Magic xpa
レコードメインからイベント化移行

マジックソフトウェア・ジャパン株式会社

はじめに

- Magicは、V10からイベント指向プログラミングに移行し、レコードメインは旧バージョンからの互換レベルで利用できるようにしておりましたが、それからすでに15年が経過いたしました。
- 本セミナーでは、レコードメインによるプログラミングのデメリットと、イベント化することによるメリットをご案内すると共に、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。
- レコードメインからイベント志向のプログラミングへの移行を考えているMagic 技術者に是非お聞きしていただきたい内容です。

目次

- Ｖ８／Ｖ９のおさらい： レコードメイン（RM）はどう動いていたか？
- Magic xpaタスクエディッタの構成
- 例① リンクの成功チェックをする
- 例② 条件により入力項目をスキップする
- 例③ 項目更新により再計算を行う
- 例④ プッシュボタンをハンドリングする
- 例⑤ ズーム処理を行う
- まとめ

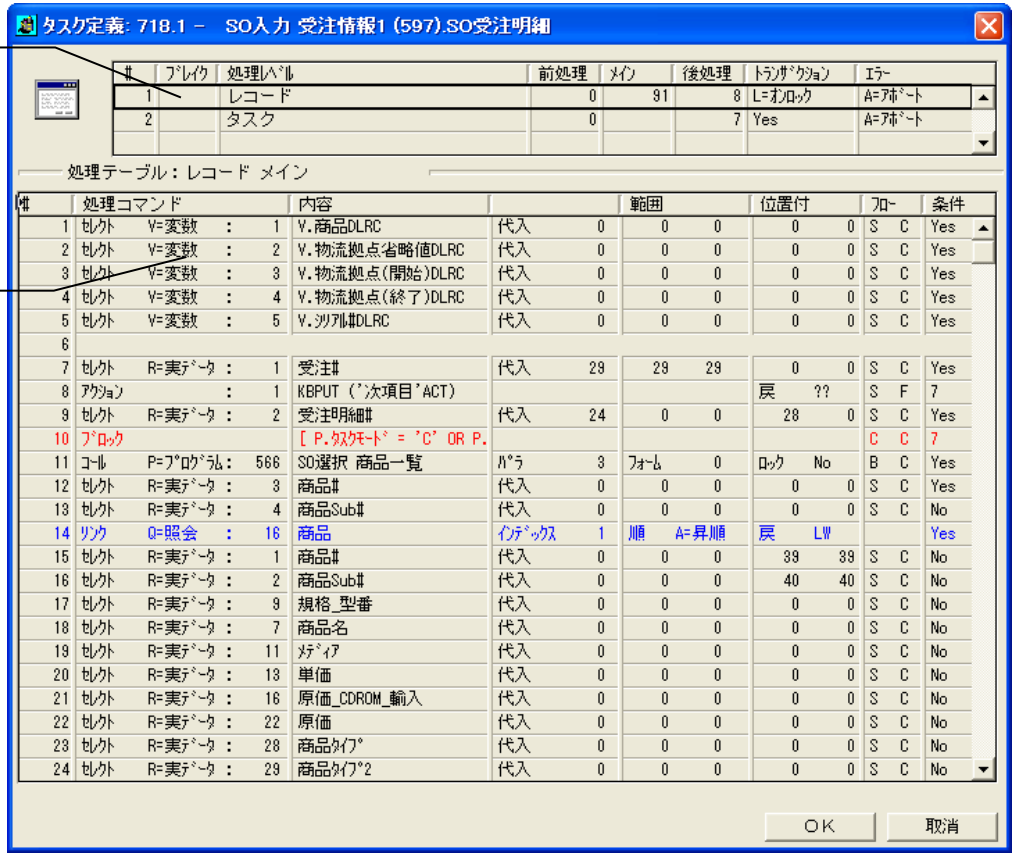
Ｖ８／Ｖ９のおさらい
レコードメイン（RM）
はどう動いていたか？

Ｖ８のタスクエディッタの構成

レベルテーブル
レコード、タスク、
ブレイクレベルなど

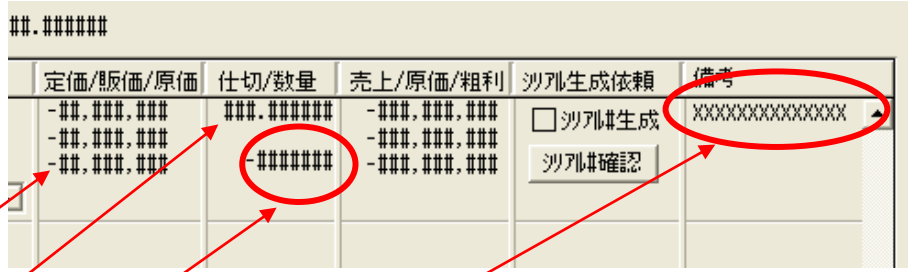
処理テーブル
レコードメインでは
- データ定義
- オンライン画面上での操作

を定義



ＲＭの基本的考え方は

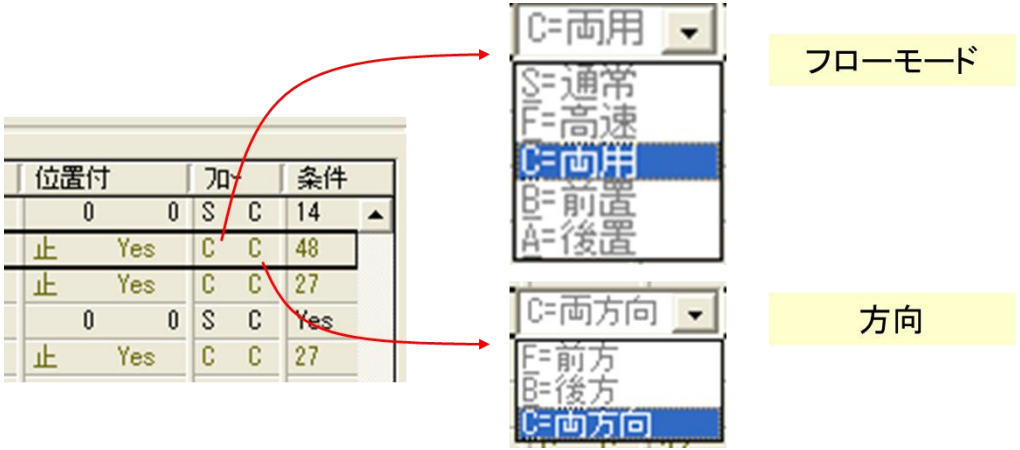
- 「カーソルの動きに応じて、ＲＭに処理を記述する」



65	ｾﾚｸﾄ	R=実ﾃﾞｰﾀ :	8	受注定価	代入	45
66	ｾﾚｸﾄ	R=実ﾃﾞｰﾀ :	9	受注単価	代入	45
67	ｾﾚｸﾄ	R=実ﾃﾞｰﾀ :	10	受注原価単価	代入	46
68	項目更新	:NU		受注原価単価	式	49
69	項目更新	:NV		仕切	式	62
70	ｾﾚｸﾄ	R=実ﾃﾞｰﾀ :	11	仕切	代入	0
71	項目更新	:NT		受注単価	式	63
72	ｾﾚｸﾄ	R=実ﾃﾞｰﾀ :	12	受注数量	代入	22
73	項目更新	:NY		受注金額	式	64
74	項目更新	:NZ		粗利金額	式	65
75	ｾﾚｸﾄ	R=実ﾃﾞｰﾀ :	21	備考	代入	67

