〔公開〕

TR－C－0050

JOKERシステム

Symbolics側ソフトウェア解説書

林　潔
KIYOSHI HAYASHI

1990. 8. 1.

ATR通信システム研究所

ＪＯＫＥＲシステム（TR-C-0027)シンボリックス側ソフトウェア解説書

はじめに

このレポートは、ソフトウェア構成の概要と関数の内容を与えるものである。

したがって、利用方法等は TR-C-0027を参照の事。

また、シンボリックス側ソフトウェアはシンボリックスのＳＹＳＴＥＭ機能を利用して

開発しているため、保守管理方法はシンボリックスのマニュアル参照の事。

なお、このレポートで解説するソフトウェアは　Genera 7.2　対応バージョンとなってます。

平成元年３月16日
ATR通信システム研究所
林　　潔

Symbolics側インタフェース・ソフトウェア概要

Symbolics側ソフトウェアは、回線から入力されてくるコードをあたかもSymbolicsのキーボードから入力された様にすり替えるプロセスが主な部分である。

すり替えは、キーボード・プロセスと同等のプライオリティで実行され、併押下キー（SHIFT,CONTROL,META,SUPER,HYPER）に関しては、Symbolicsのハードウェアキーコードに変換してmake/breakを実現している。

また、その他のキーに関しては、Symbolicsソフトウェア・キャラクタ・コード（別紙キャラクタコード表参照）を用いてSymbolicsのキーボード・プロセスにキー入力があった様に見せかけて、すり替えている。

日本語コードはＪＩＳ第二水準コードを用いて回線転送して、Symbolics漢字コードに変換している。この時の漢字シフトイン／アウト等には、本システム独自の制御コード（別紙コード表 *#xb0* ～ *#xff* ）を割当ている。

ソースリストの見方

ソースが "xx-patches.lisp"ファイルの場合、パッチ箇所は実際のソースリストの強調文字（bold）の部分です。

また、パッチを当てたSymbolicsのオリジナルファイルの格納場所は、SI:PATCH-SECTION-SOURCE-FILE　関数で定義してあります。

関数機能概要一覧

< Package:: SI / Source:: si-patches.lisp >

関数名	機能 ／ パッチ箇所
KBD-HARDWARE-CHAR-AVAILABLE	ｷｰｲﾝがあった場合にnil 以外を返す関数。 ／ ｿﾌﾄ/ﾊｰﾄﾞ･ｷｬﾗｸﾀのｷｰｲﾝがあった様に見せかける。
KBD-GET-HARDWARE-CHAR	ｷｰｲﾝされたﾊｰﾄﾞｳｪｱ･ｷｬﾗｸﾀを取り出す関数。 ／ ﾊｰﾄﾞｳｪｱ･ｷｬﾗｸﾀをすり替える。
KBD-CONVERT-TO-SOFTWARE-CHAR-original	ｷｰｲﾝされたﾊｰﾄﾞｳｪｱ･ｷｬﾗｸﾀをｿﾌﾄｳｪｱ･ｷｬﾗｸﾀに変換する関数。 ／ 関数名の変更。
kbd-convert-to-software-char	ｵﾘｼﾞﾅﾙな関数を包む形で、ｿﾌﾄｳｪｱ･ｷｬﾗｸﾀをすり替える。

< Package:: TV / Source:: tv-patches.lisp >

関数名	機能 ／ パッチ箇所
who-line-user-or-process	who-lineのuser or process 表示文字列編集。／ 入力モード表示のための文字列編集。
MOUSE-DEFAULT-HANDLER	マウス・ハンドラー関数。 ／ マウスクリック時のかな漢字変換確定処理のためのコード送出。

< Package:: ZWEI / Source:: zwei-patches.lisp >

関数名	機能
COM-単語登録	単語登録のためのコマンド。（リージョンおよびミニバッファ使用）
MAKE-JIS-STRING	*Symbolics*漢字コードをJISコード列（漢字ｼﾌﾄｲﾝ･ｱｳﾄを含む）に変換する。
MAKE-種類リスト	数字付の単語登録種別のリスト作成関数。
読み-P	読みとして登録できる文字列の判定用関数。
MEMBER-読み-P	２文字目以降の読みとして登録できる文字列の判定用関数。
MEMBER-読み-CHAR-P	読みとして登録できる文字の判定用関数。

< Package:: JOKER / Source:: reset-joker.lisp >

関数名	機能
com-reset-JOKER	システムの変数を初期設定して、システムを再起動（プロセスのkill/remake)する

< Package:: JOKER / Source:: main.process.lisp >

関数名	機能
serial-port-activate-switch	回線ストリーム及びメインプロセスのmake/close スイッチ関数（FUNCTIONキー割当て用）
make-serial-port-keyboard-process	回線ストリームのオープン　および メインプロセスの生成
close-serial-port-keyboard	回線ストリームのクローズ　および メインプロセスの抹消
入力モード設定	入力モード表示文字列編集および値設定用関数
hard-char-get-wait	ハードウェアキャラクタのすり替え待ち関数（メインプロセス内で用いウェイトをかける）
nth-of-list	リスト内の要素が何番目のものかを求める関数。（要素は文字列で比較される）
suitable-geometry	ポップアップメニューのための適切なジオメトリ計算用関数（項目数と１項目の長さで計算）
select-kanji-character	辞書参照機能用の漢字選択ポップアップメニュー関数
select-upload-file	アップロードファイル名（番号）選択用のポップアップメニュー関数
jis-to-char	JIS ⇒ *Symbolics*漢字コード変換関数のバッチ用関数。
side-effect-or	２引数のＯＲ関数で、２引数共評価してから判定するための関数。
変換文字列	who-line-file-state 行への変換対象文字列の表示のための関数
変換確定処理	変換対象文字列表示の変換確定処理
string-width	ストリングの文字幅をドット単位で求める関数。
変換対象開始	変換対象文字列表示のための初期設定
変換対象終了	変換対象文字列表示の終了処理
serial-port-keyboard-process-main-loop	回線経由のキーインすり替え用メインプロセス。（詳細は別紙１）
com-upload-ms-dos-file	MS-DOSファイルのアップロードコマンド
com-upload-oasys-file	OASYS 文書のアップロードコマンド

serial-port-keyboard-process-main-loopの構成

● メインループ関数は、以下の５つの部分から構成されている。

(1) 回線からのコード入力、*Symbolics*の自発的画面切り替え、または、マウスクリックを待ち、画面切り替え時には変換中の文字を確定させる処理をする部分。

(2) 本システムで定めた制御コードの処理を行う部分（selector関数使用）

(3) *Symbolics*特殊キー（例えば、SELECT等）の文字変換処理を行う部分。

(4) 日本語コードの変換（JIS ⇒ *Symbolics*漢字コード）を行う部分。

(5) ソフトウェアキャラクタの一文字すり替え処理を行い、すり替え完了を待つ部分。

なお、ハードウェアキャラクタの一文字すり替えは、(2)の中で行われる。
（関数機能一覧の hard-char-get-wait 参照）

● (5)において、回線から送られたコードはその使用目的（制御コードによりあらかじめ決められる）によって以下の５つのタイプの処理が行われる。

① エラーメッセージの編集。

② 辞書参照時の漢字リスト編集。

③ ファイルアップロードのためのファイル名リストの編集。

④ ファイルアップロード時のファイル内容の*Symbolics*ファイルへの書き出し処理

⑤ ソフトウェアキャラクタのすり替えと、その完了待ち。

Symbolics character code table

[0] = #\·	[10] = #\⊂	[20] = #\Space	[30] = #\0
[1] = #\↓	[11] = #\⊃	[21] = #\!	[31] = #\1
[2] = #\α	[12] = #\∩	[22] = #\"	[32] = #\2
[3] = #\β	[13] = #\∪	[23] = #\#	[33] = #\3
[4] = #\∧	[14] = #\∀	[24] = #\$	[34] = #\4
[5] = #\¬	[15] = #\∃	[25] = #\%	[35] = #\5
[6] = #\ε	[16] = #\●	[26] = #\&	[36] = #\6
[7] = #\π	[17] = #\≠	[27] = #\'	[37] = #\7
[8] = #\λ	[18] = #\←	[28] = #\(	[38] = #\8
[9] = #\γ	[19] = #\→	[29] = #\)	[39] = #\9
[A] = #\δ	[1A] = #\≠	[2A] = #*	[3A] = #\:
[B] = #\↑	[1B] = #\◇	[2B] = #\+	[3B] = #\;
[C] = #\±	[1C] = #\≤	[2C] = #\,	[3C] = #\<
[D] = #\●	[1D] = #\≥	[2D] = #\-	[3D] = #\=
[E] = #\∞	[1E] = #\≡	[2E] = #\.	[3E] = #\>
[F] = #\∂	[1F] = #\∨	[2F] = #\/	[3F] = #\?

