APPLICATION OPERATION

These three applications are designed to replace AC electromechanical keys. They must not be used for DC because of the use of triacs. All of the applications may be powered directly from AC.

Application 1: Electronic Key

The application shown in Figure 1 can replace almost all electromechanical keys. It is synchronized with the line voltage so the charge is switched on only in the beginning of the half period. It may be used in drills, fans and many electrical machines used at home.

As you can see from Figure 1, the program makes a loop, where it is first waiting for the beginning of the first half period. When this is reached, the button is tested. If it is pushed, it switches on the triac else does not switch on the triac. After that, it waits for the beginning of the second half period and so on.

Application 2: Button Dimmer Controller

The application shown in Figure 2 may be used as a dimmer or revolution controller. Button 1 increases power and button 2 decreases power. To make the input/output linear, I am using a table to convert the input value. The half period is divided to 64 areas the surface of which are equal:

\[
\frac{\int_{X_n}^{X(n+1)} \sin(x) \, dx}{\int_{X_n}^{X(n+1)} \sin(x) \, dx} = \frac{154}{156}
\]

The TMR0 is cleared every time at the beginning of the half period. The Prescaler of the timer is set to divide by 64 so when Fosc = 4MHz, the value that will be in TMR0 at the end of the half period will be:

\[
\frac{10 \text{ms (half period for 50Hz net)}}{64 \times 1 \mu\text{s}} = 156
\]

The maximum value that is returned from the table is 154. So there is a cycle that compares the value of the timer and the value returned from the table. When the value of the timer goes greater than the output, it is activated, and, in the rest of the time to the end of the half period the buttons are read. If the controller is set to minimum, there will not be time for button read. That's why the table doesn't return 156 as maximum but 154. This is not very important in the beginning and in the end of the half period.

When the button is pushed, an action is performed (increase or decrease). If the button is still held down after about 1 second, the second action will be performed. After that, the time to the next action is about 0.1 second. If the max/min value is reached, no action is performed.

Application 3: Potentiometer Dimmer Controller

The application shown in Figure 3 is similar to application 2, but there is a potentiometer instead of buttons. This uses the A/D converter from the PIC12C671. The A/D conversion is started in the end of the half period and the ADRES register is read when the GO/DOWN bit is down.
FIGURE 1: POWERING WITHOUT TRANSFORMER

- VCC = 5V
- Vz = 5V
- 220 nF
- 1 KΩ, 1 W
- 470 KΩ
- 220V AC
- GP0
- GP2
- GP3
- PIC12C508
- SW1
- 240 Ω
- 100 Ω
- 100 nF
- OUT
- +
- 3.3 KΩ
- NPN
- SYNC Signal
FIGURE 2: BUTTON DIMMER CONTROLLER

FIGURE 3: POTENTIOMETER DIMMER CONTROLLER
FIGURE 4: APPLICATION 1
ELECTRONIC KEY

FIGURE 5: APPLICATION 2 AND 3
BUTTON DIMMER AND
POTENTIOMETER
CONTROLLERS

MICROCHIP TOOLS USED:

Development Tools:
PICSTART® Plus V1.20

Assembler/Compiler Version:
MPLAB V3.22, MPASM V1.5
APPENDIX A: SOURCE CODE

;****************************************************************
; Figure1.ASM
;
;****************************************************************

LIST    p=12C508
#include "inc\p12c508.inc"

__config _WDT_ON & _IntRC_OSC & _MCLRE_OFF & _CP_OFF

Sync   equ         0
In     equ         3
Out    equ         2

org 0x00

movwf OSCCAL ; calibrating the internal oscillator
clr GPIO
movlw B'00111011'
TRIS GPIO
movlw B'01000010'
OPTION

btfsc GPIO,Sync
goto $-1
loop
clrwdt

btfss GPIO,Sync
goto $-1

btfsc GPIO,In
bcf GPIO,Out
btfss GPIO,In
bsf GPIO,Out
clrwdt

btfsc GPIO,Sync
goto $-1

btfsc GPIO,In
bcf GPIO,Out
btfss GPIO,In
bsf GPIO,Out

goto loop

end

;****************************************************************
; Figure2.ASM
;
;****************************************************************

; This application is made for power nets AC220V 50Hz
; Don't use it on 60Hz!
; It has not back reference to control the current in the triac
; so it should not be used to drive reactive charges (solenoids etc.),
; where there is big phase difference between the
; voltage and the current!
;
;****************************************************************

LIST p=12C508

#include "inc\p12c508.inc"