受注数量 から 備考 に移るときに、この二つの項目更新コマンドが実行される

ＲＭフローパラメータによる実行の制御



フローモード：

C=両用 → S=通常 ＋ F=高速

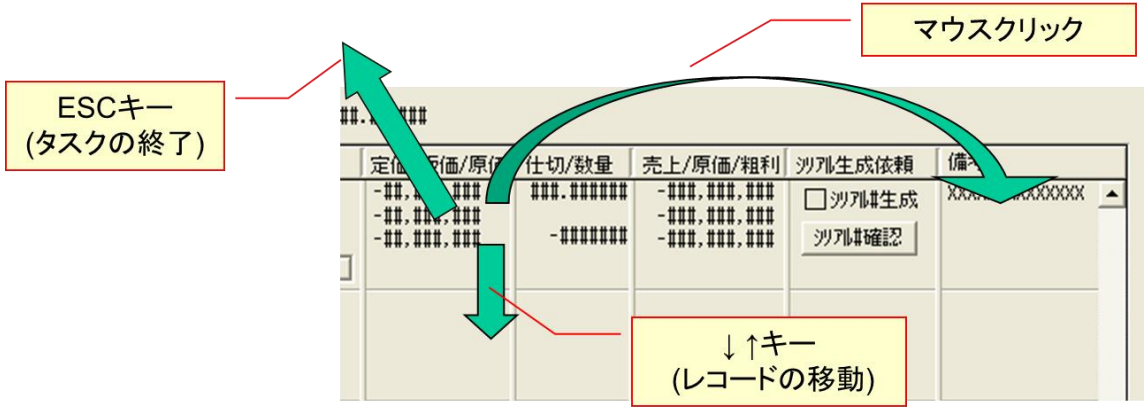
B=前置、A=後置 → ズームで使う

方向： C=両方向 → F=前方 ＋ B=後方

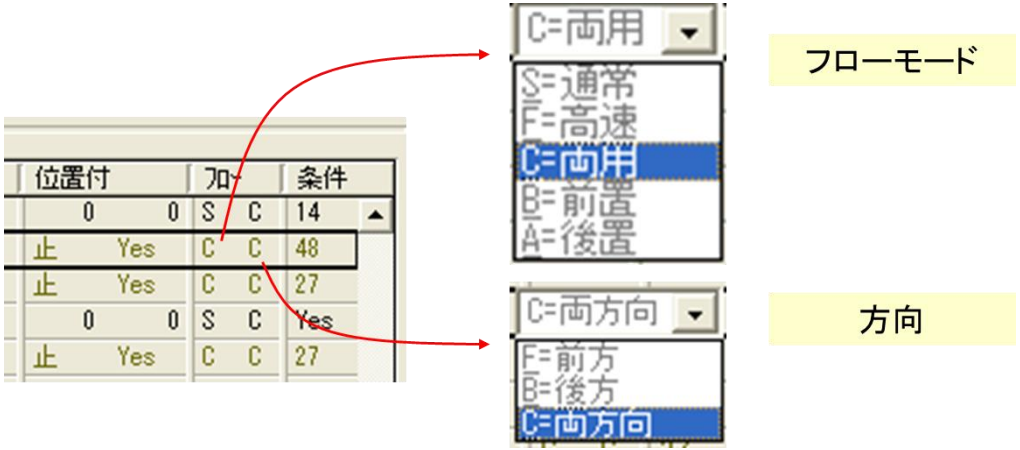
ＲＭフローモード

- S=通常モード： Enter/Tab (次項目)で動く場合に実行

- F=高速モード： マウス/↑↓キー/タスク終了などでカーソルが動く場合に実行



ＲＭフローパラメータによる実行の制御



フローモード：

C=両用 → S=通常 ＋ F=高速

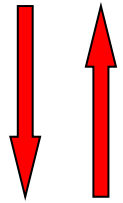
B=前置、A=後置 → ズームで使う

方向： C=両方向 → F=前方 ＋ B=後方

ＲＭカーソル移動方向

- 順方向にカーソルが動く → 上から下にコマンド実行
- 逆方向にカーソルが動く → 下から上にコマンド実行
- 特定の方向のときだけ実行したい → 方向パラメータをＦ＝前方/Ｂ=後方に設定

72	ｾﾚｸﾄ	R=実ﾃﾞｰﾀ : 12	受注数量	代入	22
73	項目更新	:NY	受注金額	式	64
74	項目更新	:NZ	粗利金額	式	65
75	ｾﾚｸﾄ	R=実ﾃﾞｰﾀ : 21	備考	代入	67



逆方向

順方向

※ 実際にはもっと複雑な動きのルールあり。

ＲＭフローモードは結構難しいブラックボックス。

ＲＭの書き方ＡＢＣ

- 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

  - [次項目]アクション（Enter/Tabキ）で動く場合のみ実行 → S=通常
  - ↓↑キー、ESC、マウス等で動く場合 → F=高速
  - 常に実行する → C=両用　※ 特定の移動方向を限定したければ、F=前方またはB=後方を設定

