 マックド行作成(1)
 マックド行作成(1)

 ア・カド
 マーカド

 マーカド
 マーカド

 マーカト
 マーカト

 マーカト
 マーカト

 マーカト
 マーカト

 マーカト
 マーカト

 マーカト
 マーカト

 マーカト
 マーカト

 マーカト
 マーカト
- ヘッダ行の2番目のカラムに進み、ハンドラのタイプとして、コンボボックスから「V=検証」を選びます。
- ヘッダ行の「コントロール名」欄に進み、ズームします。
 → このタスクのフォームで定義されているコ

	-		-		•			_				-
ントロ	_	-ม	⁄名	の -	-覧	がま	表示	され	ょ	す。	0	

38	C=3)10-1	₩=検証	ИСС	
	コントロール一覧			
	ク [*] ル-フ°名: <u>全てのコントロ-ル</u> SO受注明細			コントロール名: 仕切 受注原価単価 受注原価単価 受注原価単価 や 新品# 物品# 物流拠点#_END 物流拠点#_START メデ [*] ィア
				選択 ++ンセル

- 6. この中から「受注数量」を選択します。
 - →「コントロール名」欄に、選択したコントロール名が表示されます。

以上で、ヘッダ行ができあがりました。

	88 C=コントロー】 Y=検証 コントI 受注数量	条件 Yes
7.	次に詳細行を作成します。 詳細行を作成するには、ポップアップメニュー の「行作成」(F4)を選びます。 → 新規詳細行が作成されます。	38 C=つとの一県 V=拾目 ス ² ーム(Z) 一 () 一 () () 〇 イッネドブイキ成(D) 〇 イッネドブイキ成(D) 〇 () 〇 () 〇 ()
8.	詳細行の先頭のコマンドのところで、ズームし ます。 → コマンドを選択するためのコンボボックス が出てきます。	 38 □ C=コントロート Y=検証 コントI 受注数量 39 =コメンI ● =コメンI ● =コメンI ● =コメンI ● =コメンI =コメンII =コメンIII =コメンIII =コメンIII =コメンIII =コメンIII =コメンIII =コメンIII =コメンIII =コメンIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII

9. この中から、「項目更新」を選びます。

10.あとは、旧バージョンでのコマンドの場合と同じ方法で記述することができます。

38 E	C=3)ha-6	¥=検証	コント 受注数量				
39	傾目更新	V=項目	NS 受注金額	値:	56	受注単価*受注数量	条件:Yes

11.必要に応じ、「行作成」(F4)を繰り返し、一連のコマンドを記述してください。

38	□ C=コントa-6	Y=検証	コント 受注数量			
39	項目更新	∀=項目	NS 受注金額	値:	56	受注単価*受注数量
40	傾目更新	V=項目	NT 粗利金額	値:	57	(受注単価-受注原価ف条件:Yes

第4章 コントロールレベルハンドラの基本

本章と次章では、コントロールレベルハンドラについて扱います。

本章ではまずコントロールレベルハンドラの動作の基本について説明します。次章ではそれをもとにして、具体的な例を使って、レコードメインで書いていたものをコントロールレベルハンドラで書き直していきます。



4.1 コントロールレベルのハンドラの動作

オンライン画面のカーソル移動時の操作は、以前のバージョンではレコードメインの中で、セレクトコマンドの間 に処理コマンドを入れて定義していましたが、V10のイベント指向プログラムでは、コントロールレベルのハンド ラに記述します。

コントロールレベルハンドラは、各コントロール(表示項目)ごとに、

- コントロール前処理
- コントロール検証
- コントロール後処理

の3種類のハンドラを設定できます。

これらのハンドラは、実行されるタイミングが異なります。下図は、「受注数量」というコントロール名が付けられたエディットコントロールに、それぞれコントロール前処理、コントロール検証、コントロール後処理が定義されていた場合に、どのタイミングでそれぞれのハンドラが実行されるかを描いたものです。(「CTRL」は「コントロール」の意味です。)



最初にカーソルが「仕切」項目にあり、Enterキーなどにより「受注数量」項目にカーソルが移動したとすると、そのタイミングでコントロール前処理が実行されます。

次に、再度 Enter キーにより、カーソルが「受注数量」から「備考」項目に移動したときには、コントロール検証が 実行され、ついでコントロール後処理が実行されます。

このように、「受注数量」にカーソルがパークする場合には、コントロール前処理(入るとき)、検証・後処理(出る とき)がすべて実行されます。

ー方、カーソルが「受注数量」にパークしない場合には、前処理および後処理は実行されません。すなわち、最初にカーソルが「仕切」(あるいは、タブ順序で言ってそれ以前の項目)にあったとき、マウスなどで「備考」項目 (あるいは、タブ順序で言ってそれ以降の項目)をクリックして、カーソルが「受注数量」にパークせずに飛び越し て移動した場合には、前処理および後処理は実行されません。しかしこのような場合でもコントロール検証は実 行されます。

4.2 コントロールレベルハンドラの書き方ABC

以上のような動作をすることから、ある項目(正確には、その項目に対応するフォーム上のコントロール)に着目して、コントロールレベルハンドラの書き方は以下のようになります。

- どこで?
 - ある表示項目(例:受注数量)において・・・
- いつ?
 - カーソルが項目にパークするときにだけ実行するなら
 - 入るときの処理 → コントロール前処理に書く
 - 出るときの処理 → コントロール後処理に書く
 - ズームのときの処理 → ズームハンドラに書く(第8章ズームで説明)
 - カーソルが動くときに常に実行するなら
 - コントロール検証に書く
- 何を?
 - ハンドラの中にコマンドを記述

4.3 レコードメインとコントロールレベルハンドラとの違い

ここで、コントロールレベルのハンドラの動作と、旧バージョンでのレコードメインの動作との違いについて、一言加えておきます。

4.3.1 カーソルの移動方向

旧バージョンのレコードメインでは、カーソルの移動順序(タブ順序)がセレクトコマンドの順序と対応しており、セレクトコマンドの間に、カーソル移動時の処理内容を記述していました。

例えば、カーソルが「受注数量」にパークする際に、上(仕切)から来る場合には「仕切」のセレクトコマンドと 「受注数量」のセレクトコマンドの間の処理コマンド(項目更新①と項目更新②)が実行されますが、下(備考) から来る場合には、「備考」と「受注数量」の間のセレクトコマンドの間の処理コマンドが逆方向に(つまり、項目 更新④、項目更新③の順序で)実行されました。

逆に、カーソルが「受注数量」から出て行く場合には、下(備考)に向けて移動する場合には、「受注数量」と「備考」の間の処理コマンド(項目更新③と項目更新④)が順方向に実行されました。一方、上(仕切)に向けて逆方向に移動する場合には、「仕切」のセレクトコマンドの間にある項目更新②と項目更新①とが逆順に実行されます。



これに対し、V10のコントロールハンドラでは、カーソルの移動の方向に関わらず、つまり上から入る場合にも 下から入る場合にも、コントロール前処理が実行され、出る場合には上に行くときも下に行くときも、コントロー ル検証とコントロール後処理とが実行されます。また、各ハンドラ中のコマンドはいつも上から下に実行され、レ コードメインの時のように、逆方向に下から上に実行されることはありません。



この例外として、「エラー」コマンドを「R=復帰」モードで実行する場合があります。このときには、 エラーコマンドが実行された後、エラーコマンドからハンドラの最初のコマンドまでが逆方向に実 行されます。これは旧バージョンとの互換性を持たせるための機能で、詳細はリファレンスヘル プの ロジックエディタ > 処理コマンド > エラー を参照してください。

4.3.2 カーソル移動の原因

旧バージョンのレコードメインでは、カーソルが移動する原因 (Tab/Enter キー、マウスクリック、↑↓キー、ESC キーによるタスククローズ・・・) によって、フローモード (通常モード、高速モード) が区別されていました。

V10のコントロールハンドラでは、カーソルの移動の原因に関わらず、例えば Tab/Enter キーで入る場合にも、 マウスクリックで入る場合にも、コントロール前処理が実行され、コントロールから出る場合には、キーの種類 やマウスクリック、その他の要因の如何に関わらず、コントロール検証、コントロール後処理が実行されます。

	旧バージョンのレコードメイン	V10のコントロールハンドラ
カーソルの移動方向	影響される	関係ない
カーソル移動の原因	フローモードに影響	関係ない



旧バージョンとの互換性のために、V10のハンドラの中でもコマンド単位でフローモードを指定 できるようになっています。具体的な例は5.1「例①リンクの成功チェック」で説明します。リファ レンスマニュアルロジックエディタ>処理コマンドから、各コマンドの特性から、「フローモード」 「フロー方向」に関するページを参照してください。

第5章 コントロールレベルハンドラの例

本章では、具体的な例をあげて、コントロールレベルハンドラへ移行する例を見ていきます。

レコードメイン→コントロールハンドラの移行ポイントとしては、次のようなことがあります。

前章の説明にあったように、レコードメインとコントロールレベルハンドラとは1対1に対応しているものもあるけ れども、対応していないものも数多くあります。特に、カーソルの移動方向に関して、レコードメインは実行タイミ ングが大きく依存するのに対し、コントロールレベルハンドラは依存しませんので、カーソルの移動方向に依存 した処理をレコードメインで記述していた場合には、コントロールレベルのハンドラに移行しようとするときに面 倒なことになります。

レコードメインからコントロールレベルハンドラに機械的に変換しようとすると、一見すると必要以上に複雑で冗 長な結果となってしまい、また、100%の完全な変換は根本的には不可能であるため、いずれにしても手作業が 入ってしまうようになります。

このため、レコードメインからコントロールレベルハンドラに移行しようとする場合には、機械的に変換することよりも、「本来何をしたいのか?」の原点に立ち帰って、ハンドラ設定を考える方がはるかに良いプログラムとなります。

次節以下で、レコードメインからコントロールレベルハンドラへの移行について、いくつか例を挙げて説明していきますので、これを参考にして、プログラムを見直していってみてください。



以下に説明する内容は「考え方」を説明するために簡略化したもので、レコードメインの複雑な詳 細に依存した作りになっている場合には、全く同じ動きにすることが困難な場合もあります。

5.1 例① リンクの成功チェック

最初の例では、「『商品#』で商品マスタへのリンクを行い、マスタに登録されていない商品番号が入力された 場合にはエラーメッセージを表示させる」という、リンクの成功チェックを行うことを考えて見ます。

