Electromechanical Switch Replacement

NETSWITCH

Author: *Csaba Bardos*
RASS Ltd.
Tiszafured, Hungary
email: cbardos@mail.inext.hu

APPLICATION OPERATION

This gadget is designed for individually dimming lamps, turning on/off electrical devices, etc. It accepts commands via serial lines individually, (i.e., more than one NETSWITCH can connect to same serial line, or you can use its switches to control it).

You can handle up to 256 turning points with this device, and you can control them from one central station. (The central station description is not included in this text). There are two modes on device: Manual or Serial.

```
NODE-ADDRESSBYTE-COMMAND-COMMANDPARAM-ENDBL
```

Switching between modes is only allowed from serial commands. It starts Manual Mode when powered up.

Manual Mode

There are three buttons on the device. One for turning on/off the load, the other two are push buttons which let you dim the load. When the on/off switch is turned on, dimming is not allowed. When turned off, dimming is allowed and you have to dim to zero intensity if you want to turn off.

Serial Mode

The device is able to receive (only) data from the serial line at 1200 bps. Being connected to the same serial line and to more than one NETSWITCH is allowed, because every NETSWITCH has a unique number to identify. (This number not changeable, it is set compile time, though, this is not benefit.)

The NETSWITCH accepts blocks of bytes only as follows:

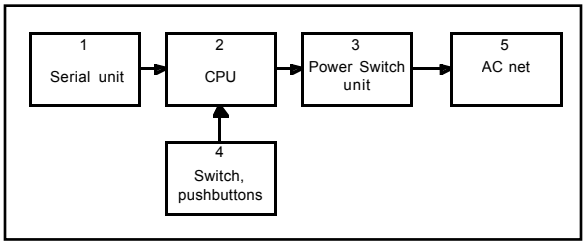
Where:

- NODE: NODE command.
- ADDRESSBYTE: This is a unique number.
- COMMAND: One of commands below.
- COMMANDPARAM: Optional, it depends from command.
- ENDBL: This is a terminator command.

The acceptable commands are:

- NODE: This command means the start of block, its value is ASCII ‘N’.
- MANU: This command puts the NETSWITCH to manual mode. When in this mode, it accepts its switches. Its value is ASCII ‘M’.
- SERI: This command puts the NETSWITCH to serial mode. When in this mode, it accepts serial commands only. Its value is ‘S’.
- SWOFF: This command turns off the NETSWITCH. Its value is ASCII ‘F’.
- SWON: This command turns on the NETSWITCH. Its value is ASCII ‘O’.
- DIMM: This command dimms the load, according to its parameter. Its value is ASCII ‘D’.
- ENDBL: This is END of BLock command, every block has to end with it. Its value is ASCII ‘E’.

BLOCK DIAGRAM



MICROCHIP HARDWARE DEVELOPMENT TOOLS USED:

Assembler/Compiler Version:

MPASM v1.40

APPLICATION OPERATION

Hardware Description

It consists of three main parts:

