

Twiddler 2 Key Map Download

Date: 09/07/2001

Version: V1.6

1. Introduction

This document details the PS/2 Key Map structure and the Download Protocol for the Handykey Twiddler 2.

The Twiddler 2 communicates with the host PC using a standard PS/2 port. Twiddler chords are mapped to PS/2 make/break sequences, which can consist of a single keystroke or multiple keystroke. There are 1020 chords which can be mapped to a standard keystroke(s) or a user defined custom keystroke(s); chords can also be unmapped altogether. The Key Map is stored in a non-volatile 8Kx8 EEPROM. Key Maps can be programmed via the Interactive Programming mode or via the K/M PS/2 connector. Note that K/M connector is intended to connect directly to a standard PS/2 port. However, this connector can also interface to an RS-232 port using a special adapter. This document will detail both download protocols.

2. Key Map Memory

The 8 kilobytes of memory used to store the Key Map is divided into four sections: (1) a 2048 (2K) byte pointer array, (2) a 16 byte constant section, (3) a 32 byte mouse key buffer, and (4) a 6096 byte block for custom scan code sequences.

2.1. Pointer Array

The pointer array maps each of the 1020 chords to a PS/2 scan code sequence using a 2 byte value. Depending on the value of the pointer, the scan code sequence can be stored in the T2's program memory (ROM) or in the Custom Scan Code Memory.

2.1.1. Chord to Pointer Mapping

The mapping from chord to pointer is based on both a finger combination value (low byte) and one of four thumb modifier values (high byte). Each of the four finger rows can generate one of four values; 0 ⇨ no key, 1 ⇨ Left, 2 ⇨ Middle, and 3 ⇨ Right. The four values for each row map into two bits in a byte value. The top row maps into bits 0 & 1, the second row maps to bits 2 & 3, the third row maps to bits 4 & 5, and the bottom row maps to bits 6 & 7.

For example ...

L000	=	0000 0001B	=	01H
000L	=	0100 0000B	=	40H
LMR0	=	0011 1001B	=	39H
ORML	=	0110 1100B	=	6CH

The four thumb modifier values are; 0 ⇨ none, 1 ⇨ Shift, 2 ⇨ Num, 3 ⇨ Func (Num+Alt). Note that Alt and Ctrl pressed independently do not directly participate in the chording to pointer mapping; Alt and Ctrl indirectly participate in the chord by automatically including the appropriate scan code to the selected scan code sequence.

For simplicity the pointer is derived by concatenating the thumb modifier (in the high byte) with the finger byte creates a 10 bit value. Note there are 255 possible finger combinations (excluding no finger keys), which yields the 1020 valid chord to pointer maps. This value is then shifted left one bit to get the address of the pointer.

Since the derived address does not take into account the four invalid combinations (i.e. the ones with no finger keys pressed), the resulting address is a value from 000H to 7FEH. The pointers at locations 200H, 400H, and 600H are not used. The 2-byte value at 000H is a 2's complement checksum of the entire key map image (see Section 2.5).

2.1.2. Key Map Pointer Values

Each of the 2-byte pointer values can have one of four formats:

1. A value of 0000H unmaps the chord, i.e. no scan code sequence will be issued.
2. A value of FFFFH maps the chord to the programmed default in ROM. See T2 User Manual for the default mapping.
3. A values of 00xxH maps the chord to a preprogrammed sequence in a ROM lookup table. The value of xxH must be within the valid preprogrammed range (see Appendix A).
4. A value of 0830H and above (the value must be less than the maximum for the memory) points to a custom scan code sequence in the fourth section of the EEPROM.

2.2. Constants

The 16 bytes from 800H to 80FH are used to store various programmable constants used by the T2. A summary of constants is provided in the table below. Note that these constants are equivalent to the ones that can be programmed using the Interactive Programming Mode.

Constant	Type	Address	Description	Default
Key Repeat Delay	Byte	800H	Delay before Key (typematic) repeat, * 0.005s.	100 (64H) = 0.5s
Key Repeat Rate	Byte	801H	Key repeat (typematic) rate, * 0.005s.	20 (14H) = 0.1s
Mouse Timeout	Word	802H	Mouse Mode timeout, * 0.005s, FFFFH ⇨ Indefinite	65535 (FFFFH) = Indefinite
Mouse Double Delay	Byte	804H	Mouse double click time (1 st) and delay (between), *0.005s.	20 (14H) = 0.1s
Mouse Key Delay	Byte	805H	Delay from alt, ctrl, shift to pressing mouse button, * 0.005s.	10 (0AH) = 0.05s
Mouse Debounce	Byte	806H	Controls debouncing of mouse activity to enable Mouse Mode.	52H (See T2 Manual for explanation)

