Remap keys for Function key usage in RDM

RDM is Remote Desktop Mobile

Intermec Keyboard Remapper

There is no need to redefine the Function Keys with the Remapper Utility. The solution uses a keyboard hook that will be only active if the Demote Desktop Mobile Window is in foreground with an active session.

iHookRemapKeys

This keyboard hook application captures the KEY_DOWN and KEY_UP messages of the hardware keyboard. The Function Keys F1 to F12 will then be consumed and 0xF1 to 0xFC Char messages will be send to the RDM keyboard target window.

It is not possible to send KEYDOWN and KEYUP messages for non-char keys to RDM. RDM only translates char messages for the Terminal Server host.

If RDM is not running in foreground, the hook application will not alter the Function Key presses.

After start a new notification icon is added to the home screen, an "U" inside a yellow box. The frame around will change the color indicating if iHookRemapKeys was able to find the RDM window and if RDM is the foreground Window.

RDM tscshift.txt

Modify \windows\tscshift.txt to map the 'chars' 0xF1 to 0xFC to index 0x70 to 0x7B:

```
0x7d 0xdd 1
0x7e 0xde 1
0x7F 0x2E 0
0x80 0x00 0
0x81 0x00 0
...
0xEF 0x00 0
0xF0 0x00 0
0xF1 0x70 0
0xF2 0x71 0
0xF3 0x72 0
0xF4 0x73 0
0xF5 0x74 0
0xF6 0x75 0
0xF7 0x76 0
0xF8 0x77 0
0xF9 0x78 0
0xFA 0x79 0
0xFB 0x7A 0
0xFC 0x7B 0
0xFD 0x00 0
0xFE 0x00 0
0xFF 0x00 0
```

The above tscshift.txt was extended with all values above 0x7F. This is not necessary. Just add the new lines as needed maintaining the sort.

Instead of mapping the 'chars' from 0xF1 to 0xFC it is also possible to map to other 'unused' 'chars'. Here 0xF1 to 0xFC make the 'chars' ñ, ò, ó, ô, õ, ö, ÷, ø, ù, ú, û, ü unusable as these are 'interpreted' as Function Key presses by RDM. See also "Upper char map" in appendix.