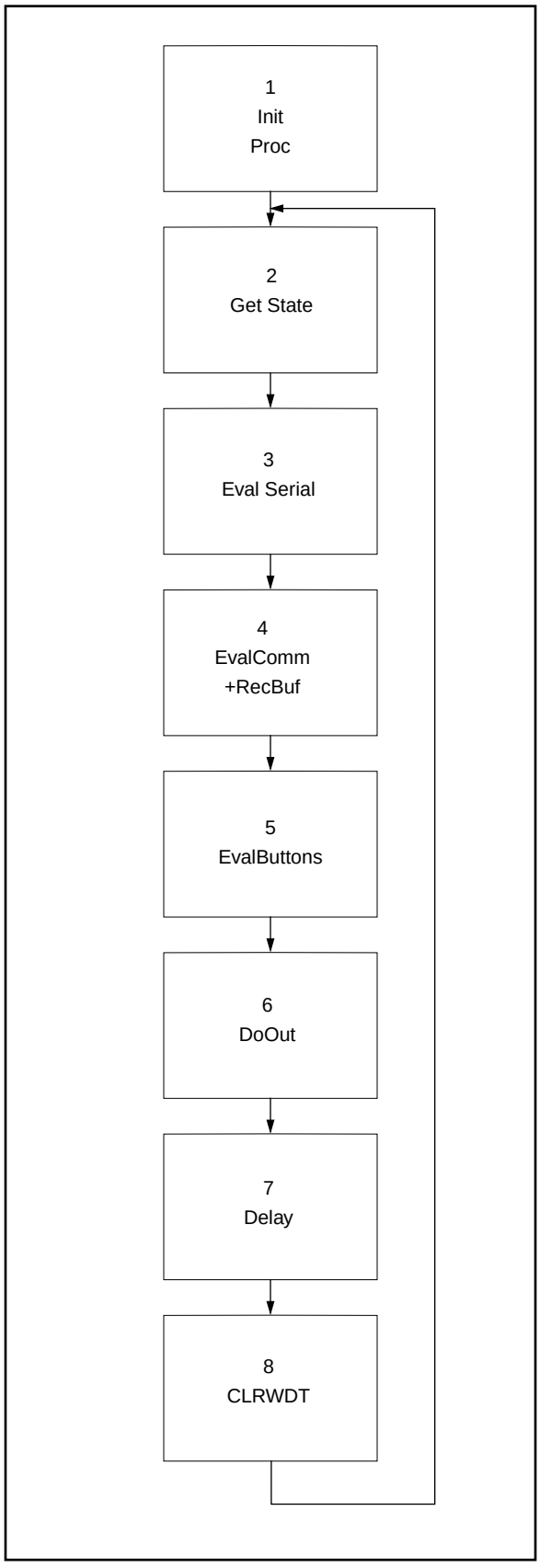
1. The power supply. The gadget obtains it from another 5V source now (e.g., it comes with serial line).
2. The PIC12C508. It is controlled in two ways.
   a)In manual mode with push buttons and switch.
   b)In serial mode with serial commands.
   The PIC12C508 runs with its on-chip (internal) RC oscillator, nominally 4 MHz. The PIC12C508 controls the power switching unit by a PWM signal. It is available on pin 7 of PIC12C508.
3. The power switching unit. This unit switches (or dimms) the load. For a half period:
   a)When the period starts, the zener stabilizes voltage at 10V.
   b)The opto-coupler receiving side transistor, NPN, and PNP transistor are voltage controlled current source. This current is available on collector of PNP. It charges a capacitor. When this capacitor reaches threshold of the UJT (uni-junction transistor), it discharges the cap and produces an ignition pulse for thyristor (SCR). You can choose a right thyristor for your requirements.