- 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。

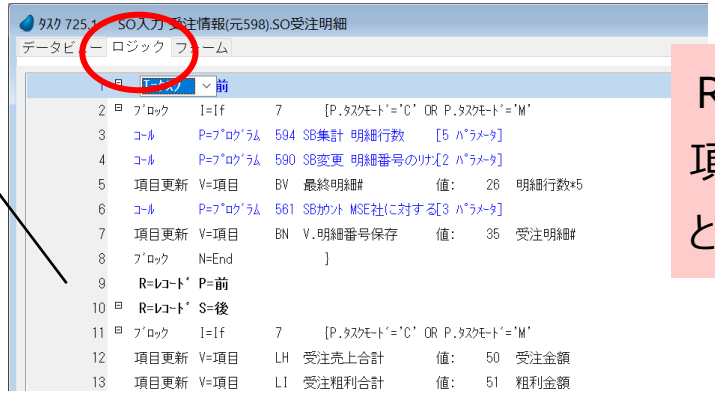
Magic xpa の
タスクエディタの構成

Magic xpaタスクエディッタの構成

データビュー：
データの定義だけ
（ＲＭのセレクト、リンクコマンド）

データビューにはロジックを書きません

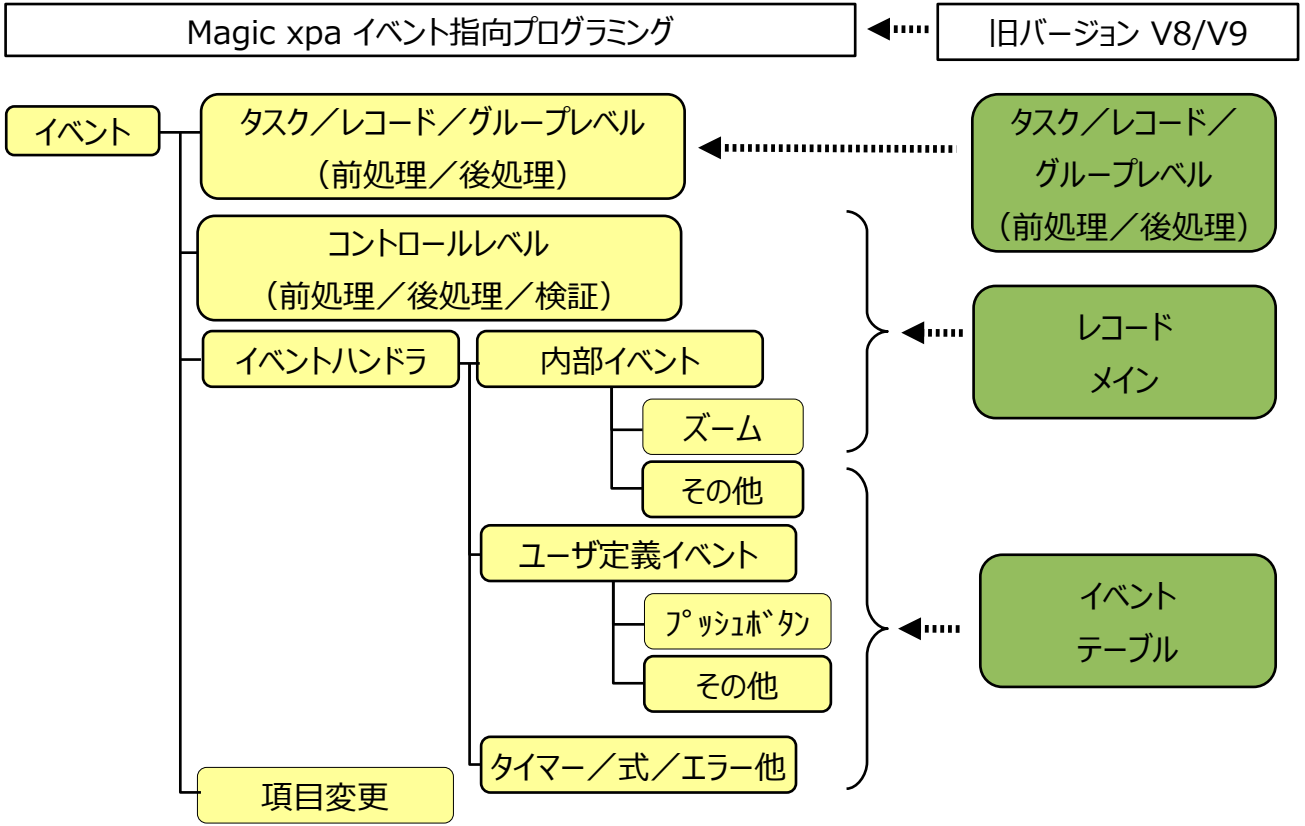
ロジック：
ＲＭ以外のレベル
（タスク前/後、レコード前/後・・・）
RMの処理コマンド