__config _WDT_ON & _IntRC_OSC & _MCLRE_OFF & _CP_OFF

RAM equ 0x07 ;Beginning of RAM
Sync equ 0
Btn1 equ 1
Btn2 equ 3
Out equ 2

cblock RAM

BtnCount1 ;Counters used to delay when button is pushed
BtnCount2
Phase ;The value got from the Table
Dimmer
Flag

endc

org 0x00

movwf OSCCAL ;calibrating the internal oscillator

goto main

; This table makes the dependence y=sin(x) linear so, if you
; use this program to control dimmers and if you increment x by 1
; up to 63 (0 - 0x3F) and measure the light with luxmeter the dependence
; will be linear
;

Table: andlw .63
addwf PCL,F
retlw .154
retlw .151
retlw .146
retlw .141
retlw .136
retlw .132
retlw .129
retlw .125
retlw .122
retlw .119
retlw .117
retlw .114
retlw .112
retlw .110
retlw .108
retlw .107
retlw .105
E lectromechanical S witch R eplacement

```
retlw .103
retlw .102
retlw .100
retlw .99
retlw .97
retlw .96
retlw .94
retlw .93
retlw .92
retlw .90
retlw .89
retlw .88
retlw .87
retlw .85
retlw .84
retlw .83
retlw .82
retlw .80
retlw .79
retlw .78
retlw .77
retlw .76
retlw .74
retlw .73
retlw .72
retlw .71
retlw .69
retlw .68
retlw .67
retlw .66
retlw .64
retlw .63
retlw .62
retlw .60
retlw .58
retlw .56
retlw .54
retlw .52
retlw .50
retlw .48
retlw .45
retlw .42
retlw .38
retlw .34
retlw .30
retlw .21
retlw .0

main
  clrф  GPIO
  movlw B'00111011'
  TRIS GPIO
  movlw B'01000101' ;clockout/64
  OPTION

;-------------------------------------------------------------------
; When 50Hz net is used the period is 20ms and
; the half period is 10ms. 64us X 156 =9.984ms
; the maximum value that the table gives is 154

  movlw .100
```
movwf BtnCount1 ;Initialize the button counters
movwf BtnCount2 ;there will be about 1 sec delay when you
;push some button. When you hold the putton
;the output will change within about 0.1 sec

btfsc GPIO,Sync
goto $-1

loop
clrwdt
btfss GPIO,Sync ;loops while Sync=0
goto $-1

clrf TMR0

; First half period

movf Dimmer,w
call Table ;converts the value in Dimmer to Phase
movwf Phase

L1
movf Phase,w ;compares the Phase with the timer
subwf TMR0,w ;when the time has come swithes the ouput on
btfs STATUS,C
goto Lb1

bsf GPIO,Out ;output on

;Button 1

Btn1
btfs GPIO,Btn1 ;skip if button 1 is pushed else
goto Btn1Enda ;inicializaes the BtnCount1

decfsz BtnCount1,F ;if the button was held down for about 1 sec
goto BtnEnda ;(when pushed) or 0.1 sec (after the
movlw .10 ;first sec) the value of Dimmer is
movwf BtnCount1 ;incremented
incf Dimmer,F ;if the highest value (0x3F) is reached
btfs Dimmer,6 ;no more incrementation is done
decf Dimmer,F
goto BtnEnda

Btn1Enda
movlw .100 ;if the button was not pushed BtnCount is
movwf BtnCount1 ;inicialized
goto BtnEnda

;the algorythm downwards is like the above

;Button 2

Btn2
btfs GPIO,Btn2
goto Btn2Enda

decfsz BtnCount2,F
goto BtnEnda
movlw .10
movwf BtnCount2

```assembly
Decf Dimmer, F
Btfsc Dimmer, 6
Incf Dimmer, F
Goto BtnEnda

Btm2Enda
Movlw .100
Movwf BtnCount2

BtnEnda
Crlwdt
Bcf GPIO, Out

; Second half period
Btfsc GPIO, Sync ; loops while Sync=1
Goto $-1
Clrf TMR0
Movf Dimmer, w
Call Table
Movwf Phase

Lb1
Movf Phase, w
Subwf TMR0, w
Btfsc STATUS, C
Goto Lb1
Bsf GPIO, Out

Btnb1
Btfsc GPIO, Btn1
Goto Btm1Endb
Decfsz BtnCount1, F
Goto BtmEndb
Movlw .10
Movwf BtnCount1
Incf Dimmer, F
Btfsc Dimmer, 6
Decf Dimmer, F
Goto BtmEndb

Btm1Endb
Movlw .100
Movwf BtnCount1

Btnb2
Btfsc GPIO, Btn2
Goto Btm2Endb
Decfsz BtnCount2, F
Goto BtmEndb
Movlw .10
Movwf BtnCount2
Decf Dimmer, F
Btfsc Dimmer, 6
Incf Dimmer, F
Goto BtmEndb
```
E lectromechanical Switch Replacement

