Z8602/14/15/C15/E23 MICROCONTROLLERS CONTROL A 101/102 PC/KEYBOARD

Scan codes, line status modes, key bounce, make/break status, scan timing... the Microcontroller does all this and much more.

INTRODUCTION

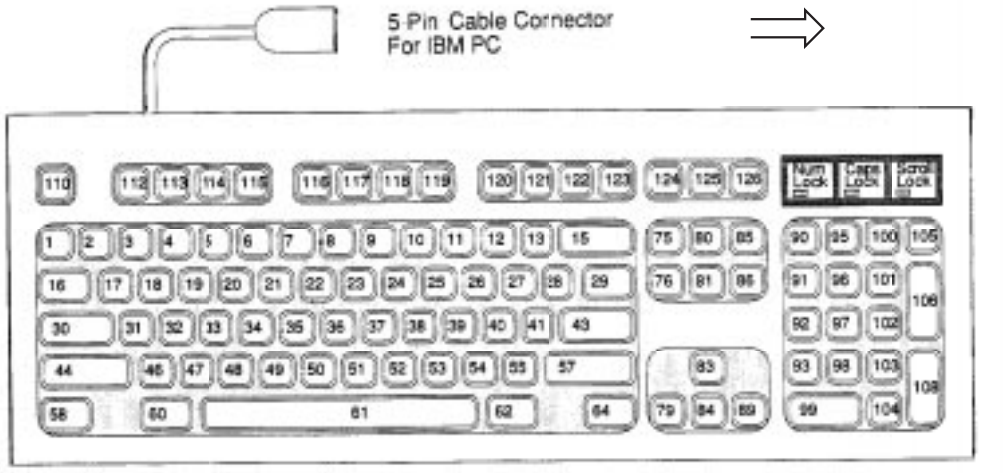
Zilog has continuously provided cost-effective solutions for the keyboard controller market. Design time to production is dramatically shortened by Zilog's easy-to-use development tools, source code availability, and steady relationships with major manufacturers that assist in defining the product line feature set.

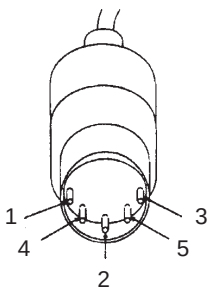
Zilog provides the following microcontrollers dedicated to keyboard operation: Z8602, Z8614, Z8615, Z86C15, and the Z86E23. The Z86E23 provides prototyping capability for the 101/102 keyboard environment; the Z8614, compared to the Z8602, provides additional ROM space for an expanded feature set; and the Z8615/C15 provides integration of external componentry.

These microcontrollers are designed into a PC keyboard to control all scan codes, line status modes, scan timing, and communications between the keyboard and PC. This application note depicts a typical method of interfacing a standard keyboard to an PC XT/AT.

KEYBOARD OVERVIEW

The 101/102 PC/Keyboard includes the following: 101/102 keypads, Zilog microcontroller, 3 LED indicators (for Num, Caps, and Scroll) for lock status, selection switch for PC/XT, and a cable between PC and keyboard. The cable provides a 5V power supply, common ground, and bi-directional Data and Clock lines for serial data communications (see Figure 1).





DIN Pins	Signal Name	Signal Type
1	*+KBD CLK*	*I/O*
2	*+KBD Data*	*I/O*
3		
4	*GND*	*Power*
5 *SHIELD*	*+5.0 V_{DC}* *Frame GND*	*Power*

Figure 1. PC Keyboard 101/102 and 5-Pin Cable Connector

KEYBOARD OVERVIEW (Continued)

PC/Keyboard Overall Flow Chart

The overall flow chart in Figure 2 shows three different diagrams involving the main program loop, keyboard scan and make/break code generation, and the PC/Keyboard communication. The following text explains the basic program flow.

Main Program Loop
Upon reset, a Basic Assurance Test on the RAM, Keyboard, and ROM is enabled. The parameters start, and test data transfers between the CPU and RAM, Keyboard and ROM. If there is no error detection, RAM receives the default values and the initialization testing is complete.

Keyboard Scan and Make/Break Code Generation
This program executes every 4.17 milliseconds (ms). If there is a current communication, the flow bypasses and returns to the main program. If there is no current communication and Key Scan (KEY_IRQ) is enabled, the main process for six buffering bounce elimination in make or break setup, goes active.

Lock Status indicators are monitored and then Make/ Break case examined. A decision determines which case is active. The active case tests for six keys. If it is the Make case and is Typematic, the Make Code Generation parameters execute. If it is a Break case, the flow returns to the main program.

Communication Between PC and Keyboard
Communication between the PC and the keyboard executes every 250 microseconds (µs). When there is an active COM_IRQ but a positive Communication Inhibit, the flow returns to the main program. If there is an active COM_IRQ and no Communication Inhibit and there is a request from the PC, the Receive Data/Command activates. Depending upon whether it is a new command, optional data, or send Resend, the pertinent logic goes active and the flow returns to the main program.

When there is no request from the PC, but there is current communication, the flow jumps to the Resend Mode. If the Resend Mode is active, the last data is sent and then flow returns to the main program. If the Resend Mode is inactive, the flow jumps to Transmit Data/Ack. If there is no more communication, the flow returns to the main program. If there is more communication, the remaining commands execute and the flow returns to the main program.

When there is a COM_IRQ but no communication inhibit or request from the PC, and not during communication, the FIFO Key Buffer is checked. If it is empty, the flow returns to the main program. If not empty, one byte is sent from the FIFO Key Buffer and one byte shifts. The flow then returns to the main program.

Scratch Pad RAM Map

Table 1 describes the RAM map. The table shows the function name, RAM address, bit position, bit name, and descriptions for all map functions.

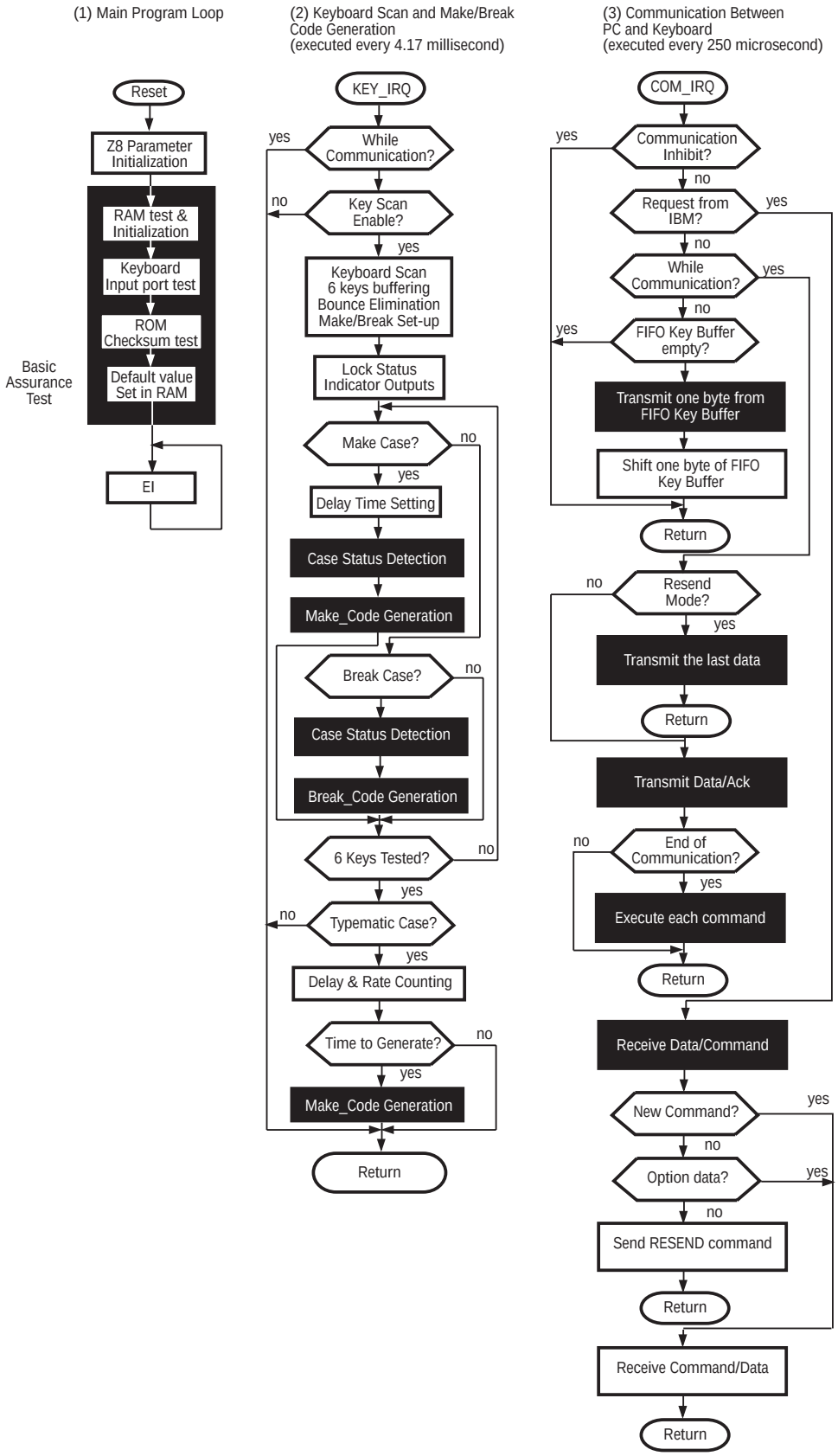


Figure 2. PC/Keyboard Program Flow Chart

KEYBOARD OVERVIEW (Continued)

