INTRODUCTION

In embedded controller applications, it is often desirable to provide a means to digitize analog signals. The MCP3201 12-bit Analog-to-Digital (A/D) Converter gives the designer an easy means to add this feature to a microcontroller with a minimal number of connections.

This Application Note will demonstrate how easy it is to connect the MPC3201 to an 8051-compatible microprocessor.

The MCP3201 is a fast 100kHz 12-bit A/D Converter featuring low power consumption and power saving standby modes. The features of the device include an onboard sample-hold and a single pseudo differential input. Output data from the MCP3201 is provided by a high speed serial interface that is compatible with the SPI® protocol. The MCP3201 operates over a broad voltage range (2.7V – 5.5V). The device is offered in 8-pin PDIP and 150mil SOIC packages.

The MCP3201 connects to the target microprocessor via an SPI-like serial interface that can be controlled by I/O commands, or by using the synchronous resources commonly found in microcontrollers. Two methods will be explored in supporting the serial format for the A/D Converter: An I/O port “bit-banging” method and a method that uses the 8051 UART in synchronous serial mode 0. An 8051 derivative processor, the 80C320, was chosen for testing since it has a second onboard serial port. This second serial port allows the A/D Converter sample data to be echoed to a host PC running an ASCII terminal program such as Hyperterm.

I/O PORT METHOD

The serial data format supported by the MCP3201 is illustrated in Figure 1. The A/D Converter will come out of its sleep mode on the falling edge of CS. The conversion is then initiated with the first rising edge of CLK. During the next 1.5 CLK cycles, the converter samples the input signal. The sampling period stops at the end of the 1.5 CLK cycles on the falling edge of CLK, and DOUT also changes from a Hi-Z state to null. Following the transmission of the null bit, the A/D Converter will respond by shifting out conversion data on each subsequent falling edge of the clock. The most significant bits are clocked out first. The micro is supplying the CS and CLK signals and the A/D Converter responds with the bit data on DOUT.

As shown in Figure 1, starting with an initial NULL bit, bits B11, B10, B9…B0 are shifted out of the A/D Converter. Following bit B0, further CLK falling edges will cause the A/D Converter to shift out bits B1…B11 in reverse order of the initial bit sequence. Continued CLKs will shift out zeros following B11 until CS returns high to signal the end of the conversion. On the rising edge of CS, DOUT will change to a Hi-Z state. The device receiving the data from the A/D Converter can use the low-to-high edge of CLK to validate (or latch) the A/D Converter bit data at DOUT.

The 8051 instruction set provides for bit manipulation to allow the use of I/O pins to serve as a serial host for the A/D Converter. By manually toggling the I/O pins and reading the resulting A/D Converter DOUT bits, the designer is free to use any I/O pin that can provide the needed function. The drawback to this method is the bandwidth limit imposed by the execution time of the opcodes supporting the A/D Converter communication. Example 1 shows a code module for a simple I/O port “bit-banging” method for supporting the MCP3201. To optimize for speed, the result is right justified in the ADRESH:ADRESL register pair.
USING THE SERIAL PORT IN SYNCHRONOUS MODE0

The UART on the 8051 supports a synchronous shift register mode that, with some software help, can be used to speed up the communications to the A/D Converter. In Mode0, the UART uses the RX pin for data I/O, while the TX pin provides a synchronization clock. The shift register is 8 bits wide and the TX pin will transition low to high to supply a clock rising edge for each bit. Figure 2 shows the typical Mode0 timing.

Since the UART was designed primarily to support RS-232 data transfers, the bit order expected is LSB first. The shift register Mode0 also uses this bit order. As shown in Figure 1, the first 12 bits of the A/D Converter data are ‘backwards’ for our application. Fortunately, the MCP3201 provides the reverse order of sampled bits after the initial transfer of bits B11…B0. Inspection of Figure 1 readily shows that working back from the last data bit transferred, 3 bytes received from the shift register will cover 24 bits of the 26 bits transferred from the A/D Converter. Conveniently, bit manipulation can be used to provide the two CLK rising edges needed during the beginning sample operation. After these two initial CLK cycles, the UART shifter can be accessed three times to read in the remainder of the data. The bit order will be correct for the third shifter byte as MSB data, the second byte will have 4 LSBs in the upper nibble (the lower nibble will be masked off), and the first byte will be tossed. Figure 3 shows the relationship between the shifted bits and SBUF data received by the UART. Example 2 shows a code module for using the synchronous port as the interface. The result is left justified in the ADRESH:ADRESL register pair.