[40] = #\@	[50] = #\P	[60] = #\`	[70] = #\p
[41] = #\A	[51] = #\Q	[61] = #\a	[71] = #\q
[42] = #\B	[52] = #\R	[62] = #\b	[72] = #\r
[43] = #\C	[53] = #\S	[63] = #\c	[73] = #\s
[44] = #\D	[54] = #\T	[64] = #\d	[74] = #\t
[45] = #\E	[55] = #\U	[65] = #\e	[75] = #\u
[46] = #\F	[56] = #\V	[66] = #\f	[76] = #\v
[47] = #\G	[57] = #\W	[67] = #\g	[77] = #\w
[48] = #\H	[58] = #\X	[68] = #\h	[78] = #\x
[49] = #\I	[59] = #\Y	[69] = #\i	[79] = #\y
[4A] = #\J	[5A] = #\Z	[6A] = #\j	[7A] = #\z
[4B] = #\K	[5B] = #\[	[6B] = #\k	[7B] = #\{
[4C] = #\L	[5C] = #\\	[6C] = #\l	[7C] = #\\|
[4D] = #\M	[5D] = #\]	[6D] = #\m	[7D] = #\}
[4E] = #\N	[5E] = #\^	[6E] = #\n	[7E] = #\~
[4F] = #\O	[5F] = #_	[6F] = #\o	[7F] = #\∫

[80] = #\Null	[90] = (Stop-Output)	[A0] = #\Complete	[B0] = {#\Shift-Key-Make}
[81] = #\Suspend	[91] = #\Abort	[A1] = #\Symbol-Help	[B1] = {#\Shift-Key-Break}
[82] = #\Clear-Input	[92] = #\Resume	[A2] =	[B2] = {#\Control-Key-Make}
[83] =	[93] = (Status)	[A3] =	[B3] = {#\Control-Key-Break}
[84] = #\Function	[94] = #\End	[A4] =	[B4] = {#\Meta-Key-Make}
[85] = (Macro)	[95] = #\Square	[A5] =	[B5] = {#\Meta-Key-Break}
[86] = #\Help	[96] = #\Circle	[A6] =	[B6] = {#\Super-Key-Make}
[87] = #\Rubout	[97] = #\Triangle	[A7] =	[B7] = {#\Super-Key-Break}
[88] = #\Back-Space	[98] = (Roman-IV)	[A8] =	[B8] = {#\Hyper-Key-Make}
[89] = #\Tab	[99] = (Hand-Up)	[A9] =	[B9] = {#\Hyper-Key-Break}
[8A] = #\Line	[9A] = #\Scroll	[AA] =	[BA] = {#\Reset-Keyboard}
[8B] = #\Refresh	[9B] = (Hand-Left)	[AB] =	[BB] =
[8C] = #\Page	[9C] = (Hand-Right)	[AC] =	[BC] =
[8D] = #\Newline	[9D] = #\Select	[AD] =	[BD] =
[8E] = (Quote)	[9E] = #\Network	[AE] =	[BE] = {シフトインコード}
[8F] = (Hold-Output)	[9F] = #\Escape	[AF] =	[BF] = {シフトアウトコード}

[C0] = {#\Jisyo-Start}	[D0] = {#\Open-MSDOS-Directory}	[E0] =	[F0] = {取消コード}
[C1] = {#\Jisyo-End}	[D1] = {#\Open-OASYS-Directory}	[E1] =	[F1] = {変換コード}
[C2] =	[D2] = {#\Directory-Start}	[E2] =	[F2] = {確定コード}
[C3] =	[D3] = {#\Directory-Sepalator}	[E3] =	[F3] = {バッファクリア}
[C4] = {#\Error-Msg}	[D4] = {#\Directory-End}	[E4] =	[F4] =
[C5] = {#\Err-Msg-End}	[D5] =	[E5] =	[F5] =
[C6] = {#\Beep}	[D6] =	[E6] =	[F6] =
[C7] = {#\通常}	[D7] =	[E7] =	[F7] =
[C8] = {#\全角}	[D8] = {#\Transfer-MSDOS-File}	[E8] =	[F8] =
[C9] = {#\半角}	[D9] = {#\Transfer-OASYS-File}	[E9] =	[F9] =
[CA] = {#\ひらがな}	[DA] = {#\File-Transfer-Start}	[EA] =	[FA] = {単語登録コード}
[CB] = {#\カタカナ}	[DB] = {#\File-Transfer-End}	[EB] =	[FB] =
[CC] = {#\英小文字}	[DC] =	[EC] =	[FC] =
[CD] = {#\英大文字}	[DD] =	[ED] =	[FD] =
[CE] = {#\ローマ字入力}	[DE] =	[EE] =	[FE] =
[CF] = {#\ローマ字解除} (親指シフト入力)	[DF] =	[EF] =	[FF] =

Symbol code key assign

[0] = #\. = Symbol-'
[1] = #\↓ = Symbol-h
[2] = #\α = Symbol-A
[3] = #\β = Symbol-B
[4] = #\∧ = Symbol-q
[5] = #\¬ = Symbol--
[6] = #\ε = Symbol-E
[7] = #\π = Symbol-P
[8] = #\λ = Symbol-L
[9] = #\γ = Symbol-G
[A] = #\δ = Symbol-D
[B] = #\↑ = Symbol-g
[C] = #\± = Symbol-:
[D] = #\⊕ = Symbol-+
[E] = #\∞ = Symbol-i
[F] = #\∂ = Symbol-p

[10] = #\⊂ = Symbol-t
[11] = #\⊃ = Symbol-y
[12] = #\∩ = Symbol-e
[13] = #\∪ = Symbol-r
[14] = #\∀ = Symbol-u
[15] = #\∃ = Symbol-o
[16] = #\⊗ = Symbol-*
[17] = #\≠ = Symbol-l
[18] = #\← = Symbol-j
[19] = #\→ = Symbol-k
[1A] = #\≠ = Symbol-=
[1B] = #\◇ = Symbol-#\Escape
[1C] = #\≤ = Symbol-,
[1D] = #\≥ = Symbol-.
[1E] = #\≡ = Symbol-‘
[1F] = #\∨ = Symbol-w

[7F] = #\∫ = Symbol-/

注: { }内 は Symbolics key code ではありません.

ソースリスト

CLM01:>local>system>joker>define-system.lisp.1

For: Kiyoshi Hayashi
Printed on: comprinter
Number of copies: 1
Data created at: 3/08/89 13:32:10
Queued at: 3/16/89 12:07:15

```
;;; -*- Mode: LISP; Package: USER; Syntax: Common-lisp; Base: 10 -*-
;;;
;;;    Jpapanese Front-End Keyboard Emulator System
;;;

(Defpackage JAPANESE-FRONT-END-KEYBOARD-EMULATOR
  (:nicknames JOKER)
  (:use SCL TV)
  (:export シフトアウトコード シフトインコード 単語登録コード 入力モード バッファクリア 変換対象
           *serial-port-hardware-char* *serial-port-software-char* *serial-port-io-stream*
           *pc-mode* *mouse-click* *kanji-select* serial-port-activate-switch))


;;; Define System

(defsystem JOKER
    (:pretty-name "Japanese Front-End Keyboard Emulator System"
     :default-pathname "JOKER:joker;"
     :default-package JOKER
     :after-patches-initializations (joker:serial-port-activate-switch 2))
  (:serial
    "Main-process"
    "SI-patches"
    "TV-patches"
    "ZWEI-patches"
    "Reset-JOKER"
    ))


;;;
;;;  end of file
;;;
```

