MUSIC BOX

Home

PIC Programmer MkV

Instruction Set for PIC12F629
PIC12F629 data (pdf)
BlankF629.asm

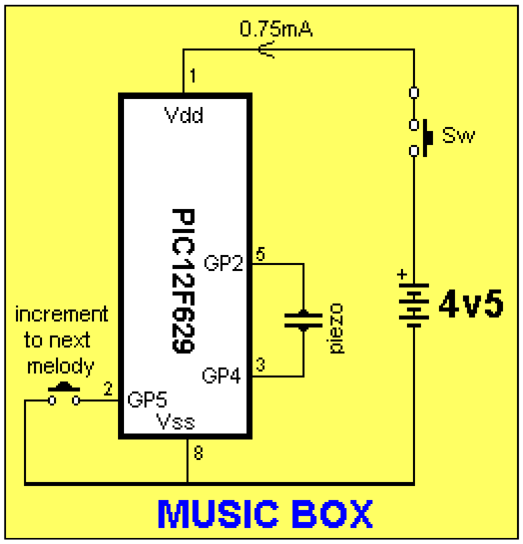
PIC12F629.inc

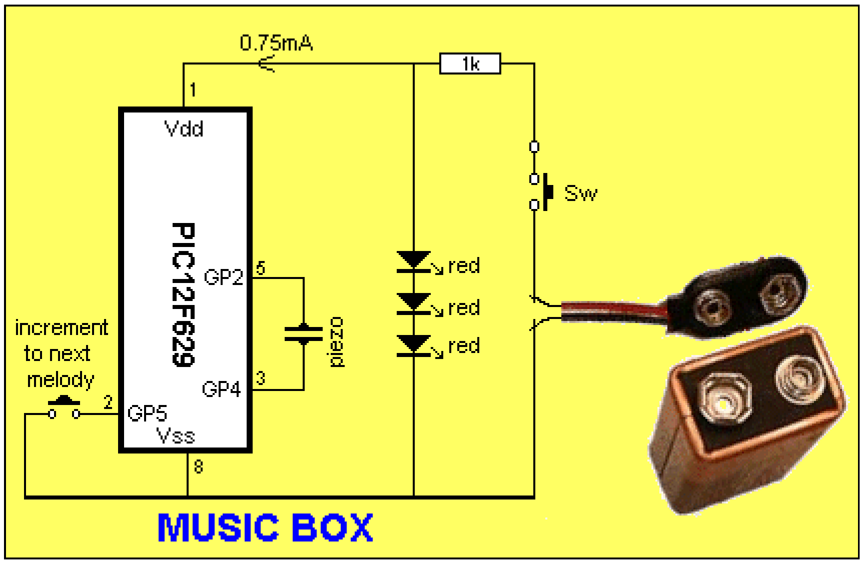
See more projects using micros:
Pic A PIC Project

Notepad2.zip Notepad2.exe
Library of Sub-routines "Cut and Paste"
Library of routines: **A-E** **E-P** **P-Z**