EXAMPLE 1: I/O PORT METHOD CODE

GET_AD:   SETB CS ; set cs hi
          MOV COUNTA,#15 ;
          NXTBIT:  CLR DCLK ; X,X,NULL,D11,D10,D9...D0
          CLR CS ; CS low to start conversion or keep low till done
          SETB DCLK ; raise the clock
          MOV C,SDAT ; put data into C flag
          RLC A ; shift C into Acc (A/D low bits)
          XCH A,ADRESH ; get ADRESH byte (save low bits in ADRESH for now)
          RLC A ; shift C into Acc (A.D high bits)
          XCH A,ADRESH ; get low bits back into Acc for next loop
          DJNZ COUNTA,NXTBIT
          MOV ADRESL,A ; put A into ADRESL
          ANL ADRESH,#0FH ; mask off unwanted bits (x,X,X,Null)
          SETB CS ; set CS hi to end conversion
EXAMPLE 2: SYCHRONOUS PORT CODE

GET_AD:

SETB CS ; set CS hi
CLR DCLK ; X,X,NULL,D11,D10,D9...D0
CLR CS ; CS low to start conversion or keep low till done
SETB DCLK ; 1st S/H clock
CLR DCLK ;
SETB DCLK ; 2nd S/H clock and leave DCLK high

SETB REN_1 ; REN=1 & R1_1=0 initiates a receive
CLR R1_1 ;

BYTE_1: JNB R1_1, BYTE_1
MOV A,SBUF1 ; toss this byte
CLR R1_1

BYTE_2: JNB R1_1, BYTE_2
MOV ADRESL,SBUF1 ; save LSbs
CLR R1_1

BYTE_3: JNB R1_1, BYTE_3
MOV ADRESH,SBUF1 ; save MSbs
SETB CS ; set CS hi to end conversion
ANL ADRESH,#0FH ; mask off unwanted LSb bits
A Quick Comparison of Results

The test circuit used was taken from the data sheet and is shown in Figure 4.

An 80C320 microprocessor clocked at a crystal frequency of 11.0592 MHz yielded the following results:

<table>
<thead>
<tr>
<th>Method</th>
<th>CS Time (Conv. time approx.)</th>
<th>Approx. Throughput</th>
<th>Resources Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>I/O Port</td>
<td>99 µs</td>
<td>10 kHz</td>
<td>3 I/O pins</td>
</tr>
<tr>
<td>Sync. Serial</td>
<td>43.4 µs</td>
<td>23 kHz</td>
<td>1 UART (Mode0)</td>
</tr>
</tbody>
</table>

Note: The 80C320 can be clocked to 33MHz, which would effectively decrease the conversion time by a factor of 3 for increased performance in demanding applications.

TABLE 1: Conversion Time Comparison.

IN SUMMARY

Both methods illustrate the ease with which the MCP3201 A/D Converter can complement a design to add functionality for processing analog signals. The synchronous serial port method provides a 2:1 performance increase over the I/O port method, but consumes one UART as a resource. The I/O port method is flexible in allowing any suitable 3 I/O pins to be used in the interface.

Potential applications include control voltage monitoring, data logging, and audio processing. The routines in the source code appendices provide the designer with an effective resource to implement the design.
APPENDIX A:  I/O PORT SOURCE CODE

; $MOD51
; $TITLE(ads)
; $DATE(7/19/98)
; $PAGEWIDTH(132)
; $OBJECT(C:\ASM51\ads.OBJ)
; Author Lee Studley
; Assembled with Metalink’s FreeWare ASM51 assembler
; Tested with NOICE emulation software.
; Tested with a DALLAS DS80C320 (8031) micro clocked @ 11.0592mhz
; This test uses a ‘bit banging’ approach yielding a conversion time
; of approximately 99uS
; The result is transmitted via the original 8051 UART to an ascii
; terminal at 19.2k baud 8N1 format
;
; ================ RESET AND INTERRUPT VECTORS ================

RSTVEC EQU 0000H;
IE0VEC EQU 0003H;
TF0VEC EQU 000BH;
IE1VEC EQU 0013H;
TF1VEC EQU 001BH;
RITIVEC EQU 0023;
TF2VEC EQU 002BH;  ( 8052 )

; ================ VARIABLES ================

DSEG

; ================ PROGRAM VARIABLES ==================

COUNTA  EQU 30H
COUNTB  EQU 31H
ADRESL  EQU 2
ADRESH  EQU 3

; ================ HARDWARE EQUATES ==================

DCLK   EQU P1.3
SDAT   EQU P1.2
CS     EQU P1.1

; ================ CONSTANTS ==================

; ================ PROGRAM CODE ==================

CSEG

; org RSTVEC
; LJMP START

ORG 4000H;  NOICE SRAM/PROGRAM SPACE

; ================ RESET AND INTERRUPT VECTORS ================

START:

; Initialize the on-chip serial port for mode 1
; Set timer 1 for baud rate: auto reload timer

SETUPUART:

MOV  PCON,#80H; SET FOR DOUBLE BAUD RATE
MOV  TMOD,#00100010B; two 8-bit auto-reload counters
MOV  TH1, #0FDH; 19.2K @ 11.059 MHZ
MOV  SCON,#01010010B; mode 1, TI set
SETB TR1; start timer for serial port
;====================================================================
; GET_AD: Initiates the A/D conversion and retrieves the AD sample into
; ADRESH,ADRESL.
; The A/D convertor is connected to port1 pins 0..2 as:
;      SDAT EQU P1.0  I/O
;      DCLK EQU P1.1  I/O
;      CS   EQU P1.2  I/O
; Uses: ADRESL,ADRESH,ACC,COUNTA
; Exits: ADRESH=(x,x,x,x,B11..B8), ADRESL(B7..B0)
;====================================================================
GET_AD:  SETB CS ; set cs hi
         MOV COUNTA,#15 ; number of bits to shift 12+X,X,NULL=15
NXTBIT: CLR DCLK ; X,X,NULL,D11,D10,D9...D0
         CLR CS ; CS low to start conversion or keep low till done
         SETB DCLK ; raise the clock
         MOV C,SDAT ; put data into C flag
         RLC A ; shift C into Acc (A/D low bits)
         XCH A,ADRESH ; get ADRESH byte(sav low bits in ADRESH for now)
         RLC A ; shift C into Acc (A/D high bits)
         XCH A,ADRESH ; get low bits back into Acc for next loop
         DJNZ COUNTA,NXTBIT
         MOV ADRESL,A ; put A into ADRESL
         ANL ADRESH,#0FH ; mask off unwanted bits (x,x,x,X,NULL
         SETB CS ; set CS hi to end conversion
;=END__GET_AD========================================================
;====================================================================
;====================================================================
PROCDIGS:
         CALL BIN16BCD
         MOV R0,#7
NXTDIG:
         MOV A,#30H
         ADD A,@R0
         CALL SENDCHAR
         DEC R0
         CJNE R0,#3,NXTDIG
         CALL RETNEWLINE ; send a carriage return and line feed
         CALL DELAY1 ; wait here awhile
         JMP START
;====================================================================
;=SUBROUTINES========================================================
;====================================================================
;====================================================================
RETNEWLINE:
         MOV A,#0AH ; *** \n newline
         CALL SENDCHAR
         MOV A,#0DH ; *** return
         CALL SENDCHAR
         RET
;====================================================================
SENDCHAR:
T_TST:   JNB TI,T_TST ; loop till output complete
         CLR TI ; clear bit
         MOV SBUF,A ; send data
         RET
;====================================================================
;************************************************************************
;  BIN16BCD
The following routine converts an unsigned integer value in the range of 0 - 9999 to an unpacked Binary Coded Decimal number. No range checking is performed.

**Inputs:**
- R3 (MSB), R2 (LSB) contain the binary number to be converted.

**Outputs:**
- R7 (MSD), R6, R5, R4 (LSD) contain the 4 digit, unpacked BCD representation of the number.

**Uses:**
- R1, R2, R3, R4, R5, R6, R7, ACC

```assembly
BIN16BCD:
    MOV R1, #16D ; loop once for each bit (2 bytes worth)
    MOV R5, #0 ; clear regs.
    MOV R6, #0
    MOV R7, #0
    BCD_16LP:
        MOV A, R2
        ADD A, R2
        MOV R2, A
        MOV A, R3
        ADDC A, R3
        MOV R3, A
    ;=======
        MOV A, R5
        ADDC A, R5
        DA   A
        MOV R5, A
        MOV A, R6
        ADDC A, R6
        DA   A
        MOV R6, A
        DJNZ R1, BCD_16LP ; loop until all 16 bits done
    ;===============
    ; unpack the digits
    ;===============
        SWAP A ; swap so that digit 4 is rightmost
        ANL A, #0FH ; mask off digit 3
        MOV R7, A ; save digit 4 in R7
        MOV A, R6
        ADDC A, R6
        DA   A
        MOV R6, A
        MOV A, R5
        ADDC A, R5
        DA   A
        MOV R5, A ; put digit 2 in R5, digit 1 -> ACC
        ANL A, #0FH ; mask off digit 2
        MOV R4, A ; save digit 1 in R4 then exit
    RET
```

```
DELAY1: DJNZ R2, DELAY1
DELAY2: DJNZ R3, DELAY1
RET
```