CLM01:>local>system>joker>si-patches.lisp.1

For: Kiyoshi Hayashi
Printed on: comprinter
Number of copies: 1
Data created at: 3/08/89 09:45:12
Queued at: 3/16/89 12:08:36

```
;;; -*- Mode: LISP; Package: SYSTEM-INTERNALS; Base: 8; Syntax: Zetalisp; Patch-File: Yes -*-
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;
;;;    SYSTEM INTERNAL PATCHES
;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(SI:BEGIN-PATCH-SECTION)
(SYSTEM-INTERNALS:PATCH-SECTION-SOURCE-FILE "SYS:SYS;LFEPIO.LISP.166")
(SI:PATCH-SECTION-ATTRIBUTES
  "-*- Mode: LISP; Package: SYSTEM-INTERNALS; Base: 8; Syntax: Zetalisp; Patch-File: Yes -*-")


(DEFUN KBD-HARDWARE-CHAR-AVAILABLE (&OPTIONAL KBD-IN-PTR)
  "Returns T if a character is available in the fep buffer"
  (OR (IF *SLB-MAIN-CONSOLE*
          (CONSOLE-HARDWARE-CHAR-AVAILABLE *SLB-MAIN-CONSOLE* KBD-IN-PTR)
          (AND (NOT (EQ KBD-BUFFER-IN-PTR KBD-BUFFER-OUT-PTR))
               ;; Be conservative about this special pointer, use it as an additional
               ;; restriction, not the only one.
               (OR (NULL KBD-IN-PTR) (NOT (EQ KBD-IN-PTR KBD-BUFFER-OUT-PTR)))))
      (KBD-TIME-TO-REPEAT)
      joker:*serial-port-software-char*
      joker:*serial-port-hardware-char*))

(DEFUN KBD-GET-HARDWARE-CHAR (&aux ch)
  "Returns the next character in the fep buffer, and NIL if there is none"
  (LET ((KBD-TIME-TO-REPEAT-P (KBD-TIME-TO-REPEAT)))
    ;; Turn this flag off unconditionally here
    (SETQ *KBD-LAST-REAL-CHAR-REPEATED-P* NIL)
    (OR (IF *SLB-MAIN-CONSOLE*
            (or (CONSOLE-GET-HARDWARE-CHAR *SLB-MAIN-CONSOLE*)
                (without-interrupts
                  (setq ch joker:*serial-port-hardware-char*)    ;hardware char
                  (setq joker:*serial-port-hardware-char* nil)   ;shift key make/break
                  ch))
            (LET ((P KBD-BUFFER-OUT-PTR))
              (UNLESS (EQ KBD-BUFFER-IN-PTR P)
                (PROG1 (LOCATION-CONTENTS P)
                       (IF (EQ P KBD-BUFFER-END)
                           (SETQ P KBD-BUFFER-START)
                           (SETQ P (%MAKE-POINTER-OFFSET DTP-LOCATIVE P 1)))
                       (SETQ KBD-BUFFER-OUT-PTR P)))))
        ;; No hardware char available, check for repetition
        (WHEN KBD-TIME-TO-REPEAT-P
          ;; Turn the flag on if character wasn't actually typed.
          (SETQ *KBD-LAST-REAL-CHAR-REPEATED-P* T)
          *KBD-LAST-REAL-CHAR-HARD*))))


(DEFUN KBD-CONVERT-TO-SOFTWARE-CHAR-original (CH &OPTIONAL (KBD-TABLE KBD-NEW-TABLE) &AUX TYPE)
  "Convert hardware character to software character, or NIL to ignore"
  (SETQ TYPE (LDB (BYTE 3 14) CH))                          ;Type code
  (SELECT TYPE
    ((0 1 3 6 7) NIL)                           ;Unused, mouse, boot code, etc.
    ;; See comment about type-2 chars in the repeat-key code below.
    (2                                          ;An all-keys-up code, just update shifts mask
     (DOTIMES (I 20) (ASET 0 KBD-KEY-STATE-ARRAY-16B I))         ;Mark all keys up
     (SETQ CH (LDB (BYTE 14 00) CH))     ;Get bits for keys or key-pairs still down
     (SETQ KBD-LEFT-SHIFTS (LOGAND KBD-LEFT-SHIFTS CH)
           KBD-RIGHT-SHIFTS (LOGAND KBD-RIGHT-SHIFTS CH)
           KBD-LEFT-SHIFTS                ;This is for keys that are down that we thought
           (LOGIOR                        ; were up, e.g. caps lock.  Boole 10 is NOR.
             (LOGAND (BOOLE 10 KBD-LEFT-SHIFTS KBD-RIGHT-SHIFTS) CH)
             KBD-LEFT-SHIFTS)
           TV:KBD-BUTTONS 0)              ;obsolete feature to let keys act as mouse buttons
     NIL)
    ((4 5)                                      ;Transition
     (LET* ((KBD-SHIFTS (LOGIOR KBD-LEFT-SHIFTS KBD-RIGHT-SHIFTS))
            (NCH (MAP-KEY-TO-SOFTWARE-CHAR
                   (DPB (LDB (BYTE 1 5) KBD-SHIFTS)      ;Symbol
                        (BYTE 1 1) (LDB (BYTE 1 0) KBD-SHIFTS)) ;Shift
                   (LDB (BYTE 7 0) CH)
                   KBD-TABLE))
            (NCH0 (AREF KBD-NEW-TABLE 0 (LDB (BYTE 7 0) CH))))
       (COND ((AND (FIXP NCH) (BIT-TEST 1_15. NCH))     ;Not a real character
```

```
            (COND ((BIT-TEST 1_14. NCH)       ;Undefined key, beep if key-down
                   (OR (= TYPE 5)
                       (SEND TV:MAIN-SCREEN :BEEP)))
                  (T                  ;A shifting key, update KBD-SHIFTS
                   (LET ((BOOLE (IF (= TYPE 5) TV:ALU-ANDCA TV:ALU-IOR))  ;Bit off, on
                         (BIT (LSH 1 (LOGAND NCH 37))))
                     (IF (BIT-TEST 40 NCH)
                         (SETQ KBD-RIGHT-SHIFTS (BOOLE BOOLE BIT KBD-RIGHT-SHIFTS))
                         (SETQ KBD-LEFT-SHIFTS (BOOLE BOOLE BIT KBD-LEFT-SHIFTS))))
                   ;; Check for Repeat key
                   (WHEN (AND (= TYPE 4) (= (LOGAND NCH 37) 6))
                     ;; When the repeat key is pressed, we want to start repeating
                     ;; the last character only if its key is still held down.
                     ;;
                     ;; There seems to be a bug that causes kbd-get-hardware-char to
                     ;; return a type-2 code when you lift a shift key (including
                     ;; Repeat) in the sequence press-char press-shift lift-shift.
                     ;; The type-2 code causes the key-state array to be cleared out
                     ;; even though a key is down.  This means that the sequence
                     ;; press-char press-repeat lift-repeat press-repeat won't start
                     ;; repeating the second time.  -York 6/8/85
                     (UNLESS (AND *KBD-LAST-REAL-CHAR*
                                  (PLUSP (AREF KBD-KEY-STATE-ARRAY
                                               *KBD-LAST-REAL-CHAR*)))
                       (SETQ *KBD-LAST-REAL-CHAR* NIL)))))
            NIL)
           ((= TYPE 5)                        ;Just an up-code
            (ASET 0 KBD-KEY-STATE-ARRAY NCH0)
            #+++IGNORE (IF (EQ *KBD-LAST-REAL-CHAR* NCH0)
                           (SETQ *KBD-LAST-REAL-CHAR* NIL))
            NIL)
           (T ;A real key depression or repetition
            ;; Don't set the key-state-array unless the key transition actually
            ;; happened.  (I.e. using the repeat key to simulate pressing F isn't
            ;; the same thing as pressing F.)
            (UNLESS *KBD-LAST-REAL-CHAR-REPEATED-P*
              (ASET 1 KBD-KEY-STATE-ARRAY NCH0))
            (SETQ *KBD-LAST-REPEAT-TIME* (TIME))
            (SETQ *KBD-LAST-REAL-CHAR* NCH0
                  *KBD-LAST-REAL-CHAR-HARD* CH)
            (WHEN (FIXP NCH)
              (SETQ NCH (CODE-CHAR NCH (LDB (BYTE 4 1) KBD-SHIFTS)))
              (COND ((NOT (ZEROP (CHAR-BITS NCH)))
                     ;; Control character, swap cases, ignore caps lock
                     (SETF (CHAR-CODE NCH)
                           (CHAR-CODE (CHAR-FLIPCASE (CODE-CHAR (CHAR-CODE NCH))))))
                    ((BOTH-CASE-P NCH)
                     ;; Not a control character, caps lock applies
                     (WHEN (LDB-TEST (BYTE 1 11) KBD-SHIFTS)
                       (IF SHIFT-LOCK-XORS
                           (SETQ NCH (CHAR-FLIPCASE NCH))
                           (SETQ NCH (CHAR-UPCASE NCH)))))))
            NCH))))))


(defun kbd-convert-to-software-char  (CH &OPTIONAL (KBD-TABLE KBD-NEW-TABLE) &AUX char)
  (if ch (kbd-convert-to-software-char-original ch kbd-table)
      (when joker:*serial-port-software-char*
          (without-interrupts
            (setq char joker:*serial-port-software-char*)
            (setq joker:*serial-port-software-char* nil))
          char)))


;;;     end of file
```