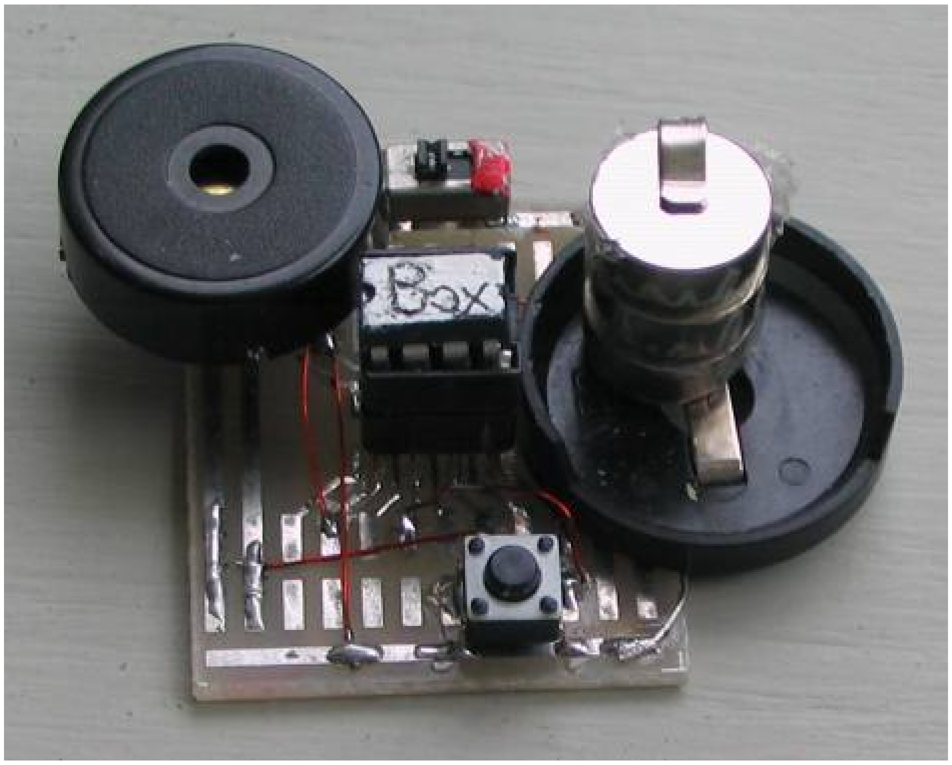
Kits: **Music Box**

This project is an extension of a number of musical projects (Happy Birthday and It's a Small World) and puts 11 melodies into a single design.
It's called EVOLUTION.
From the previous projects we learnt a lot about producing a tune.
The first thing we learnt: it takes a lot of memory.
Each note needs one, two, or even three bytes and this severely limits the length of the tune, as the PIC12F629 has a maximum of 256 bytes for a table and this can only be placed in the first page of memory (the chip does not have the facility to access a table in any other part of memory).
We had to re-design the program so all of the 1024 locations of program-space could be used.
This was done by omitting tables and using the program-space for the data-bytes.
We reduced the data-requirement further by requiring only a single byte to produce the length of the HIGH. The length of the LOW was obviously the same duration.
But now we had a problem.
To produce a note of say 300mS, we had to supply data for the number of cycles.
Obviously a high frequency needs more cycles for 300mS, than a low-frequency note.
The answer was to complement the data-bye.
This produced a result very near to 300mS.
The only other thing we had to produce was a note of a suitable length.
This was done by shifting the file right (rrf) to get a value of 50% and adding it to the original value, plus performing other operations to generate the correct length.
This has changed the capacity of the project from 2 tunes to 11, with NO bytes to spare.
You cannot use location 3FF for your program as it is used by the micro to store the oscillator value.
We have also included a button to increment through the tunes.
When the project is first turned on, it will play each tune twice. If the switch is pressed, the project will go to the first tune and it will be repeated. If it is pressed again, the second tune will be selected and repeated. See below for more details on this.
The melodies are recognisable but some of the notes are very hard to reproduce in monotone and we have left it up to an aspiring musical person to alter the tones to create an improvement.



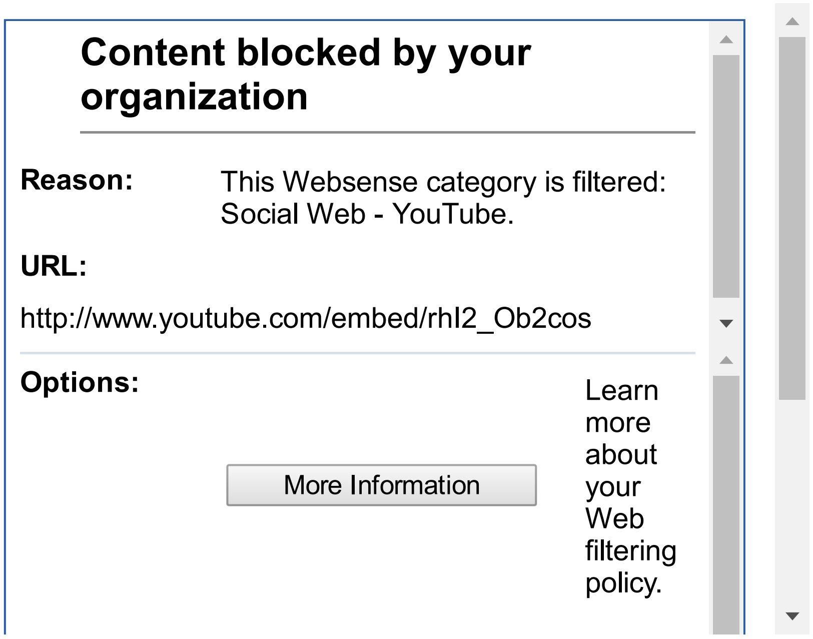


Music Box project with battery snap and 9v battery
The 3 red LEDs regulate the supply to 5v



