Electromechanical Switch Replacement

Jim's Toy

Author: *Mark R. Hahn*
Hahn Associates
Portland, OR
email: hahndo@teleport.com

"Jim's Toy" was designed to be an electronic practical joke. Jim asked me to design a device that would wake up at semi-random intervals and beep for a few seconds. In addition, the device should:

- Be quiet for a fairly long period of time after initially being powered up.
- Have a very long battery life.
- Be fairly small.

The idea was to hide a dozen of these beepers in electrical outlets, inside furniture, up the chimney, etc. The long startup time was to allow Jim time to plant the device and leave the premises, hopefully alleviating Mike's suspicions. These beepers are high enough frequency that when sounding, most people will have a hard time locating the source of the beeping. The beep should repeat itself about every six hours. This way Jim is pretty sure that Mike will be home for at least one of the daily beep cycles. Jim plans to start the beepers several hours apart, so that the overall affect will be that a mysterious beeping occurs somewhere in Mike's new house in a seemingly random manner.

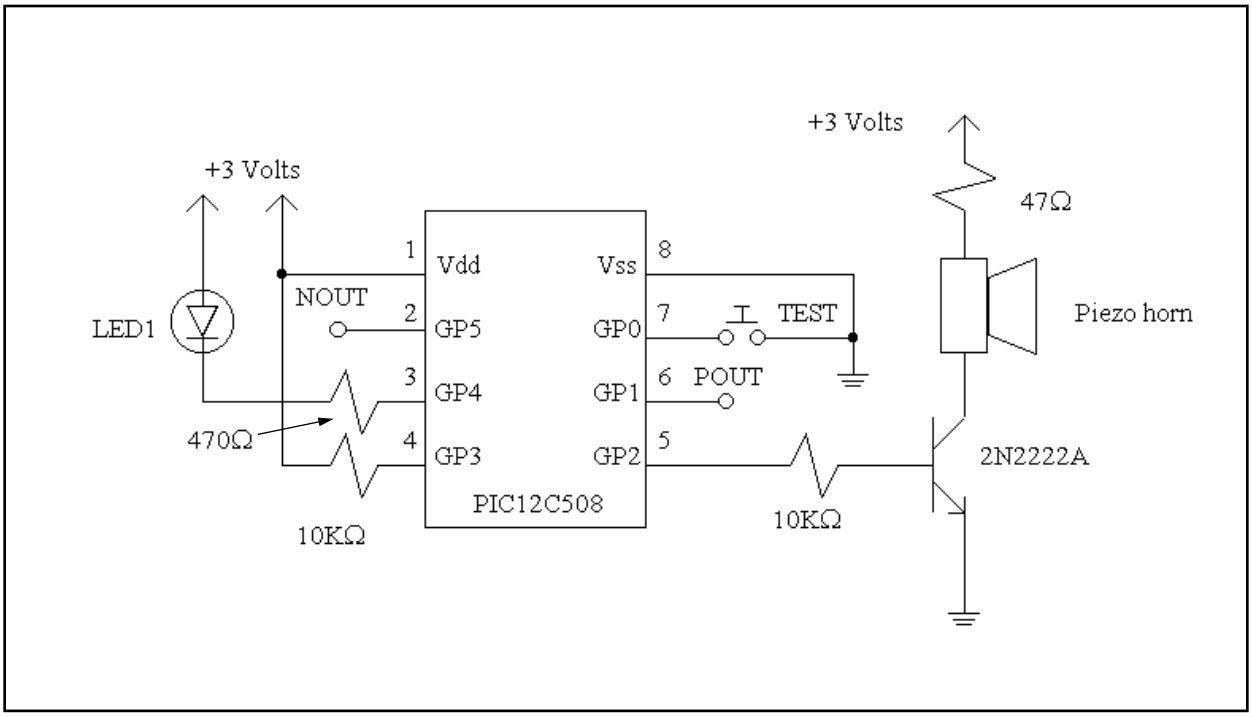
Because of the size and battery life requirements, I immediately thought of the Microchip PIC12C508. Using the PIC12C508, I was able to design a beeper that should last two years using a standard lithium coin cell as a power source. I added an LED and a test button to allow the operator to verify that the device is working. The test button can also be used to verify that the firmware is correct (see software description on following pages).