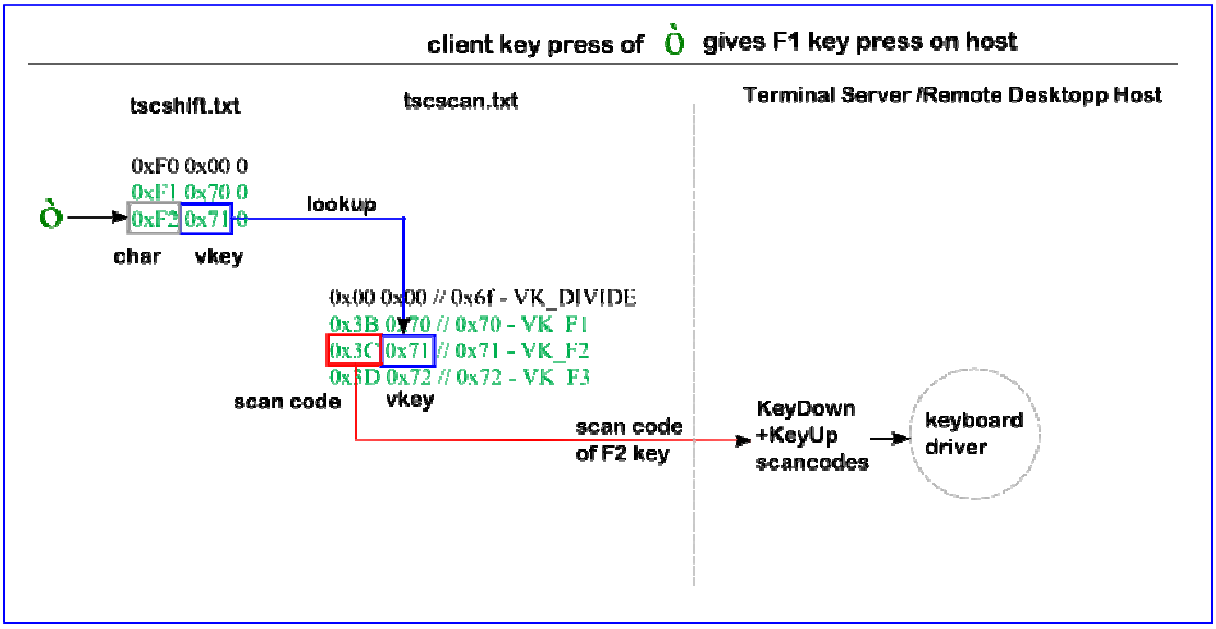
RDM tscscan.txt

Modify lines 0x70 (112) to 0x7B (115) in \windows\tscscan.txt. The line count starts at 0 and comment lines (lines beginning with "//") are not counted. The line counted is the index that tscshift.txt is pointing to.

```
...
0x00 0x00 // 0x6e - VK_DECIMAL
0x00 0x00 // 0x6f - VK_DIVIDE
0x3B 0x70 // 0x70 - VK_F1
0x3C 0x71 // 0x71 - VK_F2
0x3D 0x72 // 0x72 - VK_F3
0x3E 0x73 // 0x73 - VK_F4
0x3F 0x74 // 0x74 - VK_F5
0x40 0x75 // 0x75 - VK_F6
0x41 0x76 // 0x76 - VK_F7
0x42 0x77 // 0x77 - VK_F8
0x43 0x78 // 0x78 - VK_F9
0x44 0x79 // 0x79 - VK_F10
0x57 0x7A // 0x7a - VK_F11
0x58 0x7B // 0x7b - VK_F12
0x00 0x00 // 0x7c - VK_F13
0x00 0x00 // 0x7d - VK_F14
...
```

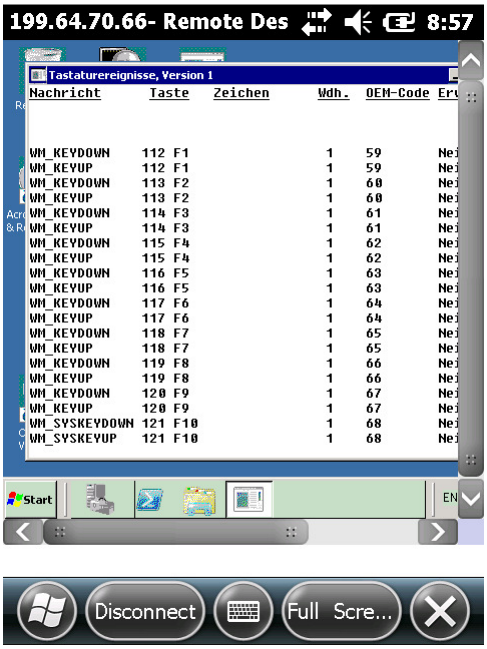
The first byte is the scancode to be used for this key. The scan codes used here are the ones for the F1 to F12 keys as defined for a PS/2 keyboard.

Remote Desktop Mobile (RDM) keyboard handling



Test against Terminal Server/Remote Desktop host

The below shows that the hardware Function Keys on the Windows Mobile keypad now are sent to the Terminal Server. The test application shows the keyboard messages received.



Note: F11 and F12 are not processed by this test application.

Define a Function Key as Pre- or Postamble

Virtual Wedge settings

To define a Function Key, for example, as Postamble, open "Start">"Settings">"System">"Intermec Settings" and scroll down to "Virtual Wedge". Ensure "Enable Virtual Wedge" is selected. The "Virtual Wedge Method" should be set to "Character Mode". Inside "Barcode Scanner Wedge" set ".*=>\1\xf1" as "Barcode Scanner Grid". This will add the 0xF1 char code at the end of every scanned barcode. This 0xF1 is routed thru tscshift.txt and the index there points to line 0x70 of tscscan.txt. Via tscscan.txt RDM will send the scan code 0x3B to the host. The host sees that scan code and translates it to the F1 key.

Additionally to "Adapt to application" a custom vwConfig.ini for "TSSHELLWND" can be used.

```
[VWConfig]
;dump config to vwCustom.log.txt and vwDefault.log.txt
DumpConfig="1"

;Only the default application is allowed to have a NULL string for the window name.
;the text between [ and ] specifies either a window title or class name. If you use
;DEFAULT_APPLICATION, then the setting is applied, if no other section matches
[DEFAULT_APPLICATION]
WindowName="TSSHELLWND"

;__GlobalXmitMethod__
;GlobalXmitMethod is a string and used similarly to the GlobalFlag.
;Setting this variable to will force all sequences from 0-255 to use this transmit
method.
;The following are the only transmit method strings allowed:
;EVENT
;POST
;SEND
;CLIP
;These strings are not case sensitive and must be surrounded by quotes.

GlobalXmitMethod="EVENT"

;__GlobalXmitType__
;GlobalXmitType is a string and used similarly to the GlobalXmitMethod.  Setting
this variable will force all sequences from 0-255 to use this transmit type.
```

```
;Characters above this can be modified using the UpperKeySeq variable.  Except for
one case, the following are the only transmit type strings allowed:
;PLAIN      - character sent with any modifiers such as shift, ctrl, alt, etc.
;SHIFT      - character sent with the "shift" modifier
;CTRL       - character sent with the "control" modifier
;CTRL_SHIFT - character sent with both the "shift" and "control" modifiers
;ALT        - character sent with the "alternate" modifier
;CTRL_ALT   - character sent with both the "control" and "alternate" modifiers
;UNSHIFT    - sends the "shift" release only
;SYNTH      - character is sent using "synthesize" to generate the character.  This
is useful mostly for characters above 127.
;PACKET     - this is used by the PostKeybdMessage() API on some platforms.  The
virtual key is set to VK_PACKET and the number value of the character is included
in the API call.
;Table      - uses a table for transmit type

GlobalXmitType="Table"

;;;exceptions to the above for TAB and CR
;RegSeq="0x09 POST PLAIN 0X09 0X00 0x0000"
;RegSeq="0x0D POST PLAIN 0X0D 0X00 0x0000"
```

The default vwConfig settings use the clipboard and that will not be compatible with the above custom tscshift.txt setup.