Software Description

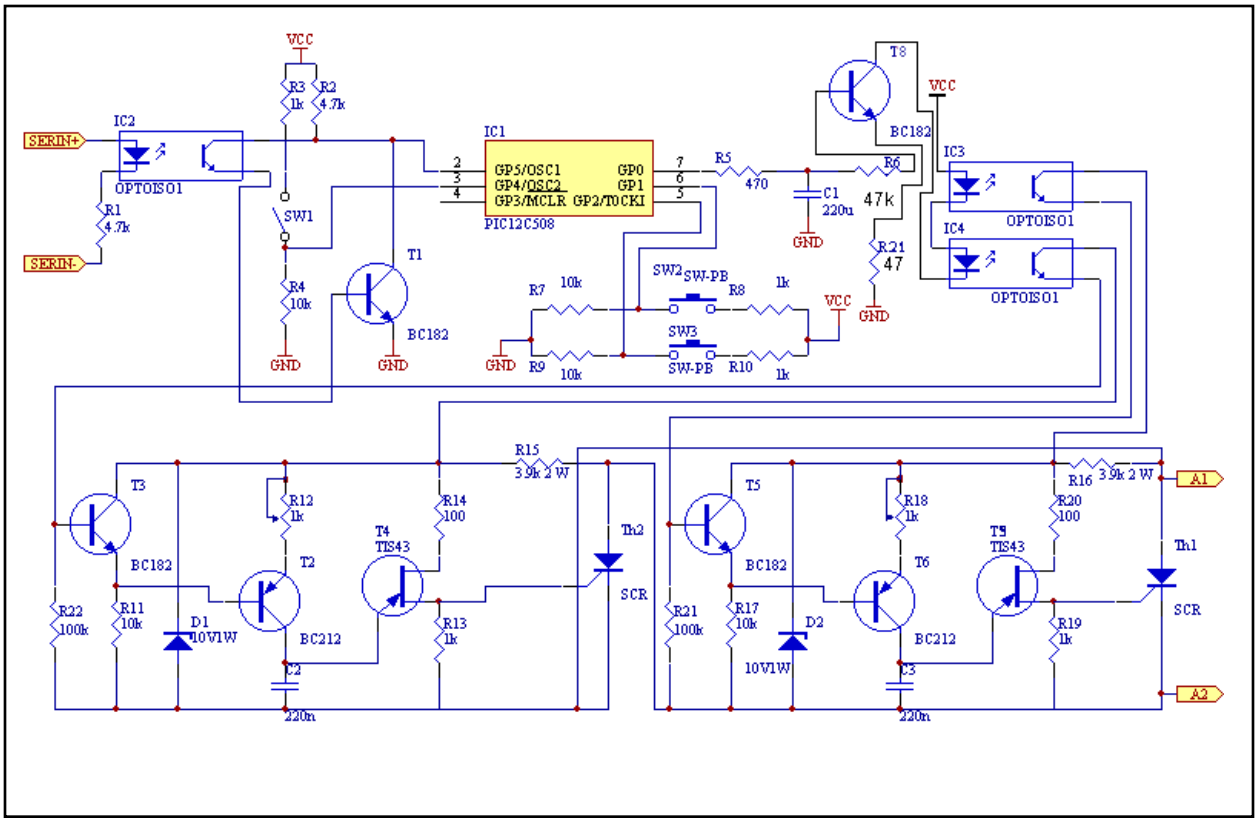
The software contains a loop which gets the state of input pins, then evaluates serial communication, the commands received via the serial line, the state of buttons and then does the appropriate action, i.e. controls SCRs with a PWM sign.

There is a delay subroutine in the loop, which controls the time usage. When the device is not receiving a serial sign, the delay routine does a half time cycle (half of one bit time of 1200 bps). When the device is receiving, it does a complete time cycle (one bit time of 1200 bps). This routine is based on the TMR0. (TMR0 is in timer mode, counts internal clock and has a prescaler 1:4). For additional information, see the file `netsw.asm`, it has many comments.