5.1.1 レコードメインでのやりかた

レコードメインの書き方 ABC「どこに?いつ?なにを?」に当てはめて考えてみると、次のようになります。

- A) どこに?「商品#」項目から次の項目に行くところで。
- B) いつ? カーソル移動の原因に関わらず(つまり、Enter/Tab キーでカーソルが移動する場合にも、マウスで移動する場合にも、ESC でタスクを終了させようとする場合でも)、リンクに失敗した場合には、常に実行しなければなりません。

C) 何を? 商品#が不正の由、エラーメッセージを表示します。(リンク結果を見て、条件付けします)

プログラム上では、次のようになります。



- レコードメインでは、リンク項目「商品#」を使ってリンクを行い、リンクの成功を戻り変数(下図では、14 行目のリンクコマンドの「戻」に設定されている変数 LW)に格納しています。
- 「どこで?」が「『商品#』項目から次の項目に行くところで」なので、「商品#」のセレクトコマンドと、次 にパーク可能なセレクトコマンドの間に書くことになります。
- ●「何を?」が「商品#が不正の由、エラーメッセージを表示」なので、「エラー」コマンドを使います。これは、「リンク終了」コマンド(41 行目)の後に設定しています。このエラーコマンドには、「条件」欄に式 37 番が設定されていますが、これは戻り変数 LW でリンク成功をチェックする式です。

リンク項目「商品#」(12 行目)から、エラーコマンド(42 行目)まで、多くのセレクトコマンドがありますが、 これらはすべてパークしない(「条件」欄が No になっている)ため、「商品#」から Enter/Tab で次に進 むと、エラーコマンドがすぐに実行されます。

● 「いつ?」は「カーソル移動の原因に関わらず常に実行」なので、フローモードは「C=両用」となります。

5.1.2 コントロールレベルハンドラでの書き方

これを V10 のコントロールレベルハンドラで書き直すと、次のようになります。 まず、コントロールレベルハンドラの書き方 ABC「どこで?いつ?何を?」に当てはめてみると、

- A) どこで? 商品#のエディットコントロールで
- B) いつ? このコントロールを通り過ぎるとき (カーソルがパークしてもしなくとも)常に。

C) 何を? 商品#が不正の由、エラーメッセージを表示します。(リンク結果を見て、条件付けします) となります。

従って、プログラム上は次のようになります。

- データビュー画面において、リンク項目「商品#」を使ってリンクを行い、リンクの成功を戻り変数に格納します。
- 「いつ?」は「コントロールを通り過ぎるとき常に」ですので、コントロール検証 ハンドラを使います。
- ●「どこで?」が「『商品#』のエディットコントロールで」ですので、このコントロール検証ハンドラの「コント ロール名」欄には、コントロール名「商品#」を指定します。
- コントロール検証ハンドラの詳細行には、「何を?」に相当するエラーコマンドを指定します。エラーコマンドの「条件」で、リンク結果を格納する変数を使って条件付けします。



0

V9 以降でプログラムを作成した場合には、コントロール名は自動的にデフォルトで項目名と同じものがつけられますが、V8 ではデフォルトで付けられないので、V8 からの移行アプリケーションではコントロール名が付いていない場合が多いです。この場合には手作業でコントロール名を指定してください。

5.1.3 微妙な動作の違い

レコードメインの場合、エラーコマンドは「商品#」のセレクトコマンドと、次にパークする項目のセレクトコマンド の間に書いたので、次項目に進む場合はエラーチェックがされましたが、前の項目に戻る場合にはエラーチェッ クがされませんでした。このため、間違って「商品#」に行ってしまったときにも、カーソルを元に戻してやりなお し、ということができました。

ー方、コントロール検証の場合には、カーソルが「商品#」から出るときにはカーソルの移動方向に関係なく(前 方向でも後方向でも)エラーチェックを行うようになります。従って、いったん「商品#」に入ったら、正しい値を入 力するまでは前にも後ろにも行けない、ということになってしまいます。

このような問題に対応するため、また旧バージョンとの互換性を保つため、V10でも、旧バージョンのフローパ ラメータの概念が残っており、各コマンドごとに「フローモード」と「フロー方向」を設定できるようになっています。

- 「フローモード」パラメータの設定値
 - S=通常
 - F=高速
 - C=両用
- 「フロー方向」パラメータの設定値
 - F=前方
 - B=後方
 - C=両方向

それぞれのパラメータの意味は、レコードメインの場合と同じです。

例えば、上記のリンクエラーチェックの問題に対応するには、カーソルが前方に移動する場合にだけチェックす るようにすればよいので、エラー コマンドの「フロー方向」パラメータを「F=前方」に設定してやればよいことにな ります。





- 1. 詳細行の「エラー」コマンドにカーソルを置きます。
- 2. 特性シートを開きます (Alt-Enter)。→ エラー処理コマンドの特性が表示されます。
- 3.「フロー方向」特性を「F=前方」に設定します。



「フロー方向」や「フローモード」特性の他に、Flow という関数があり、現在のフローモードをチェックすることができますので、条件式に使うことができます。Flow 関数の詳細については、リファレンスヘルプ 式エディタ > 関数ディレクトリ > Flow を参照してください。

5.2 例② 受注番号の後、条件により次の項目をスキップ

5.2.1 レコードメインでのやりかた

下図は、V8で作成したタスクのレコードメインの一部です。

受注#の値 KBPUT(' が	I後で、 R項目'AC	T)を実行				フローは S= 通常 F= 前方			
ģ	^见 理テー 	ブル:レコ マンド	– ۲	י אר ו	ン 内容	r	70-	「条件	
7	빈가	R=実データ	:	1	受注#		S C	Yes	-
8	アクジョン		:	1	KBPUT ('次項目	'ACT)	S F	7	
9	もしりト	R=実がら	:	2	受注明細關		S C	Yes	

ここでは、「受注#」から「受注明細#」に移動するときに、式7番で定義される条件が成立した場合に、 KBPUT('次項目')を実行します。条件式7番の内容についてはここでは立ち入りませんが、ここで、やりたい意 図は何かというと、

● 受注#から Enter /Tab で移動するとき、条件式 7 番 が成立したら、次の項目(受注明細#)をスキップ する。

ということです。フローパラメータの設定を見てみると、

- 「S=通常モード」となっているので、Enter/Tab キーで移動する場合だけ実行し、マウスやレコード移動 時には実行しない。
- ●「F=前方」となっているので、カーソルが前方向に移動する場合にだけ実行し、逆方向に(「受注明細#」 から「受注#」に)戻る場合には実行しない。

ということになります。

5.2.2 コントロールレベルハンドラでのやりかた

これと同じことをコントロールレベルハンドラで実現しようとするとどうなるでしょうか?「どこで?いつ?何を?」 で考えてみましょう。

- A) どこで? 「受注#」のエディットコントロールにおいて。
- B) いつ? カーソルがパークした後、別項目に出て行く時。(ただし、前方向に進む場合のみ)
- C) 何を? 項目をひとつスキップさせる。(条件式で条件付けする)

ということになります。

従って、コントロールレベルハンドラで書くと、次のようになります。

- ●「いつ?」は「カーソルがパークした後、別項目に出て行く時」なので、コントロール後処理 ハンドラに書きます。
- ●「どこで?」は「『受注#』のエディットコントロールにおいて」なので、ヘッダ行の「コントロール」欄には、「受注#」を設定します。
- ●「何を?」は「項目をひとつスキップさせる」なので、イベント実行コマンドで「次項目」イベントを発生さ

せます。

● 前方向にカーソルが移動するときにだけ実行させたいので、イベント実行 コマンドの「フロー方向」特性 に「F=前方」を設定します。



V10 でも KBPUT 関数はありますが、イベントを発生させるためには、イベント実行コマンドを利用 することを推奨します

5.2.3 別のやりかた

このように、レコードメインで行っていたことをコントロール後処理で 書き換えることが可能ですが、この例についてもっとよく考えてみる と、本質的には「受注明細#」のパークの有無を制御したい、という ことなので、「次項目」イベントを発行するよりは、そのものずばり、 「受注明細#」項目のパーク条件で制御した方がよいと言うことがで きましょう。

パークの有無は、レコードメインではセレクトコマンドの条件として記述していましたが、V10ではフォームエディッタで、コントロールの「パーキング可」特性に条件を記述します。(右図)

このように、レコードメインに書いていた方法を V10 で書き直す場合、 コントロールレベルのハンドラにする以外に、全く別の方法によるより直接的な方法もありえます。 V10 にするときには、別のよりよい方 法がないかを検討する良い機会でしょう。



5.3 例③ 項目更新による再計算ロジック

次の例は、「受注数量」の値をもとに、「受注金額」と「粗利金額」とを再計算するロジックを実現するものです。 再計算は、「代入式」を使えば自動的に行うことができるのですが、場合によっては、計算の条件などがあった りして、代入式ではなく、条件付きの項目更新コマンドで行うことがよくあります。ここでは、代入式ではなく、項 目更新コマンドで行う場合を考えて見ます。

5.3.1 レコードメインでのやりかた

レコードメインを使ったときには、次のように設計しました。

- A) どこで?「受注数量」と、次のパーク項目である「備考」の間で実行しますので、それぞれに対応するセレクトコマンドの間に記述します。
- B) いつ? 項目更新コマンドは、フローモードやパークの有無に関わらず常に実行しなければなりません。 従って、フローモードは「C=両用」とします。
- C) 何を?「受注金額」と「粗利金額」を計算する項目更新コマンドを実行します。



5.3.2 コントロールレベルハンドラでのやりかた

コントロールレベルハンドラを使う場合には、

- A) どこで?「受注数量」において行います。
- B) いつ? 項目を通り越した時点で。(カーソルのパーク有無には関係なく)
- C) 何を?「受注金額」と「粗利金額」を再計算します。

としたいのですから、次のように書きます。

- ●「いつ?」は「カーソルのパーク有無には関係なく項目を通り越した時点」ですから、コントロール検証 ハンドラにします。
- 「どこで?」は「『受注数量』において」ですから、ヘッダ行の「コントロール」欄には、コントロール名「受 注数量」を指定します。
- 「何を?」は「「受注金額」と「粗利金額」を再計算」ですから、レコードメインの場合と同様、項目更新コ



5.3.3 微妙な動作の違い

この例でも、レコードメインで行った場合と、コントロールレベルハンドラで行った場合とでは、表面上は出てこないけれども、微妙に動作が違うところがあります。

二つの項目更新コマンドの実行順序

第一には、「受注金額」の計算の項目更新コマンドと、「粗利金額」計算の項目更新コマンドの実行される順序 についての違いです。

レコードメインの時は、カーソルの移動方向により、コマンドの実行順序が異なりました。すなわち、カーソルが 順方向(「受注数量」から「備考」へ)に動く場合には、上から下に実行され、「受注金額」が先に計算された後に 「粗利金額」が計算されました。一方、カーソルが逆方向(「備考」から「受注数量」へ)に動く場合には、逆に、下 から上に実行され、「粗利金額」が先に計算されてから、「受注金額」が計算されます。