Appendix

Upper Char Map

hex	dec	char	hex	dec	char	hex	dec	char	hex	dec	char
0x80	128	€	0xA0	160		0xC0	192	À	0xE0	224	à
0x81	129		0xA1	161	¡	0xC1	193	Á	0xE1	225	á
0x82	130	‚	0xA2	162	¢	0xC2	194	Â	0xE2	226	â
0x83	131	ƒ	0xA3	163	£	0xC3	195	Ã	0xE3	227	ã
0x84	132	„	0xA4	164	¤	0xC4	196	Ä	0xE4	228	ä
0x85	133	…	0xA5	165	¥	0xC5	197	Å	0xE5	229	å
0x86	134	†	0xA6	166	¦	0xC6	198	Æ	0xE6	230	æ
0x87	135	‡	0xA7	167	§	0xC7	199	Ç	0xE7	231	ç
0x88	136	ˆ	0xA8	168	¨	0xC8	200	È	0xE8	232	è
0x89	137	‰	0xA9	169	©	0xC9	201	É	0xE9	233	é
0x8A	138	Š	0xAA	170	ª	0xCA	202	Ê	0xEA	234	ê
0x8B	139	‹	0xAB	171	«	0xCB	203	Ë	0xEB	235	ë
0x8C	140	Œ	0xAC	172	¬	0xCC	204	Ì	0xEC	236	ì
0x8D	141		0xAD	173		0xCD	205	Í	0xED	237	í
0x8E	142	Ž	0xAE	174	®	0xCE	206	Î	0xEE	238	î
0x8F	143		0xAF	175	¯	0xCF	207	Ï	0xEF	239	ï
0x90	144		0xB0	176	°	0xD0	208	Ð	0xF0	240	ð
0x91	145	‘	0xB1	177	±	0xD1	209	Ñ	0xF1	241	ñ
0x92	146	’	0xB2	178	²	0xD2	210	Ò	0xF2	242	ò
0x93	147	“	0xB3	179	³	0xD3	211	Ó	0xF3	243	ó
0x94	148	”	0xB4	180	´	0xD4	212	Ô	0xF4	244	ô
0x95	149	•	0xB5	181	µ	0xD5	213	Õ	0xF5	245	õ
0x96	150	–	0xB6	182	¶	0xD6	214	Ö	0xF6	246	ö
0x97	151	—	0xB7	183	·	0xD7	215	×	0xF7	247	÷
0x98	152	˜	0xB8	184	¸	0xD8	216	Ø	0xF8	248	ø
0x99	153	™	0xB9	185	¹	0xD9	217	Ù	0xF9	249	ù
0x9A	154	š	0xBA	186	º	0xDA	218	Ú	0xFA	250	ú

0x9B	155	›	0xBB	187	»	0xDB	219	Û	0xFB	251	û
0x9C	156	œ	0xBC	188	¼	0xDC	220	Ü	0xFC	252	ü
0x9D	157		0xBD	189	½	0xDD	221	Ý	0xFD	253	ý
0x9E	158	ž	0xBE	190	¾	0xDE	222	Þ	0xFE	254	þ
0x9F	159	Ÿ	0xBF	191	¿	0xDF	223	ß	0xFF	255	ÿ

Scan codes (PS/2)

Key	Make	Break	Key	Make	Break
Backspace	0E	8E	F1	3B	BB
Caps Lock	3A	BA	F2	3C	BC
Enter	1C	9C	F3	3D	BD
Esc	01	81	F4	3E	BE
Left Alt	38	B8	F7	41	C1
Left Ctrl	1D	9D	F5	3F	BF
Left Shift	2A	AA	F6	40	C0
Num Lock	45	C5	F8	42	C2
Right Shift	36	B6	F9	43	C3
Scroll Lock	46	C6	F10	44	C4
Space	39	B9	F11	57	D7
Sys Req (AT)	54	D4	F12	58	D8
Tab	0F	8F			

Key	Make	Break	Key	Make	Break
A	1E	9E	N	31	B1
B	30	B0	O	18	98
C	2E	AE	P	19	99
D	20	A0	Q	10	90
E	12	92	R	13	93
F	21	A1	S	1F	9F
G	22	A2	T	14	94
H	23	A3	U	16	96
I	17	97	V	2F	AF
J	24	A4	W	11	91
K	25	A5	X	2D	AD
L	26	A6	Y	15	95
M	32	B2	Z	2C	AC