It may be stretching the electromechanical switch aspect of this design, but I like to think of it as an electronic equivalent of a joy buzzer. I'm sure Mike will eventually suspect Jim, and will soon thereafter call me (he knows that I'm the only one of Jim's friends capable of designing something like this). Once I tell him about the cyclic nature of the beepers he probably won't have too hard a time finding them. I also suspect I'll be designing some sort of variation for Mike to use on Jim.

APPLICATION OPERATION

Jim's Toy uses a single PIC12C508. The PIC12C508 is normally in the sleep mode. It wakes on a watchdog timeout, or if the test button is pushed. The watchdog timer is normally set on its longest period (over 2 seconds). The firmware determines what caused it to wake up, either a reset (initial powerup), a watchdog timeout, or a change on one of the input lines (typically when the test button is pushed). On waking from a watchdog timeout, the firmware decrements a 16 bit counter. After initial powerup, this counter is set to a long value, to allow the beeper to sleep for a couple of days. Once the counter reaches zero, the firmware sounds a sequence of beeps. It resets the counter to a smaller value, so the beeper will now wake and make noise more frequently. The firmware can also wake on a press of the test button. This will cause the device to make three short beeps and the LED flashes. This is intended to allow the user to test how loud the beeps are. Whenever the firmware wakes from a watchdog timeout it pulses an LED for a very short period of time. This provides a visual indication to the user that the device is functioning correctly. The LED is only on for about 10 milliseconds every 2 seconds. Because the duty cycle of the LED is so low it draw very little power. In fact because of the low duty cycle of beeping, the beeper should last for several months, perhaps as long as two years. Its current draw while sleeping is less than one microamp. A special test feature of the firmware is that it runs with a much shorter watchdog timeout period if the test button is pressed while a battery is inserted. This makes it easy to test software changes, otherwise it would take as long as two days to get through the initial count decrementing to zero. When running in special test mode, the device draws considerably more power, but operates about 128 times faster than it would in normal operation.

GRAPHICAL HARDWARE REPRESENTATION



MICROCHIP TOOLS USED:

Development Tools:

PICSTART® Plus

Assembler/Compiler version:

MPASM for Windows®, version 1.50

APPENDIX A: SOURCE CODE

```
; P12C508.INC  Standard Header File, Version 1.01    Microchip Technology, Inc.
    NOLIST

; This header file defines configurations, registers, and other useful bits of
; information for the PIC12C508 microcontroller.  These names are taken to match
; the data sheets as closely as possible.

; Note that the processor must be selected before this file is
; included.  The processor may be selected the following ways:

;       1. Command line switch:
;               C:\ MPASM MYFILE.ASM /P12C508
;       2. LIST directive in the source file
;               LIST   P=12C508
;       3. Processor Type entry in the MPASM full-screen interface

;==========================================================================
;
;      Revision History
;
;==========================================================================

;Rev:   Date:    Reason:

;1.01   08/21/96 Removed VCLMP fuse, corrected oscillators
;1.00   04/10/96 Initial Release

;==========================================================================
;
;       Verify Processor
;
;==========================================================================

        IFNDEF __12C508
           MESSG "Processor-header file mismatch.  Verify selected processor."
        ENDIF

;==========================================================================
;
;       Register Definitions
;
;==========================================================================

W                           EQU     H'0000'
F                           EQU     H'0001'

;----- Register Files ------------------------------------------------------

INDF                        EQU     H'0000'
TMR0                        EQU     H'0001'
PCL                         EQU     H'0002'
STATUS                      EQU     H'0003'
FSR                         EQU     H'0004'
OSCCAL                      EQU     H'0005'
GPIO                        EQU     H'0006'

;----- STATUS Bits ---------------------------------------------------------

PA2                         EQU     H'0007'
PA1                         EQU     H'0006'
PA0                         EQU     H'0005'
NOT_TO                      EQU     H'0004'
NOT_PD                      EQU     H'0003'
Z                           EQU     H'0002'
```

```
DC                          EQU     H'0001'
C                           EQU     H'0000'

;----- OPTION Bits --------------------------------------------------

T0CS                        EQU     H'0005'
T0SE                        EQU     H'0004'
PSA                         EQU     H'0003'
PS2                         EQU     H'0002'
PS1                         EQU     H'0001'
PS0                         EQU     H'0000'

;=========================================================================
;
;       RAM Definition
;
;=========================================================================

        __MAXRAM H'1F'

;=========================================================================
;
;       Configuration Bits
;
;=========================================================================

_MCLRE_ON                   EQU     H'0FFF'
_MCLRE_OFF                  EQU     H'0FEF'
_CP_ON                      EQU     H'0FF7'
_CP_OFF                     EQU     H'0FFF'
_WDT_ON                     EQU     H'0FFF'
_WDT_OFF                    EQU     H'0FFB'
_LP_OSC                     EQU     H'0FFC'
_XT_OSC                     EQU     H'0FFD'
_IntRC_OSC                  EQU     H'0FFE'
_ExtRC_OSC                  EQU     H'0FFF'

    LIST


Software listing:
(hard copy and electronic form)


;*************************************************************************
;
; JIMBO.ASM - Long period timer, with alarm.
;
;   Uses watchdog timer to wake up about every 2 seconds. It keeps track
;   of time, and after a very long time (about 36 hours), it beeps,
;   then goes back to sleep for several hours (about 6). It also has
;   2 additional outputs that can be used to power external noisemakers.
;   POUT (pin 6) goes positive when the beeper is making noise. NOUT
;   (pin 2) is like an open collector output that sinks current when the
;   beeper is sounding. NOUT can sink about 20 mAmp, POUT can source
;   about 10 mAmp.
;
;   Every 10th time the watchdog wakes the processor, the LED blinks
;   for a few milliseconds. It will be hard to see if you are not
;   looking for it. This provides an indicator that the timer is
;   operating correctly.
;
;   The TEST button is used for 2 tests. During normal operation,
;   if the button is pressed and released, it will start a beep cycle.
;   The beeper will run thru its normal sound, then the processor will
;   go back to sleep. If the TEST button is held down while the battery
;   is attached to the processor, the watchdog timer will run much faster
;   than it would in normal operation. This allows for testing of the
```

```
;   sllep and alarm sections of the code.
;
;       Port usage:
;
;               GP0 = Test button
;               GP1 = POUT
;               GP2 = Beeper output
;               GP3 = reset
;               GP4 = LED output
;               GP5 = NOUT
;
;       Configuration bits:
;               MCLRE = TRUE
;               CP = FALSE
;               WDTE = TRUE
;               FOSC = INTRC
;
;   Notes:
;       1 week = 604800 seconds (a 20 bit value)