CLM01:>local>system>joker>main-process.lisp.1

For: Kiyoshi Hayashi
Printed on: comprinter
Number of copies: 1
Data created at: 3/16/89 13:19:26
Queued at: 3/16/89 13:25:09

```
;;; -*- Package: JOKER; Base: 10; Mode: LISP; Syntax: Common-lisp -*-
;;;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;;;
;;;
;;;       Symbolics 日本語入力フロントエンド・キーボード・エミュレーター [  JOKER  ]
;;;
;;;                                   with FMR & PC-98
;;;
;;;              designed by  K. Hayashi  1989 March 8
;;;
;;;            [ Network address :  hayashi@atr-sw.atr.junet ]
;;;
;;;          Copyright ATR Communication Systems Research Laboratories
;;;
;;;
;;;                         for Genera 7.2
;;;
;;;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

;;;  システム変数

(defvar *serial-port-io-stream*)
(defvar *serial-port-keyboard-process*)
(defvar *serial-port-software-char*     nil)
(defvar *serial-port-hardware-char*     nil)     ;shift,control,meta,super,hyper make/break
(defvar *upload-file-number* nil)
(defvar *upload-file-name* nil)
(defvar *destination-file* nil)
(defvar *error* nil)
(defvar *PC-mode* nil)
(defvar *mouse-click* nil)
(defvar *kanji-select* nil)

(defvar 入力モード          "")
(defvar 文字種別            "al")
(defvar 全角モード          "")
(defvar ローマ字入力        "")
(defvar 変換対象            nil)
(defvar *henkan-pos*        0)
(defvar *who-line-file-sheet-blinker*)
(defvar *select-escape* nil)

(defconstant シフトインコード       #xbe)
(defconstant シフトアウトコード     #xbf)

(defconstant 取消コード             #xf0)
(defconstant 変換コード             #xf1)
(defconstant 確定コード             #xf2)
(defconstant バッファクリア         #xf3)

(defconstant 単語登録コード         #xfa)


;;;   FUNCTION キー 定義

(defun serial-port-activate-switch (n &aux de)
  (cond
    ((null n) (if *serial-port-io-stream*
                  (close-serial-port-keyboard)
                  (make-serial-port-keyboard-process)))
    ((zerop n) ())
    ((= 2 n)
     (if (and (boundp '*serial-port-io-stream*) *serial-port-io-stream*)
         (com-reset-JOKER)
         (make-serial-port-keyboard-process)))
    (t (when *serial-port-io-stream*
         (close-serial-port-keyboard)
         (make-serial-port-keyboard-process))))
  (if *serial-port-io-stream* (setq de "")(setq de "de"))
  (tv:notify nil "JOKER System has been ~aactivated." de))

;;;      FUNCTIONキーの登録
(tv:add-function-key #\p #'serial-port-activate-switch
                     "JOKER System activate switch [Arg 0:Show-Status 1:Reset-System 2:Force-Acti
vate]")
```

```
;;;  ユーティリティ関数

(defun make-serial-port-keyboard-process ()
  (setq *serial-port-io-stream*
        (si:make-serial-stream :unit 0
                               :force-output t
                               :baud 9600
                               :number-of-data-bits 8
                               :parity nil
                               :number-of-stop-bits 1)
        *serial-port-keyboard-process*
        (process-run-function '(:name "Serial port Keyboard"
                                :priority 30
                                :quantum 6)
          #'serial-port-keyboard-process-main-loop))
  (send *serial-port-io-stream* ':setf-data-terminal-ready t)
  (unless (boundp '*who-line-file-sheet-blinker*)
    (setq *who-line-file-sheet-blinker*
          (tv:make-blinker tv:who-line-file-state-sheet 'tv:rectangular-blinker
                           :visibility :off :deselected-visibility :off))))

(defun close-serial-port-keyboard ()
  (si:process-kill *serial-port-keyboard-process*)
  (close *serial-port-io-stream*)
  (setq  *serial-port-io-stream* nil
         入力モード ""))


(defun 入力モード設定 (zenkaku roma moji)
  (setf 入力モード        (format nil "~a~a~a  " zenkaku roma moji)
        全角モード         zenkaku
        ローマ字入力       roma
        文字種別           moji))

(defun hard-char-get-wait (ch)
  (setq *serial-port-hardware-char* ch)
  (process-wait "wait for get-hardware-char"
    #'(lambda () (null *serial-port-hardware-char*))))

(defun nth-of-list (item item-list &optional (test-case #'string-equal))
  (let ((member-length (length (member item item-list :test test-case))))
    (if (zerop member-length) nil (- (length item-list) member-length))))

(defun suitable-geometry (item-list)
  (let* ((item (car item-list))
         (width
          (if (characterp item)
              (send tv:selected-window ':character-width item)
              (apply 'max (mapcar #'string-width item-list (mapcar #'length item-list)))))
         (number (length item-list)))
    (list (min (floor 1000 (+ width 10)) (1+ (floor (1- number) 15))))))

(defun select-kanji-character (item-list)
  (if (car item-list)
      (let* ((menu (tv:make-window 'tv:momentary-menu
                                   ':label '(:font fonts:medfnt
                                             :string " Select KANJI Character")
                                   ':geometry (suitable-geometry item-list)
                                   ':item-list item-list))
             (char (send menu ':choose)))
        (send menu ':kill)
        char)

      (tv:notify nil "Japanese Keyboard Emulator:: 参照できる漢字辞書がない (Symbolics)")))
```

```
(defun select-upload-file (item-list)
  (if (car item-list)
      (let* ((menu (tv:make-window 'tv:momentary-menu
                                   ':label '(:font fonts:medfnt
                                             :string " Select Loading File")
                                   ':geometry (suitable-geometry item-list)
                                   ':item-list item-list))
             (file-name (send menu ':choose))
             (num (nth-of-list file-name item-list))
             (number (if num (1+ num) 0)))
        (send menu ':kill)
        (values number file-name))
      (tv:notify nil "Japanese Keyboard Emulator:: 選択対象のファイルがない (Symbolics)")
      (setq *error* t)))

(defun jis-to-char (code &optional (mode (if (string= 全角モード "全角 ") nil t)))
  (selector code =
    (#x2141 #\～)
    (#x2126 #\・)
    (#x212e (if mode #\' #\¯))
    (#x2131 (if mode #\~ #\￣))
    (#x222a #\→)
    (#x222b #\←)
    (#x222c #\↑)
    (#x222d #\↓)
    (otherwise (japanese:jis-to-char code mode))))

(defun side-effect-or (x y) (or x y))
```

```
;;;  変換バッファ制御用 関数

(defun 変換文字列 (char mi-henkan)
  (let ((left-hand (subseq 変換対象 0 *henkan-pos*))
        (right-hand (subseq 変換対象 *henkan-pos*)))
    (if (not *select-escape*)
      (if (graphic-char-p char)
          (setq 変換対象 (string-append left-hand char right-hand)
                *henkan-pos* (1+ *henkan-pos*))
          (unless (and (string= 変換対象 "")(zerop *henkan-pos*))
            (case char
              (#\c-f (if (< *henkan-pos* (string-length 変換対象))
                         (setq *henkan-pos* (1+ *henkan-pos*))
                         (setq 変換対象 "" *henkan-pos* 0)))
              (#\c-b (if (< 0 *henkan-pos*)
                         (setq *henkan-pos* (1- *henkan-pos*))
                         (setq 変換対象 "" *henkan-pos* 0)))
              (#\c-d (if (< *henkan-pos* (string-length 変換対象))
                         (setq 変換対象 (string-append left-hand (subseq right-hand 1)))))
              (#\Rubout (if (< 0 *henkan-pos*)
                            (setq *henkan-pos*  (1- *henkan-pos*)
                                  変換対象 (string-append (subseq left-hand 0 *henkan-pos*)
                                                          right-hand))
                            (setq *henkan-pos* 0)))
              (#\c-k (setq 変換対象 left-hand))
              ((#\h-c-escape #\select #\function #\network))
              (otherwise (setq 変換対象 "" *henkan-pos* 0)
                         (send *serial-port-io-stream* ':tyo バッファクリア) )))))
    (if 変換対象 (tv:who-line-string tv:who-line-file-state-sheet "..." 変換対象))
    (when (boundp '*who-line-file-sheet-blinker*)
      (send *who-line-file-sheet-blinker*
            ':set-size
            (if (string= 変換対象 "") 6
                (send tv:who-line-file-state-sheet ':character-width
                      (aref (string-append 変換対象 " ") *henkan-pos*)))
            12)
      (let ((str-wid (string-width 変換対象 *henkan-pos* tv:who-line-file-state-sheet)))
        (send *who-line-file-sheet-blinker* ':set-cursorpos str-wid 0)
        (when mi-henkan
          (send tv:who-line-file-state-sheet ':draw-rectangle str-wid 1 0 11 tv:alu-xor))))
    )
  (if (and *select-escape* (not (member char '(#\select #\function #\network)))
           (or (char< char #\0)(char< #\9 char))) (setq *select-escape* nil)))

(defun 変換確定処理 ()
  (when 変換対象
      (setq 変換対象 (subseq 変換対象 *henkan-pos*) *henkan-pos* 0)
      (if 変換対象 (tv:who-line-string tv:who-line-file-state-sheet "..." 変換対象))
      (when (string≠ 変換対象 "")
        (send *who-line-file-sheet-blinker* ':set-size
              (send tv:who-line-file-state-sheet ':character-width (aref 変換対象 0)) 12))
      (send *who-line-file-sheet-blinker* ':set-cursorpos 0 0)))

(defun string-width (string pos &optional (window tv:selected-window))
  (let* ((str (subseq string 0 pos)) (width 0))
    (dotimes (i pos)
      (let* ((char (aref str i))
             (char-width (send window ':character-width char)))
        (setq width (+ width char-width))))
    width))

(defun 変換対象開始 ()
  (setq 変換対象 "" *henkan-pos* 0)
  (send tv:who-line-file-state-sheet ':clear-window)
  (send *who-line-file-sheet-blinker* ':set-visibility :blink))

(defun 変換対象終了 ()
  (setq 変換対象 nil)
  (when (boundp '*who-line-file-sheet-blinker*)
    (send *who-line-file-sheet-blinker* ':set-cursorpos 0 0)
    (send *who-line-file-sheet-blinker* ':set-visibility :off))
  (send tv:who-line-file-state-sheet ':clear-window))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;
;;;     PROCESS MAIN LOOP
;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defun serial-port-keyboard-process-main-loop
       (&aux ch-code upper (h-mode t)
        (mi-henkan nil)(window tv:selected-window)(count 0)(shift nil)(ch-bits 0)
        (error-msg nil)(dictionary nil)(file-transfer nil)(directory nil)(item ""))
  (入力モード設定 全角モード ローマ字入力 文字種別) (setq *error* nil)
  (loop
;;;     回線入力, mouseクリック, 画面切り換え待ち
;;;
    (process-wait "Waiting for serial char"
      #'(lambda () (side-effect-or
                     (and tv:selected-window (neq window tv:selected-window))
                     (setq ch-code (or *mouse-click*
                                       (send *serial-port-io-stream* ':tyi-no-hang))))))
;;;     かな入力中の画面切替え対策(変換確定処理)
;;;
    (when (and tv:selected-window (neq window tv:selected-window))
      (setq window tv:selected-window)
      (when (and 変換対象 (plusp (string-length 変換対象)))
        (send *serial-port-io-stream* ':tyo  バッファクリア)
