MICROCHIP

TB056

Demonstrating the `Set_Report` Request With a PS/2® to USB Keyboard Translator Example

Author: Reston Condit
Company: Microchip Technology Inc.

INTRODUCTION

This Technical Brief details the translation of a PS/2 keyboard to a USB keyboard using the PIC16C745/765. Although this is a fairly simple application for the PIC16C745/765, it accomplishes the intended purpose of this brief: to provide an in-depth look at the `Set_Report` Request.

Note: This Technical Brief is the third in a series of five technical briefs. This series is meant to familiarize developers with USB. For the best understanding of USB, read the briefs in order: TB054, TB055, TB056, TB057, TB058

THE SET_REPORT REQUEST

The `Set_Report` request is a HID specific command. It was the only provision in Version 1.0 of the USB Specification that could send data from the host to a peripheral. This later changed with Version 1.1 and the introduction of the interrupt OUT transfer. All low speed devices created before the introduction of Version 1.1 had to use the `Set_Report` request for host to device communication. The limitation of this is that `Set_Report` communicates over endpoint 0 (EP0). Endpoint 0 is the control endpoint. In other words, EP0 is the avenue through which the device and host send each other commands, so any outputted data from the host has to share bandwidth with administrative commands sent back and forth between the host and device. Developers wanted a way to create a dedicated OUT endpoint with one of the two endpoints available to them, EP1 and EP2. As a result, Version 1.1 of the USB specification included a provision for an interrupt OUT transfer.

Note: IN and OUT are with respect to the host.

There are many instances where the `Set_Report` request is useful in today's USB devices. In devices where the maximum amount of bandwidth is needed for device-to-host transactions both EP1 and EP2 can be set as IN endpoints. If the same device must receive occasional data packets from the host, it is better to use the `Set_Report` request to send that data from the host to the device than to change either EP1 or EP2 to an OUT endpoint. The key word here is "occasional." If the device requires that the host constantly provide it with a stream of data, then a dedicated OUT endpoint should be created. The keyboard is an example of a peripheral in which the host occasionally sends it data, namely, the status of the keyboard LEDs (Caps Lock, Num Lock, and Scroll Lock LEDs are the most common LEDs). This data is sent from the host to the device with the `Set_Report` request only when the user presses one of the corresponding keys for these LEDs (i.e., the Caps Lock key.)

Note: The `Get_Report` request, not talked about in this brief, is similar to `Set_Report` only it is used for sending data from the device to host via Endpoint 0.

DESCRIPTORS

Figure 1 shows the USB keyboard report descriptor. There are two Output items in this report descriptor. These items describe a byte of data that will be sent from the host to the device comprised of five LED status bits and three bits of padding. Because this report descriptor is associated with an IN endpoint (specified by the endpoint descriptor), the host knows all Output items describe data that will be sent to the device via a `Set_Report`. The remainder of the descriptor describes the format that data will be sent from the keyboard to the host. The output data should be ignored when trying to envision what the device-to-host report format looks like. See Figure 2 for the USB keyboard-to-host data format.