FLOW CHART



GRAPHICAL HARDWARE REPRESENTATION



APPENDIX A: SOFTWARE LISTING

```
title “NETSW.ASM V1.0 “

; Electromechanical switch replacement.
; This program controls the NETSWITCH.

                    list p=12c508

#include “p12c508.inc”

; The configuration bits are configured as follows:
; MCLRE-disabled (0)
; CP-disabled (1)
; WDTE-enabled (1)
; OSC-internal RC (10)

; The next definitions are the commands that accepted by NETSWITCH.

#define NODE  0x4E                          ; NODE address, ASCII “N”
#define MANU  0x4D                          ; MANUal mode, ASCII “M”
#define SERI  0x53                          ; SERIal mode, ASCII “S”
#define SWOFF 0x46                          ; SWitch turn OFF, ASCII “F”
#define SWON  0x4F                          ; SWitch turn ON, ASCII “O”
#define DIMM  0x44                          ; DIMMing, ASCII “D”
#define ENDBL 0x45                          ; END of BLock, ASCII “E”

; Each block that controls the NETSWITCH has to meet the next form:
; NODE-ADDRESSBYTE-COMMAND-COMMANDPARAM-ENDBL
; NODE - Command NODE
; ADDRESSBYTE - This is the NODE address
; COMMAND - COMMAND byte, see above (SERI, MANU, SWOFF, SWON, DIMM)
; COMMANDPARAM - One byte, for DIMM command only
;                                  (e.g. DIMM 0x7F means half intensity)
; ENDBL - END of BLock

NETSWOPTION         equ 0xC1       ; Bin: 11000001, Tim. presc.to TMR0, 1:4
NULL                equ 0x00
EIGHT               equ 0x08
GPSET               equ 0xFE       ; Only Bit 0 is output
BLSIZE              equ 0x05       ; BLockSIZE, it is 5 now. See above.
MYADDR              equ 0x10       ; <- This is an example
MINPWM              equ 0x00       ; MINimal PWM value, i.e. totally turn off
MAXPWM              equ 0x01       ; MAXimal PWM value
BUTTONCYCLE         equ 0x7F       ; For button delaying
HALFTIME            equ 0x92       ; This is a half cycle value. (416 usec)

ReadBuf             equ 0x08       ; Holds input info
SWStatus            equ 0x09       ; Holds all system status
NumBits             equ 0x0A       ; Number of bits
Counter             equ 0x0B       ;
EvalComBuf          equ 0x0C       ; Temporary byte for holding changes
PWMCycleCount       equ 0x0D       ; For PWM signal
PWMPulseCount       equ 0x0E       ; For PWM signal
RecBuf              equ 0x0F       ; Receive Buffer
DIMMParam           equ 0x10       ; DIMMing Parameter
ButtonCounter       equ 0x11       ; Button delaying
BLTABLE             equ 0x14       ; BLock TABLE starts here


; The next bits are used in SWStatus byte

RecInPr             equ 0x00
NewByte             equ 0x01
BLInPr              equ 0x02
ManMode             equ 0x03
Pulse               equ 0x04
```



```
NewBL               equ 0x05
PWMUpdate           equ 0x06

; The next bits are used in GPIO

Rx                  equ 0x05
UpButt              equ 0x02
DownButt            equ 0x03
SW                  equ 0x04
PWMOut              equ 0x00

MSB                 equ 0x07

                    org 0x00

Start               call Init               ; Call Init procedure
MainLoop            call GetState           ; Get all input state
                    call EvalSerial         ; Evaluate serial input
                    call RecBlock           ; Receive all byte in block
                    call EvalCommand        ; Evaluate command via serial input
                    call EvalButtons        ; Evaluate pushbuttons and switch
                    call DoOut              ; Do appropriate action
                    call Delay              ; Delay cycle
                    clrwdt                  ; Clear watchdog
                    goto MainLoop           ; Do again
Init                movlw NETSWOPTION       ;
                    option                  ;
                    movlw EIGHT+1           ;
                    movwf FSR               ;
                    movlw 0x0D              ;
                    movwf 0x08              ;
DoAgainClear        decfsz 0x08,F           ;
                    goto ClearOneRAM        ;
Continue            movlw EIGHT             ;
                    movwf NumBits           ;
                    movlw GPSET             ;
                    tris GPIO               ;
                    movlw HALFTIME          ;
                    movwf TMR0              ;
                    retlw NULL              ;
ClearOneRAM         clrf INDF               ;
                    incf FSR,F              ;
                    goto DoAgainClear       ;
GetState            movf GPIO,W             ; Read pins
                    movwf ReadBuf           ; Put to ReadBuf
                    retlw NULL              ; And return

EvalSerial          btfsc SWStatus,RecInPr  ; RECeiving IN PRogress bit test
                    goto NextBit            ; Set, the next bit comes
                    btfss ReadBuf,Rx        ; RX bit test, start condition true ?
                    bsf SWStatus,RecInPr    ; Yes, set RECeiving IN PRogress bit
                    retlw NULL              ; No, return
NextBit             btfsc ReadBuf,Rx        ; RX bit test
                    bsf RecBuf,MSB          ; If RX set, set MSB of RecBuf
                    bcf STATUS,C            ; Clear Carry
                    rrf RecBuf,F            ; Rotate Right
                    decfsz NumBits,F        ; Is there more bits ?
                    retlw NULL              ; Yes, return