        (setq mi-henkan t 変換対象 "" *henkan-pos* 0)
        (send tv:who-line-file-state-sheet ':clear-window)))
    (if ch-code
        (let ((char nil)(lower ch-code))
;;;
;;;     制御コード処理
;;;
          (selector ch-code =
            ((#x9d #x84 #x9e) (setq *select-escape* t)) ; select,function,network
            (#xb0 (hard-char-get-wait 49176))   ; shift make
            (#xb1 (hard-char-get-wait 53272))   ; shift break
            (#xb2                               ; control make
             (setq h-mode t)
             (setq ch-bits (logior ch-bits #b0001))
             (hard-char-get-wait 49163))
            (#xb3                               ; control break
             (unless (equal 全角モード "")(setq h-mode nil))
             (setq ch-bits (logand ch-bits #b1110))
             (hard-char-get-wait 53259))
            (#xb4                               ; meta make
             (setq h-mode t)
             (setq ch-bits (logior ch-bits #b0010))
             (hard-char-get-wait 49155))
            (#xb5                               ; meta break
             (unless (equal 全角モード "")(setq h-mode nil))
             (setq ch-bits (logand ch-bits #b1101))
             (hard-char-get-wait 53251))
            (#xb6                               ; super make
             (setq h-mode t)
             (setq ch-bits (logior ch-bits #b0100))
             (hard-char-get-wait 49162))
            (#xb7                               ; super break
             (unless (equal 全角モード "")(setq h-mode nil))
             (setq ch-bits (logand ch-bits #b1011))
             (hard-char-get-wait 53258))
            (#xb8                               ; hyper make
             (setq h-mode t)
             (setq ch-bits (logior ch-bits #b1000))
             (hard-char-get-wait 49154))
            (#xb9                               ; hyper break
             (unless (equal 全角モード "")(setq h-mode nil))
             (setq ch-bits (logand ch-bits #b0111))
             (hard-char-get-wait 53250))
```

```
        (#xba                                   ; reset keyboard
         (send *who-line-file-sheet-blinker* ':set-visibility :off)
         (入力モード設定 "" ローマ字入力 "al")
         (setq h-mode t      mi-henkan nil    window tv:selected-window       count 0
               shift nil     ch-bits 0        error-msg nil    dictionary nil
               file-transfer nil              directory nil    item "" *error* nil
               *upload-file-number* nil       *upload-file-name* nil   変換対象  nil
               *henkan-pos* 0        *select-escape* nil      *mouse-click* nil)
         (tv:notify nil "JOKER System has been reseted"))
;;;
        (#xbe (setq shift t count 0))           ; shift in
        (#xbf (setq shift nil count 0))         ; shift out
        (#xc0 (setq dictionary '(始め)))        ; 漢字辞書参照
        (#xc1 (setq *kanji-select* t)           ; 参照終了
         (cond ((setq char (select-kanji-character (cddr dictionary)))
                (dotimes (i (char-code (cadr dictionary)))  ; Rubout回数
                  (send tv:selected-window ':force-kbd-input #\Rubout)
                  (変換文字列 #\Rubout nil))
                (send *serial-port-io-stream* ':tyo 変換コード)(setq mi-henkan nil))
               (t (send *serial-port-io-stream* ':tyo 取消コード)(setq mi-henkan t)))
         (setq dictionary nil
               *kanji-select* nil))
;;;
        (#xc4 (setq error-msg ""))              ; error message start
        (#xc5                                   ; error message end
         (tv:notify nil "Japanese Keyboard Emulator:: ~a" error-msg)
         (setq error-msg nil *error* t))
        (#xc6 (beep))        ; BEEP音
        (#xc7 (入力モード設定 "通常 "    ローマ字入力 文字種別) (setq h-mode t))       ;通常モード
        (#xc8 (入力モード設定 "全角 "    ローマ字入力 文字種別) (setq h-mode nil))     ;全角モード
        (#xc9 (入力モード設定 ""         ローマ字入力 文字種別) (setq h-mode t))       ;半角モード
        (#xca (入力モード設定 全角モード ローマ字入力 "かな") ;ひらがな・モード
         (変換対象開始) (setq mi-henkan t))
        (#xcb (入力モード設定 全角モード ローマ字入力 "カナ") ;カタカナ・モード
         (if (null *pc-mode*) (変換対象終了)
             (変換対象開始) (setq mi-henkan t)))
        (#xcc (入力モード設定 全角モード ローマ字入力 "al") (変換対象終了))      ;英小文字モード
        (#xcd (入力モード設定 全角モード ローマ字入力 "AL") (変換対象終了))      ;英大文字モード
        (#xce        ;ローマ字入力モード
         (入力モード設定 全角モード "R-"    文字種別) (変換確定処理) (setq mi-henkan t))
        (#xcf        ;親指シフト入力モード
         (入力モード設定 全角モード ""      文字種別) (変換確定処理) (setq mi-henkan t))
;;;
        (#xd2 (setq directory '(始め)))         ; Directory open
        (#xd3                                   ; Item separator
         (setq directory (append directory (cons item nil))  item ""))
        (#xd4                                   ; Directory close
         (setq directory (cdr (append directory (cons item nil)))  item "")
         (multiple-value-setq (*upload-file-number* *upload-file-name*)
           (select-upload-file directory))
         (setq directory nil))
;;;
        (#xda (setq file-transfer t))           ;ファイル転送開始
        (#xdb                                   ;ファイル転送終了
         (setq file-transfer                    nil
               *upload-file-number*             nil
               *upload-file-name*               nil))
;;;
        (取消コード
          (setq mi-henkan t)(変換文字列 #\h-c-escape mi-henkan))
        (変換コード (setq mi-henkan nil)
                (変換文字列 #\h-c-escape mi-henkan))
        (確定コード (変換確定処理) (setq mi-henkan t))
        (バッファクリア
          (setq 変換対象 "" *henkan-pos* 0 mi-henkan t *mouse-click* nil)
          (send tv:who-line-file-state-sheet ':clear-window)))
```

```
;;;
;;;     Symbolics特殊キー変換
;;;
          (if (and (<= #x80 ch-code)(< ch-code #xb0))
              (setq char (code-char ch-code)
                    shift nil))
;;;
;;;     日本語コード変換
;;;
          (unless (or (and (<= #x80 ch-code)(<= ch-code #xff))(characterp char))
            (if shift
                (cond ((zerop count)(setq count 1       ; shift in
                                          upper lower))
                      (t (setq count 0
                               char (jis-to-char (+ (* upper #x100) lower) h-mode))
                         (unless char (setq char #\=))))
                (when (< ch-code #x80)          ; shift out
                  (setq char (code-char ch-code)))
                ))
;;;
;;;     一文字編集(すり替え)
;;;
          (when (characterp char)
            (if (and (< 0 ch-bits) (> #xb0 (char-code char)))   ;併押下キーのchar-code組立て
                (setq char (make-char (char-flipcase char) ch-bits)))
            (cond
              (error-msg
               (setq error-msg (format nil "~a~c" error-msg char)))
              (dictionary
               (if (or (char< char #\=)
                       (char< #\ヶ char)(char= char #\ゐ)(char= char #\ゑ)
                       (char= char #\ヰ)(char= char #\ヱ)(char= char #\ヴ))
                   (setq dictionary (append dictionary (cons char nil)))))
              (directory
               (setq item (format nil "~a~c" item char)))
              (file-transfer
               (format *destination-file* "~c" char))
              (t
               (if 変換対象 (変換文字列 char mi-henkan))
               (setq *serial-port-software-char* char)
               (process-wait "Wait for keyboard get"
                 #'(lambda () (null *serial-port-software-char*))))))
          ))))
```

```
;;;
;;;     File Upload Command
;;;
(define-cp-command com-upload-ms_dos-file
    ((drive '((cl:member :A :B :C :D :E :F)) :prompt "drive" :default :A))
   (setq *error* nil *upload-file-number* nil *upload-file-name* nil)
   (send *serial-port-io-stream* ':tyo #xd0)    ; OPEN-MSDOS-DIRECTORY
   (send *serial-port-io-stream* ':tyo (char-code (aref (format nil "~a" drive) 0)))  ; DRIVE
   (process-wait "Select Upload File" #'(lambda () (or *upload-file-number* *error*)))
   (unless (or *error* (zerop *upload-file-number*))
     (format t " From  ~a:~a~%" (format nil "~a" drive) *upload-file-name*)
     (let*
       ((to-file
          (accept 'fs:pathname :prompt " Into"
                  :default (pathname (format nil "~ams-dos.text" (fs:user-homedir))))))
       (with-open-file (stream to-file ':direction ':output)
         (setq *destination-file* stream)
         (send *serial-port-io-stream* ':tyo #xd8)      ; TRANSFER-MSDOS-FILE
         (do ((i 0 (1+ i))) ((= i (string-length *upload-file-name*)))  ; FILE-NAME
           (send *serial-port-io-stream* ':tyo (char-code (aref *upload-file-name* i))))
         (send *serial-port-io-stream* ':tyo #x94)      ; FILE-NAME-#\END
         (process-wait "Uploading File"
           #'(lambda () (or (null *upload-file-number*) *error*))))))
   (setq *destination-file*     nil     *error*                 nil
         *upload-file-number*   nil     *upload-file-name*      nil))

(define-cp-command com-upload-oasys-file
    ((drive '((cl:member :A :B)) :prompt "drive" :default :A))
   (if *PC-mode* (tv:notify nil "Not support this service on PC98 (Symbolics)")
     (setq *error* nil *upload-file-number* nil *upload-file-name* nil)
     (send *serial-port-io-stream* ':tyo #xd1)  ; OPEN-OASYS-DIRECTORY
     (send *serial-port-io-stream* ':tyo (char-code (aref (format nil "~a" drive) 0))) ; DRIVE
     (process-wait "Select Upload File" #'(lambda () (or *upload-file-number* *error*)))
     (unless (or *error* (zerop *upload-file-number*))
       (format t " From  ~a:~a~%" (format nil "~a" drive) *upload-file-name*)
       (let*
         ((to-file
            (accept 'fs:pathname :prompt " Into"
                    :default (pathname (format nil "~aoasys.text" (fs:user-homedir)))))
          (*mode* (accept '(integer 0 3) :prompt " Mode(0-3)" :default 3)))
         (with-open-file (stream to-file ':direction ':output)
           (setq *destination-file* stream)
           (send *serial-port-io-stream* ':tyo #xd9)    ; TRANSFER-OASYS-FILE
         (send *serial-port-io-stream* ':tyo *mode*)    ; SET-UPLOAD-MODE
;;;
;;;             mode| 改頁   | 改行
;;;             ---+-----+-----
;;;              0 | しない | しない
;;;              1 | しない | する
;;;              2 | する   | しない
;;;              3 | する   | する
;;;
;;;
           (send *serial-port-io-stream* ':tyo *upload-file-number*)    ; FILE-NUMBER
           (process-wait "Uploading File"
             #'(lambda () (or (null *upload-file-number*) *error*)))))))
   (setq *destination-file*     nil     *error*                 nil
         *upload-file-number*   nil     *upload-file-name*      nil))

;;;     グローバルコマンドの登録
(cp:install-commands cp:*global-command-table*
                     '(com-upload-ms_dos-file com-upload-oasys-file))


;;;   End of File
```