LIST    p=12C671
#include "inc\p12c671.inc"

__config _WDT_ON & _IntRC_OSC & _MCLRE_OFF & _CP_OFF

RAM equ 0x07 ;Begining of RAM

In equ 0
Sync equ 3
Out equ 2

cblock RAM

BtnCount1 ;Counters used to delay when button is pushed
BtnCount2
Phase ;The value got from the Table
Dimmer
Flag
endc

org 0x00

call EndAdd
movwf OSCCAL ;calibrating the internal oscillator
goto main

;*******************************************************************************
; Figure3.ASM
;
; This application is made for power nets AC220V 50Hz
; Don't use it on 60Hz!
; It has not back reference to control the current in the triac
; so it should not be used to drive reactive charges (solenoids etc.),
; where there is big phase difference between the
; voltage and the current!
;
;*******************************************************************************

Btn2Endb
movlw .100
movwf BtnCount2

BtnEndb

bcf GPIO,Out

goto loop

don't use this code!
| Table: andlw .63  
| addwf PCL,F  
| retlw .154  
| retlw .151  
| retlw .146  
| retlw .141  
| retlw .136  
| retlw .132  
| retlw .129  
| retlw .125  
| retlw .122  
| retlw .119  
| retlw .117  
| retlw .114  
| retlw .112  
| retlw .110  
| retlw .108  
| retlw .107  
| retlw .105  
| retlw .103  
| retlw .102  
| retlw .100  
| retlw .99  
| retlw .97  
| retlw .96  
| retlw .94  
| retlw .93  
| retlw .92  
| retlw .90  
| retlw .89  
| retlw .88  
| retlw .87  
| retlw .85  
| retlw .84  
| retlw .83  
| retlw .82  
| retlw .80  
| retlw .79  
| retlw .78  
| retlw .77  
| retlw .76  
| retlw .74  
| retlw .73  
| retlw .72  
| retlw .71  
| retlw .69  
| retlw .68  
| retlw .67  
| retlw .66  
| retlw .64  
| retlw .63  
| retlw .62  
| retlw .60  
| retlw .58  
| retlw .56  
| retlw .54  
| retlw .52  
| retlw .50  
| retlw .48  
| retlw .45  
| retlw .42  
| retlw .38  
| retlw .34  
| retlw .30  
| retlw .21 |
Electromechanical Switch Replacement

```
retlw .0

main
    clr gif GPIO
    clr gif INTCON
    bsf STATUS, RP0
    movlw B'00111011'
    movwf TRIS
    movlw B'01000101' ; clockout/64
    movwf OPTION_REG
    movlw B'00000110'
    movwf ADCON1
    bcf STATUS, RP0

;---------------------------------------
; When 50Hz net is used the period is 20ms and
; the half period is 10ms. 64us X 156 = 9.984ms
; the maximum value that the table gives is 154

    movlw .100
    movwf BtnCount1 ; Initialize the button counters
    movwf BtnCount2 ; there will be about 1 sec delay when you
                    ; push some button. When you hold the button
                    ; the output will change within about 0.1 sec
    movlw B'01000001' ; Inializes ADC cnannel 0 (GP0)
    movwf ADCON

    btfsc GPIO, Sync
    goto $-1

loop
    clrwdt
    btfss GPIO, Sync ; loops while Sync=0
    goto $-1

; First half period

    clr gif TMR0
    movf Dimmer, w
    call Table ; converts the value in Dimmer to Phase
    movwf Phase

La1
    movf Phase, w ; compares the Phase with the timer
    subwf TMR0, w ; when the time has come switch the output on
    btfsc STATUS, C
    goto Lb1
    bsf GPIO, Out ; output on
    bsf ADCON, GO
```
btfsc ADCON,GO
goto $-1

bcf STATUS,C
rrf ADRES,W
movwf Dimmer
bcf STATUS,C
rrf Dimmer,F

clrwdt
bcf GPIO,Out

; Algorythm for second half period is the same as in the first half period

; Second half period

btfsc GPIO,Sync ; loops while Sync=1
goto $-1

clrf TMR0
movf Dimmer,w
call Table
movwf Phase

Lb1
movf Phase,w
subwf TMR0,w
btfsc STATUS,C
goto Lb1
bsf GPIO,Out

bsf ADCON,GO
btfsc ADCON,GO
goto $-1

bcf STATUS,C
rrf ADRES,W
movwf Dimmer
bcf STATUS,C
rrf Dimmer,F

bcf GPIO,Out
goto loop

IFDEF __12C671
EndAdd org 0x3FF
ELSE
IFDEF __12C672
ENDIF
EndAdd
ENDIF
end