FIGURE 1: USB KEYBOARD REPORT DESCRIPTOR

0x05, 0x01	usage page (generic desktop	*Choose the usage page "keyboard" is on*
0x09, 0x06	usage (keyboard)	*Device is a keyboard*
0xA1, 0x01	collection (application)	*This collection comprises all the data words*
0x05, 0x07	usage page (key codes)	*Choose the key code usage page*
0x19, 0xE0	usage minimum (224)	*Choose key codes 224 to 231 which are modifier keys*
0x29, 0xE7	usage maximum (231)	*(left and right alt, shift, ctrl, and win)*
0x15, 0x00	logical minimum (0)	*Each of these eight key codes will report ranging in*
0x25, 0x01	logical maximum (1)	*value from zero to one*
0x75, 0x01	report size (1)	*Assign each of these keys a 1-bit report*
0x95, 0x08	report count (8)	*Report eight times*
0x81, 0x02	input (data, variable, absolute)	*The defined byte above is an IN transaction*
0x95, 0x01	report count (1)	
0x75, 0x08	report size (8)	*Report eight bits one time*
0x81, 0x01	input (constant)	*Input the byte just described as a constant*
0x95, 0x05	report count (5)	
0x75, 0x01	report size (1)	*Report five bits one time*
0x05, 0x08	usage page (page# for LEDs)	*Choose LED usage page*
0x19, 0x01	usage minimum (1)	
0x29, 0x05	usage maximum (5)	*Define five LEDs*
0x91, 0x02	output (data, variable, absolute)	*The defined bits above are an OUT transaction*
0x95, 0x01	report count (1)	
0x75, 0x03	report size (3)	
0x91, 0x01	output (constant)	*Three bit padding for the OUT transaction*
0x95, 0x06	report count (6)	
0x75, 0x08	report size (8)	*Report six bytes*
0x15, 0x00	logical minimum (0)	
0x25, 0x65	logical maximum (101)	*The byte values can range from 0 to 101*
0x05, 0x07	usage page (key codes)	*Change usage page to key codes*
0x19, 0x00	usage minimum (0)	
0x29, 0x65	usage maximum (101)	*Select key code range of 0 to 101*
0x81, 0x00	input (data, array)	*Input the above six bytes*
0xC0	end collection	*End application collection*

FIGURE 2: USB KEYBOARD DATA FORMAT

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0
Key Code 6	Key Code 5	Key Code 4	Key Code 3	Key Code 2	Key Code 1	Reserved	Modifier Keys
MSB							LSB

IMPLEMENTATION

HARDWARE

The PS/2 port is a 6-pin DIN. Only four pins are used:

- Ground
- Power
- Clock
- Data

The power and ground pins are tied directly to VDD and VSS. The Clock and Data pins are connected to RC0 and RC1, respectively, via current limiting resistors. The Clock is driven by the PS/2 keyboard. Figure 3 shows the complete system.

SOFTWARE

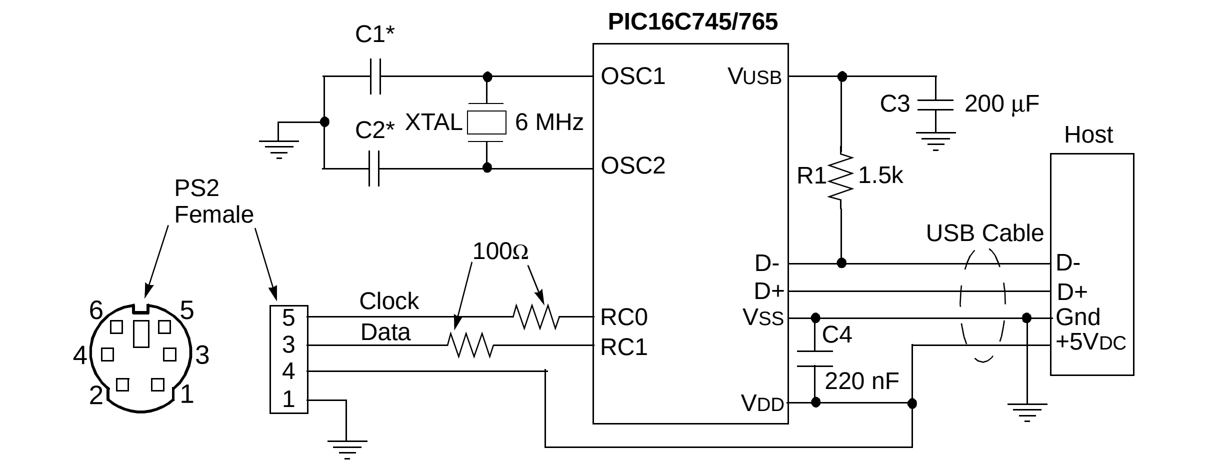
PS/2 DATA FORMAT

Before explaining how PS/2 keyboard data is translated to USB, it is necessary to touch upon the PS/2 data format. Data is sent via PS/2 one byte at a time regardless of direction, host-to-device or vice-versa. The data has the following form:

- START bit (always low)
- Data byte (Least Significant bit to Most Significant bit)
- PARITY bit (high for an even number of high bits in the data byte and low for an odd number)
- STOP bit (always high)

In the case of host-to-device communication, the STOP bit is immediately followed by an ACK bit (low), which is sent by the device to the host. The bits are read on the falling edge of the clock for device-to-host communication and on the rising edge for host-to-device communication. In the idle state, the clock and data line are held high by the device. See Figures 4 and 5 for device-to-host and host-to-device communication, respectively.