Table 1. Scratch Pad RAM Map

Name	RAM Address	Bit position	Bit Name	Function
P0	0			Port 0 Keyboard Column 8 - 15 Scan Output Ports
P1	1			Port 1 Keyboard Column 0 - 7 Scan Output Ports
P2	2	Bit0		Data line for serial data communication with IBM PC
		Bit1		Clock line for serial data communication with IBM PC
		Bit3		PC/XT or PC AT mode selection switch (high=PC AT mode)
		Bit2,4-7		Keybord Row 2, 4-7 Scan Input Ports
P3	3	Bit0,1,3		Keybord Row 0, 1,3 Scan Input Ports
		Bit4		Scroll Lock Indicator output (low=turn on LED)
		Bit5		Num Lock Indicator output (low=turn on LED)
		Bit6		Caps Lock Indicator output (low=turn on LED)
KEY_STATUS1	4	Bit0-2	bounce	3 Bit bounce counter (MAKE_CODE=3, BREAK_CODE=4)
		Bit3	RIGHT_SHIFT	Right Shift key is pressed when the key is detected
		Bit4	LEFT_SHIFT	Left Shift key is pressed when the key is detected
		Bit5	CTRL_CASE	Control key is pressed when the key is detected
		Bit6	ALT_CASE	Alternate key is pressed when the key is detected
		Bit7	NUM_LOCK2	Num Lock status is kept in this bit when the key is detected
KEY_DATA1	5	Bit0-7		1st Key Number in the schematic is kept when a key is detected
KEY_STATUS2	6			Same kind of data as KEY_STATUS1
KEY_DATA2	7			2nd Key Number in the schematic is kept when a key is detected
KEY_STATUS3	8			Same kind of data as KEY_STATUS1
KEY_DATA3	9			3rd Key Number in the schematic is kept when a key is detected
KEY_STATUS4	Ah			Same kind of data as KEY_STATUS1
KEY_DATA4	Bh			4th Key Number in the schematic is kept iwhen a key is detected
KEY_STATUS5	Ch			Same kind of data as KEY_STATUS1
KEY_DATA5	Dh			5th Key Number in the schematic is kept when a key is detected
KEY_STATUS6	Eh			Same kind of data as KEY_STATUS1
KEY_DATA6	Fh			6th Key Number in the schematic is kept when a key is detected
WORK_GRP	10h-17h			WORK_GRP working registers
TYPEMATIC_RATE	19h	Bit0-4	RATE_BITS	Typematic rate bits received from IBM PC
		Bit5-6	DELAY_BITS	Typematic delay bits received from IBM PC
SCAN_CODE_SET	1Ah	Bit0-1		Current Scan Code Set (1=scan code set 1, 2=scs 2, 3=scs 3)
LOCK_STATUS	1Bh	Bit0	SCROLL_LOCK	Current Scroll Lock status (1=scroll lock)
		Bit1	NUM_LOCK	Current Num Lock status (1=Num lock)
		Bit2	CAPS_LOCK	Current Caps Lock status (1=Caps lock)
		Bit3-4	SHIFT_CASE	Current right and left shift status (1=pressed)
		Bit5	CTRL_CASE	Current Control key status (1=pressed)
		Bit6	ALT_CASE	Current Alternate key status (1=pressed)
		Bit7	MAKE_CASE	Temporary flag to inform either Make(=1) or Break code generation
DELAY	1Ch			Delay timer (60=250 ms, 120=500ms, 180=750ms, 240=1000ms)
RATE	1Dh			Typematic rate timer (30.0/sec to 2.0/sec)
TYPEMATIC	1Eh			Current typematic key pointer to address one of KEY DATA (0=no typematic key , KEY_DATA1, 2,3,4,5,6)
FIFO_SIZE	1Fh			FIFO Buffer size (if no buffer valid, FIFO_SIZE=FIFO_GRP)
COM_GRP	20h-27h			COM_GRP working registers
COM_STATUS	28h	Bit0-6		Communication status (0=no communication, 1 to X)
		Bit7	resend_flag	Resend flag (=1)
COMMAND	29h			Current command received from IBM PC
OPTION_BYTE	2Ah			Current option byte received fro IBM PC
COMMAND_BUFFER	2Bh			Communication data includes Acknowledge byte, transmitting data. (if it is 0, then receive mode)
KEY_TYPE	2Ch	Bit0-1		Current key type (make_type_break, make_type, make_break, make)
CURRENT_BUFFER	2Dh			Curren Communication output buffer
MAIN_CONTROL	2Eh	Bit7	enable_scan	Keyboard enable scan flag
FIFO_GRP	2Fh-3Fh		17 bytes buffer	16 byte Keyboard FIFO buffer + 1 byte overrun data buffer
CODE3_GRP	40h-5Fh		32 bytes attribute	Scan Code Set 3 Attribute Data (2bits attribute x 128 keys, scan code=minimum 07, maximum 86h)
Stack Area	60h-7Fh		32 byte	32 byte Stack Area

KEYBOARD CODE GENERATION

The three program modules required to implement keyboard code generation are Keyboard Scanning, Scan Code Generation, and Make/Break/Typematic Timing Control. The modules are serviced by the Timer 1 interrupt. Each of these modules are briefly summarized below and are fully explained in the subsections that follow.

Keyboard Scanning
The keyboard scanning module cuts key bounce for both press and release, configures the First-In-First-Out (FIFO) buffer for a maximum of six keys, and allows time to generate both Make code and Break code.

Make/Break Scan Code Generation
The Make/Break scan code generation module transfers Make scan code and Break scan code into the FIFO buffer via several ROM tables.

Make/Break/Typematic Timing Control
The Make/Break Typematic timing control module checks the current key status for up to six keys; it also sets up the timing for Make Code, Break Code, and Typematic delay and rate.

Keyboard Scanning

The keyboard has three Key Scan Code Sets: Scan Code Set 1, which uses PC/XTs; Scan Code Set 2, which uses PC/ATs; and Scan Code Set 3, which is similar to Scan Code Set 2 except for the different Typematic, Make, and Make/Break defaults (refer to Table 2). Unlike Scan Code Set 2, Scan Code Set 3 is enabled by software. The initial status of the Scan Code Set is specified by the selection switch. Scan Code Set 1 activates when the switch closes; Scan Code Set 2 is selected if it is open.

The keyboard contains 101/102 keypads. All keypads are scanned every 4.17 ms by the keyboard controller. The microcontroller handles a maximum of six keys concurrently by means of the key bounce process and case conditions. (If more than six keys are pressed concurrently, they are ignored.) Quick multiple key passing for the first six keys generate the scan codes.

The key scanning is done from column output 15 toward column output 0 by keeping one of the column outputs at a low level and the remaining outputs at a high level. Whenever a low level output sets on one of the column ports, eight row inputs are tested 20 µs later. The timing chart of keyboard scanning is illustrated in Figure 3.

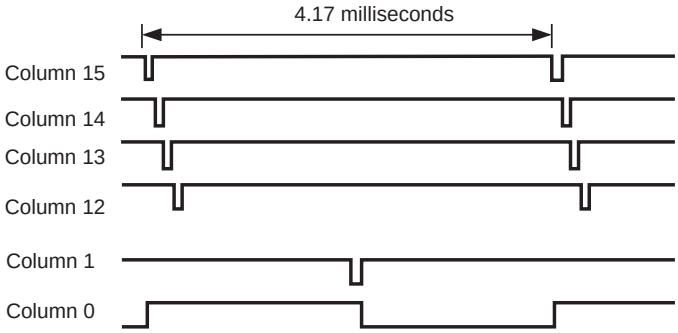


Figure 3. Keyboard Scan Timing

The key bounce ends by detecting the key pressing two times using the scanning method. The detected key converts to a key matrix number through the row and column data appearing in Figure 4a (Z8602/14/E23), Figure 4b (Z8615/C15), and Figure 5. (The key matrix number is the index address of the Scan code, as shown in Table 2.) This process repeats until all six keys fill the buffer. When the key is detected twice, Make code status sets; when the key releases twice, Break code status sets.

Key Detection
The converted key matrix number and the initial bounce data store in one of the empty KEY_DATA and KEY_STATUS registers, respectively, just after detecting the key. When the key is detected once (bounce=2), it is decremented at Make Detection to establish one bounce detection (bounce=1). The loop continues until the key is detected twice. When detected twice, the bounce bit is three (bounce=3) until the generation of the Make scan code. Then the bounce bits change to six (bounce = 6) in the Make/Break scan code generation module.

The key detection loop continues (loop from bounce=5, bounce=7, bounce=6) until two key release detections (from bounce=6 to bounce=5). The bounce bit is set to four (bounce=4) when the key release is detected two times. The number is stored until generation of the Break scan code, then it is reset to zero in the Make/Break module. The first six keys generate the Make scan code and Break scan code when multiple keys are pressed. Concurrently, the rest of the keys are ignored. (Refer to Figure 6, which illustrates the Make and Break scan code process.

KEYBOARD OVERVIEW (Continued)

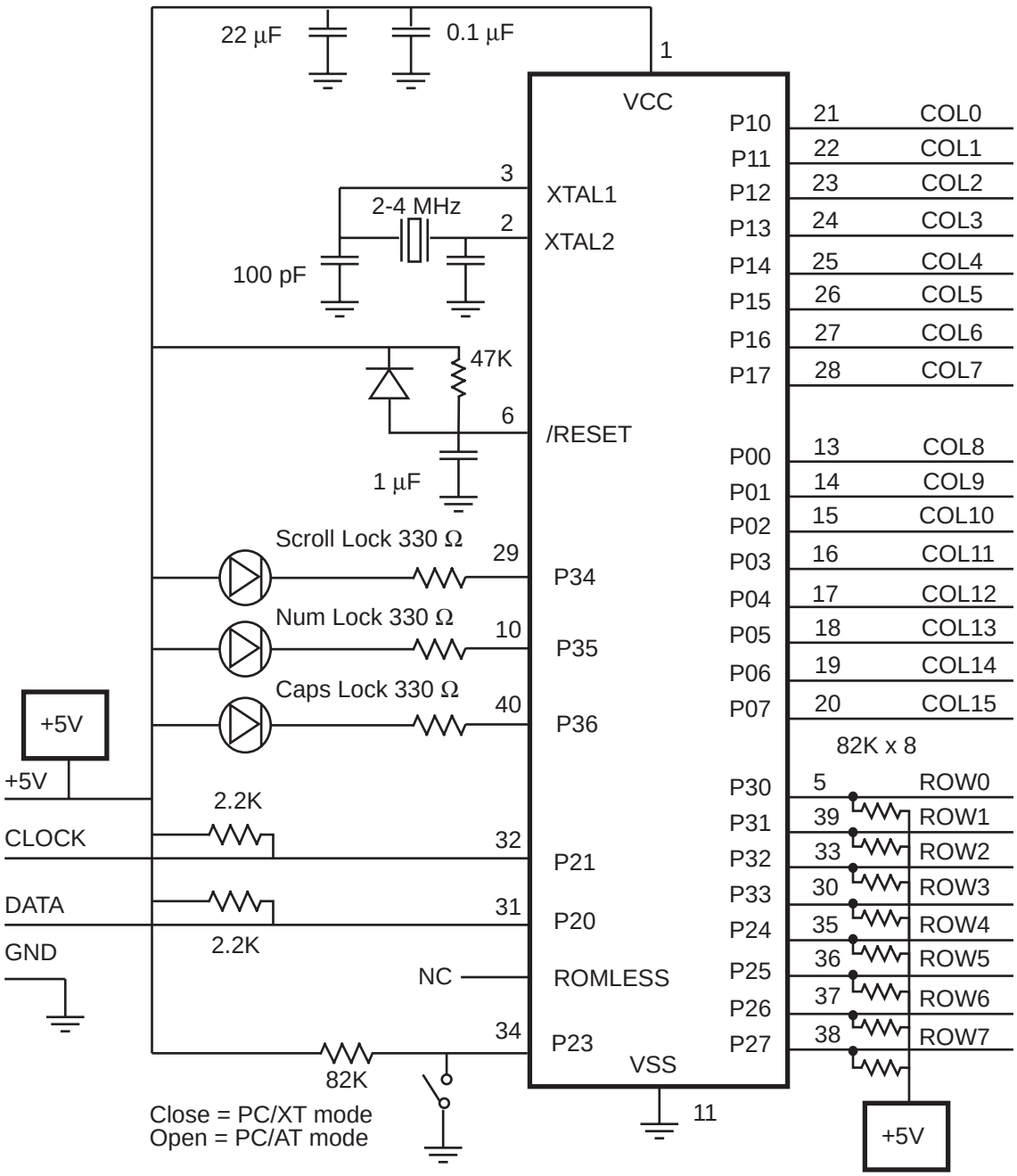


Figure 4a. Z8602/14/E23 Schematic Diagram for the Keyboard

Integration of the Z8615/C15

The Z8615/C15 generation keyboard microcontroller integrates pull-up resistors used on the scan, and Clock and Data lines. Direct drive LED ports are provided along with a cost saving RC oscillator option. The Watch-Dog Timer (WDT) provides added operational reliability in the various environments of the keyboard. The Z8615/C15 provides 4K bytes of ROM and 124 bytes of RAM.