従って、もし「粗利金額」の計算が、「受注金額」の値に依存するような形で計算式が作られていれば、項目更 新コマンドの実行順序によって計算結果が変わってくることになります。すなわち、カーソルの動きが順方向か 逆方向かで、計算結果が異なることになります。

ここで使った例では、たまたま、お互いの依存関係がないように計算式が立てられていたので、どちらを先に実 行しても結果は変わりませんでしたが、一般的には、開発者は上から下に実行することを大前提として処理を 記述していることが通常で、逆方向の動きについては考慮し忘れることが間々あります。

レコードメインに書いた場合には、このように、上から下に処理が進む場合と下から上に処理が進む場合とが あるので、開発者は常に、どちらから実行がされても、結果に矛盾が起こらないように考慮を払いながらプログ
ラムを作らなければならないことになります。これが、レコードメインで書く場合の難しさを増やす要因の一つともなっていました。

コントロール検証の時は、カーソルの移動方向に関わらず、常に上から下に実行されますので、逆方向の動き に煩わされることなく、プログラムを作成することができます。

実行タイミング

レコードメインの場合には、ユーザが「受注数量」で数量を入力した後、Enterキーで「備考」へカーソルを動か すことを念頭においており、そのタイミングで「受注金額」と「粗利金額」を計算していました。

ところが、ユーザが「受注数量」を入力した後、先に進まずに、前の「仕切」項目に戻ったとすると、項目更新コマンドは実行されませんので、「受注数量」および「粗利金額」はもとのままです。

最終的には、タスクを終了するか、次レコードに移るタイミングで(F=高速モードで)この項目更新コマンドが実 行されるため、データベースには正しい値が格納されるのですが、ユーザの画面表示上ではそのことが見えな いままとなってしまいます。

コントロール検証で書いた場合には、カーソルの移動方向に関わらず、「受注数量」項目を出るタイミングで項 目更新コマンドが実行されますから、「備考」に進む場合も、「仕切」に戻る場合も、常に「受注金額」と「粗利金 額」が更新されます。

第6章 項目変更ハンドラ

前章では、コントロールレベルハンドラについて説明しました。本章以後では、それ以外のハンドラについて説明していきます。

まず、本章では、「項目変更」ハンドラについて説明します。



6.1 項目変更ハンドラとは?

項目変更ハンドラというのは、ある特定の項目の値に注目して、その値に変更があった場合に実行されるハンドラのことです。この場合、値の変更の理由としては、ユーザの入力による場合、項目更新コマンドや VarSet 関数の実行による場合、コールプログラムのパラメータとして渡され呼び出し先のプログラムで変更があったため、などがありえますが、項目変更ハンドラは変更の理由に関わらず、値に変化のあった場合に常に実行されます。



V9/V9Plus においては、「コントロール変更」というハンドラがありました。これはオンラインフォーム 上で、ユーザ入力によって値が変更された場合に実行されるハンドラでした。V10の項目変更ハン ドラと似ていますが、V10の項目変更ハンドラとは異なり、ユーザ入力による値の変更の場合にだ け実行されます。項目更新コマンドによる値の変更などの場合には、コントロール変更ハンドラは実 行されませんでした。



V10 では項目変更ハンドラがあるので、コントロール変更ハンドラはなくなりました。ただし V9/V9Plus でのコントロール変更ハンドラとの互換性をとるために、変更の理由(ユーザ入力による ものか、それ以外か)を、ハンドラのパラメータとして渡すようになっています。詳しくは、リファレンス マニュアル Magic エンジン > エンジン実行レベル> 項目レベル を参照してください。

6.2 再計算ロジックの例

前章 5.3「例③ 項目更新による再計算ロジック」では、「受注数量」のコントロール検証イベントハンドラを使って、受注金額と粗利金額とを計算させていましたが、このようなあるデータ項目の値の変更をきっかけとして実行すべきコマンドは、「項目変更」ハンドラにした方がわかりやすくなります。

項目変更ハンドラは、次のように記述します。



- 1. 項目変更ハンドラでは、ヘッダ行で「V=項目」、「C=変更」を指定します。
- 2. 変数項目を指定する欄がありますが、ここには、値の変化を監視すべき変数のシンボルを設定します。 この例では、「受注数量」(シンボルは NY)を指定しています。
- ハンドラの明細行には、実行すべき処理コマンドを記述します。
 この例では、「受注金額」と「粗利金額」を計算する項目更新コマンドを記述しています。

このように設定しておけば、項目 NY (受注数量)の値に変化があった場合にはいつでも、この二つの項目更新 コマンドが実行されます。



- 1. ロジック タブをクリックします。→ ロジック画面が開きます。
- コンテキストメニューから「ヘッダ行作成」(Ctrl+H)を 選びます。→ ヘッダ行が作成されます。
- 3. ハンドラの先頭のコンボボックスから「V=項目」を選択 します。



→「V=項目」が設定されるとともに、タイプが「C=変更」 に自動的に設定されます。

¥=項目	C=変更	??? <mark>??</mark>	

4. 変数欄に進み、ズームします。→ 変数一覧が表示されます。

NX NY	仕切 ☞ 愛注数量	N=娄y値 N=娄y値	受注明細 受注明細	
NZ	動的原価1	N=裝y/直	受注明細	
0A	動的原価2	N=娄y值	受注明細	
OB	実質粗利金額	N=娄y值	受注明細	
00	備考	A=文字	受注明細	
OD	沙洲神自動生成依頼	L=論理	受注明細	
0E	受注#	N=裝y值	99 71 4#	
OF	受注明細葉	N=裝y/直	99 71 4#	
OG	受注明細沙개#	N=数值	9y ri l#	

5.「受注数量」(NY)を選択します。 → 右図のようなダイアログが出ます。



6. この例では、「いいえ(N)」で答えてください。
 → ヘッダ行の変数欄に、NY が設定され、その右に変数名「受注数量」が表示されます。
 これでヘッダ行の設定は終了です。

40			
41	¥=項目	C=変更	11 受注数量

次に、詳細行の記述をします。

詳細行では処理コマンドを記述します。処理コマンドの設定方法は、旧バージョンの場合とほとんど同じです。

例えば、項目更新コマンドを設定する場合には次のようにします。

7. ポップアップメニューから「行作成(R)」(F4)を選びます。

→ 空の詳細行が1行できます。

※ 受注動量
 ス^{*}-4(Z)
 行作成(P)
 ヘッタデテ作成(D)
 √
 √
 √
 √
 √

41 ⊡ ¥=項目 C=変更

NY 受注数量

8. 詳細行の先頭で、ズーム(F5 またはダブルクリック)し 日ます。

→ コマンドのコンボボックスが現れます。

¥٢	項目	C
	i Faxyi 📉	
	=コメント U=項目更業 C=コール I=外部コール B=イヘント実 A=アクション	S
	BFノアロック EFIJ- FFJオーム VF項目	

42

9.「U=項目更新」を選びます。

→「項目更新」が設定されると共に、項目更新コマンドのパラメータ入力欄が現れます。 ※ コンボボックスから選ぶ代わりに、文字「U」をキー入力しても、項目更新コマンドが設定されます。

41	∃ ¥=項目	C=変更	NY	受注螤量			
42	項目更新	V=項目	333	??	値:	0	条件: Yes

10. 更新する項目、式、条件等を設定します。

42	項目更新	V=項目	NS	受注金額	値:	56	受注単価*受注数量	条件: Yes	
----	------	------	----	------	----	----	-----------	---------	--

11.同様にして、2つ目の項目更新コマンドも設定します。

41	□ ¥=項目	C=変更	NY	受注数量			
42	項目更新	Y=項目	NS	受注金額	値:	56	受注単価*受注数量
43	項目更新	V=項目	NT	粗利金額	値:	57	(受注単価-受注原価ف条件:Yes

6.3 項目変更ハンドラの利点

項目変更ハンドラを使うと、コントロールレベルのハンドラを使うことに比べて次のような利点があります。

- 意図がわかりやすい。ここに挙げた例では、開発者の意図としては、「受注数量をもとにして受注金額 と粗利金額を常に正しい値に更新しておきたい」というものがあります。この意図をプログラム上で表 現するときに、「『受注数量』コントロールを通りすぎるときにこれこれをせよ」という意味のコントロール 検証ハンドラよりは、「『受注数量』に変化があったときに、これこれを実行せよ」という意味の項目変更 ハンドラのほうが、開発者の意図をより直接的に明確に反映していると見ることが出来ます。これにより、プログラムを作りやすくなるし、読む側も意図を明瞭に知ることができるようになります。
- 設定忘れがなくなる。項目変更ハンドラは、指定された項目の値に変更があった場合には、いつ、どこで、どのようにして(例:ユーザ入力、項目更新コマンド、コールプログラムなど)変更されたかに関わりなく、常に実行されます。そのため、何箇所かで変更される可能性がある項目の場合には、同じ再計算ロジックをあちこちに繰り返す必要がなく、また、開発者にとっても、漏れが出る可能性がなくなります。
- 値の変化がない場合には実行されない。コントロールレベルのハンドラを使ったやりかたでは、「受注数量」の値の変化の有無に関わらず、項目更新コマンドが常に実行されます。ここの例のように、四則演算だけの計算ならば、毎回計算してもオーバーヘッドは無視できる程度ですが、時間のかかるプログラムの呼び出しなどを伴う処理であればオーバーヘッドは無視できない場合もあります。項目変更ハンドラでは、値に変化のない場合には、ハンドラが実行されません。例えば、ユーザがカーソルを「受注数量」に移動したのち、そのまま別の項目に移動した場合、あるいは、ユーザが「受注数量」にデータを入力しても、前と同じ値だった場合などには、ハンドルが実行されません。従って、再計算は必要最低限にだけ限られることになり、オーバヘッドも最小にすることができます。

項目更新ハンドラは Magic V10 に備わった強力な機能です。ぜひ活用して簡潔明快なプログラムを作るようにしてください。

第7章 プッシュボタン

本章と次章では、イベントハンドラを使った例を説明します。

イベントハンドラを利用する最も一般的なケースはプッシュボタンです。プッシュボタンを押したときに、ある処理 を行う、というユーザインターフェースは、昨今の GUI インターフェースでは非常に多用されますので、本章では まず、プッシュボタンについて説明をしていきます。