CLM01:>local>system>joker>tv-patches.lisp.1

For: Kiyoshi Hayashi
Printed on: comprinter
Number of copies: 1
Data created at: 3/08/89 09:45:59
Queued at: 3/16/89 12:09:04

```
;;; -*- Mode: LISP; Package: TV; Base: 8; Syntax: Zetalisp; Patch-File: Yes -*-
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;
;;;     TV PATCHES
;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(SI:BEGIN-PATCH-SECTION)
(SYSTEM-INTERNALS:PATCH-SECTION-SOURCE-FILE "SYS:WINDOW;WHOLIN.LISP.281")
(SI:PATCH-SECTION-ATTRIBUTES
  "-*- Mode: LISP; Package: TV; Base: 8; Syntax: Zetalisp; Patch-File: Yes -*-")

;;;     WHO-LINE      :USER field

(defun who-line-user-or-process (who-sheet state ignore)
  (who-line-string who-sheet state
                   (block nil
                     (let ((force-process
                             (who-line-screen-force-process (sheet-screen who-sheet))))
                       (when force-process
                         (return
                           (FORMAT NIL "~A~A" JOKER:入力モード (process-name force-process)))))
                     (when (and current-process *show-current-process-in-wholine*)
                       (return
                         (FORMAT NIL "~A~A" JOKER:入力モード (process-name current-process))))
                     (return (FORMAT NIL "~A~A" JOKER:入力モード user-id)))))


;;;   バッファクリア コード送出  when mouse click

(SI:BEGIN-PATCH-SECTION)
(SYSTEM-INTERNALS:PATCH-SECTION-SOURCE-FILE "SYS:WINDOW;MOUSE.LISP.393")
(SI:PATCH-SECTION-ATTRIBUTES
  "-*- Mode: LISP; Package: TV; Base: 8; Syntax: Zetalisp; Patch-File: Yes -*-")

(DEFUN MOUSE-DEFAULT-HANDLER (WINDOW &OPTIONAL SCROLL-BAR
                                               (MOUSE (IF (INSTANCEP WINDOW)
                                                          (SHEET-MOUSE WINDOW)
                                                          MAIN-MOUSE))
                                      &AUX MOVE-HANDLER MOVE-METHOD BUTTONS-METHOD
                                           (WINDOW-X-OFFSET 0) (WINDOW-Y-OFFSET 0)
                                           WINDOW-X WINDOW-Y
                                           WINDOW-WIDTH WINDOW-HEIGHT)
  ;; Backwards compatibility for the scroll-bar argument. It used to be a
  ;; single symbol, T meant what (:LEFT) means, :IN meant what :IN-LEFT means.
  (COND ((EQ SCROLL-BAR T) (SETQ SCROLL-BAR '(:LEFT)))
        ((EQ SCROLL-BAR ':IN) (SETQ SCROLL-BAR ':IN-LEFT)))
  (MULTIPLE-VALUE (MOVE-METHOD BUTTONS-METHOD)
    (IF (AND SCROLL-BAR (SYMBOLP SCROLL-BAR))
        (VALUES ':MOUSE-MOVES-SCROLL ':MOUSE-BUTTONS-SCROLL)
        (VALUES ':MOUSE-MOVES ':MOUSE-BUTTONS)))
  (SETQ MOVE-HANDLER (IF (AND (NOT (SYMBOLP WINDOW)) (GET-HANDLER-FOR WINDOW MOVE-METHOD))
                         WINDOW
                         #'(LAMBDA (&REST IGNORE)
                             (DECLARE (DOWNWARD-FUNCTION))
                             (MOUSE-SET-BLINKER-CURSORPOS-INTERNAL MOUSE))))
  (UNLESS (SYMBOLP WINDOW)
    (MULTIPLE-VALUE (WINDOW-X-OFFSET WINDOW-Y-OFFSET)
      (SHEET-CALCULATE-OFFSETS WINDOW (MOUSE-SHEET MOUSE)))
    (SETQ WINDOW-WIDTH (SHEET-WIDTH WINDOW)
          WINDOW-HEIGHT (SHEET-HEIGHT WINDOW)))
  (DO ((DX) (DY) (BU) (BD) (HAND) (X) (Y)
       (OLD-OWNER (WINDOW-OWNING-MOUSE-INTERNAL MOUSE) (WINDOW-OWNING-MOUSE-INTERNAL MOUSE))
       (X-OFFSET 0) (Y-OFFSET 0)
       (WAIT-FLAG NIL T))
      ((MOUSE-RECONSIDER MOUSE))
    (MULTIPLE-VALUE (DX DY BD BU X Y) (MOUSE-INPUT WAIT-FLAG MOUSE))
    ;; If asked to reconsider, do so immediately.
    ;; Don't bother updating blinker since it is likely to change soon, and
    ;; in any case we are going to be called back shortly.
    (IF (MOUSE-RECONSIDER MOUSE) (RETURN NIL))
    ;; Update console-idle time when buttons pushed
    (unless (ZEROP BD) (SETF (CONSOLE-LAST-ACTIVITY-TIME (MOUSE-CONSOLE MOUSE)) (TIME))
        ;;   バッファクリア コード送出
        (when (and joker:*serial-port-io-stream* joker:変換対象 (null joker:*kanji-select*))
            (send joker:*serial-port-io-stream* ':tyo joker:バッファクリア)
```

```
        (setq joker:*mouse-click* joker:バッファクリア)
        )   )
(SETQ WINDOW-X (- X WINDOW-X-OFFSET)
      WINDOW-Y (- Y WINDOW-Y-OFFSET))
;; X-OFFSET is how far out the sides of the window the mouse has moved, or 0 if the
;; mouse is inside the window. If x-offset is negative the mouse has moved outside
;; the left of the window, if it is positive the mouse has moved outside the right
;; of the window. Y-OFFSET is similar, with negative => top, positive => bottom.
;;
;; If the side of the window the mouse is moving out of is at an edge of the screen,
;; MOUSE-X/MOUSE-Y will not move out the edge of the window, but DX/DY will. This
;; extra movement is accumulated into X-OFFSET/Y-OFFSET.
(UNLESS (SYMBOLP WINDOW)
  (COND ((≤ WINDOW-X 0)      ;≤ not < if you want scroll bars at screen edge to work
         (SETQ X-OFFSET (IF (MINUSP X-OFFSET)
                            (MIN (+ X-OFFSET DX) -1)
                            -1)))          ;First time, don't use all of DX
        ((≥ WINDOW-X (1- WINDOW-WIDTH))
         (SETQ X-OFFSET (IF (PLUSP X-OFFSET)
                            (MAX (+ X-OFFSET DX) 1)
                            1)))           ;First time, don't use all of DX
        ((AND (> WINDOW-X SCROLL-BAR-RELUCTANCE)
              (≤ WINDOW-X (- WINDOW-WIDTH SCROLL-BAR-RELUCTANCE)))
         (SETQ X-OFFSET 0)))
  (COND ((≤ WINDOW-Y 0)      ;≤ not < if you want scroll bars at screen edge to work
         (SETQ Y-OFFSET (IF (MINUSP Y-OFFSET)
                            (MIN (+ Y-OFFSET DY) -1)
                            -1)))          ;First time, don't use all of DY
        ((≥ WINDOW-Y (1- WINDOW-HEIGHT))
         (SETQ Y-OFFSET (IF (PLUSP Y-OFFSET)
                            (MAX (+ Y-OFFSET DY) 1)
                            1)))           ;First time, don't use all of DY
        ((AND (> WINDOW-Y SCROLL-BAR-RELUCTANCE)
              (≤ WINDOW-Y (- WINDOW-HEIGHT SCROLL-BAR-RELUCTANCE)))
         (SETQ Y-OFFSET 0)))
  (UNLESS (or (mouse-warp-internal mouse)
              (AND (ZEROP X-OFFSET) (ZEROP Y-OFFSET)))
    (SEND-IF-HANDLES WINDOW :MOUSE-EXIT X-OFFSET Y-OFFSET WINDOW-X WINDOW-Y)))
;; Consider entering the scroll bar. [Perhaps this should be changed so that it is
;; in the move-handler rather than here. The problem would be communicating the
;; values of X-OFFSET/Y-OFFSET to the move handler.]
;;
;; If there is a scroll bar and we are entering it, activate it. However, the mouse
;; must move at least a certain distance past the edge of the window in order to
;; qualify for scrolling (this is set by the SCROLL-BAR-RELUCTANCE variable in the