The Z86C15, in particular, is a CMOS part, so less power is consumed in Normal mode. Additionally, the Z86C15 can be put in STOP mode, where the power savings is even more dramatic.

The Figure 4b. represents the integrated diagram of the Z8615/C15 Keyboard implementation. Direct LED drive ports, direct-connect Row/Scan lines, direct-connect PC communication, and the RC oscillator option reduces the component count significantly.

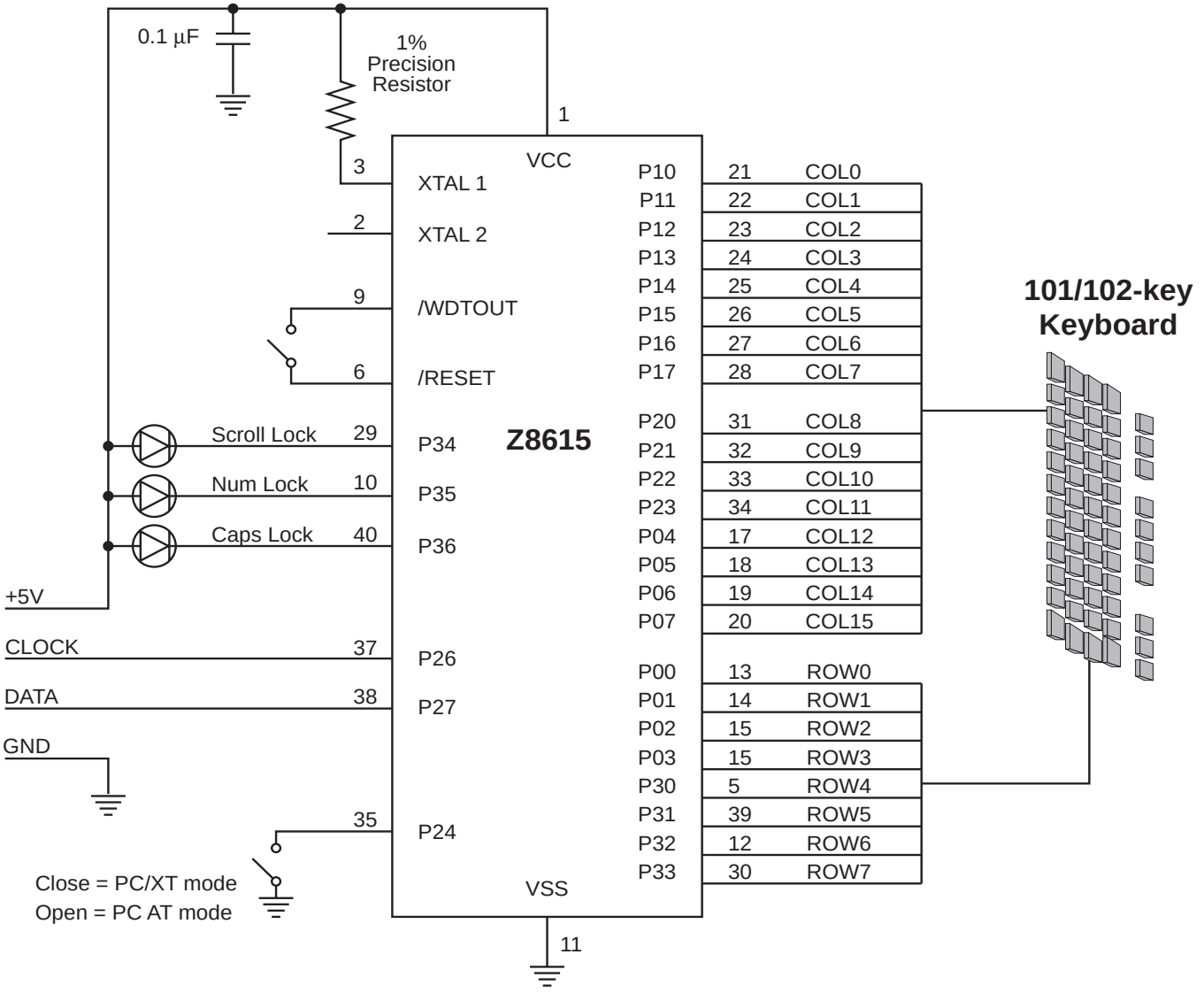


Figure 4b. Z8615/C15 Schematic Diagram for the Keyboard

KEYBOARD CODE GENERATION (Continued)

Table 2. Scan Code Set

Key Number	Scan Code Set 1		Scan Code Set 2	
	Make Code	Break Code	Make Code	Break Code
1	29	A9	0E	F0 0E
2	02	82	16	F0 16
3	03	83	1E	F0 1E
4	04	84	26	F0 26
5	05	85	25	F0 25
6	06	86	2E	F0 2E
7	07	87	36	F0 36
8	08	88	3D	F0 3D
9	09	89	3E	F0 3E
10	0A	8A	46	F0 4E
11	0B	8B	45	F0 45
12	0C	8C	4E	F0 4E
13	0D	8D	55	F0 55
15	0E	8E	66	F0 66
16	0F	8F	0D	F0 0D
17	10	90	15	F0 15
18	11	91	1D	F0 1D
19	12	92	24	F0 24
20	13	93	2D	F0 2D
21	14	94	2C	F0 2C
22	15	95	35	F0 35
23	16	96	3C	F0 3C
24	17	97	43	F0 43
25	18	98	44	F0 44
26	19	99	4D	F0 4D
27	1A	9A	54	F0 54
28	1B	9B	5B	F0 5B
29	2B	AB	5D	F0 5D
30	3A	BA	58	F0 58
31	1E	9E	1C	F0 1C
32	1F	9F	1B	F0 1B
33	20	A0	23	F0 23
34	21	A1	2B	F0 2B
35	22	A2	34	F0 34
36	23	A3	33	F0 33
37	24	A4	3B	F0 3B
38	25	A5	42	F0 42
39	26	A6	4B	F0 4B
40	27	A7	4C	F0 4C
41	28	A8	52	F0 52
42	2B	AB	5D	F0 5D
43	1C	9C	5A	F0 5A
44	2A	AA	12	F0 12
45	56	D6	61	F0 61
46	2C	AC	1A	F0 1A
47	2D	AD	22	F0 22
48	2E	AE	21	F0 21
49	2F	AF	2A	F0 2A
50	30	B0	32	F0 32
51	31	B1	31	F0 31
52	32	B2	3A	F0 3A
53	33	B3	41	F0 41
54	34	B4	49	F0 49
55	35	B5	4A	F0 4A
57	36	B6	59	F0 59
58	1D	9D	14	F0 14
60	38	B8	11	F0 11
61	39	B9	29	F0 29
62	E0 38	E0 B8	E0 11	E0 F0 11
64	E0 1D	E0 9D	E0 14	E0 F0 14
90	45	C5	77	F0 77
91	47	C7	6C	F0 6C
92	4B	CB	6B	F0 6B
93	4F	CF	69	F0 69
96	48	C8	75	F0 75
97	4C	CC	73	F0 73
98	50	D0	72	F0 72
99	52	D2	70	F0 70
100	37	B7	7C	F0 7C
101	49	C9	7D	F0 7D
102	4D	CD	74	F0 74
103	51	D1	7A	F0 7A
104	53	D3	71	F0 71
105	4A	CA	7B	F0 7B
106	4E	CE	79	F0 79
108	E0 1C	E0 9C	E0 5A	E0 F0 5A
110	01	81	76	F0 76
112	3B	BB	05	F0 05
113	3C	BC	06	F0 06
114	3D	BD	04	F0 04
115	3E	BE	0C	F0 0C
116	3F	BF	03	F0 03
117	40	C0	0B	F0 0B
118	41	C1	83	F0 83
119	42	C2	0A	F0 0A
120	43	C3	01	F0 01
121	44	C4	09	F0 09
122	57	D7	78	F0 78
123	58	D8	07	F0 07
125	46	C6	7E	F0 7E

Key Number	Scan Code Set 1		
	Base Case, Shift + Num Lock Make/Break	Shift Case * Make/Break	Num Lock On Make/Break
75	E0 52 / E0 D2	E0 AA E0 52 /E0 D2 E0 2A	E0 2A E0 52 /E0 D2 E0 AA
76	E0 53 / E0 D3	E0 AA E0 53 /E0 D3 E0 2A	E0 2A E0 53 /E0 D3 E0 AA
79	E0 4B / E0 CB	E0 AA E0 4B /E0 CB E0 2A	E0 2A E0 4B /E0 CB E0 AA
80	E0 47 / E0 C7	E0 AA E0 47 /E0 C7 E0 2A	E0 2A E0 47 /E0 C7 E0 AA
81	E0 4F / E0 CF	E0 AA E0 4F /E0 CF E0 2A	E0 2A E0 4F /E0 CF E0 AA
83	E0 48 / E0 C8	E0 AA E0 48 /E0 C8 E0 2A	E0 2A E0 48 /E0 C8 E0 AA
84	E0 50 / E0 D0	E0 AA E0 50 /E0 D0 E0 2A	E0 2A E0 50 /E0 D0 E0 AA
85	E0 49 / E0 C9	E0 AA E0 49 /E0 C9 E0 2A	E0 2A E0 49 /E0 C9 E0 AA
86	E0 51 / E0 D1	E0 AA E0 51 /E0 D1 E0 2A	E0 2A E0 51 /E0 D1 E0 AA
89	E0 4D / E0 CD	E0 AA E0 4D /E0 CD E0 2A	E0 2A E0 4D /E0 CD E0 AA

* If the left shift Key is held down, the AA/2A shift make and break is sent with the other scan codes. If the right Shift key is held down, B6/36 is sent. If both Shift keys are down, both sets of codes are sent with the other scan code.

Table 2. Scan Code Set (Continued)

Scan Code Set 1		
Key No.	Make/Break Code	Shift Case Make/Break *
95	E0 35 / E0 B5	E0 AA E0 35 / E0 B5 E0 2A
*If the left Shift key is held down, the AA/2A shift make and break is sent with the other scan codes. If the right Shift key is held down, B6/36 is sent. If both Shift keys are down, both sets of codes are sent with the other scan code.		