FIGURE 3: PS/2 TO USB KEYBOARD TRANSLATOR HARDWARE DIAGRAM



***Note:** C1 and C2 values selected according to crystal load capacitance.

FIGURE 4: DEVICE-TO-HOST COMMUNICATION (DATA BIT READ ON FALLING EDGE OF CLOCK)

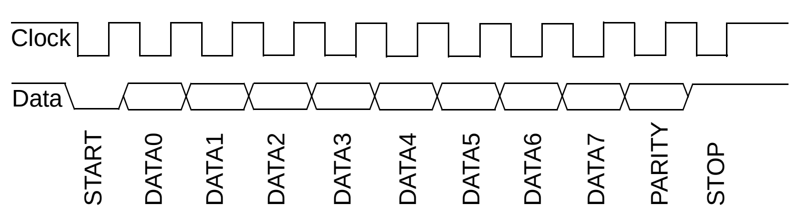
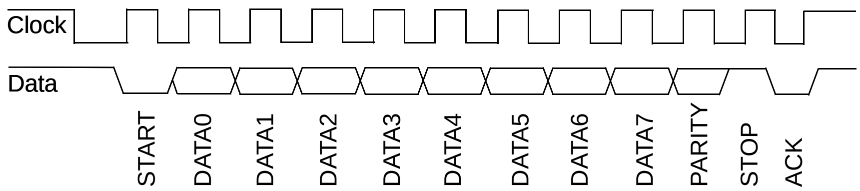


FIGURE 5: HOST-TO-DEVICE COMMUNICATION (DATA BIT READ ON FALLING EDGE OF CLOCK)



PS/2 KEYBOARD REPORT FORMAT

The PS/2 keyboard data report format is summarized for every key in Appendix A. Make codes are the byte or bytes that the PS/2 keyboard sends to the host when a certain key is pressed. Break codes are the bytes that the PS/2 keyboard sends when the user releases a key. If the user does not release a specific key for several hundreds of milliseconds, the make code will be sent repeatedly until the user releases the key. At this point, the break code is sent.

INTERRUPT ROUTINE

The translator firmware is entirely interrupt-driven (with the exception of sending the data via USB to the host). An interrupt is generated when the PS/2 START bit is received, at which time the firmware will begin its receive routine. In addition to this interrupt, every 683 μs a timer overflow interrupts the main program and implements one state of the keyboard state machine. This state machine handles sending bytes to (and translating bytes received from) the PS/2 device automatically. All of this is done in the background while the main program runs in the foreground. The only operation that the main program implements is sending keyboard data to the PC via USB.

TRANSLATION TO USB

Incoming PS/2 keyboard data can be one to eight bytes in length, depending on the button that is pressed. No pattern or mathematical expression can be used to convert incoming PS/2 data to a USB key code. USB key codes are one byte in length. There is one USB key code for every key on the keyboard. A lengthy lookup table, found in file `table_kb.asm`, is used to implement the translation from PS/2 to USB key codes. The USB key codes are listed in Section 9 of the USB HID Usage Tables (see References).

MAKING THE KEYBOARD LEDS WORK

As mentioned earlier, the state of the LEDs on a USB keyboard is passed from the host to the device via the `Set_Report` request. The `Set_Report` request will be transferring the data specified by the Output items in the report descriptor listed in Figure 1. The data consists of one byte, in which, the first five bits represent the LED status. This data is sent from the host to the PICmicro® MCU only when the host receives a key code that modifies the state of a LED, such as the CAPS LOCK key code.