<u>Constant</u>	<u>Type</u>	<u>Address</u>	<u>Description</u>	<u>Default</u>
Mouse On Key	Word	808H	Mouse On Chord, binary number format as in pointer map.	03AAH = FUNC MMMM
Mouse Off Key	Word	80AH	Mouse Off Chord, binary number format as in pointer map.	0355H = FUNC LLLL
Mouse Flags	Byte	80CH	Mouse operation flag(s), bit 0 = (1) 0 ⇒ (no) exit mode after click.	00H ⇒ exit mouse mode after click
Unused/Reserved	Byte	80DH	Not used, reserved for future use.	00H
Unused/Reserved	Byte	80EH	Not used, reserved for future use.	00H
Unused/Reserved	Byte	80FH	Not used, reserved for future use.	00H

2.3. Mouse Keys

The 32 bytes located from 810H through 82FH are used to stored the Mouse Key Array. Each entry in the array consists of two bytes: (1) the finger chord byte, and (2) the mouse action byte. Up to 15 mouse keys can be assigned. The array is null terminated so the last two bytes (82EH & 82FH) are always 0. If fewer than 15 mouse keys are assigned, the unused bytes should all be set to 0.

The finger chord byte follows the same format as the least significant byte described in section 2.1.1. Note that because thumb buttons automatically take the T2 out of mouse mode, these buttons can not be use in the chord allowing for only a single byte to represent the finger chord. Alt, Ctrl, and Shift key scan codes can be issued prior to executing the mouse function by properly defining the mouse action byte.

The mouse action byte is a bit-mapped value that tells the T2 what mouse function to perform as a result of seeing the finger chord when in mouse mode. The table below defines the bit map for the mouse action byte. These bits correlate to the mouse key definitions in the Interactive Programming Mode.

<u>Bit</u>	<u>Value</u>	<u>Name</u>	<u>Action</u>	<u>Description</u>
0	01	LEFT	Left Click	Issues a Left Mouse Button Click
1	02	MIDDLE	Middle Click	Issues a Middle Mouse Button Click
2	04	RIGHT	Right Click	Issues a Right Mouse Button Click
3	08	DBL	Double Click	Execute a Double Click – See Note
4	10	SHIFT	Shift Key	Issue Shift prior to Click
5	20	CTRL	Ctrl Key	Issue Ctrl prior to Click
6	40	ALT	Alt Key	Issue Alt prior to Click
7	80	TOGGLE	Toggle	Execute mouse button toggle – See Note

Note: Double and Toggle are mutually exclusive. If both bits are set, this creates the Invert Exit (INV_EXIT, 88H) mode flag.

The default mouse keys are defined in Appendix B.

2.4. Custom Scan Code Memory

The remaining memory 6096 bytes of memory are used to store the custom make/break scan code sequences and text strings. The value stored in the pointer array (see Section 2.1) is the actual starting memory location of the make/break sequence or text string. The first byte is always a type and length byte. The most significant bit identifies the type, and the least-significant 7 bits are the length. If the most-significant bit is 1, the data is a text. Otherwise, the data is a make/break scan code sequence.

2.4.1. Text Strings

The maximum length for a text string is 127 characters (a zero length string, type/length = 80H, is ignored). Text strings are assumed to be ASCII strings consisting of the printable ASCII character set (space, 20H, through tilde, ~ 7EH) plus carriage return (CR, 0DH) and the tab character (HT, 09H). When the T2 processes a text string, each character is converted to the standard PS/2 scan code sequence. First the make sequence is issued, then the break sequence is issued, the T2 proceeds to the next character. The upper case letters include the make/break code for the left shift key (see Appendix A).

2.4.2. Make/Break Sequences

A make/break scan code sequence consists of two sequences of binary scan codes. Both the make and the break sequence are limited to 63 bytes each, and must be at least 1 byte.

The distinction between the make and break sequence is when they are issued, not as to what scan code values are permitted. The make sequence is issued when a key is pressed; while the break sequence is issued when the key is released. The make sequence will also be re-issued during the typematic function. Since, unlike a regular keyboard, the T2 acts on key releases, the make and break sequences are issued in succession. However, when the chord is repeated and held, the T2 will re-issue the make sequence, and repeat it at the Key Repeat Rate (see Section 2.2). It is perfectly permissible to include both make and break scan codes in both the make sequence and in the break sequence.

Caution must be exercised when creating scan code sequences to insure that proper make/break relationship is preserved, and the only valid scan codes are included. The T2 does not error check the custom scan code sequences, and erroneous PC operation can occur if the make/break relationship is violated and/or if invalid scan codes are issued. Repeated make codes are always permissible so long as there is one and only one corresponding break code.