Scan Code Set 1			
Key No.	Make/Break Code	Ctrl, Shift Case Make/Break	Alt Case Make/Break
124	E0 2A E0 37 /E0 B7 E0 AA	E0 37 /E0 B7	54/D4

Scan Code Set 1		
Key No.	Make Code	Ctrl Key Pressed
126	E1 1D 45 E1 9D C5	E0 46 E0 C6
*This key is not typematic. All associated scan codes occur on the make of the key.		

Key Number	Scan Code Set 2		
	Base Case, Shift + Num Lock Make/Break	Shift Case * Make/Break	Num Lock On Make/Break
75	E0 70 / E0 F0 70	E0 F0 12 E0 70 /E0 F0 70 E0 12	E0 12 E0 70 /E0 F0 70 E0 F0 12
76	E0 71 / E0 F0 71	E0 F0 12 E0 71 /E0 F0 71 E0 12	E0 12 E0 71 /E0 F0 71 E0 F0 12
79	E0 6B/E0 F0 6B	E0 F0 12 E0 6B /E0 F0 6B E0 12	E0 12 E0 6B /E0 F0 6B E0 F0 12
80	E0 6C/E0F0 6C	E0 F0 12 E0 6C /E0 F0 6C E0 12	E012 E0 6C /E0 F0 6C E0 F0 12
81	E0 69 / E0 F0 69	E0 F0 12 E0 69 /E0 F0 69 E0 12	E0 12 E0 69 /E0 F0 69 E0 F0 12
83	E0 75 / E0 F0 75	E0 F0 12 E0 75 /E0 F0 75 E0 12	E0 12 E0 75 /E0 F0 75 E0 F0 12
84	E0 72 / E0 F0 72	E0 F0 12 E0 72 /E0 F0 72 E0 12	E0 12 E0 72 /E0 F0 72 E0 F0 12
85	E0 7D/E0 F0 7D	E0 F0 12 E0 7D /E0 F0 7D E0 12	E012 E0 7D /E0 F0 7D E0 F0 12
86	E0 7A/E0 F0 7A	E0 F0 12 E0 7A /E0 F0 7A E0 12	E0 12 E0 7A /E0 F0 7A E0 F0 12
89	E0 74 / E0 F0 74	E0 F0 12 E0 74 /E0 F0 74 E0 12	E0 12 E0 74 /E0 F0 74 E0 F0 12
* If the left shift Key is held down, the F0 12/12 shift make and break is sent with the other scan codes. If the right Shift key is held down, F0 59/59 is sent. If both Shift keys are down, both sets of codes are sent with the other scan code.			

Scan Code Set 2		
Key No.	Make/Break Code	Shift Case Make/Break *
95	E0 4A / E0 F0 4A	E0 F0 12 4A / E0 12 F0 4A
*If the left Shift key is held down, the AA/2A shift make and break is sent with the other scan codes. If the right Shift key is held down, B6/36 is sent. If both Shift keys are down, both sets of codes are sent with the other scan code.		

Scan Code Set 2			
Key No.	Make/Break Code	Ctrl, Shift Case Make/Break	Alt Case Make/Break
124	E0 12 E0 7C /E0 F0 7C E0 F0 12	E0 7C /E0 F0 7C	84/F0 84

Scan Code Set 2		
Key No.	Make Code	Ctrl Key Pressed
126	E1 14 77 E1 F0 14 F0 77	E0 7E E0 F0 7E
*This key is not typematic. All associated scan codes occur on the make of the key.		

Scan Code Set 3							
T=Typematic, M/B=Make/Break, M=Make only							
Key No.	Make Code	Break Code	Default Stratus	Key No.	Make Code	Break Code	Default Status
1	0E	F0 0E	T	51	31	F0 31	T
2	16	F0 16	T	52	3A	F0 3A	T
3	1E	F0 1E	T	53	41	F0 41	T
4	26	F0 26	T	54	49	F0 49	T
5	25	F0 25	T	55	4A	F0 4A	T
6	2E	F0 2E	T	57	59	F0 59	M/B
7	36	F0 36	T	58	11	F0 11	M/B
8	3D	F0 3D	T	60	19	F0 19	M/B
9	3E	F0 3E	T	61	29	F0 29	T
10	46	F0 46	T	62	39	F0 39	M
11	45	F0 45	T	64	58	F0 58	M
12	4E	F0 4E	T	75	67	F0 67	M
13	55	F0 55	T	76	64	F064	T
15	66	F0 66	T	79	61	F0 61	T
16	0D	F0 0D	T	80	6E	F0 6E	M
17	15	F0 15	T	81	65	F0 65	M
18	1D	F0 1D	T	83	63	F0 63	T
19	24	F0 24	T	84	60	F0 60	T
20	2D	F0 2D	T	85	6F	F0 6F	M
21	2C	F0 2C	T	86	6D	F0 6D	M
22	35	F0 35	T	89	6A	F0 6A	T
23	3C	F0 3C	T	90	76	F0 76	M
24	43	F0 43	T	91	6C	F0 6C	M
25	44	F0 44	T	92	6B	F0 6B	M
26	4D	F0 4D	T	93	69	F0 69	M
27	54	F0 54	T	95	77	F0 77	M
28	5B	F0 5B	T	96	75	F0 75	M
29	5C	F0 5C	T	97	73	F0 73	M
30	14	F0 14	M/B	98	72	F0 72	M
31	1C	F0 1C	T	99	70	F0 70	M
32	1B	F0 1B	T	100	7E	F0 7E	M
33	23	F0 23	T	101	7D	F0 7D	M
34	2B	F0 2B	T	102	74	F0 74	M
35	34	F0 34	T	103	7A	F0 7A	M
36	33	F0 33	T	104	71	F0 71	M
37	3B	F0 3B	T	105	84	F0 84	M
38	42	F0 42	T	106	7C	F0 7C	T
39	4B	F0 4B	T	108	79	F0 79	M
40	4C	F0 4C	T	110	08	F0 08	M
41	52	F0 52	T	112	07	F0 07	M
42	53	F0 53	T	113	0F	F0 0F	M
43	5A	F0 5A	T	114	17	F0 17	M
44	12	F0 12	M/B	115	1F	F0 1F	M
45	13	F0 13	T	116	27	F0 27	M
46	1A	F0 1A	T	117	2F	F0 2F	M
47	22	F0 22	T	118	37	F0 37	M
48	21	F0 21	T	119	3F	F0 3F	M
49	2A	F0 2A	T	120	47	F0 47	M
50	32	F0 32	T	121	4F	F0 4F	M
				122	56	F0 56	M
				123	5E	F0 5E	M
				124	57	F0 57	M
				125	5F	F0 5F	M
				126	62	F0 62	M

KEYBOARD CODE GENERATION (Continued)