The microcontroller firmware services the `Set_Report` request. When the `TOK_DNE` interrupt is generated by a Set_Report request, the `ServiceUSBInt` routine directs the transaction to be handled by the `HidSetReport` routine. `HidSetReport` copies the Set_Report data from the EP0 OUT buffer to the EP1 OUT buffer. This is how the default USB support firmware treats a Set_Report request -- it makes a Set_Report look like an interrupt EP1 OUT transfer. A developer using the support firmware needs only to decide how to service the interrupt EP1 OUT transfer. For the keyboard example, this trick was not utilized. The keyboard state machine simply reads the EP0 OUT buffer to find out the LED states. Special function register `BD0OAL` contains the address of this buffer. `BD0OBC`, also a special function register, specifies the number of bytes received. The LED status report comprises one byte so only the first byte of the buffer needs to be read in order to determine the state of the keyboard LEDs. Once the status is known, this information is relayed to the keyboard via PS/2.

CONCLUSION

The `Set_Report` request is useful in low speed USB applications where the maximum amount of bandwidth available is required for IN transactions, yet it is necessary to occasionally support OUT transactions. In addition to providing an example of the `Set_Report` request, the firmware included with this technical brief provides ready-made routines for communicating via PS/2.

MEMORY USAGE

In the PIC16C765, the following memory was used

Data Memory: 50 Bytes

Program Memory: 2.1K Bytes

REFERENCES

1. *USB Specification*, Version 1.1: Chapter 9 (located at www.usb.org)
2. *Device Class Definition for Human Interface Devices* (located at www.usb.org)
3. *HID Usage Tables* (located at www.usb.org)
4. *USB Firmware User's Guide* (located in USB Support Firmware zip file at www.microchip.com)
5. *USB Complete, Second Edition***,** Jan Axelson; Lakeview Research, 2001 (www.lvr.com)
6. PS/2 Mouse/Keyboard Protocol, Adam Chapweske,

   http://panda.cs.ndsu.nodak.edu/~achapwes/PICmicro/PS2/ps2.htm
7. TB054: *An Introduction to USB Descriptors with a Gameport to USB Gamepad Translator*
8. TB055: *PS/2 to USB Mouse Translator*
9. TB057: *USB Combination Devices Demonstrated by a Combination Mouse and Gamepad device*
10. TB058: *Demonstrating the Soft Detach Function with a PS/2 to USB Translator Example*

APPENDIX A: PS/2 KEYCODES

Key	Make	Break	Key	Make	Break	Key	Make	Break
A	1C	F0,1C	9	46	F0,46	[	54	F0,54
B	32	F0,32	‘	0E	F0,0E	INSERT	E0,70	E0,F0,70
C	21	F0,21	-	4E	F0,4E	HOME	E0,6C	E0,F0,6C
D	23	F0,23	=	55	F0,55	PG UP	E0,7D	E0,F0,7D
E	24	F0,24	\	5D	F0,5D	DELETE	E0,71	E0,F0,71
F	2B	F0,2B	BKSP	66	F0,6	END	E0,69	E0,F0,69
G	34	F0,34	SPACE	29	F0,29	PG DN	E0,7A	E0,F0,7A
H	33	F0,33	TAB	0D	F0,0D	U ARROW	E0,75	E0,F0,75
I	43	F0,43	CAPS	58	F0,58	L ARROW	E0,6B	E0,F0,6B
J	3B	F0,3B	L SHFT	12	F0,12	D ARROW	E0,72	E0,F0,72
K	42	F0,42	L CTRL	14	F0,14	R ARROW	E0,74	E0,F0,74
L	4B	F0,4B	L WIN	E0,1F	E0,F0,1F	NUM	77	F0,77
M	3A	F0,3A	L ALT	11	F0,11	KP /	E0,4A	F0,4A
N	31	F0,31	R SHFT	59	F0,59	KP *	7C	F0,7C
O	44	F0,44	R CTRL	E0,14	E0,F0,14	KP -	7B	F0,7B
P	4D	F0,4D	R WIN	E0,27	E0,F0,27	KP +	79	F0,79
Q	15	F0,15	R ALT	E0,11	E0,F0,11	KP EN	E0,5A	F0,5A
R	2D	F0,2D	APPS	E0,2F	E0,F0,2F	KP .	71	F0,71
S	1B	F0,1B	ENTER	5A	F0,5A	KP 0	70	F0,70
T	2C	F0,2C	ESC	76	F0,76	KP 1	69	F0,69
U	3C	F0,3C	F1	5	F0,05	KP 2	72	F0,72
V	2A	F0,2A	F2	6	F0,06	KP 3	7A	F0,7A
W	1D	F0,1D	F3	4	F0,04	KP 4	6B	F0,6B
X	22	F0,22	F4	0C	F0,0C	KP 5	73	F0,73
Y	35	F0,35	F5	3	F0,03	KP 6	74	F0,74
Z	1A	F0,1A	F6	0B	F0,0B	KP 7	6C	F0,6C
0	45	F0,45	F7	83	F0,83	KP 8	75	F0,75
1	16	F0,16	F8	0A	F0,0A	KP 9	7D	F0,7D
2	1E	F0,1E	F9	1	F0,01	]	5B	F0,5B
3	26	F0,26	F10	9	F0,09	;	4C	F0,4C
4	25	F0,25	F11	78	F0,78	’	52	F0,52
5	2E	F0,2E	F12	7	F0,07	,	41	F0,41
6	36	F0,36		E1,14,77		.	49	F0,49
7	3D	F0,3D	PAUSE	E1,F0,14	NONE	/	4A	F0,4A
8	3E	F0,3E		F0,77		PRNT	E0,12,	E0,F0,7C
SCROLL	7E	F0,7E				SCRN	E0,7C	E0,F0,12