2.5. Key Map Checksum

As was mentioned in Section 2.1.1, a word (two bytes) 2's complement checksum is stored at address location 0000H in the Key Map memory. This checksum is verified on startup. If an invalid checksum is detected, the T2 will not use the Key Map data stored in the EEPROM, but will instead use the default Key Map stored in its program memory. The checksum is calculated by summing all bytes from address location 0002H through address location 1FFFH modulo 65536. The resulting sum is 2's complemented (complement and increment), and stored in address locations 0000H (most significant

byte) and 0001H (least significant byte). When the T2 verifies the checksum, it sums the bytes modulo 65536 from 0002H to 1FFFH, then adds this word to the checksum value, and checks to see the result is zero.

3. Key Map Download

Downloading the Key Map data from the PC can be accomplished via the PS/2 port or an RS-232 comm port. However, the RS-232 comm port download requires a special adapter to convert the TTL level signals to RS-232 levels. Regardless of the transfer protocol used, data is sent in 32 byte binary blocks. While it is possible to program just a portion of the Key Map EEPROM, it is highly recommended that all 8K bytes are programmed to insure the integrity of the checksum. With only 8K bytes to be transmitted and programmed, the programming time is minimal (typically less than 10 seconds).

The following two sections will detail the specifics of the PS/2 and RS-232 download options.

3.1. PS/2 Download

The PS/2 Download is somewhat complicated by the fact that there is no provision in the standard PS/2 protocol to download large amounts of data to a keyboard as is required to by the T2's Key Map memory. Also, in order to remain compliant with the PS/2 specification, new commands could not be added as the PS/2 specification requires a device to respond with a RESEND (FE – note all PS/2 commands are in hex) command when it receives an invalid command. Unfortunately some laptop PCs use these commands to determine if a new device has been connected to the PS/2 port.

To solve these problems, and still remain PS/2 compliant required implementing an algorithm that whereby the host to sends a NOP command (EF) four times in rapid succession to place the T2 into the programming mode. The first three times the NOP is received, the T2 responds with RESEND (FE) as required by the specification. However, the fourth time the NOP is received, the T2 assumes the host want to initiate a programming sequence and responds with an ACK (FA). The host must send each of the four NOPs within 25mS of the previous NOP.

3.1.1. PS/2 Programming Commands

Once the program mode has been enabled the commands shown in the table below can be used to program and/or verify the Key Map. Note that the PS/2 communication protocol is still followed, each byte received is ACK'd by the T2. When data is requested of the T2, the command is ACK'd first, then the data byte(s) are sent.

When the program mode is first entered, the starting address for programming or verifying is set to 0000H (i.e. the first 32 byte page). Since each program or verify command always handles a 32 byte page block, the internal programming address is automatically incremented to the start of the next page after each program or verify command. In other words, as long as the data is to be programmed or verified sequentially, it's not necessary to issue the Set Page Block command.

Program commands and verify commands can not be interspersed. The entire image should programmed and then verified. Programming is truly done after each 32 byte

block is received. Once programming has begun, aborting the programming sequence can result in an invalid checksum.

<u>Command</u>	<u>Value</u>	<u>Description</u>
Set Defaults	E9	Instructs the T2 to set the Key Map to its Default state and exit the program mode.
Set Page Block	EA	Set the starting page address for the next block of data. Host sends a byte value which the T2 shifts left by 5 bits to get the starting EEPROM address.
Program Verify	EB	Instructs the T2 to reply with a 32 byte block of data for verification.
Program Done	EC	Instructs the T2 to exit the Program mode and resume normal operation. The new Key Map will be active provided the checksum is valid.
Program Data	EF	Receive and program a 32 byte page. Note this is the same command used to enter the Program mode. (NOP in the PS/2 protocol)
Get Version	F1	Instructs the T2 to send its program version as a BCD byte (e.g. 16H). (NOP in the PS/2 protocol)

Note: The Get Version command (F1) can also be sent outside the Program mode to get the T2's program version using the same sequence as is used to enable the program mode. If the host sends the F1 command 4 times in rapid succession, on the fourth receipt the T2 will ACK the command and send the version byte.

3.1.2. PS/2 Programming Sequence Summary

The simplest sequence to program the Key Map memory is as follows ...