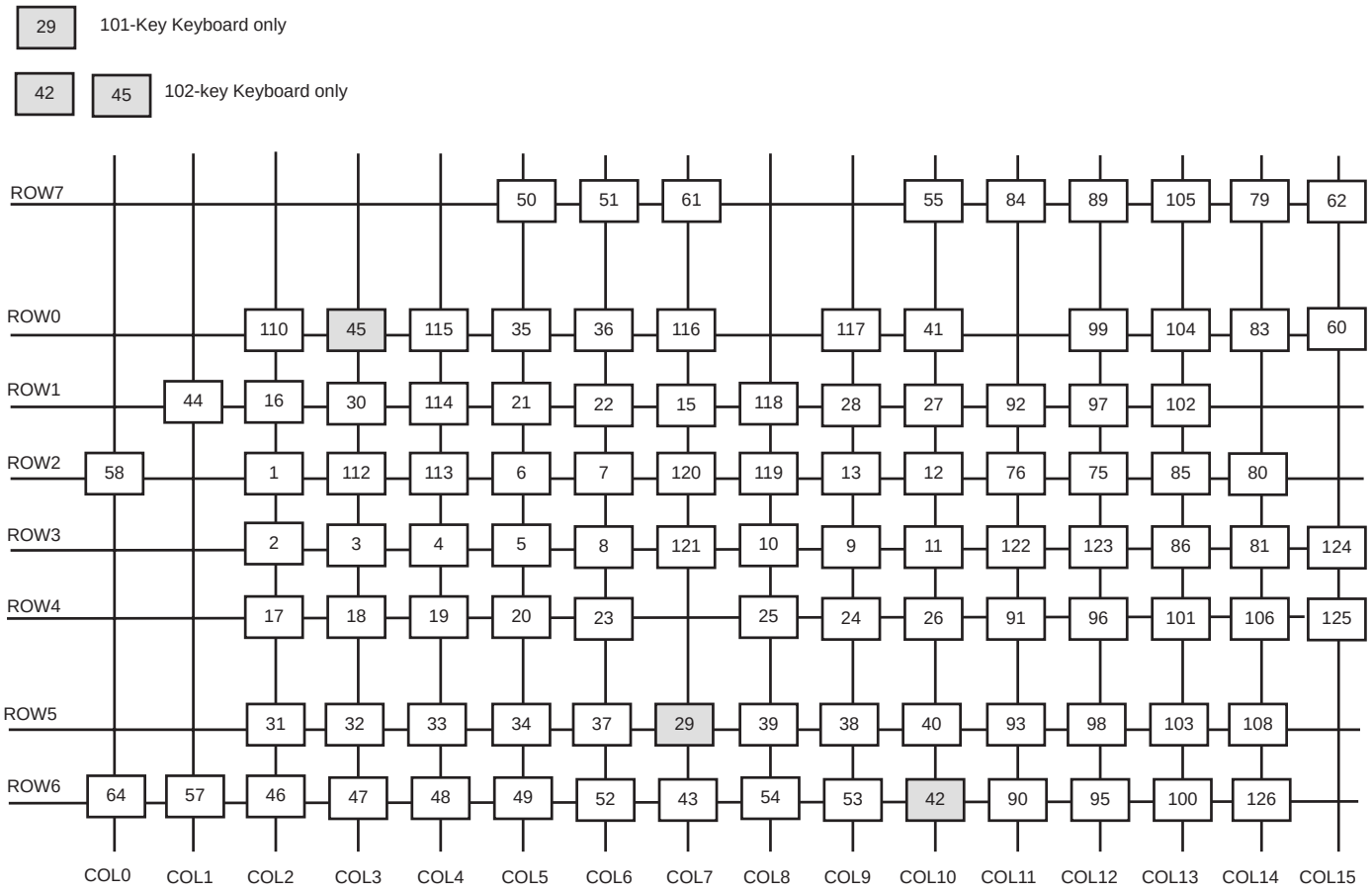


Figure 5. 101/102 Keyboard Matrix

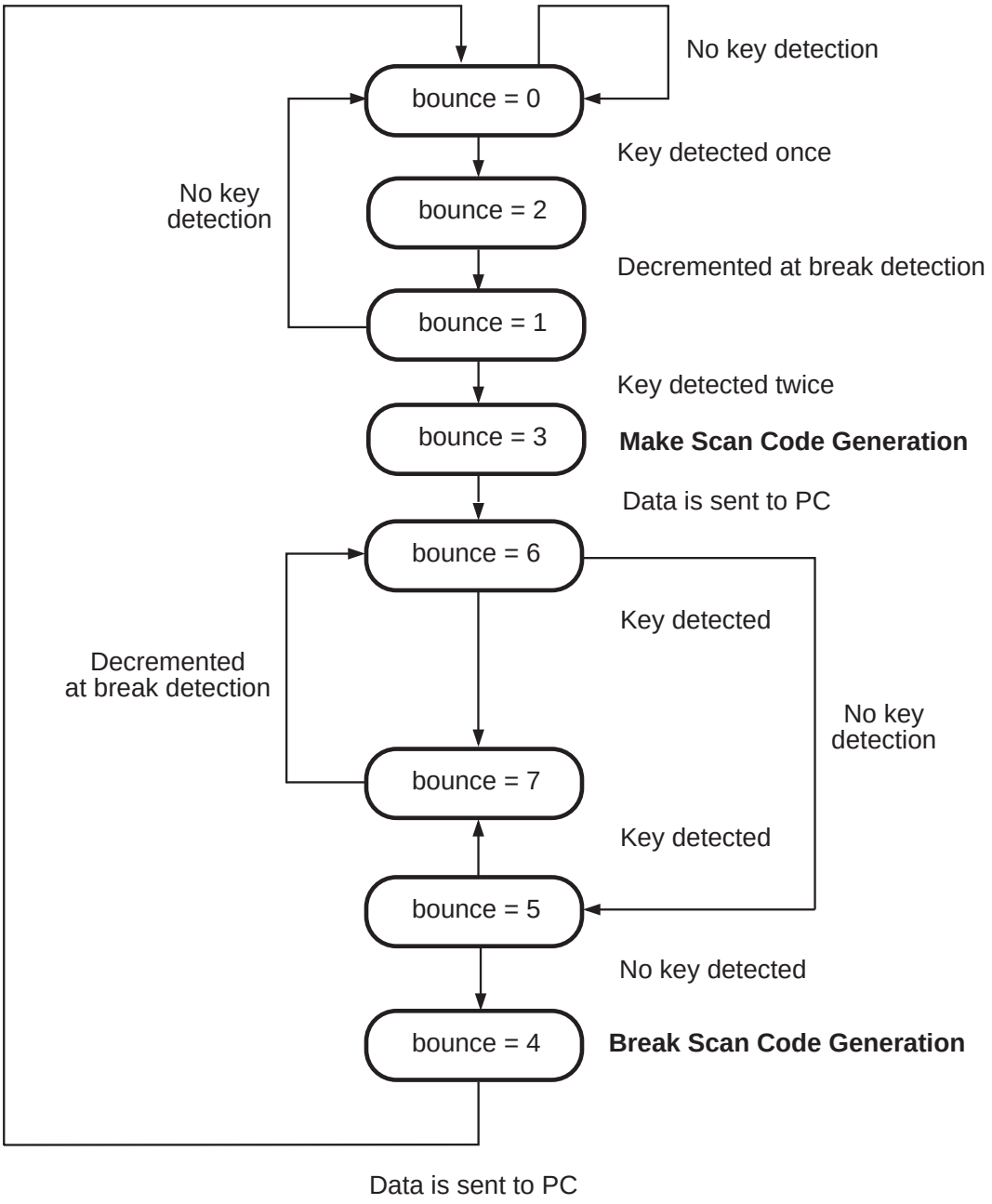


Figure 6. Key Detection

KEYBOARD CODE GENERATION (Continued)

Keyboard Buffer

Six pairs of working registers (KEY_STATUS and KEY_DATA) are manipulated in the keyboard scanning program. The key bounce is handled in the bounce counter of KEY_STATUS (Figure 7).

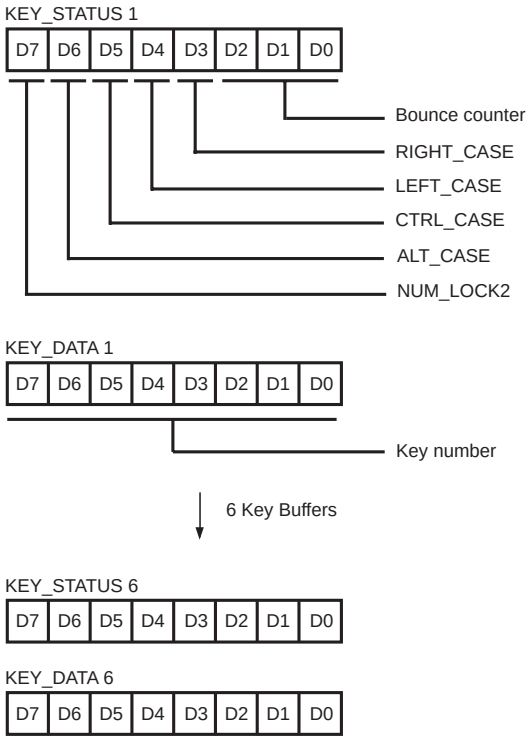


Figure 7. Six Pairs of Key Buffers

FIFO Dynamics

The overrun code appears at the last position of the buffer. When the full 16 bytes of scan code are in the FIFO buffer and a further scan code appears, the overrun code goes into the 17th byte of the last occupied buffer register. This produces an audible "beep" warning. The FIFO buffer pointer points to the working register plus one. Therefore, if there are no scan codes available in the buffer, the pointer is the same address as the top of the FIFO buffer. The keyboard buffer only contains the scan codes and does not include any commands from the PC or acknowledges any data.

Make/Break Scan Code Generation

Each keypad sends multiple data bytes to the PC under the control of the keyboard controller. There are two kinds of Scan codes: Make scan code and Break scan code. The Make scan code is sent to the PC when the key is pressed; Break scan code is sent when the key releases. Additionally, these keys are Typematic, which means that when a key is pressed and held down, the keyboard sends the Make scan code with a particular delay and rate. The typematic delay and rate are specified by the F3H command sent by the PC.

Macro Description

The system has one macro command to expand one byte of Make code to multiple scan codes. The macro is data_gen. The macro structure appears in Figure 8.

After getting one byte of Make code, the data_gen macro command expands to multiple scan codes. The data_gen macro contains a total number of bytes generated (lower four bits) minus the offset value. This offset value addresses a byte that is XORed with a Make-Byte code and other generated data bytes.

The previously kept "index address" indicates the entry point of the data_gen macro command when the data is generated for Break code. If it is generated for Make code, then the index address is decremented by one and the ROM data at the address is read to specify the entry pointer for the Make code generation.

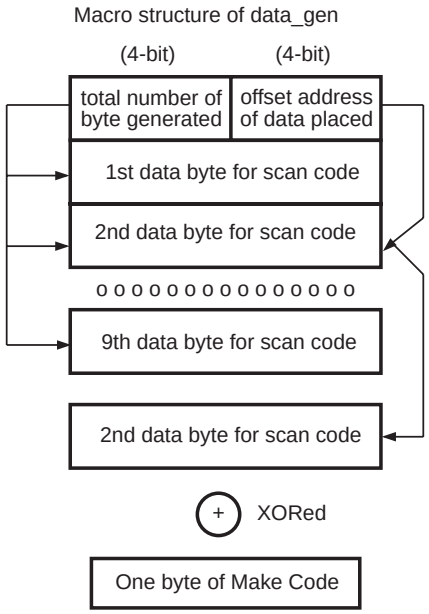


Figure 8. Macro Structure of Data_Attr and Data_Gen

Multiple Scan Codes

The multiple scan codes are stored into a 16-byte First-In-First-Out (FIFO) buffer until the PC is ready to receive them. A buffer-overrun condition occurs when more than 16 bytes remain in the FIFO buffer. This FIFO uses 16 general-purpose registers.

Generating Multiple Scan Codes

The following steps explain how multiple scan codes are generated.

1. Get data for the total number of bytes generated (lower four bits of first byte).

2. Temporarily store the offset address, which shows the position to be XORed with a Byte-Make code.

3. Store all the bytes from the ROM table to the FIFO buffer while incrementing the FIFO_SIZE and the ROM address pointer.

4. Subtract the offset address from the temporary register containing the same value of FIFO_SIZE. Change the FIFO data by taking the XOR with the Byte-Make code.

Multiple Scan Code Organization

The multiple scan code generation starts from conversion of a key matrix number that points at the Key Matrix Table. The Key Matrix Table has the index address offset of the Scan Code Table. The Scan Code Set Table contains all Make Code keys. The Scan Code Table Map organization is shown in Table 2 and Figure 10.

Typematic Attribute

The Scan Code Set 3 is similar to Scan Code Set 2. It is the scan_code set flag (1=Scan Code 1, 2 = Scan Code Set 2, and 3 = Scan Code Set 3), which specifies the Scan Code Table to use. If the scan_code_ set flag is 3 (Scan Code Set 3), its data checks for typematic *attributes. The typematic attribute for scan codes (addresses 7-83_H) store in the scratch pad RAM. The four typematic attributes include; Typematic, Make/Break, Make and Typematic/Make/Break. To select one of the attributes, two bits decode to determine which one of the four attributes to use.

Phantom Keys

If the key matrix number is none of the valid keys, it is designated the "Phantom" key. The Phantom key is the result of multiple keypads depressed concurrently. The Phantom key is zero in the Keyboard Matrix. If the microcontroller sees a zero from the Key Matrix Table, it sends no code to the PC. (Data conversion for Phantom key is ignored.)

Make/Break/Typematic Timing Control

Make Code Timing Control

The Make scan code is generated when the bounce bits of KEY_STATUS is three. Then, the delay timer is calculated by using the TYPEMATIC_RATE register (see Figure 9) and sets to one of the periods: 250 ms, 500 ms, 750 ms, or 1 second. The keys for the left/right shift case, control case, and alternate case test to hold the current key configuration case. The Num Lock, Caps Lock, and Scroll Lock status cases are stored in three of the six KEY_STATUS registers to keep the same pair of Make and Break codes for each key. The Make scan code is stored in the FIFO keyboard buffer. The lock status transfers back from the PC soon after the key scan code transmits.