ＲＭに書いていたコマンド（コール、項目更新、・・・）は、すべて**ハンドラ**として記述。

Magic xpaでのハンドラの種別とタイプ

すべて「ロジック」タブにイベントハンドラとして記述します

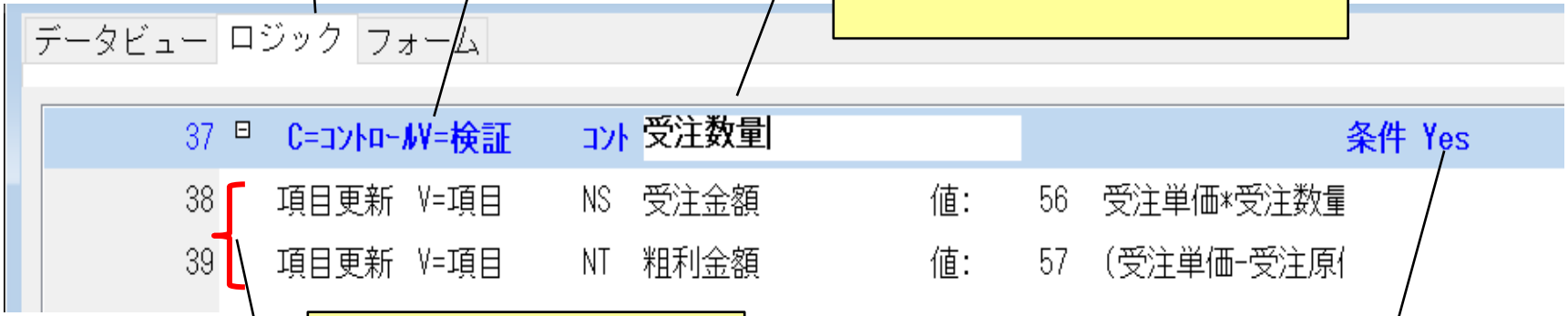


ハンドラの書き方

ロジック タブに記述

ヘッダ行： ハンドラごとに、種別とタイプ他を定義。
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

コントロール名： カーソルがここにあるときに、
ハンドラが有効になる。

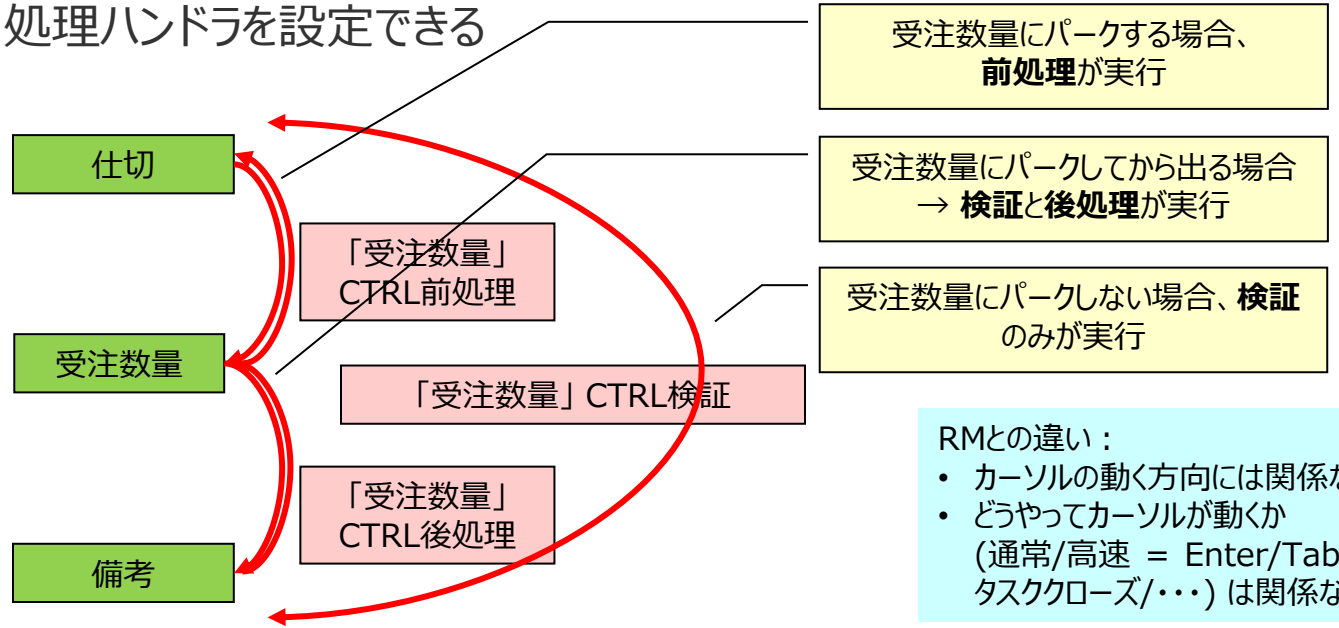


詳細行： 処理内容を記述。
項目更新、コール、アクション、 ・・・
などのコマンド

条件： ハンドラが有効に
なる条件を式で指定

コントロールレベルのハンドラ

- オンライン画面のカーソル移動時に実行すべきコマンドは、コントロールレベルのハンドラに記述する。
- 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ハンドラを設定できる



RMとの違い：
- カーソルの動く方向には関係ない
- どうやってカーソルが動くか
(通常/高速 = Enter/Tab/マウス、↑↓キー/タスククローズ/・・・) は関係ない。

コントロールレベルハンドラの書き方ＡＢＣ

- A) どこで？ →ある表示項目 (例： 受注数量) において・・・
- B) いつ？
  - カーソルが項目にパークする場合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
  - 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
- C) 何を？ → ハンドラの中にコマンドを記述

ＲＭ→コントロールハンドラのTIPS

- 機械的に変換しようとしても、却って不必要に複雑になってしまう。
- 「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。
- フローモード、方向により実行を制御したい場合、パラメータ設定もできる。(後述)

※ 注意：

- 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す。
- RM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。