                    bcf SWStatus,RecInPr    ; No, clear status bit
                    bsf SWStatus,NewByte    ; This is new byte
                    movlw EIGHT             ; Prepare next serial receiving
                    movwf NumBits           ;
                    retlw NULL              ; Return

; This routine called RecBlock waits until all element of the command
; block is received by setting BLInPr (BLock receiving In PRogress) bit in
```

```
; SWStatus. It write the received bytes to BLTable.

RecBlock            btfss SWStatus,NewByte              ; Is there new byte ?
                          retlw NULL                    ; No, return
                          bcf SWStatus,NewByte          ; Clear new byte status
                          btfsc SWStatus,BLInPr         ; Is receiving in progress ?
                          goto NextByte                 ; Yes, go to NextByte
                          bsf SWStatus,BLInPr           ; No, sets BLInPr bit
                          movlw BLTABLE                 ; BLTABLE to
                          movwf FSR                     ; FSR
                          movlw BLSIZE                  ; BLSIZE to
                          movwf Counter                 ; Counter
NextByte            movf RecBuf,W                       ; Put contents to
                          movwf INDF                    ; (FSR), i.e. to BLTable
                          incf INDF,F                   ; Points to next element
                          decfsz Counter,F              ; Is there more byte ?
                          retlw NULL                    ; Yes, return
                          bcf SWStatus,BLInPr           ; No, clears BLInPr bit
                          bsf SWStatus,NewBL            ; This is new block
                          retlw NULL                    ; Return, the BLTable is ready to use

; The routine called EvalCommand evaluates the BLTable.

EvalCommand         btfss SWStatus,NewBL      ; Is there new block ?
                          retlw NULL                    ; No
                          btfsc SWStatus,BLInPr; Is block receiving in progress ?
                          retlw NULL                    ; Yes, return
                          movlw BLTABLE; No, BLTABLE to
                          movwf FSR                     ; FSR
                          movlw NODE                    ; This must be the first command
                          call EvalCell                 ; Is the command rigth ?
                          btfss STATUS,Z                ;
                          goto ClearTable               ; No, go to ClearTable
                          movlw MYADDR                  ; Yes,
                          call EvalCell                 ; Check, NODE ADDRESS is matching ?
                          btfss STATUS,Z                ;
                          goto ClearTable               ; No, go to ClearTable
                          movlw MANU                    ; Yes,
                          call EvalCell                 ; Is the next command MANU ?
                          btfss STATUS,Z                ;
                          goto ChkSERI                  ; No, go to ChkSERI
                          bsf EvalComBuf,ManMode; Yes, sets ManMode
                          goto ChkEND                   ; Go to ChkEND
ChkSERI             movlw SERI                ;
                          call EvalCell                 ; Is the command SERI ?
                          btfss STATUS,Z                ;
                          goto ChkSWON                  ;
                          bcf EvalComBuf,ManMode; Yes, clears ManMode
                          goto ChkEND                   ;
ChkSWON             movlw SWON                ;
                          call EvalCell                 ; Is the command SWON ?
                          btfss STATUS,Z                ;
                          goto ChkSWOFF;
                          movlw MAXPWM;
                          movwf DIMMParam; DIMMParam sets to MAXPWM
                          goto ChkEND                   ;
ChkSWOFF            movlw SWOFF               ;
                          call EvalCell                 ; Is the command SWOFF ?
                          btfss STATUS,Z                ;
                          goto ChkDIMM                  ;
                          movlw MINPWM                  ;
                          movwf DIMMParam; DIMMParam sets to MinPWM
                          goto ChkEND                   ;
ChkDIMM             movlw DIMM     ;
                          call EvalCell                 ; Is the command DIMM ?
                          btfss STATUS,Z                ;
```

```
                                goto ChkEND           ;
ChkParam            movf INDF,W                       ; Yes, the Parambyte is
                                movwf DIMMParam       ; transferred to DIMMParam
                                incf FSR,F            ;
ChkEND                          movlw ENDBL           ; Is the command ENDBL ?
                                call EvalCell         ;
                                btfss STATUS,Z        ;
                                goto ClearTable       ; No, go to ClearTable
                                movf EvalComBuf,W     ; Yes, all changes are written to
                                iorwf SWStatus,F      ; SWStatus