7.1 旧バージョンでのやりかた

旧バージョンでは、プッシュボタンの処理を行うのに、タスクイベントテーブルをよく使っていました。

例えば、「シリアル#確認」というプッシュボタンを押したときに、シリアル#確認プログラムを呼び出す、というプログ ラムを考えてみます。より正確には、「どこで?いつ?何を?」に従って書いてみると、次のような仕様になりま す。

- A) どこで?「シリアル#確認」というラベルのあるプッシュボタンにおいて。
- B) いつ? プッシュボタンが押されたとき。
- C) 何を? シリアル#確認処理を行う。

これを実現するためのプログラムは、内部アクション「ユーザアクション1」を使って、

- フォーム上のプッシュボタンの設定:実行時、ユーザがプッシュボタンを押したときに、「ユーザアクション1」を発生させる。
- イベントテーブルでの設定:ユーザアクション1が発生したときに、それをキャッチして、プログラム 588 番「SO 照会 シリアル#」を呼び出すようにする。

という2箇所での設定が必要です。

フォーム上のプッシュボタンの設定は、次の通りです。

- フォームエディッタ上にプッシュボタンを配置します。
- このプッシュボタンのコントロール特性を開きます。
- 「ラベル/書式」特性に「シリアル#確認」を設定します。
- 「アクション」特性に「ユーザアクション1」を設定します。

フォームエディッタでの設定		
ラベルは プッシュボタンを 配置 売上/原価/粗利 沙州生成依頼 備考 *** -###,### ジッパは生成 *** -###,### ジッパは生成	は「シリアル # 確認」 ■ コントロール特性: ブッシュボタン 詳細 入 力 外 額 位置とりな [*] パラメータ 値 式 ボタン スタイル PP7 [®] ・ジュボ [*] ダン ● ブータ ?? 回 ヨ ?? 回 コントロール名 ラベルノ書式 ?? アクション ■ ーザアクション1	
アクション には、 「ユーザアクション1」を設定	OK	

一方、タスクイベントテーブルでの設定は、次のようになります。

- タスクイベントテーブルを開きます。
- ●「アクション」欄に、同じく「ユーザアクション1」を指定します。
- 「Prog/Task」欄には、呼び出すプログラムの番号(588番)を指定します。

● 「パラメータ」欄には、呼び出すプログラムに渡すパラメータを設定します。

タスクイベントテーブルの設定 /	<mark>アクション には、 「ユーザアクション1」を設定</mark>	
 	情報1 (597).SO受注明細 経過時間 式 スコーブ コール Prog/Task パランータ ロック 00:00:00 0 T=タスク P=プロヴラム 588 2 No 88 番を呼び出す	

7.2 イベントハンドラでのやりかた

V8 でのタスクイベントテーブルは、「ベントが起こった場合に、どのプログラムを呼び出すか」という関連付けを 定義するものですが、V10 ではこれを**イベントハンドラ** で行います。

今回の例で使うイベントとしては、V8の場合と同じく「ユーザアクション1」などでも良いのですが、ユーザ定義イベントを使った方が、アプリケーションにとって意味のある名前でイベントを扱うことができ、プログラムがわかりやすくなるので、ここではユーザ定義イベントを使います。

7.2.1 ユーザ定義イベントの利用

ユーザ定義イベントというのは、

- V9 で導入された機能で、V10 でもサポートされています。
- 文字通り、開発者が必要に応じて新しく作成することができるイベントで、自由な名前をつけることができます。
- V8 での「ユーザアクション n」と同様、それ自体では何の機能を持ちません。必ず、イベントハンドラと 共に用います。
- タスクのイベントテーブルで定義します。
- ひとつのタスク内で定義できる数に制限はありません。
- スコープ(有効範囲)があります。あるタスクに定義したユーザ定義イベントは、そのタスクおよびその 子孫タスクで有効です。ただし、「メインプログラム」(プログラム番号1番)に定義したユーザ定義イベン トは、そのアプリケーション全体で有効なグローバルなものとなります。

例えば、次の図は、あるタスクのイベントテーブルに「U_シリアル番号発行」という名前のユーザ定義イベントを定義した例です。

ㄱ	らイベットの空差	_						
			イベント: 725.1 - SO入力 受注情報(元598).SO受注明細					
	好きな名前で イベントを作成	_	#5	名前 11 沙测番号發行	トリカ [*] タイフ [®] N= なし	「トリカ゛	በ በ	強制終了
			, in the second s		10.00		Ť	10.00



V8 の「(タスク)イベントテーブル」と、V9Plus/V10 での「イベントテーブル」とは、名前は同じで すが内容は異なります。

V8の「(タスク)イベントテーブル」というのは、上記の例のように、イベントが起こった場合に、 どのプログラムを呼び出すか、という関連付けを定義するものです。

V9Plus/V10 での「イベントテーブル」は、ユーザ定義イベントを定義するものです。V9Plus/V10 で、イベントとそれに対する処理とを関連づけるのは、イベントハンドラになります。



- 1. タスクを開きます。
- 2. イベントテーブルを開きます (Ctrl+U)。
- ポップアップメニューから「行作成(R)」
 (あるいは F4 キー)で新規行を作成します。

#		名前		「トリカ゛タイフ゜	[トリカ*
	11	0_納品書		NF73CU	
		MH TO L			
Θ,	ズト	-4(<u>Z</u>)		r	
*	行	作成(<u>R</u>)			
F .	19	划行作成(I)	N		
Æ	行道	削除(<u>D</u>)			

イペント: 725.1 - SO入力 受注情報(元598).SO受注明細

N°ラメータ 強制終了

小个	ント・	725.1 -	SO入力 受注情報(元598).S	0受注明細	
#		名前	「トリカ゛タイフ゜ 「トリカ゛	い。 ラメータ	強制終了
	11	U_納品書	N=なし	0	N≕なし
	12		S=ジステム	0	NHなし

- 4. 「名前」欄には、利用目的がわかるような任意の名前をつけます。 ここでは、「U_シリアル番号発行」とします。
- 5. その他のパラメータについては、次のように設定します。
 - トリガタイプ: 「N=なし」
 - トリガ: (空欄のまま)
 - パラメータ: (0 のまま)
 - 強制終了:「N=なし」

これらのパラメータの意味については、第8章「ズーム」で、ズーム機能に関連したものについて簡単 に説明します。

イペント: 725.1 - SO入力 受注情報(元598).SO受注明細				
# 名前	トリカ゛タイフ゜ トリカ゛	「パラメータ「強	制終了	
11 U_納品書	N≕なし	0 N=	なし	
12 U_沙州番号発行	N=なし	0 N=	なし	



V8 で「終了」「取消」「選択」「削除」等々、Magic エンジンに何らかの動作を起こさせるものは「ア クション」と呼んでいましたが、V9 から「内部イベント」と呼ばれるようになりました。

「ユーザアクション n」は、エンジンに何らの動作を起こさせるものではありませんが、既定義の アクションとして、内部イベントとして分類されています。

7.2.2 プッシュボタンの設定

プッシュボタンでは、実行時にユーザがプッシュボタンを押すと、「U_シリアル番号発行」イベントが発行されるように設定します。

これは、フォームのプッシュボタンの特性シートで、

- 実行時のボタンラベルとして、「書式」特性には、「シリアル#確認」と設定。
- プッシュボタンを押したときにに発行させるイベントとして、「実行イベント」特性には、上で定義したユー ザ定義イベント「U_シリアル番号発行」を設定。
- イベントハンドラで、プッシュボタンを区別するため、「コントロール名」特性には「PB_シリアル#確認」と設定。

とします。





プッシュボタンの命名規約について:上の例では、プッシュボタンのコントロール名として「PB」シ リアル#確認」という名前が付けられていました。このように、プッシュボタンには「PB」」という接頭 辞をつけておくと、プログラム中で「これはプッシュボタンだな」とすぐに区別できるので便利です。



- 1. フォームエディッタを開きます。
- プッシュボタンを配置します。(V8 からの移行 アプリケーションであれば、すでにフォーム上 に配置されているはずです)。

売上/原価/粗利	沙別生成依頼	備考
-###,###,###	□ シリア雌生成	XXXXXXXXXXXXXXXX
-###,###,###		

コントロール特性:プッシュボタン - PB_シリアル... 🔀

PB_シリアル非確認 シリアル非確認

区分(C) 全体(A)

ントレート名

注着

- 3. プッシュボタンを選択して、コントロール特性を 開きます (Alt+Enter)。
- 4. 「コントロール名」特性に「PB_シリアル#確認」と指 定します。
- 5.「書式」特性に「シリアル#確認」と設定します。
- 6.「実行イベント」特性にカーソルを合わせます。

- 7. ズームします(F5 キーあるいはダブルクリック)
 → イベントダイアログが開きます。
- 8. 「イベントタイプ」は「U=ユーザ」とします。

ホッタンスタイル	P=7°-994#*\$2
デジフォレトイメージ	Ti#190 💼
実行へい	
実行元	C=コンテナ タスク
コントロール特性:フ	パッシュホッシー PB_シリアル 🔀
区分(<u>C</u>) 全体	5(<u>A</u>)
コントロート名	PB_9974建建22 🔍 🔨
た書	997 は確認 0
型	A=文字
ボタンスタイル	Ρ=Ͻ°ϧϿϫϮʹʹϙϽ
デフォルトイナージ]‡°y)
■実行へい	
実行元	0-3777 727
-	