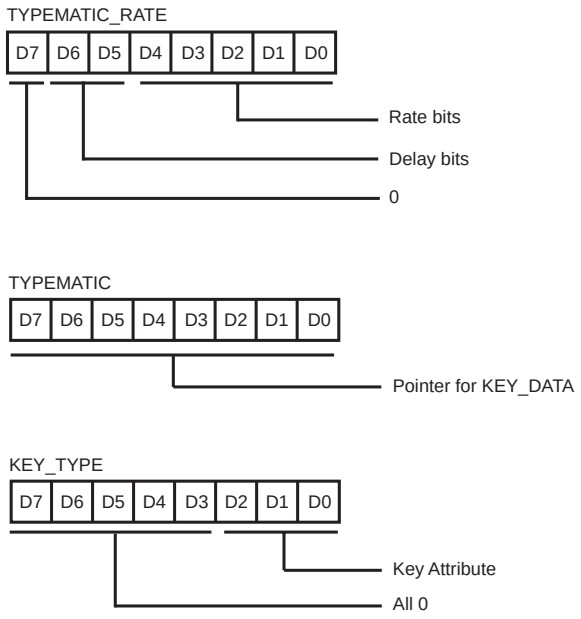


Figure 9. Key Control Registers

KEYBOARD CODE GENERATION (Continued)

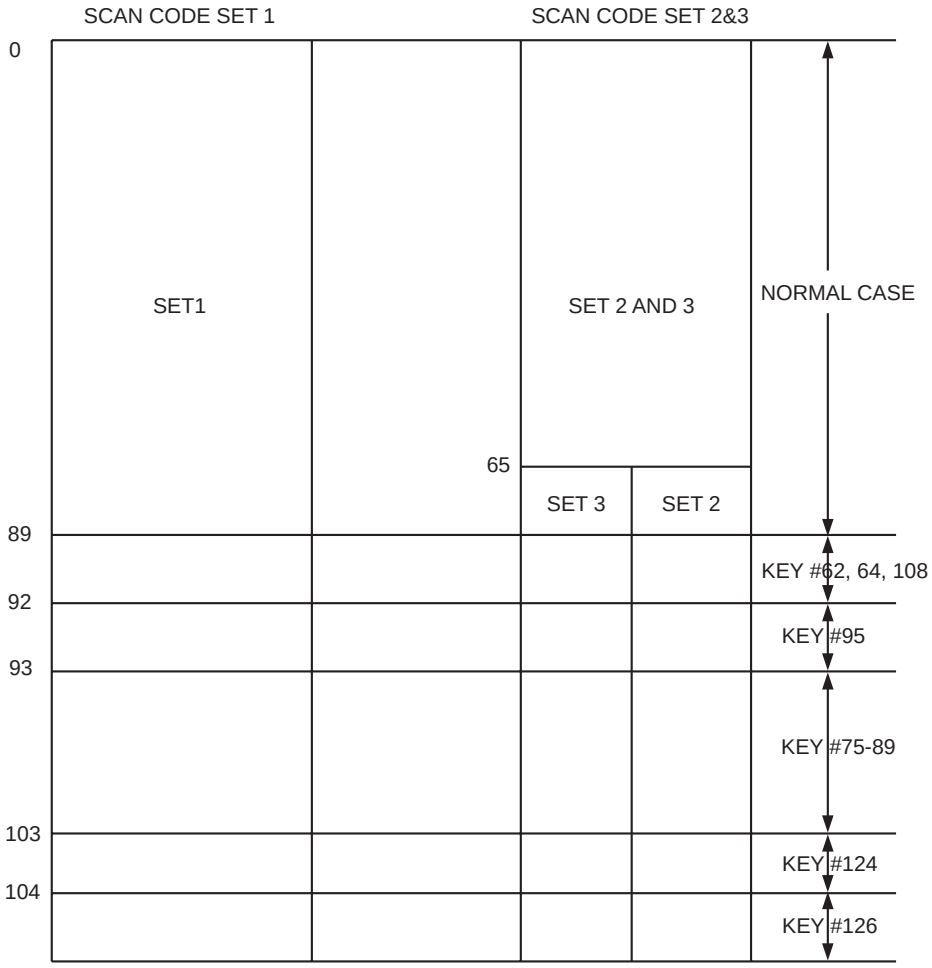


Figure 10. Scan Code Table Map

Make/Break/Typematic Timing Control

The current pointer of KEY_DATA stores in the TYPEMATIC pointer register to generate Make scan code when typematic time-out occurs. When the Make code is generated, two flag bits in the KEY_TYPE register are set to determine the key attributes: Make-Typematic-Break (00), Make-Typematic (01), Make-Break (10) and Make only (11).

Break Code Timing Control

Code is generated when the bounce bits of KEY_STATUS are set to four, which means the key is released. The TYPEMATIC pointer register is reset when the current typematic key releases. The case key tests and the case flag is reset when released. Then, Break code is stored in the FIFO keyboard buffer and KEY_DATA plus KEY_STATUS reset to show an empty key. This procedure repeats for the six KEY_STATUS registers.

Typematic Code Timing Control

To generate typematic code, the key attribute tests to be sure the key is typematic. If the current TYPEMATIC pointer is not zero, the typematic process is carried out. The delay timer decrements every 4.17 ms. When it reaches 0, there is Make code generation. Once the delay timer sets to 0, the rate timer takes over the typematic code generation, which continues whenever the rate timer decrements to 0. The rate timer than reloads from the TYPEMATIC_RATE register (see Figure 11).

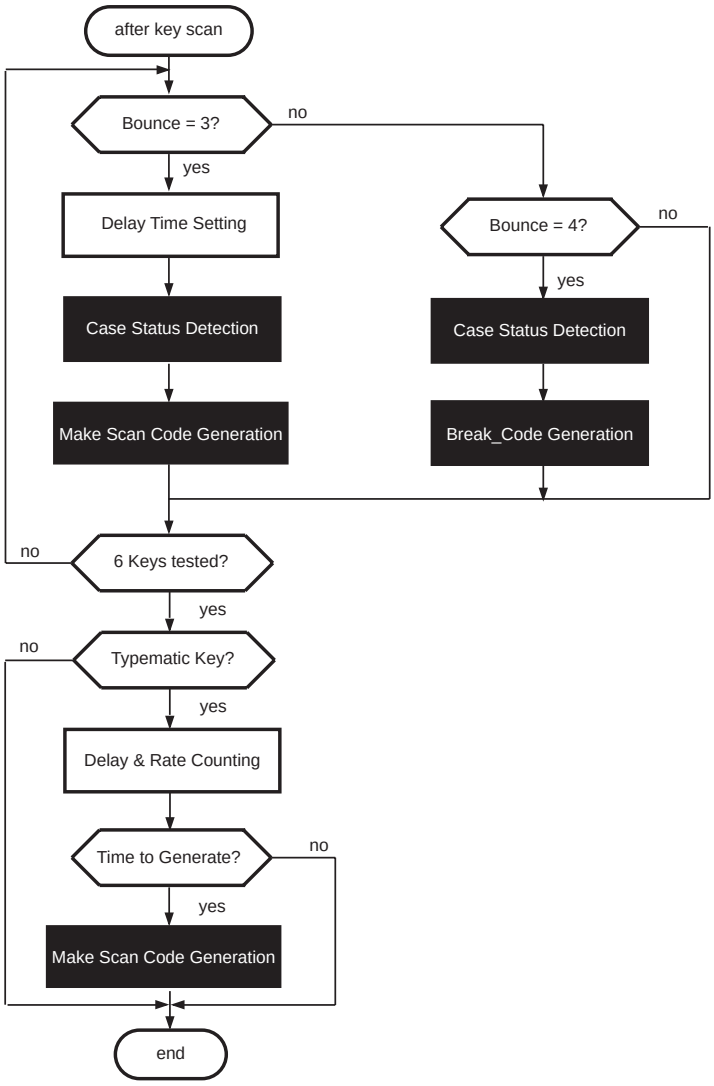


Figure 11. Flow Chart to Make/Break/Typematic Timing

KEYBOARD MICROCONTROLLER CONTROL AND INTERFACE

The following subsections define the parameters and explain the control and interfacing of the microcontroller (located in the keyboard) to the PC.

Pin Descriptions and Assignments

Figure 12 and Table 3 show the pin assignment and pin descriptions, respectively. Figure 13 shows the communications format between the PC and the keyboard.

Communication Between the PC and Keyboard

Before the keyboard microcontroller drives the keyboard Clock and Data lines, it sets both lines to a high level to check the current line status. The three line status modes between the PC and the keyboard are the following:

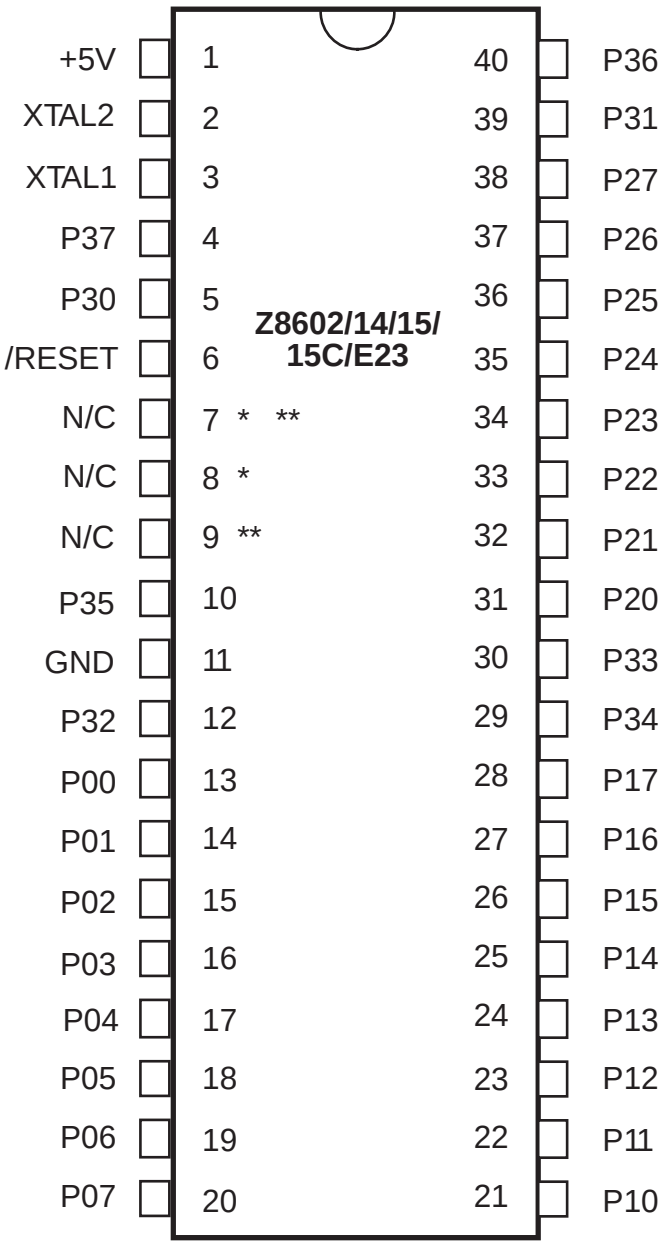
- Two-way communication between Keyboard and PC if both Clock and Data lines are high
- PC sends to Keyboard if Data line is low
- PC inhibits communication if Clock is low

The PC always drives the Clock and inhibits the communication by forcing the Clock line to a low level (inactive level), which keeps the keyboard from sending any data packet and from generating Clock pulses during this stage.