ClearTable          movlw BLTABLE                     ; BLTABLE to
                                movwf FSR             ; FSR
                                movlw BLSIZE          ; BLSIZE to
                                movwf Counter         ; Counter
ClearCycle          clrf INDF               ; Clears one element
                                incf FSR,F            ; Points to next element
                                decfsz Counter,F; Is there more byte ?
                                goto ClearCycle       ; Yes,
                                movlw NULL            ; No, all changes are discarded
                                movwf DIMMParam;
                                bcf SWStatus,NewBL;
                                retlw NULL            ; Return
EvalCell                        subwf INDF,W          ; W and INDF are equal ? ( affects Z)
                                btfss STATUS,Z        ;
                                retlw NULL            ; No, (Z remains)
                                incf FSR,F            ; Yes, points next element
                                movlw NULL            ; Sets Z, since NULL = 0
                                retlw NULL            ; return (Z remains)

; The routine called EvalButtons evaluates status of the buttons, i.e.
; it sets DIMMParam according to the status of buttons. It works manual
; mode (MANU bit set) of course.

EvalButtons         btfss SWStatus,ManMode; Are buttons enabled ?
                                retlw NULL            ; No
                                btfss ReadBuf,SW      ; Yes, Does SWitch turn on ?
                                goto ChkUpButt        ; No, go to check pushbuttons
SWTurnOn            movlw MAXPWM            ; Yes, turn on
                                movwf DIMMParam;
                                retlw NULL            ;
ChkUpButt           btfss SWStatus,UpButt             ;
                                goto ChkDownButt      ;
                                decfsz ButtonCounter,F;
                                retlw NULL            ;
                                movlw MAXPWM          ;
                                subwf DIMMParam,W     ;
                                btfss STATUS,Z        ;
                                decf DIMMParam,F      ;
ChkDownButt         btfss SWStatus,DownButt           ;
                                goto RetFromButt      ;
                                decfsz ButtonCounter,F;
                                retlw NULL            ;
                                movlw MINPWM          ;
                                subwf DIMMParam,W     ;
                                btfss STATUS,Z        ;
                                incf DIMMParam,F      ;
RetFromButt         movlw BUTTONCYCLE                 ;
                                movwf ButtonCounter   ;
                                retlw NULL            ;

; The routine called DoOut makes a PWM sign depending on DIMMParam.
; If DIMMParam is 0, no PWM sign exist, the output pin is HIGH.

DoOut               btfss SWStatus,PWMUpdate  ; Does PWM need to update ?
                                retlw NULL            ; No
                                decfsz PWMCycleCount,F; Yes, Decrement PWM cycle
```

```
                                goto PWMPulse             ;
                                bsf GPIO,PWMOut           ; Prepare ...
                                movf DIMMParam,W          ; ... next ...
                                movwf PWMPulseCount       ; ... pulse
                                bsf SWStatus,Pulse;
                                retlw NULL                ;
PWMPulse            btfss SWStatus,Pulse      ; Need a pulse cycle ?
                                retlw NULL                ; No
                                decfsz PWMPulseCount,F    ; Yes, do
                                retlw NULL                ;
                                bcf GPIO,PWMOut; clear output and ...
                                bcf SWStatus,Pulse        ; ... status
                                retlw NULL                ;
Delay                           btfss SWStatus,RecInPr    ; Is receiving in progress ?
                                goto HalfCycle            ; No, polling rate is about 416 us
                                call HalfCycle;Yes, poll. rate is 833 us first half
                                movlw 0x10 ;
                                addwf TMR0,F;
                                call HalfCycle            ; Second half
                                bsf SWStatus,PWMUpdate;
                                retlw NULL                ;
HalfCycle           movf TMR0,W                           ;
                                btfss STATUS,Z            ; Need to wait ?
                                goto HalfCycle            ; Yes
                                movlw HALFTIME            ; No
                                movwf TMR0 ; Reload
                                btfss SWStatus,PWMUpdate;
                                goto SetPWMUpdate;
ClearPWMUpdate      bcf SWStatus,PWMUpdate                ;
                                retlw NULL                ;
SetPWMUpdate        bsf SWStatus,PWMUpdate                ;
                                retlw NULL                ;
                                end                       ;
_
```