Music Box Project on a prototype PC board

Here is the Music Box added to a Fox-Hunt transmitter by Bjorn Dinse PA4BWD of the Netherlands email: bwewdinse@quicknet.nl



Content blocked by your organization

Reason: This Websense category is filtered: Social Web - YouTube.

URL:

http://www.youtube.com/embed/rhI2_Ob2cos

Options:

Learn more about your Web filtering policy.

THE CIRCUIT

The circuit uses just a few components on a small prototype PC board. It's easy to construct and ideal for a beginner. It's just a PIC chip, a piezo diaphragm and a switch. You can put it together in a few minutes on a prototype board and download the program from the website.
The pull-up resistor for the switch is inside the chip and this saves one component. We have not placed a 100n across the chip and this is another component saved. The circuit takes less than 1mA and everything else is inside the chip via a program.
The photo shows the prototype.
If you want to buy a kits, there are two versions:

Version 1 comes as a piece of matrix board consisting of solder lands. After placing and soldering the components to the lands, they are joined with fine tinned copper wire to represent the tracks of a printed circuit board.
Version 2 has a PC board and a battery snap for a 9v battery. Three LEDs on the board regulate the voltage to 5v and also act as an "ON" indicator.
Music Box has a memory feature. Any of the tunes can be selected and when the project is turned on again, the selected melody will repeat.
Simply increment through the melodies via the button and turn the project off. Turn the project on to hear the selected melody.
To erase the selection, push the button when the project is off and turn it on. Release the button immediately and the program will start at the first melody and play each twice.
This feature uses the EEPROM and the program below shows exactly how to use these instructions to perform read and write operations.
When creating your own program, these instructions should always be copied and pasted to prevent a mistake.
The full 1024 program locations have been used in this project plus the first location in EEPROM.

PRODUCING A TONE

Producing a tone is very easy. All you have to do is make an output go HIGH then LOW at a controlled rate.
The output goes HIGH and stays HIGH for a certain number of microseconds then goes LOW and stays LOW for the same time. If the HIGH and LOW times are not the same, the tone is not very "clean" and does not have a "ring" about it.
In our case, we have connected the piezo diaphragm to two outputs to increase the volume.
When one output goes HIGH the other goes LOW and vise versa. This causes the piezo to see an increased voltage because the piezo is actually a capacitor of about 22n and when it gets a voltage on one direction, it charges to a voltage equal to the applied voltage.
To make the discussion easy to understand, let's say one lead is fixed at 0v, and the other lead is now supplied with a reverse voltage. The potential of the other lead will change from say a positive voltage to a negative voltage. This is a swing of twice the value of the supply and the additional voltage produces a louder output.

THE PROGRAM

The program is large because each note in a melody requires two bytes. The first instruction loads w with a value and the second sends the micro to the "oscillator" sub-routine.
If a table was used, it would require only one byte, but a table in a PIC12F629 can only occupy the first 256 bytes of the program-space and we need 4 times this length.
The instructions for producing a tone are very simple.
Turn ON an output and create a delay. Turn OFF the output and create the same delay.
When a switch is introduced to a program, we need to be able to detect it very quickly and the only place is to look is within the tone routine.
Next we had to analyse the frequency of the notes and work out the number of microseconds for the HIGH and LOW for each note. This has already been mentioned above and everything is straight-forward until the "Memory" section is created.
This involves using the EEPROM. This is a separate section within the microcontroller consisting of 128 locations that hold information and retain it after power is removed. The program memory consists of 1024 locations that cannot be altered after the chip is "burnt." The chip also has 64 locations called files or registers that can be altered during the running of the program, but lose memory when power is removed. These are the three types of memory.
The first EEPROM location is loaded with 00 when the chip is burnt and this location is looked at to see if the program goes to a requested melody or if the 11 tunes are played in sequence.
If the button is pressed, the current melody number is stored in EEPROM so that if the project is turned off, it will return to the selected tune, when turned on.
Although these EEPROM routines are simple, they must be copied from data sheets to be sure the instructions are correct.
We have used only easy-to-understand instructions for all the other subroutines and once this is done, the program is complete.