例① リンクの成功チェック

例① リンクの成功チェック （例： 商品#）昔のやりかた

（A） どこで？
→「商品Sub＃」項目の後で

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
11	ｺｰﾙ	P=ﾌﾟﾛｸﾞﾗﾑ： 566	SO選択_商品一覧	ﾊﾟﾗ 3	ﾌｫｰﾑ 0	ﾛｯｸ No	B C	Yes
12	ｾﾚｸﾄ	R=実ﾃﾞｰﾀ ： 3	商品#	代入 0	0 0	0 0	S C	Yes
13	ｾﾚｸﾄ	R=実ﾃﾞｰﾀ ： 4	商品Sub#	代入 0	0 0	0 0	S C	No
14	ﾘﾝｸ	Q=照会 ： 16	商品	ｲﾝﾃﾞｯｸｽ 1	順 A=昇順	戻 LW		Yes
15	ｾﾚｸﾄ	R=実ﾃﾞｰﾀ ： 1	商品#	代入 0	0 0	39 39	S C	No
16	ｾﾚｸﾄ	R=実ﾃﾞｰﾀ ： 2	商品Sub#	代入 0	0 0	40 40	S C	No
17	ｾﾚｸﾄ	R=実ﾃﾞｰﾀ ： 9	規格_型番	代入 0	0 0	0 0	S C	No

この間、パークしない項目（リンク項目）

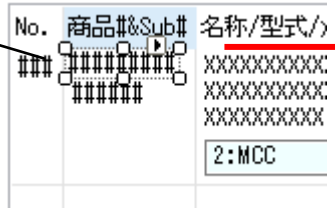
39	ｾﾚｸﾄ	R=実ﾃﾞｰﾀ ： 49	ｼﾘｱﾙ#自動生成対象	代入 0	0 0	0 0	S C	No
40	ｾﾚｸﾄ	R=実ﾃﾞｰﾀ ： 53	商品有効F	代入 0	0 0	0 0	S C	No
41	ﾘﾝｸ終了							
42	ｴﾗｰ	： 4	'$無効な商品ﾏｽﾀが選択さ	ﾓｰﾄﾞ W=警告			C C	37

（C）何を？ →
エラーコマンド

（B） いつ？ → 常に実行
→ C=両用、C=両方向
（リンクフラグで条件付け）

例① リンクの成功チェック： Magic xpaでは

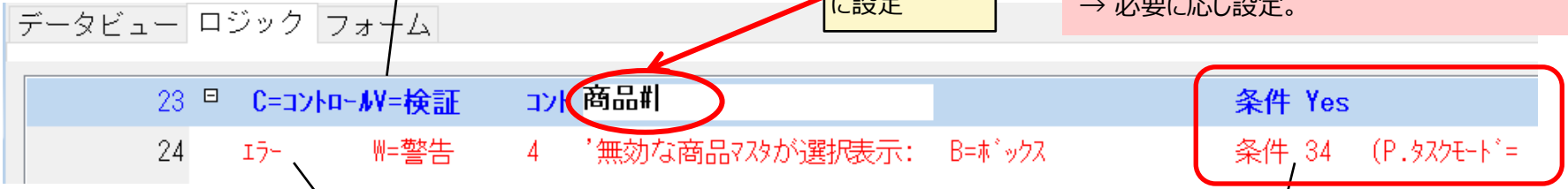
（A） どこで？
→「商品＃」項目



（B） いつ？ → 常に実行
→ コントロール検証

コントロール欄に設定

※ V8 APだと、コントロール名がついていないこと多い
→ 必要に応じ設定。

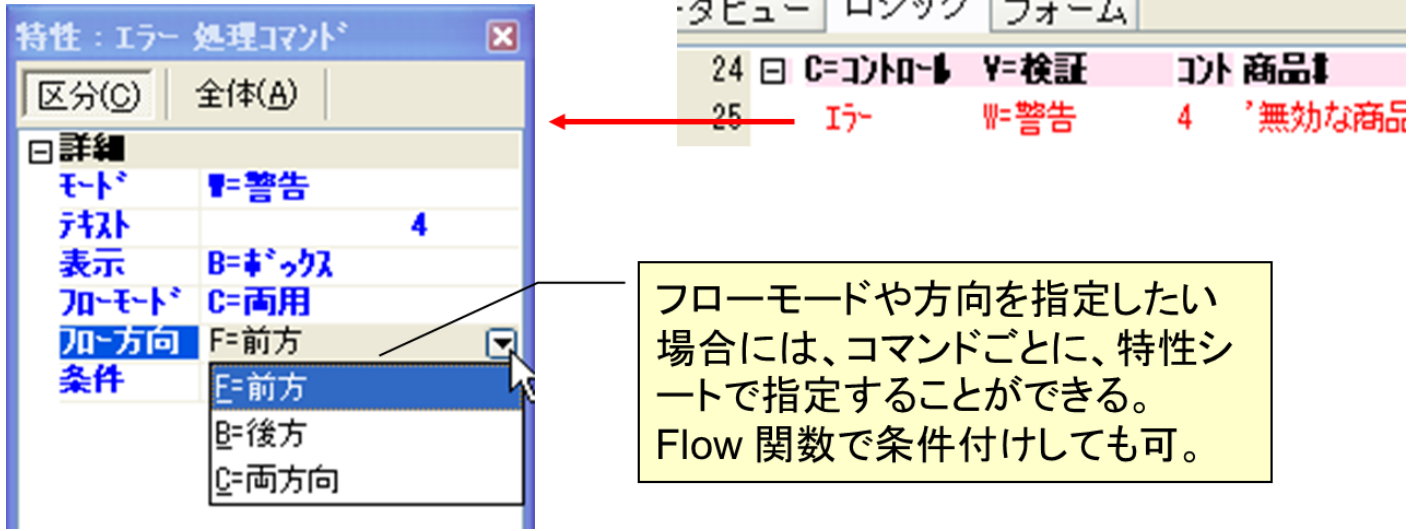


（C）何を？（複数のコマンド指定可）
→ エラーコマンド

コマンドの実行の条件付け

例① 微妙な違い

- RMの場合、前の項目に戻る場合にはエラーチェックを行わなかった。
- Magic xpa コントロール検証の場合には、前方向でも後方向でもエラーチェックを行う。