END
### APPENDIX B: SYNCHRONOUS PORT SOURCE CODE

$MOD51
$TITLE(ads2)
$DATE(7/29/98)
$PAGEWIDTH(132)
$OBJECT(C:\ASM51\ads2.OBJ)

Author: Lee Studley
Assembled with Metalink's FreeWare ASM51 assembler
Tested with NOICE emulation software.
Tested with a DALLAS DS80C320 (8031) micro clocked @ 11.0592mhz
This micro has a 2nd UART resource at pins P1.2,P1.3
This test uses a the UART MODE0 approach yielding a conversion
time of approximately 43.4uS
The result is transmitted via the original 8051 UART to an ascii
terminal at 19.2k baud 8N1 format

;------------------- RESET AND INTERRUPT VECTORS -------------------
RSTVEC EQU 0000H
IE0VEC EQU 0003H
TF0VEC EQU 000BH
IE1VEC EQU 0013H
TF1VEC EQU 001BH
RITIVEC EQU 0023H
TF2VEC EQU 002BH

;------------------- VARIABLES -------------------
DSEG

;------------------- HARDWARE EQUATES -------------------
DCLK EQU P1.3
SDAT EQU P1.2
CS EQU P1.1

;2nd Uart equates
SCON1 EQU 0C0H
SBUF1 EQU 0C1H
REN_1 BIT SCON1.4
R1_1 BIT SCON1.0

;------------------- CONSTANTS -------------------

;------------------- PROGRAM CODE -------------------
CSEG

ORG RSTVEC
LJMP START
ORG 4000H ; NOICE SRAM/PROGRAM SPACE
START:
; Initialize the on-chip serial port for mode 1
; Set timer 1 for baud rate: auto reload timer

SETUPUART:
    MOV PCON,#80H ; SET FOR DOUBLE BAUD RATE
    MOV TMOD,#00100010B ; two 8-bit auto-reload counters
    MOV TH1,#0FDH ; 19.2K @ 11.059 MHZ
    MOV SCON,#01010010B ; mode 1, TI set
    SETB TR1 ; start timer for serial port

SETUPUART2:
    MOV SCON1,#00000000B ; 2nd uart mode 0, TI set
    ; Shift clk(TX)=Tosc/12

GET_AD: Initiates the A/D conversion and retrieves the AD sample into
; ADRESH,ADRESL.
; The A/D convertor is connected to port1 pins 1..3 as:
; DCLK EQU P1.3 Tx(synchronous clock)
; SDAT EQU P1.2 Rx(synchronous data)
; CS EQU P1.1 I/O
; Uses: ADRESL,ADRESH,ACC,COUNTA
; Exits: ADRESH=(B11..B4), ADRESL(B3..B0,x,x,x,x)

GET_AD:
    SETB CS ; set cs hi
    CLR DCLK ; X,X,NULL,D11,D10,D9...D0
    CLR CS ; CS low to start conversion or keep low till done
    SETB DCLK ; 1st S/H clock
    CLR DCLK ;
    SETB DCLK ; 2nd S/H clock and leave DCLK high
    SETB REN_1 ; REN-1 & R1_1=0 initiates a receive
    CLR R1_1 ;

BYTE_1: JNB R1_1,BYTE_1 ; toss this byte
    MOV A,SBUF1
    CLR R1_1

BYTE_2: JNB R1_1,BYTE_2 ; save labs
    MOV ADRESL,SBUF1
    CLR R1_1

BYTE_3: JNB R1_1,BYTE_3 ; save mabs
    MOV ADRESH,SBUF1 ; set CS hi to end conversion
    ANL ADRESL,#0F0H ; mask off unwanted lsb bits

PROCDIGS:
    CALL BIN16BCD
    MOV R0,#7

NXTDIG:
    MOV A,#30H
    ADD A,@R0
    CALL SENDCHAR
    DEC R0
    CJNE R0,#3,NXTDIG
CALL RETNEWLINE  ; send a carriage return and line feed
CALL DELAY1      ; wait here awhile
JMP START

;====================================================================
;=SUBROUTINES========================================================
;====================================================================
;====================================================================
;====================================================================
;====================================================================

RETNEWLINE:
    MOV  A,#0AH ; *** \n newline
    CALL SENDCHAR
    MOV  A,#0DH ; *** return
    CALL SENDCHAR
    RET

;====================================================================
;====================================================================

SENDCHAR:
T_TST:  JNB  TI,T_TST ; loop till output complete
    CLR  TI ; clear bit
    MOV  SBUF,A ; send data
    RET

;====================================================================
;====================================================================