Note	Frequency (Hz)
C_3	130.81
$C^{\#}_3/D^b_3$	138.59
D_3	146.83
$D^{\#}_3/E^b_3$	155.56
E_3	164.81
F_3	174.61
$F^{\#}_3/G^b_3$	185.00
G_3	196.00
$G^{\#}_3/A^b_3$	207.65
A_3	220.00
$A^{\#}_3/B^b_3$	233.08
B_3	246.94
C_4	261.63
$C^{\#}_4/D^b_4$	277.18
D_4	293.66
$D^{\#}_4/E^b_4$	311.13
E_4	329.63
F_4	349.23
$F^{\#}_4/G^b_4$	369.99
G_4	392.00
$G^{\#}_4/A^b_4$	415.30
A_4	440.00
$A^{\#}_4/B^b_4$	466.16
B_4	493.88
C_5	523.25
$C^{\#}_5/D^b_5$	554.37
D_5	587.33
$D^{\#}_5/E^b_5$	622.25
E_5	659.26
F_5	698.46
$F^{\#}_5/G^b_5$	739.99
G_5	783.99
$G^{\#}_5/A^b_5$	830.61
A_5	880.00
$A^{\#}_5/B^b_5$	932.33
B_5	987.77
C_6	1046.50

The frequency for each note

C_4 = middle C

Here are the files:
MusicBox.asm
MusicBox.hex

You must use the .hex file to "burn" the chip or the .asm file (if you want to modify the program) as these are laid out so the compiler and programmer can understand the data. The following program is just for viewing and includes only the first melody.

```
;************************************************************
;MUSIC BOX TUNES                                            *
;  3-6-2010                                                 *
;Multi tune Music Box without tables - uses whole of memory!*
;Press Sw to increment to next tune                         *
;************************************************************
;
;       --+------------ +5v
;         |                 |  |
;    +--- | ----------------|[]|----+
;    |    |Vdd ---v---      |  |    |
;    |    +---|1   Gnd|     piezo   |
;    |        |       |             |
;    |  +-----|GP5 GP0|             |
;    |  |     |       |             |
;    +--------|GP4 GP1|             |
;       |     |       |             |
;       |     |GP3 GP2|--------------+
;       |      -------
;       o     PIC12F629
;  Sw    /
;       /
;      |
;     -+------------- 0v

        list    p=12F629
        radix   dec
        include "p12f629.inc"

        errorlevel      -302    ; Don't complain about BANK 1 Registers

        __CONFIG        _MCLRE_OFF & _CP_OFF
                     & _WDT_OFF & _INTRC_OSC_NOCLKOUT  ;Internal osc.


;========================
;
;    Equates
;
;========================

note    equ     21h ;value of HIGH and LOW for note
gap     equ     22h ;gap between notes - uses delay "gap_1"
loops   equ     23h ;loops of HIGH/LOW for 250mS or other duration
temp1   equ     24h ;temp file for note
melody  equ     25h ;counter for melody for switch
D1      equ     26h ;used in 250mS delay
D2      equ     27h ;used in 250mS delay

gapDel  equ     29h ;used in gap delay
tempA   equ     2Ah ;used in gap delay


;****************************************************************
;Beginning of program
```

```
;****************************************************************
                org             0x00
SetUp   bsf             status, rp0     ;Bank 1
        movlw   b'00101011'             ;Set TRIS
        movwf   TRISIO              ;GP2,4 outputs      GP5 input
        bcf             option_reg,7    ;pull-ups enabled
        bcf             status, rp0             ;bank 0
        movlw   07h             ;turn off Comparator
        movwf   CMCON           ;must be placed in bank 0
        clrf    melody          ;jump value for melody table
        btfsc   gpio,5
        goto    readEEPROM      ;read EEPROM at start-up
        movlw   00      ;If switch pressed at turn-on; clear EEPROM
        call    write
        goto    SetUp

;*******************
;* Delays          *
;*******************


        ;gap_1 produces the gap between notes each unit is 1mS

gap1    movlw   .30
        movwf   gapDel
        nop
        decfsz  tempA,1
        goto    $-2
        decfsz  gapDel,1 ;produces loops
        goto    $-4
        retlw   00

        ;extra pause between notes

pause   movlw   .60
        movwf   D2
        nop
        decfsz  D1,1
        goto    $-2
        decfsz  D2,1
        goto    $-4
        retlw   00


        ;250mS second delay

_250mS  nop
        goto    $+1
        decfsz  D1,1
        goto    _250mS
        decfsz  D2,1
        goto    _250mS
        retlw   00

_1Sec   call    _250mS
        call    _250mS
_500mS  call    _250mS
        call    _250mS
        retlw   00


;***********************
;* Subroutines         *
;***********************
```

```
        ;produces note length

length1 movwf   temp1   ;put note length HIGH into "temp1"