<u>Host Command</u>	<u>T2 Response</u>	<u>Notes</u>
EF	FE	PS/2 NOP followed by RESEND
EF	FE	PS/2 NOP followed by RESEND
EF	FE	PS/2 NOP followed by RESEND
EF	FA	PS/2 NOP followed by ACK
EF <data>	FA ...	Each Byte is ACK'd – Block 0
EF <data>	FA ...	Each Byte is ACK'd – Block 1
.	.	“
.	.	“
EF <data>	FA ...	Each Byte is ACK'd – Block 255
EC	FA	Program Done followed by ACK

If it is desired to verify the programming, then prior to sending the Program Done command, the host should send the following sequence ...

<u>Host Command</u>	<u>T2 Response</u>	<u>Notes</u>
EA, 00	FA, FA	T2 ACK's Each Byte – Set Block 0
EB	FA <data>	T2 ACK's Command, Sends 32 bytes, Block 0

EB	FA <data>	T2 ACK's Command, Sends 32 bytes, Block 1
.	.	“
.	.	“
EB	FA <data>	T2 ACK's Command, Sends 32 bytes, Block 255
EC	FA	Program Done followed by ACK

To return the T2 to its default map the host can issue the following sequence ...

Host <u>Command</u>	T2 <u>Response</u>	<u>Notes</u>
EF	FE	PS/2 NOP followed by RESEND
EF	FE	PS/2 NOP followed by RESEND
EF	FE	PS/2 NOP followed by RESEND
EF	FA	PS/2 NOP followed by ACK
E9	FA	Set Defaults followed by ACK
EC	FA	Program Done followed by ACK

3.2. RS-232 Download

To perform an RS-232 download, a special adapter is required to convert the TTL level signals from the T2's keyboard PS/2 lines to RS-232 levels. Once the T2 is properly interfaced to the PC's serial port, the Key Map data can be downloaded in a similar fashion to the PS/2 download detailed in the previous two sections.

3.2.1. RS-232 Adapter Description

The simplest way to make the adapter is to use a standard TTL-to-RS-232 interface IC (e.g. a Maxim MAX232). Note that in order for the T2 to function it still must be powered from the Mouse's PS/2 port (M/K connector). Since the +5V power and ground lines are commoned between the M/K and K/M connectors, this same supply can be used to power the interface IC. Then all that needs to be done is to route the PC's serial transmit line through one of the interface IC's receive channels and take its output into the T2's keyboard data line. Similarly, the PC's serial receive line is connected to one of the interface IC's transmit channels; the input side of the transmit channel is connected to the T2's keyboard clock line.

In short, the T2's PS/2 keyboard clock and data lines can automatically detect and function as asynchronous serial transmit and receive lines, respectively. The sole function of the interface IC in the adapter is to provide the requisite voltage level shift from TTL to RS-232.

3.2.2. RS-232 Download Protocol

The data transmission protocol uses the standard RS-232 format with 8 data bits, no parity bit, and one stop bit. The BAUD rate is fixed at 28.8K. All commands are initiated by the host and are acknowledged by the T2.

3.2.3. RS-232 Download Command Set

Once the T2 is properly interfaced to the host PC's comm port via the adapter, the following commands can be used to program the Key Map. All commands except for the Load command use ASCII format for the data. Commands issued from the host to the T2 do not use a terminal character, and must be the exact number of bytes specified. Command acknowledgements from the T2 to the host are terminated by an ASCII carriage return (CR, 0DH) character.

<u>Command</u>	<u>Code</u>	<u>Description</u>
Set Defaults	B	Instructs the T2 to set the Key Map to its Default. The B command must be followed by the ASCII characters "Twiddler" to be valid.
Program Done	D	Instructs the T2 to exit the Programming mode and resume normal operation. The new Key Map will be active provided the checksum is valid.
Load Page Block	L	T2 receives 32 binary bytes and programs it into the Key Map EEPROM at the current address.
Program Enable	P	Enables the Program mode and sets the address pointer to 0000H. The P command must be followed by the ASCII characters "Twiddler" to enable the program mode.
Read EEPROM	R	Instructs the T2 to read and return 8 bytes of data beginning at a specified address. Address and data are in ASCII Hex.
Get Version	V	Instructs the T2 to send its program version. Version is two ASCII characters (e.g. "16").

Note: The T2 ignores all other command codes. Valid command codes with invalid data are flagged by the T2 responding with the "E" command.

3.2.4. RS-232 Download Programming Sequence Summary

Note that the RS-232 command set does not include a command to set the address pointer. Accordingly, the RS-232 download procedure must begin at address 0H and should continue until all 8K bytes have been sent. Below is the required programming sequence.

<u>Host Command</u>	<u>T2 Response</u>	<u>Notes</u>
PTwiddler	P	Initiates programming mode, address pointer = 0H
L <data>	L	Load data into Block 0 - <data> is 32 binary bytes
L <data>	L	Load data into Block 1 - <data> is 32 binary bytes
.	.	"
.	.	"
L <data>	L	Load data into Block 256 - <data> is 32 binary bytes
D	D	Programming complete.