; BIN16BCD
;   The following routine converts an unsigned integer value in the
;   range of 0 - 9999 to an unpacked Binary Coded Decimal number. No
;   range checking is performed.
;
;   INPUT: R3 (MSB), R2 (LSB) contain the binary number to be
;          converted.
;
;   OUTPUT: R7 (MSD), R6, R5, R4 (LSD) contain the 4 digit, unpacked BCD
;          representation of the number.
;
;   Uses: R1, R2, R3, R4, R5, R6, R7, ACC

;************************************************************************
BIN16BCD:
    MOV  A,ADRESL ; right justify the
    SWAP A        ; R3:R2 pair for bin16bcd routine
    MOV  ADRESL,A
    MOV  A,ADRESH
    SWAP A
    ANL  A,#0F0H
    ORL  ADRESL,A
    MOV  A,ADRESH
    SWAP A
    ANL  A,#0FH
    MOV  ADRESH,A

;=======
    MOV  R1,#16D ; loop once for each bit (2 bytes worth)
    MOV  R5,#0   ; clear regs.
    MOV  R6,#0
    MOV  R7,#0

; BCD_16LP:
    ADD  A,R2
    MVI  A,R3
    ADDC A,R3
    MOV  R3,A

;=====
MOV A,R5  
ADDC A,R5  
DA A  
MOV R5,A  

MOV A,R6  
ADDC A,R6  
DA A  
MOV R6,A  
DJNZ R1,BCD_16LP ; loop until all 16 bits done  

; unpack the digits  

SWAP A ; swap so that digit 4 is rightmost  
ANL A,#0FH ; mask off digit 3  
MOV R7,A ; save digit 4 in R7  
MOV A,R6 ; get digits 3,4 again  
ANL A,#0FH ; mask off digit 4  
MOV R6,A ; save digit 3  

MOV A,R5 ; get digits 1,2  
SWAP A ; swap so that digit 2 is rightmost  
ANL A,#0FH ; mask off digit 1  
XCH A,R5 ; put digit 2 in R5, digit 1 => ACC  
ANL A,#0FH ; mask off digit 2  
MOV R4,A ; save digit 1 in R4 then exit  

RET

;====================================================================
;====================================================================
;====================================================================

DELAY1: DJNZ R2,DELAY1  
DELAY2: DJNZ R3,DELAY1  
RET

;====================================================================
;====================================================================
;====================================================================

END
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.

Information contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. No representation or warranty is given and no liability is assumed by Microchip Technology Incorporated with respect to the accuracy or use of such information, or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Microchip's products as critical components in life support systems is not authorized except with express written approval by Microchip. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights.

Trademarks

The Microchip name and logo, the Microchip logo, FilterLab, KEELOQ, microID, MPLAB, PIC, PICmicro, PICMASTER, PICSTART, PRO MATE, SEEVAL and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

dsPIC, ECONOMONITOR, FanSense, FlexROM, fuzzyLAB, In-Circuit Serial Programming, ICSP, ICEPIC, microPort, Migratable Memory, MPASM, MPLIB, MPLINK, MPSIM, MXDEV, PICC, PICDEM, PICDEM.net, rIPIC, Select Mode and Total Endurance are trademarks of Microchip Technology Incorporated in the U.S.A.

Serialized Quick Turn Programming (SQTP) is a service mark of Microchip Technology Incorporated in the U.S.A.

All other trademarks mentioned herein are property of their respective companies.

© 2002, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

Printed on recycled paper.
**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: 770-640-0075

**Boston**  
2 Lan Drive, Suite 180  
Itasca, IL  60143  
Tel: 630-285-0071  
Fax: 630-285-0075

**Chicago**  
333 Pierce Road, Suite 180  
Addison, TX 75001  
Tel: 972-818-7423  
Fax: 972-818-2924

**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  92812  
Tel: 949-263-1338  
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: 408-436-7955  
Fax: 408-436-7955

**Toronto**  
6285 Northave 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  
Beijing, 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  
No. 71 Wusi Road  
Fuzhou 350001, China  
Tel: 86-591-7503506  
Fax: 86-591-7503521

**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, 189890  
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  
Regerus Business Centre  
Lastrup Hoej 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 - 1er 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: 49-89-6275-700  
Fax: 49-89-6275-5060

**Hong Kong**  
Microchip Technology Hong Kong Ltd.  
Unit 901-6, Tower 2, Metroploza  
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  
Divyasee Chambers  
1 Floor, Wing A (A3/A4)  
No. 11, O’Shaugnessy Road  
Bangalore, 560 025, India  
Tel: 91-80-2290061  
Fax: 91-80-2290062

**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