ተላንኮ	
~~~>> ***	イベントのタイプと実行するイベントを指定してください. イベントタイプ: イベント:
	0K \$+>>tel

9. 「イベント」欄でズームします。(F5 キー あるい はダブルクリック)

→ 定義されているユーザ定義イベントの一覧 が表示されます。

4421-	-覧 🛛 🔀
表示:	全て
#	名前
1	GU_一覧
2	GU_実行N
3	GU_実行E
4	GU_X*-4
5	U
6	U_住人先注文書
/	U_于配者 11 今时在西
0	□_七印□広奈
3 10	□
10	
(12	0_11111日日 11 沙孔番号發行
	X
- 説明 —	
	選択 [ キャンセル

10.「U_シリアル番号発行」を選択します。

→ イベント ダイアログに設定されます。

14.21	X
	イベントのタイプと実行するイベントを指定してください。
	ለ*`ントタイフ°: <u>U= <u>ז</u>-ザ</u>
	ペット: U 沙川番号発行
	ОК <b>†</b> +>>t ,

11.「OK」を押します。

→ プッシュボタン特性シートの「実行イベント」 特性に、「U_シリアル番号発行」が設定されます。

コントロール特性:フ	*ァシュネジン - PB_シリアル 🛛			
区分(C) 全体(A)				
コントロート名	PB_シリアル非確認 🛛 🔨			
た書	<u> シリアは確認 0 </u>			
型	A=文字			
ボタンスタイル	P=7°-992#*\$92			
7*7411-17-7**	+ + + + + + + + + + + + + + + + + + + +			
回実行へい	U_沙川番号発行 🔵 🛄			
実行元	<u>C-3)77 424</u>			

以上で、プッシュボタンの設定は終了です。 フォームエディッタを閉じてください。

## 7.2.3 イベントハンドラの設定

このプッシュボタンに対応するイベントハンドラは、次のようなものとなります。

- A) どこで?「PB_シリアル#確認」という名前のコントロール(プッシュボタンコントロール)において。
- B) いつ?「U_シリアル番号発行」というユーザ定義イベントが発行されたときに。
- C) 何を? プログラム 589 番「SO 照会 シリアル#」を呼び出します。

これを実現するために、イベントハンドラは次のように設定します。



- 1. イベント発生時に実行すべきハンドラなので、ヘッダ行は「E=イベント」とします。
- 2. 「いつ?」を指定するものを、「トリガ」と呼びます。この例では、ユーザ定義イベント「U_シリアル番号発行」 がトリガです。トリガは、ヘッダ行の第2パラメータとして指定します。
- 3. 「どこで?」は、ヘッダ行の「コントロール」欄に、コントロール名を指定します。ここでは、プッシュボタン のコントロール名「PB_シリアル#確認」を指定します。
- 4. 「何を?」は、詳細行に記述します。プッシュボタンを押したときに実行すべきプログラムを、コールコマンドで呼び出します。



コントロールレベルハンドラと異なり、イベントハンドラでは「コントロール名」欄の指定は任意で す。「コントロール名」欄を指定した場合には、カーソルがそのコントロールにある場合にだけイ ベントハンドラが有効になります。一方、「コントロール名」欄を指定しない場合には、カーソルが どこにあっても常に有効となります。

今の例では、イベント「U_シリアル番号発行」は、プッシュボタン「PB_シリアル#確認」を押した場合にし か発行されないので、「コントロール名」欄を空白にしたままでも、動作は同じになります。

同じイベントが複数のプッシュボタンに割り当てられている場合など、コントロールを区別しなけ ればならない場合には「コントロール名」を設定します。



- ロジック タブをクリックします。
   → ロジック画面が開きます。
- ポップアップメニューで、「ヘッダ行作成(I)」を選びます。
   → 空のヘッダ行が作成されます。



ヘッダ行の先頭で、ハンドラの種別を、コンボボックスから選びます。ここでは、「E=イベント」を選択します。



→ イベントハンドラのヘッダ行が作成されるとと もに、トリガを設定するための イベント ダイアロ グが表示されます。



プッシュボタンの設定と同様に、トリガとなるイベントを設定します。

- 4. 「イベントタイプ」は「U=ユーザ」とします。
- 5.「イベント」欄でズームします。(F5 キー あるい は ダブルクリック)
   → イベントー覧が表示されます。



「U_シリアル番号発行」を選択します。
 →「イベント」欄に、選択したイベント名が表示されます。

	14.21	Σ	<
Ŧ	~~~>F	イベントのタイプと実行するイベントを指定してください.	
	"	/ペントタイプ: U=2-ザ	
		べいた: № 沙川番号発行	
		OK (++)tell	)

7. OK ボタンを押します。

→ イベントハンドラのヘッダ行に戻り、トリガ欄には選択したイベント名が表示されます。

41	E=イベント (	19_刘71番号発行	) ACE	スコーブ S=サブクリー 条件 Yes

8. コントロール欄に移り、ズームします。(F5 キー あるいは ダブルクリック)
 → このタスクのフォームにあるコントロールの一覧が表示されます。(コントロール名が設定されているもののみ)

41	E=イベント	U_シリアル番号発行		ד. בצ אין	=ワブウリー 条件 Yes
	コントロール一覧				
	ケループ名:  全てのコントロール		1)加州名:		
	SO受注明細		PB 沙川雄確認 在力 旁注原便用便		
			物流拠点#_END 物流拠点#_START メディア		
			[]]] 選打	R ++>U	

9.「PB_シリアル#確認」を選択します。

→ ヘッダ行に戻り、「コントロール」欄には選択したコントロール名が表示されます。

41	EFATON	U_シリアル番号発行	<u>コント</u> PB_シリアは確認	スコーフ* T=タスク	条件 Yes
----	--------	------------	----------------------	-------------	--------

以上で、ヘッダ欄の設定は終了です。 次に、詳細行の設定にはいります。詳細行は、処理コマンドを記述します。操作方法としては、旧バー ジョンの場合とほとんど同じです。この例では、コールコマンドで、プログラム 589 番を呼び出すようにし ます。

10.ポップアップメニューで、「行作成(R)」を選択しま
 す。(あるいは、F4 キーを押します)
 → 詳細行が1行作成されます。

뭎	<b>発行</b>	TUHI
0	λ ス [×] −Δ( <u>Ζ</u> )	
1	■行作成(B)	
	小ッダ行作成     小     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ     マ	
4	≥ 行削除( <u>D</u> )	

11.詳細行の先頭でズーム (F5 キー、あるいはダブ **E=イベント U** ルクリック)をします。 → コマンドの一覧がコンボボックスで表示され ます。



12.「C=コール」を選択します。

→ コールコマンドが表示され、パラメータ類が表示されます。

41	⊡ E=イベント	U_シリアル番号発行	コントI PB_シリアは確認	スコーフ* T=タスク	
42	a-N	PED 0217. 🕶 0	[0 パラメータ]	戻り値: ???	- 条件: Ye

13.コールのタイプは、「P=プログラム」とします。

14. プログラム番号欄でズームします。→ プログラム一覧が表示されます。

41 🖸 L-1 V /r	U_ソリノル番亏発行	プログラムー	- 覧					
42 ⊐~lk	P=7°07°56	<b>主二</b> ,						
		उत्तरन ः	ΞU					
			名前					<u> </u>
		589	0照会	99711#				
		590	SB変更	明細番号	のリナンハ	1 [°]		
		591	SB修正	粗利情報	2			
		592	SB修正	粗利情報	1			
		593	MB修正	粗利情報。	ALL			
		594	SB集計	明細行数				
		595	SO照会	受注情報				
		596	SB変更	受注罪(受)	注備考	·)		
		597	SB変更	受注罪(受)	注連絡	先)		
		598	SO入力	受注情報				
		599	SB制御	受注情報	入力			
		600	SBED刷	合計請求	書			_
		601	SB合計	請求書ファイ	∥生成			
		602	MO集計	- 合計請	求額			
		603	MO照会	- 合計請	求書依	頼顧客		
		- 004 	on±+±L	ਙ±=⊥ ∟				
		- 190H						
						计建护	] [ ±+`	741
プログラム 589	番「SO 照会 シリアル#」	を選択しま	Ξ	E=イベン	۱ŀ I	U_9976番号	弓発行	
						D-T2-b/st	1000	0087 0

16.パラメータに移動し、ズームします。→ パラメータテーブルが表示されます。

す。→ 選択したプログラム番号とプログラム名

とが表示されます。

٦

	40									
	41	⊡ E= <b>1</b>	ベント	U_シリアル番号3	発行		1)    PB_9 <b>9764</b> 5	認わ	1-7* T=935	
	42	3-	ll.	P=7°05°56	589 SC	0照会 沙개배	( [O //*	'ラメータ]	戻り値: ???	条件:Yes
V,	ラメー	}: SO∰	会 うりフ	7ルま						
Ħ	:	項目	」 式	説明			スキッフ°	~	コール先パラ	X-9
								8		
	<b>X</b> UH									

17.必要なパラメータを設定します。

パラメー	<del>ያ</del> : SO	요승 기	J7N#	
Ħ	項目	〕五	説明	スキッフ、
1	MB		0 受注#	
2	MC		0 受注明細難	

#### 18.OK ボタンを押します。

→ 詳細行に戻ります。パラメータ欄には、パラメータの個数が表示されています。

41	⊟ E=イベント	U_シリアル番号発	衍	:	1)   PB_)  7   確認	スコーフ* T=タス	<u>,</u>	
42	14V	P=プログラム	589 SO	0照会 沙개構	▶[2 /\°ラ⊁-タ]	戻り値:	333	条件: Yes

以上で、できあがりです。

# 7.3 イベントハンドラ方式の利点

レコードメインを使った方式に比べ、イベントハンドラを使った方式は、次のような利点があります。

- ボタンとイベントの関連がわかりやすい。V8 では「ユーザアクション n」しかなく、どのボタンにどのユー ザアクションが割り当てられているのかがわからなかったのですが、V10 では、アプリケーションにとっ て意味のある名前のユーザ定義イベントを使って記述できるので、意図がすぐにわかるようになりまし た。
- プログラムの見通しが良い。V8では、ズームや画面の操作はレコードメインに、プッシュボタン等の操作はタスクイベントテーブルにそれぞれ記述していましたが、V10ではすべて「ロジック」タブに記述するようになりました。
- 処理内容がわかりやすい。V8のタスクイベントテーブルは、プログラム番号のみしか見ることができず、 プログラム名はプログラム番号欄からズームしなければなりませんでした。このため、タスクイベントテー ブルだけを見ても、どのボタンで何が呼び出されるのかわかりませんでした。V10では処理内容は詳 細行に記述されるので、プログラム名、パラメータ、条件などがわかりやすくなっています。