APPENDIX B: SOURCE CODE

Due to the length of the source code for the PS/2 to USB Keyboard Translator example, the source code is available separately. The complete source code is available as a single WinZip archive file, `tb056sc.zip`, which may be downloaded from the Microchip corporate Web site at:

www.microchip.com

NOTES:

Note the following details of the code protection feature on PICmicro® MCUs.

- The PICmicro family meets the specifications contained in the Microchip Data Sheet.
- Microchip believes that its family of PICmicro microcontrollers is one of the most secure products of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the PICmicro microcontroller in a manner outside the operating specifications contained in the data sheet. The person doing so may be engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as “unbreakable”.
- Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our product.

If you have any further questions about this matter, please contact the local sales office nearest to you.

Trademarks

Printed on recycled paper.

Microchip received QS-9000 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona in July 1999. The Company’s quality system processes and procedures are QS-9000 compliant for its PICmicro® 8-bit MCUs, KEELOQ® code hopping devices, Serial EEPROMs and microperipheral products. In addition, Microchip’s quality system for the design and manufacture of development systems is ISO 9001 certified.

WORLDWIDE SALES AND SERVICE

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200 Fax: 480-792-7277
Technical Support: 480-792-7627
Web Address: http://www.microchip.com

Rocky Mountain
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7966 Fax: 480-792-7456

Atlanta
500 Sugar Mill Road, Suite 200B
Atlanta, GA 30350
Tel: 770-640-0034 Fax: 770-640-0307

Boston
2 Lan Drive, Suite 120
Westford, MA 01886
Tel: 978-692-3848 Fax: 978-692-3821

Chicago
333 Pierce Road, Suite 180
Itasca, IL 60143
Tel: 630-285-0071 Fax: 630-285-0075

Dallas
4570 Westgrove Drive, Suite 160
Addison, TX 75001
Tel: 972-818-7423 Fax: 972-818-2924

Detroit
Tri-Atria Office Building
32255 Northwestern Highway, Suite 190
Farmington Hills, MI 48334
Tel: 248-538-2250 Fax: 248-538-2260

Kokomo
2767 S. Albright Road
Kokomo, Indiana 46902
Tel: 765-864-8360 Fax: 765-864-8387

Los Angeles
18201 Von Karman, Suite 1090
Irvine, CA 92612
Tel: 949-263-1888 Fax: 949-263-1338

New York
150 Motor Parkway, Suite 202
Hauppauge, NY 11788
Tel: 631-273-5305 Fax: 631-273-5335

San Jose
Microchip Technology Inc.
2107 North First Street, Suite 590
San Jose, CA 95131
Tel: 408-436-7950 Fax: 408-436-7955

Toronto
6285 Northam Drive, Suite 108
Mississauga, Ontario L4V 1X5, Canada
Tel: 905-673-0699 Fax: 905-673-6509