;       6 hours = 21600 seconds (a 16 bit value)
;       1 week = 28 6 hour periods
;
; History:
;       Version         Date            Author
;
;       0.01            5/31/97        M R Hahn
;   Tested in TEST mode (about 128 times as fast as regular mode).
;   Appears to work. Tests indicate that the first beep should
;   happen between 35 and 51 hours after powerup. Subsequent beeps
;   should happen at 6 to 9 hour intervals. Haven't added the POUT
;   and NOUT signals to the code yet.
;
; Authors:
;
;       Mark R. Hahn
;       503-286-6125
;       hahndo@teleport.com
;
;*****************************************************************************
        ;
        ;
    list p=PIC12C508    ;
        ;
    include "p12c508.inc";
        ;
       __CONFIG        _IntRC_OSC & _WDT_ON & _MCLRE_ON & _CP_OFF
        ;
;       __IDLOCS        1234h ;
        ;
;INDF   equ     000h       ;index register
;TMR0   equ     001h        ;real time clock/counter
;PCL    equ     002h       ;program counter
;STATUS equ     003h       ;status register
;FSR    equ     004h        ;file select register
;OSCCAL equ     005h       ;oscilator calibration
;GPIO   equ     006h          ;IO port
        ;
        ;define STATUS flags
C_FLG           equ     0     ;carry
DC_FLG          equ     1     ;decimal carry
Z_FLG           equ     2     ;zero
PD_FLG          equ     3     ;power down
TO_FLG          equ     4     ;time out
RP0_FLG         equ     5     ;register page 0
RP1_FLG         equ     6     ;register page 1
GPWU_FLG        equ     7     ;wake up flag
GPWUF           equ     7     ;wake up flag
        ;
```

```
;W      equ     0           ;
;F      equ     1           ;
           ;
;special function registers
indf            equ     00h
tmr0            equ     01h
pcl             equ     02h
status          equ     03h
fsr             equ     04h
osccal          equ     05h
gpio            equ     06h

;*****************************************************************************
;
; RAM Definitions
;
;*****************************************************************************
               ;
Tmp0     equ007h     ;a temporary location
Tmp1     equ 008h      ;another temp location
Delay_cnt equ009h      ;
           ;
SixLoequ00Bh;low byte of sixes counter
SixHiequ00Ch;hi byte of sixes counter
           ;
Old_stateequ00Dh;
Stateequ00Eh;
#definesPowerupState,0;initial powerup state
#definesTestState,1;do a test beep
#definesCodeTestState,2;code test state
#definesAlarmState,3;keep track of time
           ;
Flagsequ00Fh;
#definefTestFlags,0;
           ;
fA  equ 010h;used by tdelay
fB  equ 011h;
           ;
Flash_cntequ012h;
Beep_cntequ013h;
           ;
SIXES_STARTequd'41';41 512 second periods = 6 hrs
SIXES_PWRUPequd'250';64000 2 second periods = 36 hrs
           ;
;bit assignments
c   equ     0       ;carry bit
w   equ     0       ;to indicate working register
z   equ     2       ;zero bit

;io assignments
#define         iTestgpio,0
#define         oPoutgpio,1
#define         oBeepgpio,2
#define         iResetgpio,3
#define         oLEDgpio,4
#define         oNoutgpio,5

;*****************************************************************************
;
; RAM Definitions
;
;*****************************************************************************
             ;
;*****************************************************************************
;
; VECTORS:
```

```
;       Stick some executable code in the RESET location.
;
;*****************************************************************************
        ;
    org     0           ;RESET vector
    goto    Start       ;jump to Start label
        ;
;*****************************************************************************
;
; Delay:
;       Delay a bit.
;
;*****************************************************************************
        ;
Delay   ;
    movlw   0C0h     ;set up length of delay
    movwf   Delay_cnt  ;
        ;
dellp   movlw   001h      ;set up length of delay
    movwf   Tmp0         ;
        ;
dell1   decfsz  Tmp0,1      ;dec count
    goto    dell1        ;loop till done
        ;
    clrwdt;keep watchdog happy
        ;
    decfsz  Delay_cnt,1  ;dec count
    goto    dellp  ;loop till done
        ;
    retlw   000h         ;
        ;
        ;
Big_delay             ;
        ;
    movlw   d'100'       ;
    movwf   Tmp1         ;
        ;
wloop   call    Delay     ;
    decfsz  Tmp1,F       ;
    goto    wloop        ;
           ;
    retlw   000h         ;
           ;
;*****************************************************************************
;
; tdelay:
;       Short delay routine.
;
;       Delays ((3 * 6) + 5) * W cycles.
;
;       W contains a value from 0 to 255
;
;*****************************************************************************
        ;
tdelay            ;
    movwf    fB    ;save count in fB
        clrwdt;keep watchdog happy
        ;
td1                           ;
    nop          ;timing fix for 4 Mhz clock
    nop             ;
    nop                ;
        ;
    movlw   d'6'   ;loop 6 times
    movwf   fA      ;
        ;
```

```
td2                                 ;
    decfsz  fA,1         ;dec loop counter
    goto    td2          ;3 * 6 cycles
            ;
    decfsz  fB,1        ;((3 * 6) + 5) * fB cycles total
    goto    td1          ;
            ;
    retlw   0            ;return
            ;