Appendix A: T2 Pre-Programmed Key Code Map

The table below provides the T2's pre-programmed key code map. Setting the most-significant byte to 00H and the least-significant byte in a key map pointer will configure the corresponding chord to issue the scan code sequence for the key/macro shown in the table. Note the mapping is somewhat correlated to the ASCII character set.

Code	Key/Macro	Code	Key/Macro	Code	Key/Macro	Code	Key/Macro
		30	n	60	`	90	SHIFT+HOME
01	INS	31	1	61	a	91	SHIFT+END
02	HOME	32	2	62	b	92	SHIFT+PGUP
03	END	33	3	63	c	93	SHIFT+PGDN
04	PGUP	34	4	64	d	94	SHIFT+UPAR
05	PGDN	35	5	65	e	95	SHIFT+DNAR
06	UPAR	36	6	66	f	96	SHIFT+LTAR
07	DNAR	37	7	67	g	97	SHIFT+RTAR
08	BKSP	38	8	68	h	98	ALT+F1
09	TAB	39	9	69	i	99	ALT+F2
0A	BTAB	3A	:	6A	j	9A	ALT+F3
0B	LTAR	3B	;	6B	k	9B	ALT+F4
0C	RTAR	3C	<	6C	l	9C	ALT+F5
0D	ENTER	3D	=	6D	m	9D	ALT+F6
0E	CTRL	3E	>	6E	n	9E	ALT+F7
0F	ALT	3F	?	6F	o	9F	ALT+F8
10	"the"	40	@	70	p	A0	ALT+F9
11	"of"	41	A	71	q	A1	ALT+F10
12	"to"	42	B	72	r	A2	ALT+F11
13	"ed"	43	C	73	s	A3	ALT+F12
14	"and"	44	D	74	t	A4	CTRL+F1
15	in	45	E	75	u	A5	CTRL+F2
16	ion	46	F	76	v	A6	CTRL+F3
17	ing	47	G	77	w	A7	CTRL+F4
18	CAPS	48	H	78	x	A8	CTRL+F5
19	NUM	49	I	79	y	A9	CTRL+F6
1A	SCROLL	4A	J	7A	z	AA	CTRL+F7
1B	ESC	4B	K	7B	{	AB	CTRL+F8
1C	PRINT	4C	L	7C		AC	CTRL+F9
1D	PAUSE	4D	M	7D	}	AD	CTRL+F10
1E	SHIFT	4E	N	7E	~	AE	CTRL+F11
1F	BREAK	4F	O	7F	DEL	AF	CTRL+F12
20	SPACE	50	P	80	F1	B0	SHIFT+F1
21	!	51	Q	81	F2	B1	SHIFT+F2
22	"	52	R	82	F3	B2	SHIFT+F3
23	#	53	S	83	F4	B3	SHIFT+F4
24	\$	54	T	84	F5	B4	SHIFT+F5
25	%	55	U	85	F6	B5	SHIFT+F6
26	&	56	V	86	F7	B6	SHIFT+F7
27	'	57	W	87	F8	B7	SHIFT+F8
28	(58	X	88	F9	B8	SHIFT+F9
29)	59	Y	89	F10	B9	SHIFT+F10
2A	*	5A	Z	8A	F11	BA	SHIFT+F11
2B	+	5B	[8B	F12	BB	SHIFT+F12
2C	,	5C	\	8C	WIN	BC	RALT
2D	-	5D]	8D	RWIN	BD	RCTRL
2E		5E	^	8E	APP	BE	SHIFT

2F	/	5F	_	8F	"The"		
----	---	----	---	----	-------	--	--

Appendix B: Default Mouse Key Assignments

The table below shows the default mouse key assignments for the T2. Included in the table are the hex code bytes for the finger chord and for the mouse action. Note that only 12 of the 15 possible assignments are used.

Chord	Chord Byte	Mouse Action	Action Byte
L000	01	RIGHT	04
M000	02	LEFT+INV_EXIT	89
R000	03	LEFT	01
0L00	04	RIGHT+DBL	0C
0M00	08	MIDDLE+INV_EXIT	8A
0R00	0C	LEFT+DBL	09
00L0	10	LEFT+ALT	41
00M0	20	LEFT+SHIFT	11
00R0	30	LEFT+CTRL	21
000L	40	LEFT+DBL+SHIFT	19
000M	80	RIGHT+TOGGLE	84
000R	C0	LEFT+TOGGLE	81