Once the Clock line releases high (active level), the keyboard has to check the Data line. When the Data line is inactive, the PC requests to send the serial data to the keyboard. The keyboard has to send serial Clock streams to receive the data packet from the PC. When both data and Clock lines are active, the keyboard sends the scan codes to the PC at any time.

Serial Data Stream

The serial data bit stream consists of 11 bits, which include a start bit, 8 data bits, odd parity bit, and a stop bit. When bit stream sends data to the PC, all the data bits are guaranteed while the Clock line is low. The data changes during a high level of the Clock line. When the bit stream arrives from the PC, the data fetches at the leading edge of the Clock. After the stop bit detects high, the KBC forces the Data line to a low level for one bit period. The start bit is always low and the stop bit is high. The 8-bit data transmits from the least significant bit (LSB). The odd parity bit means that the number of 1s for the data bit and the parity bit must be odd all the time.



* Pin 7&8 are used for test purposes and must be floating pins.

**For the Z8615/15C: Pin 7 = AGnd
**For the Z8615/15C: Pin 9 = Watch-Dog Timer (WDT) Output

Figure 12. Pin Assignments

KEYBOARD MICROCONTROLLER CONTROL AND INTERFACE (Continued)

Table 3. Pin Assignments

Pin	No.	I/O	Pin Description
+5V	1	IN	+5V Power Supply
GND	11	IN	Common Ground
XTAL2	2	OUT	8 MHz Ceramic Resonator
XTAL1	3	IN	8 MHz Ceramic Resonator
RESET	6	IN	Reset Input (active Low)
N/C	7*	N/C	No Connection
N/C	8	N/C	No Connection
N/C	9*	N/C	No Connection
ROMless	12	IN, PUR	ROMless Selection (=GND)
P00	13	OUT,OD	Column 8 Low Output Scan Line
P01	14	OUT,OD	Column 9 Low Output Scan Line
P02	15	OUT,OD	Column 10 Low Output Scan Line
P03	16	OUT,OD	Column 11 Low Output Scan Line
P04	17	OUT,OD	Column 12 Low Output Scan Line
P05	18	OUT,OD	Column 13 Low Output Scan Line
P06	19	OUT,OD	Column 14 Low Output Scan Line
P07	20	OUT,OD	Column 15 Low Output Scan Line
P10	21	OUT,OD	Column 0 Low Output Scan Line
P11	22	OUT,OD	Column 1 Low Output Scan Line
P12	23	OUT,OD	Column 2 Low Output Scan Line
P13	24	OUT,OD	Column 3 Low Output Scan Line
P14	25	OUT,OD	Column 4 Low Output Scan Line
P15	26	OUT,OD	Column 5 Low Output Scan Line
P16	27	OUT,OD	Column 6 Low Output Scan Line
P17	28	OUT,OD	Column 7 Low Output Scan Line
P20	31	IN/OUT,OD	Data line for PC Communication
P21	32	IN/OUT,OD	Clock line for PC Communication
P22	33	IN	Row 2 Input Scan Line
P23	34	IN	PC/XT or AT (=high) Selection
P24	35	IN	Row 4 input Scan Line
P25	36	IN	Row 5 input Scan Line
P26	37	IN	Row 6 input Scan Line
P27	38	IN	Row 7 input Scan Line
P30	5	IN	Row 0 input Scan Line
P31	39	IN	Row 1 input Scan Line
P33	30	IN	Row 3 input Scan Line
P34	29	OUT	Scroll Lock Indicator
P35	10	OUT	Num Lock Indicator
P36	40	OUT	Caps Lock Indicator
P37	4	N/C	No Connection

Notes:
IN = Input Port
PUR = Pull-up Resistor
OUT = Output Port
OD = Open-Drain Output
N/C = No Connection
* For the Z8615/C15 only: Pin7 = AGnd; Pin 9 = WDT Output.

Serial Data Communication from PC to Keyboard

Three kinds of data are handled between the keyboard and the PC:

- Command and Acknowledge
- Optional Data
- Key Scan Code from the Keyboard FIFO Buffer

The command reception has the highest priority of the data communication. A new command always overrides to the old command even during communication. The PC starts the command transfer by lowering the Data line. Then, the microcontroller sends eleven Clock pulses to receive the serial data packet from the PC. When the data arrives, the microcontroller sends an acknowledge (0FAh). The optional data arrives or departs after sending the acknowledge. The key scan code only sends from the keyboard FIFO buffer when no data is coming from the PC.

Transmission Mode	Clock	Data
Transmission Inhibit	Low	High
System to send	High	Low
Keyboard to send	High	High

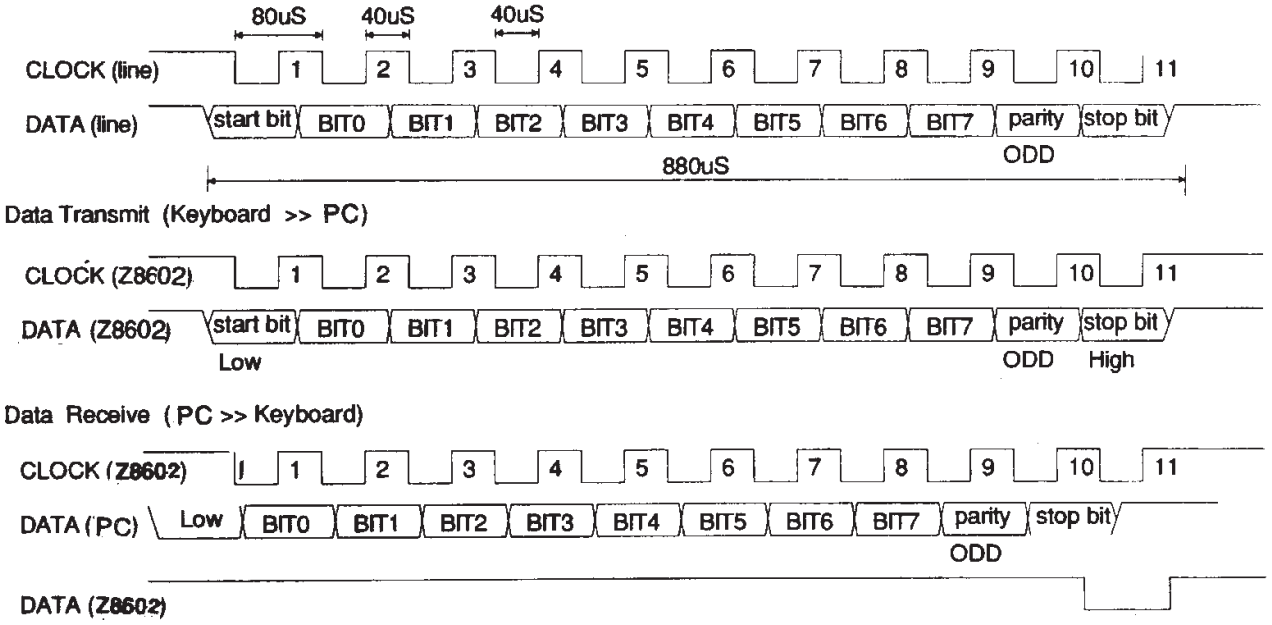


Figure 13. Data Communication Format Timing Between Microcontroller/PC

KEYBOARD MICROCONTROLLER CONTROL AND INTERFACE (Continued)

Table 4. Commands Between the Keyboard and the PC

Name	Hex Values	Function
STATUS_IND	ED,ACK,XX,ACK	Set/Reset Lock Status indicators
ECHO	EE,EE (=ACK)	Echo Command
ALT_SCAN	F0,ACK,XX,ACK	Select Alternate Scan Codes
TYPE_RATE DELAY	F3,ACK,XX,ACK	Set Typematic Rate/Delay
ENABLE	F4,ACK	Enable Key Scanning
DISABLE	F5,ACK	Default Disable
SET_DEFAULT	F6,ACK	Set Default Value
ALL_MAKE_TYPE	F7,ACK	Set All Keys - Typematic
ALL_MAKE_BREAK	F8,ACK	Sent All Keys - Make/Break
ALL MAKE	F9,ACK	Set All Keys - Make
ALL_M_T_B	FA,ACK	Set All Keys Typematic/Make/Break
KEY_MAKE_TYPE	FB,ACK,XX,ACK	Set Key Type - Typematic
KEY_MAKE_BREAK	FC,ACK,XX,ACK	Set Key Type - Make/Break
KEY MAKE	FD,ACK,XX,ACK	Set Key Type - Make
RESEND	FE	Resend Command
RESET	FF,ACK,YY	Reset Command
READ ID	F2 ACK, AB,83	Read ID Command

Notes:
ACK = Acknowledge Data to PC (0FAh)
XX = Received Data from PC.
YY = Result of Basic Assurance Test
AB 83 = ID Number