NOTES:

NOTES:

WORLDWIDE SALES & SERVICE

AMERICAS

Corporate Office
Microchip Technology Inc.
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 602-786-7200 Fax: 602-786-7277
Technical Support: 602 786-7627
Web: http://www.microchip.com

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

Boston
Microchip Technology Inc.
5 Mount Royal Avenue
Marlborough, MA 01752
Tel: 508-480-9990 Fax: 508-480-8575

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

Dallas
Microchip Technology Inc.
14651 Dallas Parkway, Suite 816
Dallas, TX 75240-8809
Tel: 972-991-7177 Fax: 972-991-8588

Dayton
Microchip Technology Inc.
Two Prestige Place, Suite 150
Miamisburg, OH 45342
Tel: 937-291-1654 Fax: 937-291-9175

Los Angeles
Microchip Technology Inc.
18201 Von Karman, Suite 1090
Irvine, CA 92612
Tel: 714-263-1888 Fax: 714-263-1338

New York
Microchip Technology Inc.
150 Motor Parkway, Suite 416
Hauppauge, NY 11788
Tel: 516-273-5305 Fax: 516-273-5335

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

Toronto
Microchip Technology Inc.
5925 Airport Road, Suite 200
Mississauga, Ontario L4V 1W1, Canada
Tel: 905-405-6279 Fax: 905-405-6253

ASIA/PACIFIC

Hong Kong
Microchip Asia Pacific
RM 3801B, Tower Two
Metroplaza
223 Hing Fong Road
Kwai Fong, N.T., Hong Kong
Tel: 852-2-401-1200 Fax: 852-2-401-3431

India
Microchip Technology Inc.
India Liaison Office
No. 6, Legacy, Convent Road
Bangalore 560 025, India
Tel: 91-80-229-4036 Fax: 91-80-559-9840

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

Shanghai
Microchip Technology
RM 406 Shanghai Golden Bridge Bldg.
2077 Yan'an Road West, Hong Qiao District
Shanghai, PRC 200335
Tel: 86-21-6275-5700
Fax: 86 21-6275-5060

Singapore
Microchip Technology Taiwan
Singapore Branch
200 Middle Road
#07-02 Prime Centre
Singapore 188980
Tel: 65-334-8870 Fax: 65-334-8850

Taiwan, R.O.C
Microchip Technology Taiwan
10F-1C 207
Tung Hua North Road
Taipei, Taiwan, ROC
Tel: 886 2-717-7175 Fax: 886-2-545-0139

EUROPE

United Kingdom
Arizona Microchip Technology Ltd.
Unit 6, The Courtyard
Meadow Bank, Furlong Road
Bourne End, Buckinghamshire SL8 5AJ
Tel: 44-1628-851077 Fax: 44-1628-850259

France
Arizona Microchip Technology SARL
Zone Industrielle de la Bonde
2 Rue du Buisson aux Fraises
91300 Massy, France
Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79

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

Italy
Arizona Microchip Technology SRL
Centro Direzionale Colleoni
Palazzo Taurus 1 V. Le Colleoni 1
20041 Agrate Brianza
Milan, Italy
Tel: 39-39-6899939 Fax: 39-39-6899883

JAPAN

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

8/29/97