- イベント数に制限がない。V8 での「ユーザアクション n」は、「ユーザアクション 20」までの 20 個しかあり ませんでした。それ以上のボタンを使いたい場合には、同じユーザアクションを複数のプッシュボタンに 割り当て、CTRLNAME()で条件付けして区別していました。V10 ではユーザ定義イベントの数に上限が ないので、必要なだけ定義して利用することができます。
- 別途プログラムを作る必要がない。V8のタスクイベントテーブルでは、必ずプログラムあるいはサブタスクを呼び出すようになっており、実際の処理内容がコマンド1行だったとしても、別途にタスクを作る必要がありました。V10のイベントハンドラでは、詳細行にさまざまな処理コマンドを記述できるので、 例えば、項目更新コマンドひとつで済む場合には、わざわざ別タスクを作成する必要はありません。

# 第8章 ズーム

ズーム機能は、ある項目に選択肢がある場合に、F5キー、あるいはダブルクリックによって、一覧選択プログラムを表示させて、その中から欲しいものを選択する、というもので、Magic アプリケーションでは非常に多く使われます。

V8 では、ズームはレコードメインのフローモード「B=前置」あるいは「A=後置」を使って実現していました。V10 では、イベントハンドラを用いて実現します。



本章では、ズーム機能の実現方法と注意点などについて説明します。



ズームを実現するには、データ項目の「選択プログラム」特性にプログラム番号を指定する方法も あります。この方法は簡便で使いやすく、お勧めなのですが、プログラムに渡すことのできるパラメー タ数が1個に限定される、という制限があります。

「選択プログラム」特性を使ってズームを実現する方法は、旧バージョンでも V10 でも全く同じように 使うことができますので、本書では説明を省略します。

旧バージョンでのズーム機能は、一覧選択プログラムの呼び出しだけでなく、

- プッシュボタンに「ズーム(Z)」イベントを関連づけて、ボタンを押した場合の処理をレコードメイン に記述する。
- タブコントロールで、ユーザがタブをクリックした場合に「ズーム(Z)」イベントが発生することを利用して、タブ選択変更時の処理をレコードメインに記述する

などの使いかたもしていました。

V10 では、プッシュボタンについては第7章「プッシュボタン」に説明したようにユーザ定義イベントを 利用することがお勧めです。また、タブ選択変更時の処理は、第6章「項目変更ハンドラ」に説明し た、項目変更ハンドラを利用することをお勧めします。

# 8.1 レコードメインでのやりかた

旧バージョンでは、ズームを実現するのに、レコードメインのフローモードを使って実現していました。これには 大きく分けて二つの方法があります。

#### 8.1.1 「B=前置」を利用

ーつは、フローモード「B=前置」を利用する方法です。

- ズームを実装したい項目(ズーム可能項目)のセレクトコマンドの直後に、ズーム時に実行させたい処 理コマンド(通例、選択プログラムを呼び出すコールコマンド)を記述します。
- 処理コマンドのフローモードは「B=前置」とします。

例えば、次の図は、ズーム可能項目として「商品#」があり、そのセレクトコマンド(第12行目)の直前(第11行目)に、商品選択プログラムを呼び出すコールコマンドがあります。そして、コールコマンドのフローモードはB (B=前置)となっています。

「B=前置」を使う方法										
11 コール P=プログラム: 566 SO選択商品一覧	Λ°5 3	7#~6 0	ロック No B C Yes							
12 也小 R=実データ: 3 商品#	代入 0	0 0	0 /0 S C Yes							
<mark>ズーム項目</mark> 「商品#」	ズーム項目の コールコマンド	前に、B= 前置 を入れる	: C							

このように設定されている場合には、次のような動作になります。

- ズーム可能項目「商品#」にカーソルがパークしたときに、ズーム可能となります。
- ユーザが F5 キーを押すか、マウスのダブルクリックを行うと、直前のコールプログラムが実行され、プログラム 566 番「SO 選択 商品一覧」が呼び出されます。
- 呼び出されたプログラムが終了したときには、カーソルは再びズーム可能項目「商品#」でパークします。

#### 8.1.2 「A=後置」を利用

もうひとつの方法は、これと似ていますが、フローモードは「A=後置」を使うものです。

- ズーム可能項目のセレクトコマンドの直前ではなく、直後に、ズーム時の処理コマンドを置きます。
- 処理コマンドのフローモードは「A=後置」とします。

#### 「A=後置」を使う方法 11 也外 R=実が-り: 3 商品# 2 0 代入 0 0 Yes Û 0 С 12 그네 P=プログラム: 566 SO選択 商品一覧 3 77-6 ロック No №÷. 0 Yes ズーム項目の後に、A=後置で コールコマンドを入れる

このように設定されている場合には、ズーム可能項目「商品#」にカーソルがパークしたときに、ズーム可能となることは同じですが、選択プログラムが終了したときには、カーソルはズーム可能項目「商品#」の次の項目に 進んでパークします。

#### 8.1.3 ズーム時に複数のコマンドを実行

場合によっては、ズーム時に実行したいコマンドが一つではなく、複数のコマンドからなる場合があります。この ような場合には、

- 実行したい一連の処理コマンドをブロックコマンドで囲みます。
- ブロックコマンドのフローモードとして、「B=前置」あるいは「A=後置」を設定します。

下図は、フローモード「B=前置」を指定したブロックコマンドを使って複数のコマンドをズーム時に実行させるよう にした例です。「営業実績配分」項目がズーム可能項目であり、ここでズームすると、ブロックコマンド内の一連 の処理コマンドが実行されます。

ズーム時に複数のコマンドを実行させる方法										
ズーム時に ブロックコマ	実行するコマンドを ンドで囲む	<mark>ブロックコマン</mark> 「B=前置」に	ッドのフローモー ニする							
300 311 コール P=フ°ロク°うム: 587 322項目更新 :H 33項目更新 :I 34 コール T=ななり : 3 35 フ°ロック終了	[Yes S0入力 営業実績配分率 V.子奴り終了F V.子奴り画面消去F 営業実績配分 ]	パ*う 2 式 3 式 4 パ*う 0	<mark>フォーム 0</mark> 計 N=代入 計 N=代入 フォーム 0	ロック Yes 止 Yes 止 Yes ロック Yes	B C C C C C C C C C C C C C C C C C C C	Yes Yes Yes Yes				
376 世小 v=変致 : 45 ズーム項目 「営業実績配分」	¥.呂耒美楨即分		ーム時には、ブ 身が実行される	^で ロックの る	S C	Tes				

# 8.2 イベントハンドラを使うと・・・

これと同じズーム機能を、レコードメインを使わずにイベントハンドラで実現する方法を以下に説明します。 イベントの考え方で整理すると、ズームの扱いは次のようになります。

- A) どこで? ズーム項目で。
- B) いつ? ズームが発生したときに。
- C) 何を? 選択プログラムなどをコールする。

この考え方で、プログラムを作ってみましょう。

### 8.2.1 ズーム項目

「どこで?」の指定は、イベントハンドラの「コントロール」欄に、ズーム項目のコントロール名を指定することによって行います。そのため、まずズーム項目にコントロール名を設定する必要があります。

下図の例では、「商品#」がズーム項目となっている例です。「商品#」を表示しているエディットコントロールの コントロール特性で、「コントロール名」が「商品#」に設定されています。



# 8.2.2 イベントハンドラ (その1)

「いつ?」は、「ズームが発生したとき」です。もうすこし正確に言うと、F5 キーを押したり、ダブルクリックしたり すると、内部イベント「ズーム(Z)」が発生しますので、それをそのままイベントハンドラにすると、イベントハンドラ のヘッダ行のトリガとして内部イベント「ズーム(Z)」を指定することになります。

実際のところは、内部イベント「ズーム(Z)」を直接ハンドルすると、後述するような問題が出てきますので、ユー ザ定義イベントをひとつ介して行わなければならないのですが、まずは、直接ハンドルすることの問題点を実地 に見ていくために、内部イベント「ズーム(Z)」をトリガとするハンドラを作ってみます。その後、正しい動作をさせ るためにどうするかを解説します。

内部イベント「ズーム(Z)」をトリガとするハンドラは、次のようになります。

- 内部イベントに対するハンドラなので、ヘッダ行は「E=イベント」になります。
- ズーム時に実行させたいので、トリガは、内部イベント「ズーム(Z)」になります。
- ズーム項目「商品#」でだけ有効にしたいので、コントロール名に「商品#」を指定します。
- 詳細行では、選択プログラムを呼び出すコール コマンドを実行します。





上記のイベントハンドラは、次のような操作で作成します。

•

🔍 ズーム(Z)

🗐 行作成(<u>B</u>)

💊 🗤 🛛 🖉 行削除(D)

- 1. ロジック画面を開きます。
- 2. ポップアップメニューで、「ヘッダ行作成 ロジック フォーム (I)」を選択します。(あるいは Ctrl+H キー) → 新規ヘッダ行が作成されます。
- 3. イベント種別のコンボボックスから、 「E=イベント」を選択します。 → ヘッダ行に「E=イベント」と表示され、 トリガを設定するイベントダイアログが 表示されます。
- 41 E={\\$')} 23-7* S= 과 小小 イベント イベントのタイプと実行するイベントを指定してください. (ベントタイプ: 1=内部) べつた: OK ) ( <del>+</del>+>)tili
- 4. 「イベントタイプ」としては、「I=内部」を 選びます。
- 5.「イベント」欄でズームすると、内部イベ ントー覧が表示されます。



6.「ス´ーム(Z)」を選択します。
 → イベントダイアログに選択したイベン
 ト名が表示されます。

14.51	
	イベントのタイプと実行するイベントを指定してください.
	ペットタイプ: I=内部
	ለ ^ና ንኑ፡ ፲ ^፻ -ሬ(Z)
	OK \$75761
41	$E=f^{(x)}F(-R^{(x)}-L(Z))$

7. OK ボタンを押します。

→ トリガ欄に、選択したイベント名が表 示されます。

8. コントロール欄に移って、ズームします。

→ このタスクのフォーム上のコントロール一覧(コントロール名が設定されているものだけ)が表示され ます。

41	E={*`}	λ°−₽(Σ)	<b>ارت</b> )		スコーフ* S=サフ*ウリー	条件 Yes
		コントロール一覧				
		がW-プ名:		コントロール名:		
		50受注明細		PB 刘744確認 仕切 受注原価単価		
			(	文/11 11 商品# 物流规点#_END 物法规点#_END	г	
				1707/LLD2.RC#_011411 メデドィア	I	
					選択	77)tll

9.「商品#」を選択します。

→ 選択したコントロール名が設定されます。

41	EFAがント	ን∗-₽(S)	は 11日 日本 11日 11日	אלאבי זיריבג (גער איר	条件 Yes

以上で、ヘッダ行の設定は完了です。

詳細行の作成方法は、旧バージョンの場合とほとんど同じです。

10.ポップアップメニューで、「行作成(R)」
 を選びます。(あるいは、F4 キー)
 → 詳細行が新規作成されます。