フローモードや方向を指定したい場合には、コマンドごとに、特性シートで指定することができる。
Flow 関数で条件付けしても可。

例② 条件により、入力項目を飛ばす

例② 受注番号の後でＫＢＰＵＴ（'次項目'）を行う

受注＃の直後で、
KBPUT('次項目' ACT)を実行

フローは
S=通常
F=前方

どういう動作になるか？

受注＃から Enter／Tabで移動するとき、

条件 7 が成立したら、次の項目（受注明細＃）をスキップする。

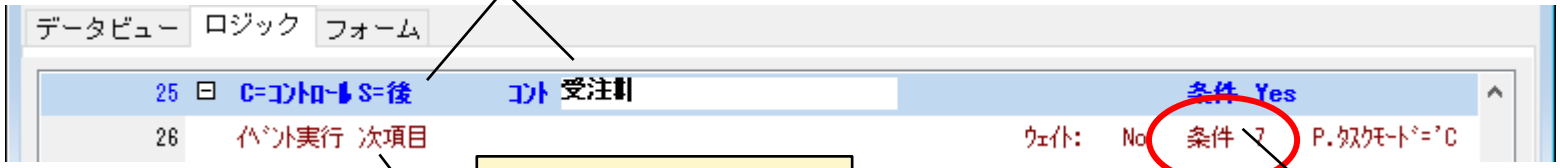
※ (S=通常モード のみ) マウスやレコード移動時は実行しない

※ (F=前方 のみ) 受注明細# から 受注# に戻る場合も実行しない

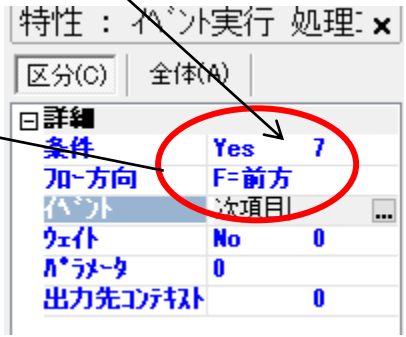
例② xpa コントロールレベルハンドラで書くと・・・

（A） どこで？
「受注＃」で

（B） いつ？ → パークして出るときだけ
→ コントロール後処理



（C） 何を？
「次項目」イベントを実行



「条件」と「フロー方向」で条件付け

※ KBPUTでなく、イベント実行 コマンドを使いましょう。

※ ただ、この例について言えば、受注明細＃ 項目のパーク条件で制御した方がよい。
→ 別の方法による解も考えてみましょう。

例③ 項目更新による再計算

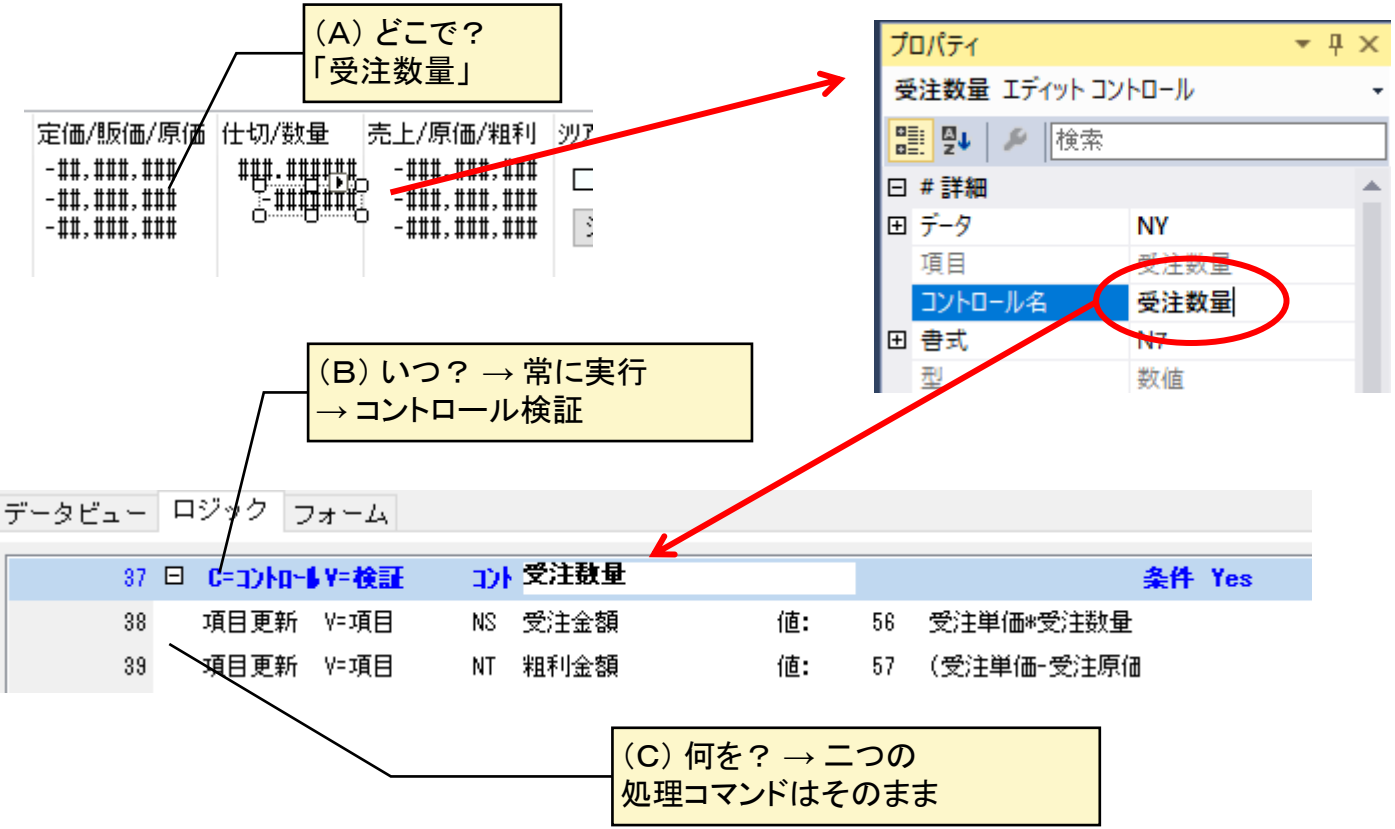
例③ 項目更新による再計算ロジック

処理テーブル：レコード メイン

#	処理コマンド		内容		
72	セレクト	R=実データ : 12	受注数量	代入	22
73	項目更新	:NY	受注金額	式	64
74	項目更新	:NZ	粗利金額	式	65
75	セレクト	R=実データ : 21	備考	代入	67

- 「受注数量」から「備考」に移る時、受注金額と粗利金額の再計算を行う。

例③ Magic xpa コントロール ハンドラで書くと…



例③ 微妙な違い

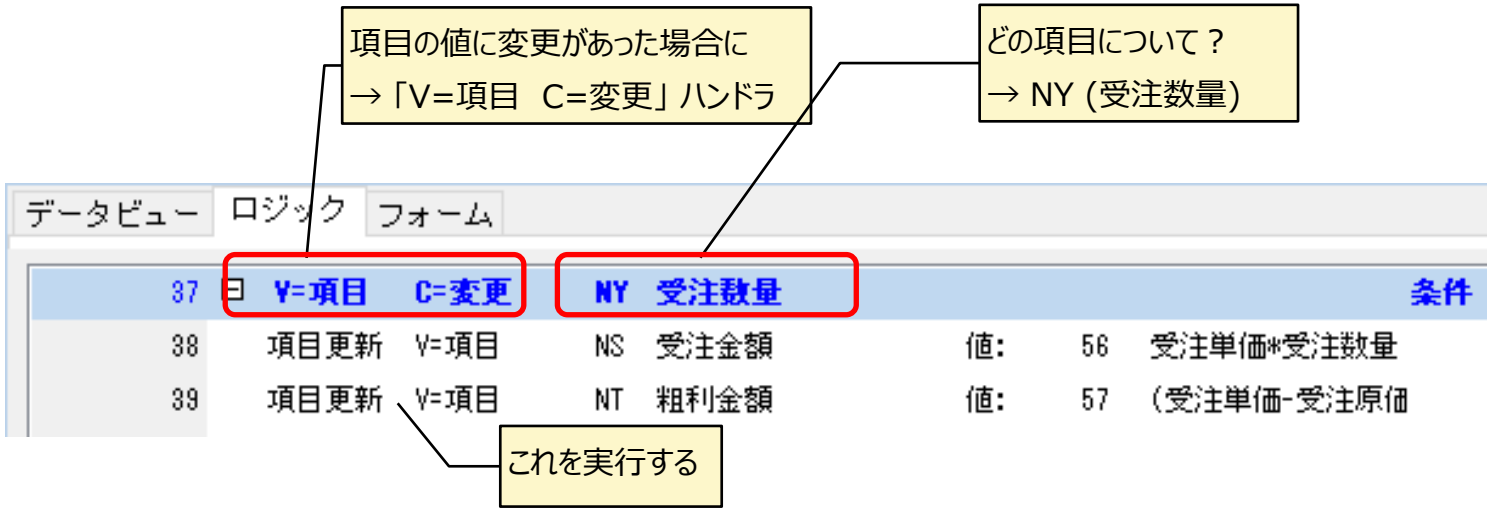
- 項目更新コマンドが二つある (受注金額と粗利金額)
- RMの時は、カーソルの移動方向により、コマンドの実行順序が異なる。
  (逆方向の場合は、下から上へ)
- Magic xpa コントロール検証の時は、カーソルの移動方向に関わらず、上から下に実行される。

✓ この例では、どちらでも結果は同じになるので変更する必要はないが、依存関係のある場合には実行結果が異なるので注意。

✓ 必要なら方向パラメータなどで条件づけ。

項目変更ハンドラ （V10～ 新機能）

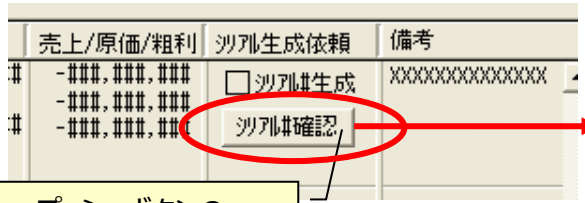
- 例③のように、項目（ex. 受注数量)の値の変更をきっかけとして実行すべきコマンドは、「項目変更」ハンドラにした方がわかりやすい。



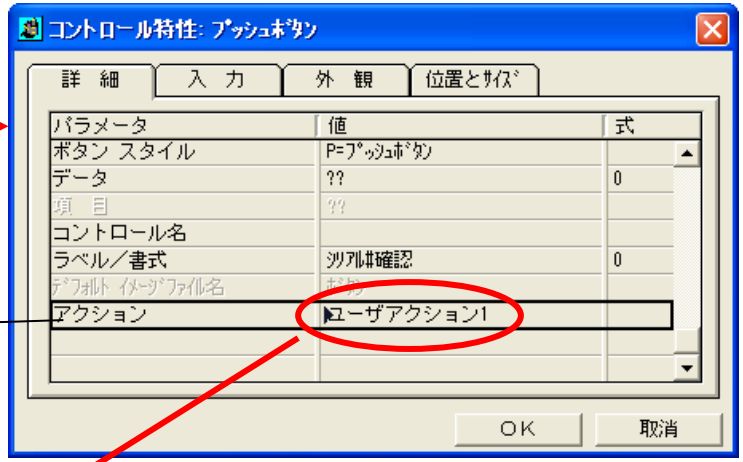
- 意図がわかりやすい。
- 設定忘れがなくなる： いつ、どこで、どのようにして変更されたかには関係なく、変更があったら常に実行される。
（ex. ユーザ入力、項目更新コマンド、プログラムへのパラメータ、など。画面に表示されていなくともよい）
- 値に変更がない場合には実行されない → 無駄な再計算をなくす