;; window).  Before entering scroll bar, send a :MOUSE-MOVES message in order to let
;; the window know what's happening.
(UNLESS (OR OLD-OWNER
            (WINDOW-OWNING-MOUSE-INTERNAL MOUSE)
            (NULL SCROLL-BAR)
            (MOUSE-WARP-INTERNAL MOUSE)
            )                              ;These disable scroll bar
  (COND ((LISTP SCROLL-BAR)
         (LET ((*IN-SCROLL-MARGIN*
                (CAR (OR (AND (MINUSP X-OFFSET) (MEMQ ':LEFT   SCROLL-BAR))
                         (AND (PLUSP X-OFFSET)  (MEMQ ':RIGHT  SCROLL-BAR))
                         (AND (MINUSP Y-OFFSET) (MEMQ ':TOP    SCROLL-BAR))
                         (AND (PLUSP Y-OFFSET)  (MEMQ ':BOTTOM SCROLL-BAR))))))
           (WHEN *IN-SCROLL-MARGIN*
             (COND ((AND SCROLL-BAR-MAX-SPEED (> (MOUSE-SPEED MOUSE)
                                                 SCROLL-BAR-MAX-SPEED))
                    (RETURN NIL))          ;Too fast, pass right through
                   ((OR (> (ABS X-OFFSET) SCROLL-BAR-RELUCTANCE)
                        (> (ABS Y-OFFSET) SCROLL-BAR-RELUCTANCE))
                    (FUNCALL MOVE-HANDLER MOVE-METHOD WINDOW-X WINDOW-Y)
                    (RETURN (FUNCALL WINDOW ':HANDLE-MOUSE-SCROLL)))
                   (T
                    (COND ((MINUSP X-OFFSET) (SETQ WINDOW-X 0))
                          ((PLUSP X-OFFSET) (SETQ WINDOW-X (SHEET-INSIDE-WIDTH WINDOW)))
                          ((PLUSP Y-OFFSET) (SETQ WINDOW-Y (SHEET-INSIDE-HEIGHT WINDOW)))
                          ((MINUSP Y-OFFSET) (SETQ WINDOW-Y 0))))))))
        ((SYMBOLP SCROLL-BAR)
         ;; We are in a scroll bar.  Moving the mouse faster than the exit speed,
         ;; or moving it towards the center of the window by more than the scroll
         ;; bar width will escape. Cannot escape by moving outside the window.
         ;; [This is different from the previous algorithm, which involved a
```

```
            ;; "hidden variable" which let you out when you moved the mouse (but
            ;; not the cursor) more than a certain amount to the "left or right".]
            (COND ((AND SCROLL-BAR-MAX-EXIT-SPEED
                        (> (MOUSE-SPEED MOUSE) SCROLL-BAR-MAX-EXIT-SPEED))
                   ;; Moving like a bat, let the guy out of the scroll bar
                   (RETURN NIL))
                  ((NOT (AND (≤ 0 WINDOW-X) (< WINDOW-X WINDOW-WIDTH)
                             (≤ 0 WINDOW-Y) (< WINDOW-Y WINDOW-HEIGHT)))
                   (WITHOUT-INTERRUPTS
                     (OPEN-MOUSE-CURSOR MOUSE)
                     (CL:SETF WINDOW-X (MIN (MAX WINDOW-X 0) (- WINDOW-WIDTH 1))
                              WINDOW-Y (MIN (MAX WINDOW-Y 0) (- WINDOW-HEIGHT 1))
                              (MOUSE-LAST-X MOUSE) (CL:SETF (MOUSE-X MOUSE)
                                                            (+ WINDOW-X-OFFSET WINDOW-X))
                              (MOUSE-LAST-Y MOUSE) (CL:SETF (MOUSE-Y MOUSE)
                                                            (+ WINDOW-Y-OFFSET WINDOW-Y))
                              (MOUSE-CURSOR-STATE MOUSE) (MOUSE-CURSOR-CLOSED-STATE MOUSE)
                              PREPARED-SHEET NIL)))
                  ((OR (AND (EQ SCROLL-BAR ':IN-LEFT)  (> WINDOW-X SCROLL-BAR-WIDTH))
                       (AND (EQ SCROLL-BAR ':IN-TOP)   (> WINDOW-Y SCROLL-BAR-WIDTH))
                       (AND (EQ SCROLL-BAR ':IN-RIGHT)
                            (< WINDOW-X (- WINDOW-WIDTH SCROLL-BAR-WIDTH)))
                       (AND (EQ SCROLL-BAR ':IN-BOTTOM)
                            (< WINDOW-Y (- WINDOW-HEIGHT SCROLL-BAR-WIDTH))))
                   (RETURN NIL))))))
    ;; Update the position of the mouse before checking for button clicks, so
    ;; that button clicks get processed with knowledge of where the mouse
    ;; was when the button was first clicked.  The arguments to the move handler
    ;; may be where the mouse was when the button was clicked, whereas the
    ;; mouse cursor follows MOUSE-X and MOUSE-Y, which may be different.
    (CL:SETF (MOUSE-WARP-INTERNAL MOUSE) NIL)
    (FUNCALL MOVE-HANDLER MOVE-METHOD WINDOW-X WINDOW-Y)
    ;; Check for all the ways of losing control of the mouse.
    (IF (COND ;; The move handler may have decided to warp the mouse so that it will not
          ;; move out of the window.  This test is a crock but should work.
          ((MOUSE-WARP-INTERNAL MOUSE) NIL)
          ;; Check for mouse seized, becoming seized, or ceasing to be seized
          ((OR (WINDOW-OWNING-MOUSE-INTERNAL MOUSE) (EQ WINDOW T))
           (NOT (WINDOW-OWNS-MOUSE-P WINDOW X Y MOUSE)))
          ;; Check for moving into a window when not in any
          ((NULL WINDOW)
           (WINDOW-OWNING-MOUSE X Y MOUSE))
          ;; Check for leaving the boundaries of the current window
          ;; HYSTERETIC-WINDOW-MIXIN requires that we wait at least once before returning
          ((NOT (AND (SHEET-EXPOSED-P WINDOW)
                     (≥ WINDOW-X 0)
                     (< WINDOW-X (SHEET-WIDTH WINDOW))
                     (≥ WINDOW-Y 0)
                     (< WINDOW-Y (SHEET-HEIGHT WINDOW))))
           WAIT-FLAG)
          ;; Check for moving into an inferior of the current window
          ((NEQ (LOWEST-SHEET-UNDER-POINT WINDOW WINDOW-X WINDOW-Y ':HANDLE-MOUSE ':EXPOSED)
                WINDOW)
           T))
        ;; Return to overseer, saving any pending button click.
        (RETURN (MOUSE-DEFER-BUTTONS BU BD MOUSE)))
    ;; Now process button pushes if mouse is not seized
    (COND ((OR (ZEROP BD) (EQ WINDOW T) OLD-OWNER))
          ;; If over an exposed window, send it the button-push
          (WINDOW (FUNCALL WINDOW BUTTONS-METHOD BD WINDOW-X WINDOW-Y))
          ;; Default action for left button is to select what mouse is pointing at
          ((BIT-TEST 1 BD)
           (AND (SETQ HAND (WINDOW-UNDER-MOUSE-INTERNAL MOUSE ':MOUSE-SELECT ':ACTIVE X Y))
                ;; Next line temporarily papers over a bug with :MOUSE-SELECT
                (GET-HANDLER-FOR HAND ':SELECT)
                (MOUSE-SELECT HAND)))
          ;; Default action for middle button is to switch to the main screen
          ((BIT-TEST 2 BD)
           (WHEN (AND (EQ MOUSE (SHEET-MOUSE DEFAULT-SCREEN))
                      (TYPEP (MOUSE-SHEET MOUSE) 'SCREEN))
             (PROCESS-RUN-FUNCTION "Set mouse sheet" #'MOUSE-SET-SHEET DEFAULT-SCREEN)))
          ;; Default action for right button is to call the system menu
          ((BIT-TEST 4 BD)
           (CONSIDER-MOUSE-CALL-SYSTEM-MENU (MOUSE-BUTTON-ENCODE BD MOUSE) NIL X Y MOUSE)))))
```

```
;;;     end of file
```