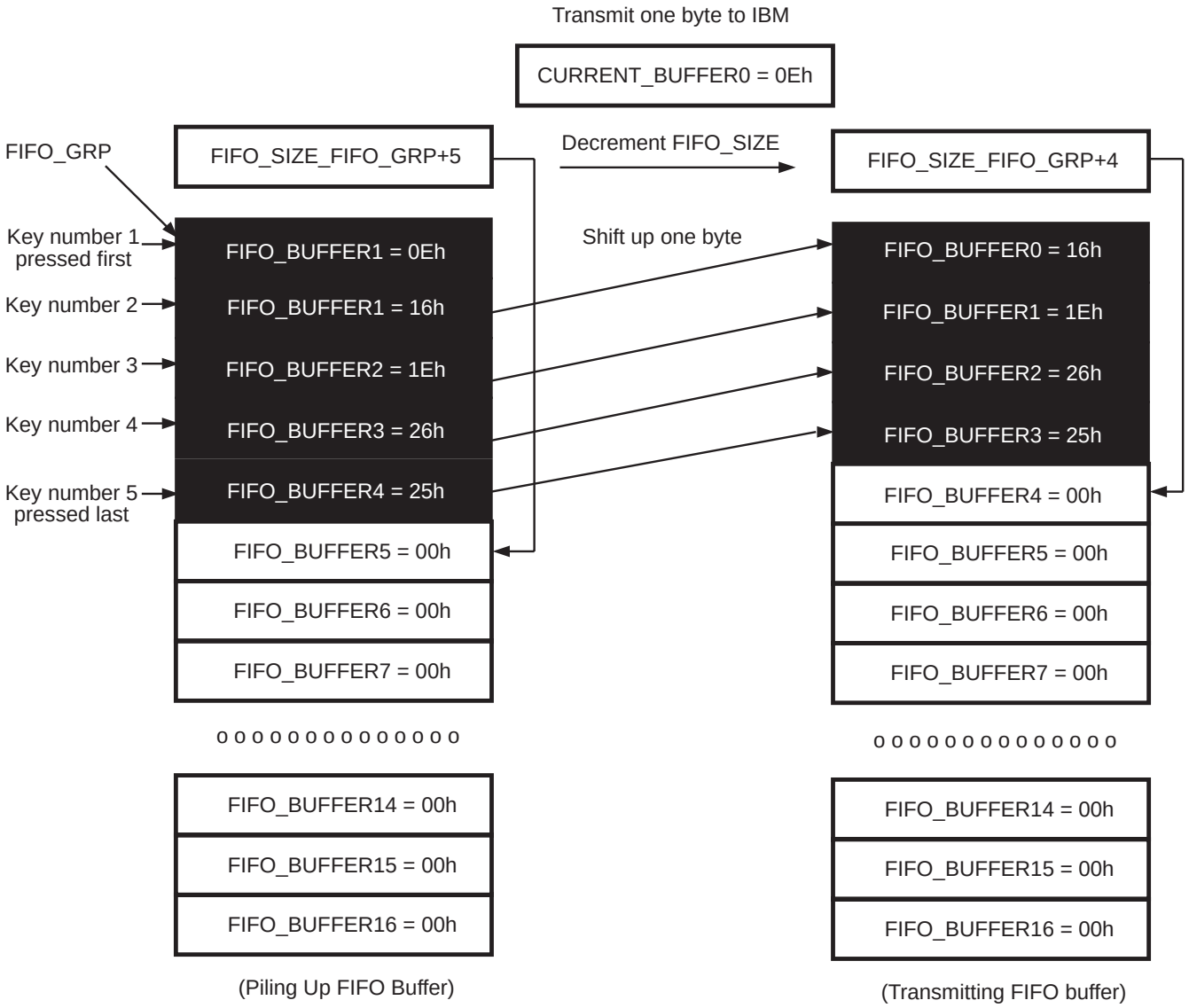


Figure 14. Stacking and Sending Status of Keyboard FIFO Buffer

KEYBOARD MICROCONTROLLER CONTROL AND INTERFACE (Continued)

Command Communication

To do the command communication, use the macro command com_gen. The com_gen macro contains three parameters which ensures the number of bytes handled after command reception, command data, and Jump address at the end of the communication (see Figure 15).

Whenever Data line goes low (from PC), the microcontroller receives the data by driving the Clock line eleven times. The received data is always checked whether or not it is a new command. The command compares to the Command data defined in com_gen macro.

If it is a new command, then it is stored in a COMMAND register. The first parameter of com_gen saves to the COM_STATUS register. The COM_STATUS shows how many bytes are handled in the current command. The acknowledge data (0FAh) sets to COMMAND_BUFFER and departs in the next Timer0 interrupt.

After that transmission, COM_STATUS decrements by one and tests for zero. If it is zero, the communication is over and the program executes by getting a jump address from the com_gen macro table.

If it is not zero, the communication remains active. The new command buffer sets to 0 unless the current command is READ_ID. The command buffer at 0 means data receive; non-zero means a data transmission to the PC. After the proper data transfer, the acknowledge data is sent. Now, each command is executable.

Basic Serial Data Input and Output Drivers

This module includes a parity generation for data transmission; 11 bits of data transmission with detection of line contention and 11 bits of data reception with an 11th bit acknowledge pulse.

Table 5 shows working registers specifying the initial values used to handle serial data transfers.

Table 5. Serial Data Transfer Working Registers

Register	Function
Serial Data Output	
CF (carry flag)	0 to set low start bit
serial data hi	bit7-1=1 and bit0 = odd parity bit
serial data lo	8 bits data to be transmitted
bit count	11 = number of bit transmitted
serial bit	temporary register for P2 I/O port
transmit	0ffh to specify transmit mode
com_delay	to set up 80 µsec/bit timing
P2	P2.1 = Clock to make pulse, P2.0=Data to transmit data
Serial Data Input	
CF (carry flag)	1 to set low high output for input mode
serial data hi	0ffh to receive 8 bits data
serial data lo	0ffh to receive 3 bits data
bit_count	11 = number of bits received
serial bit	temporary register for P2 I/O port
transmit	0 to specify receive mode
com_delay	to set up 80 µsec/bit timing
P2	P2.1 = Clock to make pulse, P2.0=Data to receive

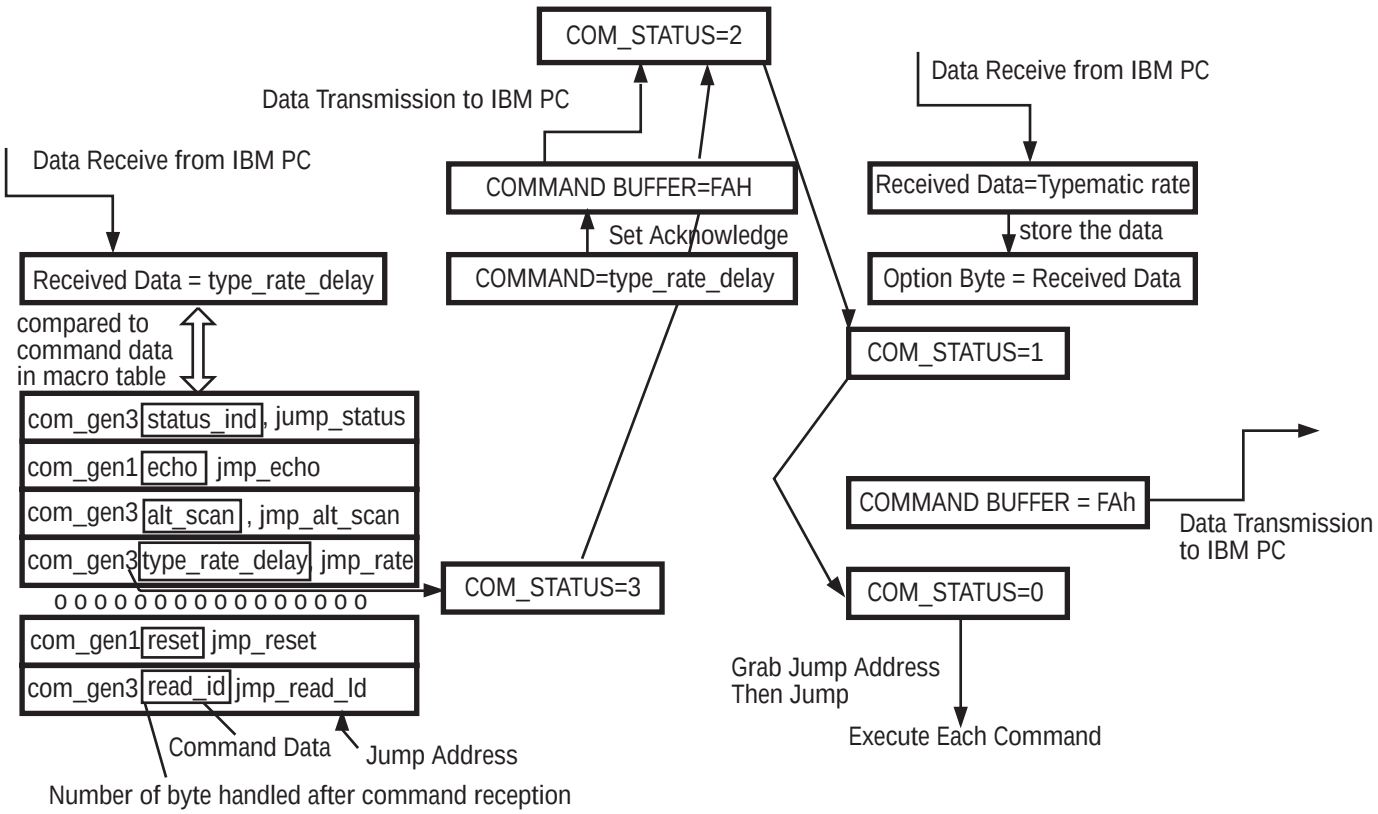


Figure 15. Register Manipulation in Command Communication

KEYBOARD MICROCONTROLLER CONTROL AND INTERFACE (Continued)

After receiving all 11 bits, the serial_data_hi and serial_data_lo shift right five times to set up 8 bits of data into serial_data_lo. Now, parity is in bit 0 of serial_data_hi and is checked by the subroutine of parity_gen (see Figure 16).

The period of one data bit is 80 µs; the data must change when the Clock pulse is high. In fact, the switching is done 20 µs after the leading edge of the Clock pulse.

Transmit Mode

In the second mode, line contention is always detected by the Clock line during a high level. If the Clock line goes low, the PC drives the line. If line contention appears before the 10th bit transmission, the system immediately quits the process and sets both Data and Clock lines to high. If detection occurs after the 10th bit, sending continues until complete.

Receive Mode

In the Receive mode, the 10th bit tests for high. If high, the PC sends a low level for the 11th bit period to show an acknowledge. This means successful data reception from the PC. Otherwise, the PC sends multiple Clock pulses until it receives the correct stop bit.

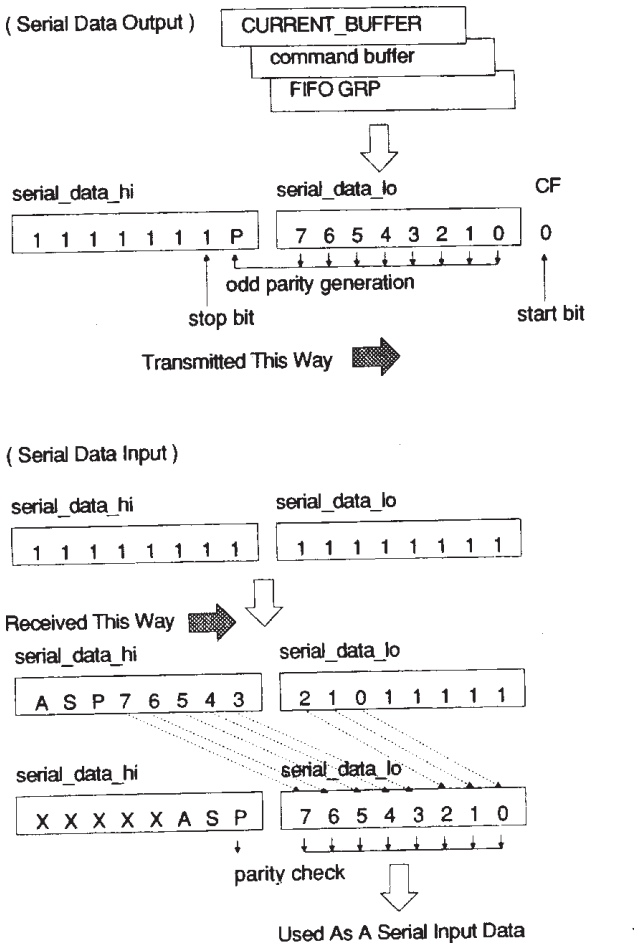


Figure 16. Register Manipulation of the Serial Data Buffer

Zilog, Inc. 210 East Hacienda Ave.
Campbell, CA 95008-6600
Telephone (408) 370-8000
Telex 910-338-7621
FAX 408 370-8056
Internet: http://www.zilog.com