;*****************************************************************************
;
; Beep:
;       Sound the buzzer.
;
;*****************************************************************************
                ;
Beep                               ;
    movlwd'200';
    movwfBeep_cnt;
                ;
Beep_loop    ;
    bsf oBeep   ;
    movlwd'25'  ;
    calltdelay  ;
    bcf oBeep   ;
    movlwd'25'  ;
    calltdelay  ;
    decfszBeep_cnt,f;
    gotoBeep_loop;
                ;
    retlw   0                  ;return
                ;
;*****************************************************************************
;
; Flash:
;       Turn on LED.
;
;*****************************************************************************
                ;
Flash           ;
    movlwd'200';
    movwfFlash_cnt;
                ;
Flash_loop   ;
    bcf oLED    ;
    callDelay   ;
    decfszFlash_cnt,f;
    gotoFlash_loop;
                ;
    bsf oLED    ;
                ;
    retlw   0;return
                ;
;*****************************************************************************
;
; Start:
;       Start of the program. We start way up here since we need the low
;       page of memory for data tables and subroutines.
;
;*****************************************************************************
              ;
Start                          ;
    movf    STATUS,W;save status before it changes
    movwf   Old_stat    ;
              ;
    movlw   b'00001001'  ;set GP0, GP3 as inputs
```

```
    tris    GPIO          ;
              ;
    movlw   b'00010000'   ;turn off all outputs
    movwf   GPIO         ;
              ;
    btfssfTest;check if in TEST mode
    gotoNoTest;no, setup long watchdog
              ;
    movlwb'01001000';very short watchdog timeout
    option   ;
              ;
    gotoPassTest;
              ;
NoTest       ;
    movlw   b'01001111'  ;enable pullups, assign prescaler to WDT
    option               ;note: disables wake on input change
              ;
PassTest   ;
    clrwdt   ;clear the watch dog
              ;
    btfscOld_stat,GPWUF;check for input change caused reset
    gotoState_check;
              ;
    btfss   Old_stat,TO_FLG;check for a watchdog timeout
    goto    State_check    ;if TO_FLAG = FALSE (WDT happened) check state
              ;
    gotoPowerup;go do powerup state
              ;
;*****************************************************************
;
; State_check:
;   Figure out what state we are in, and then go to the handler
;   for that state.
;
;*****************************************************************
              ;
State_check;
    btfsciTest;check for test button pushed
    gotoTime_check;not pushed, check time
              ;
    btfscfTest;check if we are in test mode
    gotoTime_check;in test mode, do timekeeping
              ;
    callFlash;make some noise
    callBeep ;
    callFlash;
    callBeep ;
    callFlash;
    callBeep ;
              ;
    gotoBig_sleep;
              ;
;*****************************************************************
;
; Time_check:
;   Update timers, beep if it's time.
;
;*****************************************************************
              ;
Time_check  ;
    decfszSixLo,f;
    gotoNextTime;
              ;
    decfszSixHi,f;
    gotoNextTime;
              ;
```

```
    movlwSIXES_START;41 * 256 = 10500 2 sec periods = 6 hrs
    movwfSixHi;
    movlwd'0';
    movwfSixLo;
            ;
    callBeep;make some noise
    callBig_delay;
    callBeep;
    callBig_delay;
    callBeep;
    callBig_delay;
    callBig_delay;
    callBeep;
    callBig_delay;
    callBeep;
    callBig_delay;
    callBig_delay;
    callBig_delay;
    callBeep;
    callBig_delay;
    callBeep;
    callBig_delay;
    callBig_delay;
    callBig_delay;
    callBig_delay;
    callBeep;
    callBig_delay;
    callBeep;
    callBig_delay;
    callBeep;
    gotoBig_sleep;
            ;
NextTime   ;
    bcf oLED;turn LED on
    callDelay;
    bsf oLED;turn LED off
    gotoBig_sleep;
            ;
;*****************************************************************
;
; Powerup:
;   Flash LED, make beeper noise, check if TEST button pushed.
;
;*****************************************************************
            ;
Powerup      ;
    callFlash;make some noise
    callBeep ;
    callFlash;
    callBeep ;
    callFlash;
    callBeep ;
             ;
    bcf fTest ;
    btfssiTest;
    bsf fTest ;
             ;
    movlwSIXES_PWRUP;250 * 256 = 64000 2 sec periods = 36 hrs
    movwfSixHi;
    movlwd'0';
    movwfSixLo;
    gotoBig_sleep;
             ;
;*****************************************************************
;
; Big_sleep:
```

```
;   Enable wake on change, and read and latch inputs. Then put
;   processor to sleep.
;
;*****************************************************************
            ;
Big_sleep                   ;
;   call    Delay;delay for debouncing (make shorter later)
            ;
    clrwdt  ;
            ;
    movf    GPIO,W        ;read and latch inputs
            ;
    nop                   ;
    nop                  ;
            ;
    sleep                ;goto sleep
            ;
    nop                  ;these probably are not necessary
    nop                   ;
    goto    Start        ;start over
            ;
    end                ;
```

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