CLM01:>local>system>joker>zwei-patches.lisp.1

For: Kiyoshi Hayashi
Printed on: comprinter
Number of copies: 1
Data created at: 3/16/89 12:06:00
Queued at: 3/16/89 12:09:37

```
;;; -*- Mode: LISP; Package: ZWEI; Base: 10; Syntax: Zetalisp -*-
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;
;;;     ZWEI PATCHES
;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(SI:BEGIN-PATCH-SECTION)
(SI:PATCH-SECTION-ATTRIBUTES
  "-*- Mode: LISP; Package: ZWEI; Base: 10; Syntax: Zetalisp -*-")


(DEFCONST  種類リスト '(接続なし 一般名詞 姓 名 地名 サ変名詞 形容動詞))

(DEFCOM COM-単語登録 "かな漢字変換用辞書への単語登録ユティリティ

本コマンドは、リージョンで指定した単語を登録するためのコマンドである。

注) 読みとして登録できる文字は、ひらがなで始まり、つぎの文字から構成されている必要がある。

{ ひらがな、数字、空白、ピリオド、カンマ、//、$ }

                                                    以 上" NIL
  (LET* ((単語 (WITH-EDITOR-STREAM (STREAM :START :REGION :WINDOW *WINDOW*)
                (READLINE STREAM)))
         (読み (TYPEIN-LINE-ACTIVATE
                (SEND *TYPEIN-WINDOW* ':FORCE-KBD-INPUT #\Clear-Input)
                (TYPEIN-LINE-READLINE (FORMAT NIL "単語 /"~A/" の読みを入力して下さい。" 単語)))))
    (DO NIL ((読み-P 読み)
             (LET* ((種類 (TYPEIN-LINE-ACCEPT
                           '((SCL::INTEGER 1 7))
                           :PROMPT (FORMAT NIL "言葉の種類を番号で指定して下さい。 ~A"
                                           (MAKE-種類リスト 種類リスト 1))
                           :DEFAULT 1)))
               (TYPEIN-LINE-MORE-DURABLE (FORMAT NIL "~&単語登録 : /"~A/"  /"~A/"  /"~A/"~%"
                                                 単語 読み (NTH (- 種類 1) 種類リスト)))

               (WHEN JOKER:*SERIAL-PORT-IO-STREAM*
                 (LET* ((STR (FORMAT NIL "~C~A" (CODE-CHAR JOKER:単語登録コード)
                                 (MAKE-JIS-STRING (FORMAT NIL
                                                          "~A:~A:~A:" 単語 読み 種類))))
                   (DO ((I 0 (1+ I))) ((= I (STRING-LENGTH STR)))
                     (SEND JOKER:*SERIAL-PORT-IO-STREAM* ':TYO (CHAR-CODE (AREF STR I)))))
                 )))
     (BEEP)
     (SETQ 読み
           (TYPEIN-LINE-READLINE
             (FORMAT NIL
                     "~&[読みの指定誤り ~A は登録できません! ] => 単語 /"~A/" の読みを再入力して下さい。"
                     読み 単語))))
  DIS-NONE)

(DEFUN MAKE-JIS-STRING (STRING &AUX (JIS-STRING "")
  (LOOP FOR CH BEING THE ARRAY-ELEMENTS OF STRING DO
    (IF (NULL (JAPANESE:CHAR-TO-JIS CH T))
        (SETQ JIS-STRING (FORMAT NIL "~A~C~C" JIS-STRING (CODE-CHAR #x00) CH))
        (MULTIPLE-VALUE-BIND (UPPER LOWER)
            (FLOOR (JAPANESE:CHAR-TO-JIS CH T) #x100)
          (SETQ JIS-STRING
                (FORMAT NIL "~A~C~C" JIS-STRING
                        (CODE-CHAR UPPER)(CODE-CHAR LOWER))))))
  (FORMAT NIL "~C~A~C"
          (CODE-CHAR JOKER:シフトインコード) JIS-STRING (CODE-CHAR JOKER:シフトアウトコード)))

(DEFUN MAKE-種類リスト (LIST I)
    (COND ((CDR LIST) (FORMAT NIL "~A:~A ~A" I (CAR LIST)(MAKE-種類リスト (CDR LIST)(+ I 1))))
          (T (FORMAT NIL "~A:~A" I (CAR LIST)))))
```

```
(DEFUN 読み-P (STR)
  (IF (AND (< 0 (LISP:LENGTH STR))
           (JAPANESE:HIRAGANA-CHAR-P (AREF STR 0)))
      (MEMBER-読み-P  (LISP:SUBSEQ STR 1))
      NIL))

(DEFUN MEMBER-読み-P (STR)
  (COND ((= 0 (LISP:LENGTH STR)) T)
        ((MEMBER-読み-CHAR-P (AREF STR 0))
         (MEMBER-読み-P (LISP:SUBSEQ STR 1)))
        (T NIL)))


(DEFUN MEMBER-読み-CHAR-P (CH)
  (IF (OR (JAPANESE:HIRAGANA-CHAR-P CH)
          (AND (<= #x30 (char-code ch))(>= #x39 (char-code ch))) ; 数字
          (CHAR= CH #/ー)
          (CHAR= CH #/.)
          (CHAR= CH #/,)
          (CHAR= CH #/SPACE)
          (CHAR= CH #/$)
          (CHAR= CH #//))
      T
      NIL))

;;;     Zmacsコマンドの登録
(COMMAND-STORE 'COM-単語登録 #\R *ZMACS-CONTROL-X-COMTAB*)

;;;
;;;     Zmacs Mouse-R Menu Patch
;;;
(SI:BEGIN-PATCH-SECTION)
(SYSTEM-INTERNALS:PATCH-SECTION-SOURCE-FILE "SYS:ZWEI;ZMACS.LISP.1036")
(SI:PATCH-SECTION-ATTRIBUTES
  "-*- Mode: LISP; Package: ZWEI; Base: 8; Syntax: Zetalisp; Patch-File: Yes -*-")

(SET-COMTAB *ZMACS-COMTAB*      ;mouse 右クリック ポップアップメニュー(コマンドテーブル)の追加
            (LIST #\Mouse-R
                  (MAKE-MENU-COMMAND '(COM-ARGLIST COM-EDIT-DEFINITION
                                       COM-LIST-CALLERS COM-LIST-DEFINITIONS COM-LIST-BUFFERS
                                       COM-KILL-OR-SAVE-BUFFERS COM-SPLIT-SCREEN
                                       COM-COMPILE-REGION COM-INDENT-REGION
                                       COM-CHANGE-TYPEIN-STYLE COM-CHANGE-STYLE-REGION
                                       COM-UPPERCASE-REGION COM-LOWERCASE-REGION
                                       COM-MOUSE-INDENT-RIGIDLY COM-MOUSE-INDENT-UNDER
                                       COM-単語登録))))


;;;     end of file
```

CLM01:>local>system>joker>reset-JOKER.lisp.1

For: Kiyoshi Hayashi
Printed on: comprinter
Number of copies: 1
Data created at: 3/10/89 09:08:56
Queued at: 3/10/89 17:11:40

```
;;; -*- Mode: LISP; Base: 10; Syntax: Common-lisp; Package: JOKER -*-

(define-cp-command com-reset-JOKER ()
   (send (car (send tv:who-line-file-state-sheet ':blinker-list)) ':set-visibility ':off)
   (send tv:who-line-file-state-sheet ':clear-window)
   (setq 文字種別          "al"
         全角モード        ""
         ローマ字入力      ""
         変換対象          nil)
   (if *serial-port-io-stream*
       (send *serial-port-io-stream* ':tyo #xba)) ; Reset Keyboard
   (serial-port-activate-switch 1))

(cp:install-commands cp:*global-command-table* '(com-reset-JOKER))

;;; end of file
```