例④ プッシュボタン

例④ プッシュボタン： V8のタスクイベントテーブル

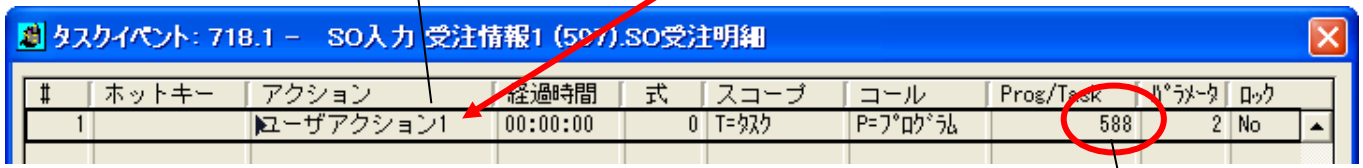


プッシュボタンの
コントロール特性で



アクションを設定
（ユーザアクション１）

タスクイベントテーブルにア
クションを指定

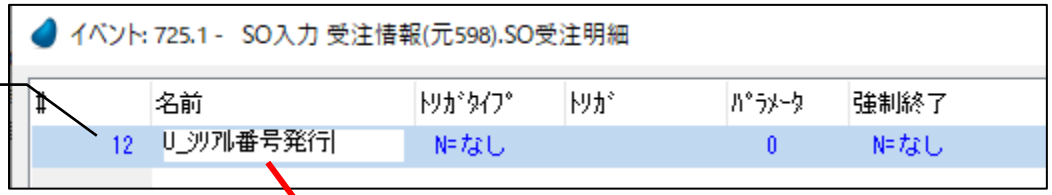


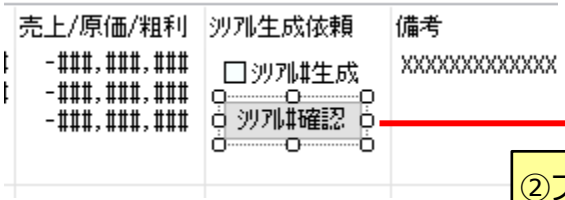
処理内容は別タスク/プロ
グラムに記述

例④ プッシュボタン： Magic xpa ユーザ定義イベントを利用

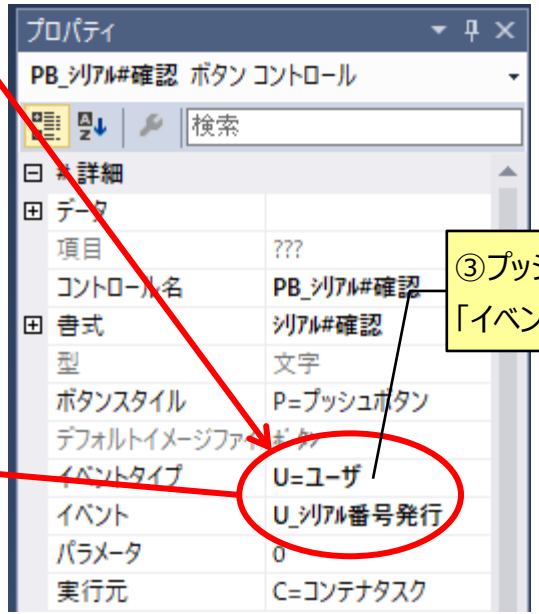
「ユーザアクションn」の代わりに、**ユーザ定義イベント** を使うのが便利です

① ユーザイベントテーブルにわかりやすい名前でイベントを作成





②フォーム デザイナ上でコントロール特性を開く



③プッシュボタン特性に「イベント」を設定

ボタンを押したときに、このイベントが発行されます。
タスクイベントテーブルで定義していた処理は、Magic xpaでは**イベントハンドラ**で行います。

Magic xpa イベントハンドラの書き方ＡＢＣ

ハンドラの種別は
「E=イベント」

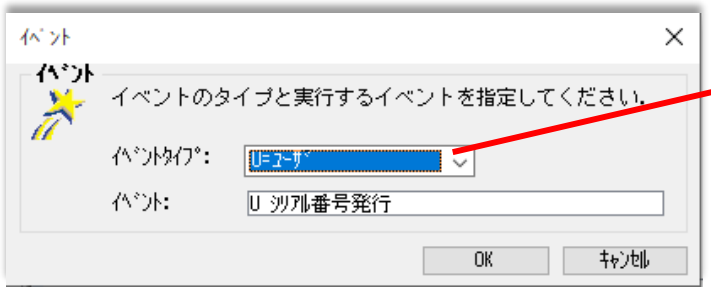
(A)どこで？
「ｼﾘｱﾙ＃確認」プッシュボタンで。
→ プッシュボタンのコントロール名
「PB_ｼﾘｱﾙ＃確認」を指定

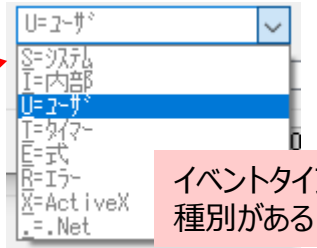
E=イベント のハンドラでは省略も可。
省略すると「どこでも有効」になる

(C) 何を？
実行するコマンドを記述

(B)いつ？→ トリガとなる
ユーザ定義イベント「U_ｼﾘｱﾙ
番号発行」を指定

ズームして詳細設定





イベントタイプにはこれだけの
種別がある

ユーザ定義イベントを使うイベントハンドラ方式の利点

- わかりやすい。
  - V8では「ユーザアクションn」しかなく、意味がわからなかった。
  - Magic xpaでは意味のある名前 (ex. U_ｼﾘｱﾙ番号発行) で記述できる。
- イベント数に制限なし。
  - 「ユーザアクションn」は20個までだった。
  - それ以上の場合には、CTRLNAME()で条件付けしていた。
- 別途プログラムを作る必要なし。
  - Ｖ８では中身がコマンド１行でも、別途タスクを作る必要があった。

V8 vs. Magic xpa

タスクイベント: 718 – SO入力 受注情報1 (597)

#	ホットキー	アクション	経過時間	式	スコープ	コール	Prog/Task	パラメータ	ロック
1		ユーザアクション1	00:00:00	0	T=タスク	P=プログラム	626	1	No
2		ユーザアクション2	00:00:00	0	T=タスク	P=プログラム	620	2	No
3		ユーザアクション3	00:00:00	0	T=タスク	P=プログラム	619	2	No
4		ユーザアクション4	00:00:00	0	T=タスク	P=プログラム	618	2	No
5		ユーザアクション5	00:00:00	0	T=タスク	P=プログラム	616	3	No
6		ユーザアクション6	00:00:00	0	T=タスク	P=プログラム	621	2	No
7		ユーザアクション7	00:00:00	0	T=タスク	P=プログラム	561	1	No

V8：どこで何が実行されるかわかりにくい。

タスク 725 - SO入力 受注情報(元598)

データビュー | ロジック | フォーム

51	⊟	E=イベント	U_受注書		コント PB_受注書	スコープ T=タスク	条件	Yes
52		コール	P=プログラム	629	SBE印刷 受注書	[1 パラメータ]		
53	⊟	E=イベント	U_仕入先注文書			スコープ T=タスク		
54		コール	P=プログラム	562	SBE印刷 MSE社に対する発[1 パラメータ]			
55	⊟	E=イベント	U_手配書		コント PB_手配書	スコープ T=タスク		
56		コール	P=プログラム	624	SBE印刷 手配書	[2 パラメータ]		
57	⊟	E=イベント	U_宅配伝票		コント PB_宅配伝票	スコープ T=タスク		
58		コール	P=プログラム	617	SBE印刷 宅配伝票	[3 パラメータ]		
59	⊟	E=イベント	U_請求書		コント PB_請求書	スコープ T=タスク		
60		コール	P=プログラム	619	SBE印刷 請求書	[2 パラメータ]		
61	⊟	E=イベント	U_送り状		コント PB_送り状	スコープ T=タスク		
62		コール	P=プログラム	621	SBE印刷 送り状	[2 パラメータ]		
63	⊟	E=イベント	U_納品書		コント PB_納品書	スコープ T=タスク		
64		コール	P=プログラム	623	SBE印刷 納品書	[2 パラメータ]		

xpa: 意図は明瞭

例⑤ ズーム処理

例⑤ ズーム処理　（昔のやりかた）

- セレクトコマンドの直前にコマンド記述
- コマンドのフローモードをB＝前置

11	ｺｰﾙ	P=ﾌﾟﾛｸﾞﾗﾑ：	566	SO選択 商品一覧	ﾊﾟﾗ	3	ﾌｫｰﾑ	0	ﾛｯｸ	No	B	C	Yes
12	ｾﾚｸﾄ	R=実ﾃﾞｰﾀ ：	3	商品#	代入	0	0	0	0	0	S	C	Yes