        movwf   loops   ;create number of loops to produce .25sec
        comf    loops,f ;complement note value to get loops value
        clrc            ;clear carry before shifting
        rrf     loops,f ;halve the value of loops
        clrc
        rrf     loops,w ;halve the value of loops again and put into w
        addwf   loops,w   ;to get 0.75 of original
len1_a  movf    temp1,w
        movwf   note
        bsf     gpio,2
        bcf     gpio,4
        goto    $+1
        goto    $+1
        goto    $+1
        nop
        decfsz  note,1
        goto    $-5
        movf    temp1,w
        movwf   note
        bcf     gpio,2
        bsf     gpio,4
        goto    $+1
        btfss   gpio,5
        goto    switch
        goto    $+1
        nop
        decfsz  note,1
        goto    $-6
        decfsz  loops,f
        goto    len1_a
        call    gap1    ;gap between notes
        retlw   00

        ;produces note length



length2 movwf   temp1   ;put note length HIGH into "temp1"
        movwf   loops   ;create number of loops to produce .25sec
        comf    loops,f ;complement note value to get loops value
        goto    len1_a



        ;produces note length

lengthX movwf   temp1   ;put note length HIGH into "temp1"
        movlw   60h
        movwf   loops   ;
        goto    len1_a

        ;produces long note length  for Happy Birthday


length2X
        movwf   temp1   ;put note length HIGH into "temp1"
        movlw   0FFh
        movwf   loops   ;
        goto    len1_a

        ;read melody number from EEPROM at turn-on
```

```
readEEPROM
        bsf     status,rp0
        clrf    EEADR           ;to read first location in EEPROM !!!
        bsf     EECON1,0  ;start EEPROM read operation - result in EEDATA
        movf    EEDATA,w        ;move read data into w
        bcf     status,rp0
        movwf   melody          ;jump value for melody table
        movlw   00
        xorwf   melody,w
        btfss   status,z
        goto    sw_M
        goto    Main


switch  call    _500mS
        incf    melody,f
        incf    melody,f        ;jump 2 bytes at a time on table
        movf    melody,w
        call    write           ;store melody value in EEPROM
sw_M    movf    melody,w
        addwf   02,1            ;Add W to the Program Counter for jump
        call    M1
        goto    $-1
        call    M2
        goto    $-1
        call    M3
        goto    $-1
        call    M4
        goto    $-1
        call    M5
        goto    $-1
        call    M6
        goto    $-1
        call    M7
        goto    $-1
        call    M8
        goto    $-1
        call    M9
        goto    $-1
        call    M10
        goto    $-1
        call    M11
        goto    $-1
        nop
        btfss   gpio,5
        goto    $-1
        call    _250mS
        clrf    melody
        goto    Main

write   bsf     status,rp0      ;select bank1
        clrf    eeadr           ;to load into first location
        movwf   eedata          ;w will have melody value
        bsf     eecon1,wren     ;enable write
        movlw   55h             ;unlock codes
        movwf   eecon2
        movlw   0aah
        movwf   eecon2
        bsf     eecon1,wr       ;write begins
        bcf     status,rp0      ;select bank0
writeA  btfss   pir1,eeif       ;wait for write to complete
        goto    writeA
        bcf     pir1,eeif
        bsf     status,rp0      ;select bank1
        bcf     eecon1,wren     ;disable other writes
        bcf     status,rp0      ;select bank0
        retlw   00


;************************
;*Melodies              *
;************************

        ;It's A Small World

M1      movlw   .151            ;It's
        call    length1
        movlw   .142            ;a
        call    length1
        movlw   .128            ;world
        call    length2
        movlw   .75             ;of
        call    length2
        movlw   .95             ;laugh-
        call    length2
        movlw   .84             ;-ter
        call    length1
        movlw   .95             ;a
        call    length1
        movlw   .95             ;world
        call    length2
        movlw   .102            ;of
        call    length2
        movlw   .102            ;tears
        call    length2
        movlw   .172            ;It's
        call    length1
        movlw   .151            ;a
        call    length1