-6(7)	٦
	.[
( 🔚 行作成( <u>R</u> ) 🔪 )	
	1

11.詳細行の先頭でズームして、コンボボッ 41 日 E=イベント ズーム(Z) クスから「C=コール」コマンドを選択しま す。

→ 詳細行がコールコマンドとなり、パラ
 メータが表示されます。



12.コールコマンドのパラメータ類を設定します。

41 E	∃ E=イベント	ズーム(Z)	コント 商品#	えコーフ* T=タスク	
42	그네	P=プログラム	- 567 SO選択 商品一覧 - ▶️S パラメータ]	戻り値: ???	条件: Yes

以上で、イベントハンドラができあがりました。

## 8.2.3 内部イベント「ズーム」を直接ハンドルするときの問題

このようにして直接内部イベント「ズーム(Z)」をハンドルすると、実行時に予想しない動作となってしまいます。 実際に実行させて見てみましょう。



Enter キーを押して、次項目に移ると、商品名称が「dbMAGIC V8 クライアント実行 5 ユーザ版」に変わってしまいます。
 結局、最初に手で入力した番号が有効になり、商品一覧で選択したものは無視されてしまっています。



このように、内部イベント「ズーム(Z)」を直接ハンドルすると、アプリケーションの動作として極めて不都合なことになってしまいます。

このような現象が起こる理由は、表示フォーム上のエディットコントロールのデータと、内部のデータ項目のデータとが、ユーザ入力途中の状態においては一時的に異なっているためです。

ー般的に、オンラインのフォームにおいて、入力途中の状態ではユーザがキー入力はエディットコントロールに 蓄えられるだけで、内部のデータ項目にすぐに反映されません。ユーザが入力を終わり、Enter キーなどで別 の項目に移った時点で、はじめてエディットコントロールの内容が、内部のデータ項目に反映されます。

このことを念頭において、内部でどういうことが起こっているかを示したのが、次の図です。



- 受注明細画面において、新規行を作成した時点では、「商品#」項目の値はデフォルト値0です。 ここで、ユーザがキー入力し、482078021と入れます。 ユーザのキー入力中は「入力途中の状態」であり、フォーム上のエディットコントロールに入力データが 蓄えられていきますが、「商品#」項目にはすぐには反映されません。従って、「商品#」項目の値は0 のままです。
- 2. ここで、F5 キーを押すと、「入力途中の状態」のまま、「ズーム(Z)」アクションが発生します。 「ズーム(Z)」ハンドラが定義されているので、ハンドラが実行され、商品一覧プログラムがコールコマン ドで呼び出されます。

この間、エディットコントロールは「入力途中の状態」のままですので、ユーザ入力はまだ項目に反映されていません。従って、商品一覧プログラムに渡されるパラメータは0のままです。このために、商品 一覧プログラムの初期画面で、最初のレコードに位置付けられることになります。

3. 商品一覧プログラムで、482078103を選択します。

すると、「商品#」項目のデータが、この番号(482078103)になります。これにともなってリンク再計算が なされ、「商品名称」が「dbMAGIC V8 エンタープライス、サーハ、35 スレット、」となります。

ここでもまだ、「商品#」エディットコントロールは「入力途中の状態」のままで、項目の値の変更はすぐにはエディットコントロールに反映されません。

4. 受注明細プログラムに戻った後、ユーザが Enter キーを押します。

ここで初めて「商品#」エディットコントロールの「入力途中の状態」が終了し、エディットコントロールの データ(482078021)が「商品#」項目に反映されます。

これに伴いリンクの再計算が行われ、商品名称は「dbMAGIC V8 クライアント実行 5 ユーザ」となります。

このような内部動作となっているので、結果として、商品一覧プログラムでは正しく位置付されないし、また選択 した商品#が上書きされ無視されてしまう、という現象となります。

### 8.2.4 ユーザ定義イベント「GU_ズーム」

この問題に対応するには、ズーム発生の時点で、エディットコントロールの「入力途中の状態」を解除してやる 必要があります。そのためには、内部イベント「ズーム(Z)」を直接ハンドリングするのではなく、一段、ユーザ定 義イベントを介して処理するようにします。

ここでは、その目的のためのユーザ定義イベントを以下のように定義します。

- 「名前」:任意の名前でいいのですが、ここでは「GU_ズーム」とします。
- 「トリガタイプ」および「トリガ」: ここで指定されたイベントをトリガとして、このユーザ定義イベントが発行されます。

この例では、内部イベント「ズーム(Z)」が発生したら、このユーザ定義イベント「GU_ズーム」を発生させたいので、「トリガタイプ」が「I=内部」、「トリガ」が「ズーム(Z)」とします。

- 「パラメータ」は必要ないのでそのまま (0 のまま)にします。
- 「強制終了」: イベントが発生するに当たって、エンジンのサイクルを実行することを指示します。

この例では、「E=編集」となっていますが、これは、「入力途中の状態」を抜けて、エディットコントロール などのフォーム上のコントロールの内容を、対応する項目に反映させることを意味します。

#### ユーザ定義イベント「GU_ズーム」の定義 イベント: 1 ー メインプログラム │名前 1 GU_一覧 トリカ゛タイフ゜ ニトリカ゛ パラメータ「強制終了 公開名 公開 # N⊨なし E=編集 2 GU_実行N N⊨なし Û. N=なし 3 GU 実行E N=trl. F= 編生 Û 4 GU_ズーム λ°-μ(Ζ) I=内部 E=編集 0



「強制終了」には、「E=編集」のほかに、次のような設定ができます。

- N=なし: イベントハンドラが実行される前に、エンジンサイクルは何も実行されません。この場合には、エディットコントロールの入力途中の状態はそのままでイベントハンドラが実行されます。これがデフォルトの動作です。
- C=コントロール: イベントハンドラが実行される前に、コントロールレベルのサイクル(コント ロール検証、コントロール後処理 → コントロール前処理)が実行されます。
- R=レコード更新の前 あるいは P=レコード更新の後:イベントハンドラが実行される前に、レ コードレベルのサイクル(レコード後処理、レコード更新、レコードフェッチ、レコード前処理) を実行します。

これらの設定については、本書では説明を省略します。

このユーザ定義イベントは、アプリケーションで共通に使えるグローバルイベントとして定義すると、ズームを使うプログラムごとにいちいち定義せずにすむので便利です。

グローバルイベントとするには、プログラムリポジトリの最初のプログラム「メインプログラム」のイベントテーブ ルで定義します。



これはアプリケーション全体で有効なグローバル ユーザ定義イベントとしたいので、「メインプロ グラム」に定義します。

- プログラムリポジトリの先頭にある「メイ ンプログラム」を開きます。
   フログラムリポジトリ
- メニュー「タスク環境(K) → ユーザイベ ント(U)」を選択します(あるいは Ctrl+U キーを押します)。
   → イベントテーブルが開きます。

#		フォルタ゛
( 1 メインプロ	コグラム	
2 #########		
タスク環境( <u>K)</u> オフジョ	ン(0) デバック	^y
- 🗊 タスク特性(丁)	Ctrl+P	1
- 🗊 データビュー(V)	Ctrl+1	
🙏 🛯 🕹 🔥	Ctrl+2	
🎾 7ォ−ム(E)	Ctrl+3	
- <b>∮</b> * 式(E)	Ctrl+E	-
🛓 🛃 入出力ファイル(1)	Ctrl+I	
増加 ソートテーフかん(3)	OUHT	
[●] ユーザイヘント( <u>U</u> ) [●]	Ctrl+U	
→ 範囲/位置付(B)	<b>NOWIN</b>	
אַרקדבא SQL	Ctrl+Q	

 ポップアップメニューから「行作成(R)」を 選びます(あるいは、F4キーを押します)
 新しい行が作成されます。
 3. ポップアップメニューから「行作成(R)」を 1 GU_一覧 2 GU_実行)



- 4.「名前」欄は「GU_ズーム」、「トリガタイプ」は「I=内部」とします。
- 5. 「トリガ」欄に移り、ズームします → 内部イベントー覧が表示されます。

470	ット:1 ー・メインプ	ログラム					
#	名前	■トリカ*タイフ* ■	ኑሀታ*	IN°ラメータ	強制終了	公開名	公開
	1 GU_→覧 2 GU 実行N	N=なし N=なし		U O	EF編集 N=なし		
	3 GU_実行E	N⊨なし		0	E=編集		
	4 GU_ズーム	I=内部		) •	N⊨なし		
	イベント一覧						
	ヴループ名:			(ベント名:			
	全イベント 実行モード			サブッツー削除 サブッツー縮小	余(D) ト(L)		^
	ナビゲーション 編集、、、			サブ:ツリー上書     サブ:ツリー展開	<u>駻(₩)</u> <u>乳(E)</u>		
	アフリケーション タスク	/ 		リアンオーム冉 シークレット名(	表示 R)		
	マルチマニキン? ユーザアクション 開発す: ゼ	7	(	λτ97 <u>λ° - λ(Z)</u>			
	開発モート			9-5(T) 9-5(T)	2		~
	ļ			17 17 7 IN	37		
					選	択 <del>れ</del>	))till

- 6. 「ズーム(Z)」を選択します。
   →「トリガ」欄にイベント名が表示されます。
- 7.「強制終了」欄で、「E=編集」を選択しま す。

	名前	トリカ゛タイフ゜	トリカ	「A°ラメータ	強制終]	$7 \pm 2$
1	GU_一覧	N≕なし		0	E=編集	
2	GU_実行N	N=なし		0	N⊨なし	
3	GU_実行E	N≕なし		0	E=編集	
- 4	GU_ズーム	I=内部	λ°~μ(Ζ)	0	EF編集	~
				(	NFなし E=編集 日本に P=加-トう P=加-トう	更新行

以上で、ユーザ定義イベントの作成ができました。

## 8.2.5 イベントハンドラ (その2)

このユーザ定義イベント「GU_ズーム」を使って、イベントハンドラを書き直してみましょう。変更するのは、ハンドラのトリガの設定だけです:

ハンドラの「トリガ」欄の設定を、内部イベント「ズーム(Z)」ではなく、ユーザ定義イベント「GU_ズーム」とします。



トリガの設定は、次の操作によって変更します。

- 1. イベントハンドラのヘッダ行のトリガの欄(第2番目のパラメータ)でズームします。
  - → イベント ダイアログが表示されます。

41 🖂	E={ <b>1</b> *`}}	ז∗-⊾(z)		コント商品		<b>スコ−フ* T=タス</b>	ኃ	
42	_ <b>_</b> −⊮	P=7°02°56	567 SO選択 商品	計覧	🚺 N°5X-9]	戻り値:	333	条件: Yes
		ント イベントの イベントが7*: イベント	0タイプと実行するイ U=2-サ [*] <del>D=3774</del> T-5078 <b>U=2-ザ[*]</b> T=94マー E=式 E=式	ベントを指定し マ OK	区 てください. 			

- 2.「イベントタイプ」として、「U=ユーザ」を選びます。
- 3.「イベント」欄でズームします。

ユーザ定義イベントの一覧が表示されます。

GU_ズーム」を選択します。
 「イベント」ダイアログに戻り、選択したイベント名が表示されます。

<u> </u>	イベントのタ	イブと実行す	「るイベントマ	を指定してく	ださい.
7	11°51977°: 11°515:	U=2-ቻ° GU ス°-ሌ			