ASIA/PACIFIC

Australia
Microchip Technology Australia Pty Ltd
Suite 22, 41 Rawson Street
Epping 2121, NSW
Australia
Tel: 61-2-9868-6733 Fax: 61-2-9868-6755

China - Beijing
Microchip Technology Consulting (Shanghai) Co., Ltd., Beijing Liaison Office
Unit 915
Bei Hai Wan Tai Bldg.
No. 6 Chaoyangmen Beidajie
Beijing, 100027, No. China
Tel: 86-10-85282100 Fax: 86-10-85282104

China - Chengdu
Microchip Technology Consulting (Shanghai) Co., Ltd., Chengdu Liaison Office
Rm. 2401, 24th Floor,
Ming Xing Financial Tower
No. 88 TIDU Street
Chengdu 610016, China
Tel: 86-28-6766200 Fax: 86-28-6766599

China - Fuzhou
Microchip Technology Consulting (Shanghai) Co., Ltd., Fuzhou Liaison Office
Unit 28F, World Trade Plaza
No. 71 Wusi Road
Fuzhou 350001, China
Tel: 86-591-7503506 Fax: 86-591-7503521

China - Shanghai
Microchip Technology Consulting (Shanghai) Co., Ltd.
Room 701, Bldg. B
Far East International Plaza
No. 317 Xian Xia Road
Shanghai, 200051
Tel: 86-21-6275-5700 Fax: 86-21-6275-5060

China - Shenzhen
Microchip Technology Consulting (Shanghai) Co., Ltd., Shenzhen Liaison Office
Rm. 1315, 13/F, Shenzhen Kerry Centre,
Renminnan Lu
Shenzhen 518001, China
Tel: 86-755-2350361 Fax: 86-755-2366086

Hong Kong
Microchip Technology Hongkong Ltd.
Unit 901-6, Tower 2, Metroplaza
223 Hing Fong Road
Kwai Fong, N.T., Hong Kong
Tel: 852-2401-1200 Fax: 852-2401-3431

India
Microchip Technology Inc.
India Liaison Office
Divyasree Chambers
1 Floor, Wing A (A3/A4)
No. 11, O'Shaugnessey Road
Bangalore, 560 025, India
Tel: 91-80-2290061 Fax: 91-80-2290062

Japan
Microchip Technology Japan K.K.
Benex S-1 6F
3-18-20, Shinyokohama
Kohoku-Ku, Yokohama-shi
Kanagawa, 222-0033, Japan
Tel: 81-45-471- 6166 Fax: 81-45-471-6122

Korea
Microchip Technology Korea
168-1, Youngbo Bldg. 3 Floor
Samsung-Dong, Kangnam-Ku
Seoul, Korea 135-882
Tel: 82-2-554-7200 Fax: 82-2-558-5934

Singapore
Microchip Technology Singapore Pte Ltd.
200 Middle Road
#07-02 Prime Centre
Singapore, 188980
Tel: 65-334-8870 Fax: 65-334-8850

Taiwan
Microchip Technology Taiwan
11F-3, No. 207
Tung Hua North Road
Taipei, 105, Taiwan
Tel: 886-2-2717-7175 Fax: 886-2-2545-0139

EUROPE

Denmark
Microchip Technology Nordic ApS
Regus Business Centre
Lautrup hoj 1-3
Ballerup DK-2750 Denmark
Tel: 45 4420 9895 Fax: 45 4420 9910

France
Microchip Technology SARL
Parc d'Activite du Moulin de Massy
43 Rue du Saule Trapu
Batiment A - ler Etage
91300 Massy, France
Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79

Germany
Microchip Technology GmbH
Gustav-Heinemann Ring 125
D-81739 Munich, Germany
Tel: 49-89-627-144 0 Fax: 49-89-627-144-44

Italy
Microchip Technology SRL
Centro Direzionale Colleoni
Palazzo Taurus 1 V. Le Colleoni 1
20041 Agrate Brianza
Milan, Italy
Tel: 39-039-65791-1 Fax: 39-039-6899883

United Kingdom
Arizona Microchip Technology Ltd.
505 Eskdale Road
Winnersh Triangle
Wokingham
Berkshire, England RG41 5TU
Tel: 44 118 921 5869 Fax: 44-118 921-5820

01/18/02