        movlw   .142            ;world
        call    length2
        movlw   .84             ;of
        call    length2
        movlw   .102            ;hopes
        call    length2
        movlw   .95             ;and
        call    length1
        movlw   .102            ;a
        call    length1
        movlw   .113            ;world
        call    length2
        movlw   .128            ;of
        call    length2
        movlw   .128            ;fears
        call    length2
        movlw   .151            ;There's
        call    length1
        movlw   .142            ;so
        call    length1
        movlw   .128            ;much
        call    length2
        movlw   .95             ;that
        call    length1
        movlw   .84             ;we
        call    length1
        movlw   .75             ;share
        call    length2
        movlw   .84             ;that
        call    length1
        movlw   .95             ;it's
        call    length1
        movlw   .113            ;time
        call    length2
        movlw   .84             ;we're
        call    length1
        movlw   .75             ;a
        call    length1
        movlw   .71             ;ware
        call    length2
        call    pause
        movlw   .75             ;It's
        call    length1
        movlw   .102            ;a
        call    length1
        movlw   .113            ;small
        call    length2
        movlw   .71             ;world
        call    length2
        movlw   .75             ;aft-
        call    length2
        movlw   .84             ;-ter
        call    length1
        movlw   .95             ;all
        call    length2
        call    _250mS
        movlw   .95             ;It's
        call    length2
        movlw   .95             ;a
        call    length1
        movlw   .75             ;small
        call    length2
        movlw   .95             ;world
        call    length2
        movlw   .84             ;aft-
        call    length2
        movlw   .84             ;-ter
        call    length1
        movlw   .84             ;all
        call    length2
        call    _250mS
        movlw   .84             ;It's
        call    length2
        movlw   .84             ;a
        call    length1
        movlw   .71             ;small
        call    length2
        movlw   .84             ;world
        call    length2
        movlw   .75             ;aft-
        call    length2
        movlw   .75             ;-ter
        call    length1
        movlw   .75             ;all
        call    length2
        call    _250mS
        movlw   .75             ;It's
        call    length2
        movlw   .75             ;a
        call    length1
        movlw   .64             ;small
        call    length2
        movlw   .71             ;world
        call    length2
        movlw   .71             ;aft-
        call    length2
        movlw   .71             ;-ter
        call    length1
        movlw   .71             ;all
        call    length2
        movlw   .75             ;It's
        call    length1
        movlw   .84             ;a
        call    length1
        movlw   .128            ;small
        call    length2
        movlw   .102            ;small
        call    length2
        movlw   .95             ;world
        call    length2
        call    _1Sec
        retlw   00



;************************
;*Main                  *
;************************


Main    call    M1
        call    M1
        call    M2
        call    M2
        call    M3
        call    M3
        call    M4
        call    M4
        call    M5
        call    M5
        call    M6
        call    M6
        call    M7
        call    M7
        call    M8
        call    M8
        call    M9
        call    M9
        call    M10
        call    M10
        call    M11
        call    M11
        goto    SetUp


;************************************
;*EEPROM                            *
;************************************

        org             2100h

        de              00h,

        END
```

GOING FURTHER

This project provides you with the tools to create your own tune or download one of the "old favourites" and annoy everyone in the household with its incessant playing. The program-space is completely full so you will have to delete one of the melodies to create space for your new tune.

3/6/2010