				OK C	402011

5. OK ボタンを押します。

→ トリガ欄に、選択したイベント名が表示されます。

		$\frown$			
21	⊟ E=イベント	GU_X*~&		HCE	商品
22	a-14	P= 7"07"76	567	SO選択 商品一覧	[3 /*əx-9]

以上で、トリガの設定変更は完了です。

### 8.2.6 実行時の動作

このようにイベントハンドラを定義すれば、次のような動作になり、意図したとおりのズーム動作が実現できます。

1. ユーザがデータをキー入力します。

この時点では、まだ「入力途中の状態」なので、項目に反映されていません。

- 2. ユーザが F5 キーを押します。
  - ① 内部イベント「ズーム(Z)」が発生します。
  - これをトリガとして、ユーザ定義イベント「GU_ズーム」が発生します。
  - ③ 「強制終了」が「E=編集」なので、入力途中の状態が解除され、ユーザが入力したデータが項目に 反映されます。
  - ④ イベントハンドラが起動されます。
  - ⑤ イベントハンドラの中で、コールコマンドが実行され、「商品一覧」プログラムが呼び出されます。 このとき、パラメータとしては、すでにユーザ入力が反映されたデータが渡されるので、初期画面の 位置付けは正しく行われます。


- 3. 商品一覧から適当な商品を選択して、受注明細画面に戻ります。
  - ① パラメータを介して、受注明細の「商品#」項目が更新されます。
  - ② それに伴い、リンク再計算が行われます。
  - ③ 入力途中の状態はすでに終了しているので、項目の変更はすぐにエディットコントロールに反映されます。
- その後、Enter キーを押して、次の項目にカーソルを移動させます。
  項目のデータとエディットコントロールの内容とは同期がとれているので、特に何も起こりません。



このように、ユーザイベントをひとつ間にいれてやることにより、期待していた通りの動作となることがわかります。

## 第9章 おわりに

最後にまとめとして、V8のレコードメインで作った場合と、V10のイベント指向で書き直した場合とを、プログラムリポジトリで見比べてみることにします。

下図は、V8のレコードメインとタスクイベントテーブルの例です。

## V8 のレコードメインとタスクイベントテーブル

		, 🏼	「ブレイク	」 処3	理/ベ	ļ		前処理	ж)	後処理	「トランザ・ク	990	ΙĴ	-					
1020			1			!		0	91	1	3 L=オンロッ	ケー	A=)	味~~	<u>۲</u>	-			
		'⊢	2	37	スク			0			Yes		A=)	府~~	۲.				
																_			
	一义	処理テー	ブル:レコ	1 – K	- ×1	ン ―													
	[	処理コ	マンド			内容	[		範囲		位置作	ţ	170	1-	条件				
	1	也外	V≕変数	:	1	V.商品DLRC	代入	, O	0	0	0	0	S	С	Yes				
	2	也外	V≕変数	:	2	V.物流拠点省略値DLRC	代入	. 0	0	0	0	0	S	С	Yes				
	3	반가	V≕変数	:	3	V.物流拠点(開始)DLRC	代入	. 0	0	0	0	0	S	С	Yes				
	4	也外	V=変数	:	4	V.物流拠点(終了)DLRC	代入	. 0	0	0	0	0	S	С	Yes				
	5	也外	V≕変数	:	5	V.99714DLRC	代入	. 0	0	0	0	0	S	С	Yes				
	6	11.61					40.7												
_	-/	20/21	REET	: :	1	文)田	147	. 29	29	29		0	8	U	Yes				
	8	パクジョン	n-=	:	1	KBPUT (*)欠項目 ACT) 会社BBAm#	442.7			0	_ 戻	17	2	F	/	-			
	9	2025	ドー夫アン	: :	2	文7士 ⁴ 月前田県   「 P わわわましい ³ - 101 00 D	IT.A	. 24	- 0	U	- 28	U	0	0	Tes				
	11	7 H97 Hell	D- Դ°ոհ*։		500	LF.WYTTP - U UKF. SO避視 帝日二階	825		7451	0		No		<u>с</u>	Vac	-			
	12	⊣_⊪ ⊎ՐԴԻ	- FE ( 17 17 1	1 •	000	30)温が、Mana一見 商品#	14° 7	0	74-6	0	- H97 0	NU 0	0	с С	Ves	-			
	13	机水	、 R=宝デジ	· ·	4	高品Subt	1 (th )	<u> </u>	- 0	0	- 0	0	8	0	No	-			
	14	970	0=昭会		16	商品	15	- 		A=昇順	Ē	LW	Ť		Yes	-			
	15	地外	R=実デー	1:	1	商品#	代入	. 0	0	0	39	39	s	С	No	-			
	16	也外	R=実デー	::	2	商品Sub#	代入	. 0	0	0	40	40	S	С	No	-			
	17	11/31	R=実デー	ı:	9	規格_型番	代入	, O	0	0	0	0	S	С	No	-			
	18	也外	RE実がら	ł :	7	商品名	代入	. 0	0	0	0	0	S	С	No				
	19	地外	RE実がら	ł :	11	¥ディア	代入	. 0	0	0	0	0	S	С	No				
	20	也小 🧹		• •	10	946 / TT	767	•	-	0	-	0	0	^	N.				
	21	也外 .	着 タスク	巜	ント・	718 - SO入力 受测	主情報	g1 (597)	)										
	22	也外																	_
	23	也外	# 7	5ツ	<u> </u>	-   アクション		経過時間	元   1	ース	コーブ	<u> </u>	<u>- 1</u>	/	Pr	rog/1	fask	ハッシントダ	
	24	划外	1			ユーザアクション	1	00:00:00		0 T=3	12.7	P=7	ים״ן	1.21			626	1	1
			2			ユーザアクション	2	00:00:00		0 T=3	れたり こうしん しんしょう しんしょ しんしょ	P=7	ים°י	パラム			620	2	
			3			ユーザアクション	3	00:00:00		0 T={	れた	P=7	ים°י,	パラム			619	2	1
			4			ユーザアクション・	4	00:00:00		0 T={	れた	P=7	ים°ע	パラム			618	2	
			5			ユーザアクション	5	00:00:00	1	0 T=3	7,7,5	P=7	)° D'	パラム			616	3	i lī
			6			ユーザアクション	6	00:00:00	I	0 T=4	772	P=7	°ם?	1.51			621	2	1
			7			コーザマクション	7	00.00.00		0 T=2	174	P=T	1°n/	1851			561	1	

これを見てわかるように、プログラムの内容を理解するのが難しいものとなっています。

- どの項目において、何をすると何が起こるのかが、一見しただけではわからない。レコードメインの行の前後をフローモードや条件(パークの有無)に注意しながら追ってみて、始めて理解することができる。
- プッシュボタンとその処理との関係がタスクイベントテーブルだけからはわからない。プッシュボタンと ユーザアクションnの関係、および、プログラム番号とその内容の関係がわかって、始めて、タスクイベントテーブルの内容も理解できるようになる。

これに対し、V10のイベント指向のやりかたで書き直したのが、次のページの図です。

10の	イ	ベント指向	で書き直した	:もの	)						
8479	72	25.1 – SO,	入力 受注情報	(元5	98).SO受注明細						X
データ	ビュ	. ー ロジック	フォーム								
1	9 E	5 E=11°)1	U_9971番号3	発行	ጋንኑ PB_୬	リアし 非確認	!	スコーフ* T=タスク	条件 Yes		~
2	20	a-N	P=7°ロク°ラム	589	SO照会 沙개雄	[2 //°5%	勿				
2	21 E	3 E=イベント	GU_X*~&		コント商品			スコーフ* T=タスク			
2	22	a-N	P=プログラム	567	SO選択 商品一覧	[3 //*5×	-妇				
2	23 E	0 C=3)ha-6	¥=検証	- <b>1</b> 2H	商品						
2	24	15-	₩警告	4	?無効な商品マスタが選択さ	表示:	B=‡°	9 <b>9</b> 7	条件: 34	(P. ዓአ/ት-ኑ*=")	
2	25 E	0 C=3)ha-6	S=後	324	商品						
2	26	べい実行	次項目					ウェイト: No	条件: 7	P.\$Z\$Ŧ~ト*="C	
2	27 E	0 C=3)ha-6	Y=検証	3)1	ケディア						
2	28	15-	E=I5-	3	'Magic製品として許され	,表示:	S=54	2	条件:20	商品如7°='MGI	
2	29 E	3 C=3)ha-6	¥=検証	- 1) H	物流拠点#_START						
3	0	15-	E=I5-	5	'\$物流拠点#(開始)が正(	.表示:	S=54	2	条件:10	Ⅴ.物流拠点(開	
3	11 E	0 C=3)ha-6	¥=検証	3)1	物流觀点#_END						
3	32	15-	E=I5-	6	'\$物流拠点#(終了)が正(	.表示:	S=54	2	条件: 11	Ⅴ.物流拠点(約	
3	33 E	C=3)10-1	P=前	3)1	仕切						
3	34	項目更新	V=項目	NX .	仕切	値:	54	受注単価/受注定価*1	条件:27	(P.\$スクモード=゚)	
3	85 E	C=3)10-1	¥=検証	- 1) H	仕切						
3	86	項目更新	V=項目	NV	受注単価	值:	55	受注定価*仕切/100	条件:27	(P.\$298-h*=1)	
3	17 E	∃ ¥=項目	C=変更	NY	受注数量						
3	88	項目更新	V=項目	NS	受注金額	值:	56	受注单価*受注数量			
3	39	項目更新	V=項目	NT	粗利金額	値:	57	(受注単価-受注原価質			

V8 のものと比較すれば明らかなように、V10 のイベント指向の記述は、プログラムの意図がずっと明瞭になっていて、どこでいつ何が起こるかが一見して理解することができます。

例えば、上から見ていくと、

- 「PB_シリアル#確認」という名前のコントロール (プッシュボタン)で、「U_シリアル番号発行」というイベントが 発生すると、プログラム 589 番「SO 照会 シリアル#」が呼び出される。
- 「商品#」でズームすると、プログラム 567 番「SO 選択 商品一覧」が呼び出される。
- 「商品#」を通りすぎるときに(パークの有無に関わらず)、条件34に合致すればエラーが表示される。

などのことがすぐにわかります。

このように、イベント指向プログラムは、Magic プログラムの作成・メンテナンスを容易にしてくれる大きな可能性を持っています。ぜひとも、イベントハンドラに熟達してください。

## Magic eDeveloper V10



Magic eDeveloper V10

レコードメインからイベントへ

Copyright  $\ensuremath{\mathbb{C}}$  2007, Magic Software Japan K.K., All rights reserved.

第1版 2007年10月12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/