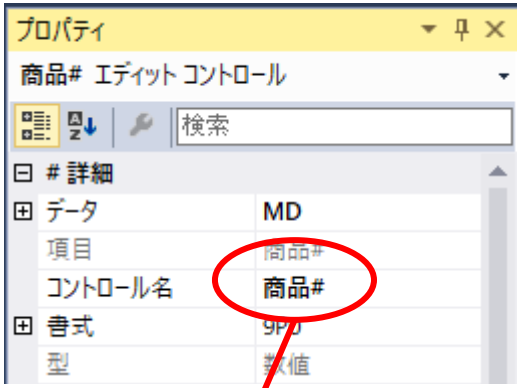
- 複数のコマンドのある場合は、ブロックコマンドで囲み、B＝前置を設定

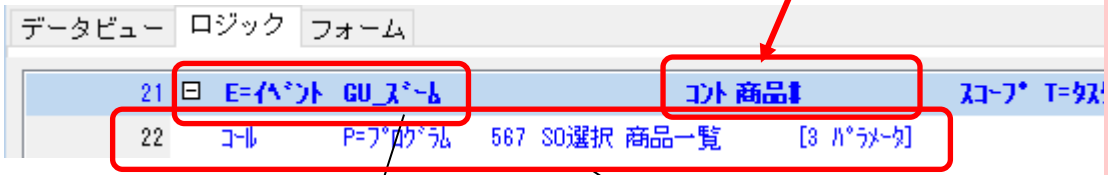
370	ﾌﾞﾛｯｸ			[Yes							B	C	Yes
371	ｺｰﾙ	P=ﾌﾟﾛｸﾞﾗﾑ：	587	SO入力 営業実績配分率	ﾊﾟﾗ	2	ﾌｫｰﾑ	0	ﾛｯｸ	Yes	S	C	Yes
372	項目更新	:H		V.子ﾀｽｸ終了F	式	3	計	N=代入	止	Yes	C	C	Yes
373	項目更新	:I		V.子ﾀｽｸ画面消去F	式	4	計	N=代入	止	Yes	C	C	Yes
374	ｺｰﾙ	T=ﾀｽｸ ：	3	営業実績配分	ﾊﾟﾗ	0	ﾌｫｰﾑ	0	ﾛｯｸ	Yes	S	C	Yes
375	ﾌﾞﾛｯｸ終了			]									
376	ｾﾚｸﾄ	V=変数 ：	45	V.営業実績配分	代入	13	0	0	0	0	S	C	Yes

例⑤ ズーム処理 ： Magic xpa イベントハンドラを使うと・・・

（A） どこで？
「商品＃」の項目で

フォームデザイナ上で
ズーム項目を選択





「ズーム」を直接ハンドルするのではなく、ユーザ定義イベント「GU_ズーム」をハンドルしている。
（理由 → 次へ）

（B） いつ？
ユーザ定義イベント
「GU_ズーム」が起きたら

（C） 何を？
プログラムをコール

ユーザ定義イベント「ＧＵ_ズーム」を使う理由

メインプログラムのイベント
テーブルで定義。
1箇所の定義でAP全体に有効。

イベント: 1 - メインプログラム

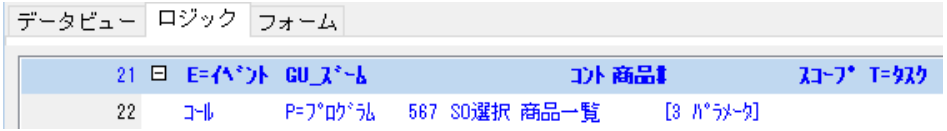
順	名前	ﾄﾘｶﾞﾀｲﾌﾟ	ﾄﾘｶﾞ	ﾊﾟﾗﾒｰﾀ	強制終了
1	GU_一覧	N=なし		0	E=編集
2	GU_実行N	N=なし		0	N=なし
3	GU_実行F	N=なし		0	E=編集
4	GU_ｽﾞｰﾑ	I=内部	ｽﾞｰﾑ(Z)	0	E=編集

F5キーを押す → ズーム アクションがトリガとなり、**GU_ズーム** イベントが発行される。

「強制終了」が E=編集 の場合には、まず項目に入力途上の値が反映される。

フォーム上のエディットコントロール
※ 入力途中では、まだ項目に値が反映されていない。

入力途中のデータを項目に反映させるため、強制終了 E=編集 のユーザ定義イベントを介在させる必要がある。
(そうしないと、入力途中のデータが反映されないまま、以前の値がコールコマンドのパラメータに渡される)

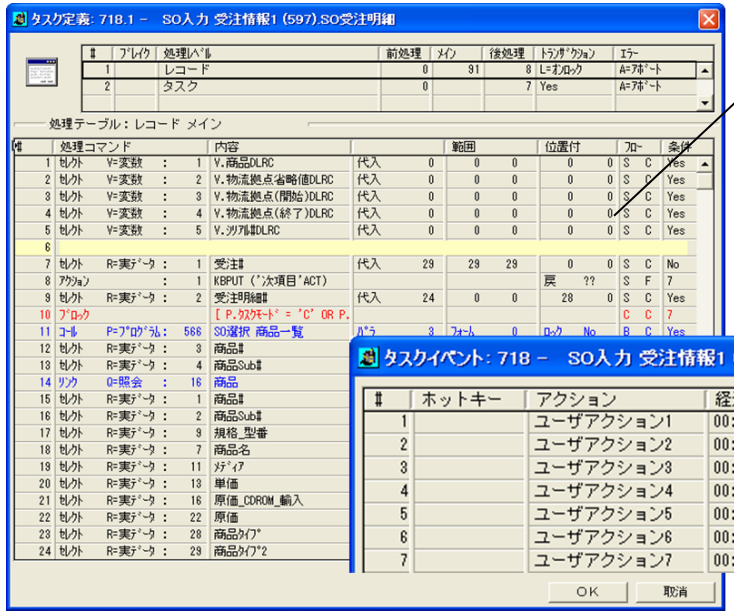


その後ハンドラを実行

まとめ

ＲＭの問題点

- キーボードのみで操作していた時代にはＲＭでよかったが、ＧＵＩ全盛のいまどきでは制御が複雑である。



レコードメイン

- どこでいつ何が起こるのか、見通しが悪い。

タスクイベントテーブル

- プッシュボタンの動作はイベントテーブルに記述
- どのボタンで何が呼び出されるかわからない。

まとめると・・・

- Magic xpaイベント方式でプログラムを書くと、どこで、いつ、何が起こるのか？が明瞭にわかるようになる。
- 他人のプログラムを読んでいる人にもわかりやすい。
- イベントの考え方は、ＲＭフローよりも単純。

Thank You!

magicsoftware.com