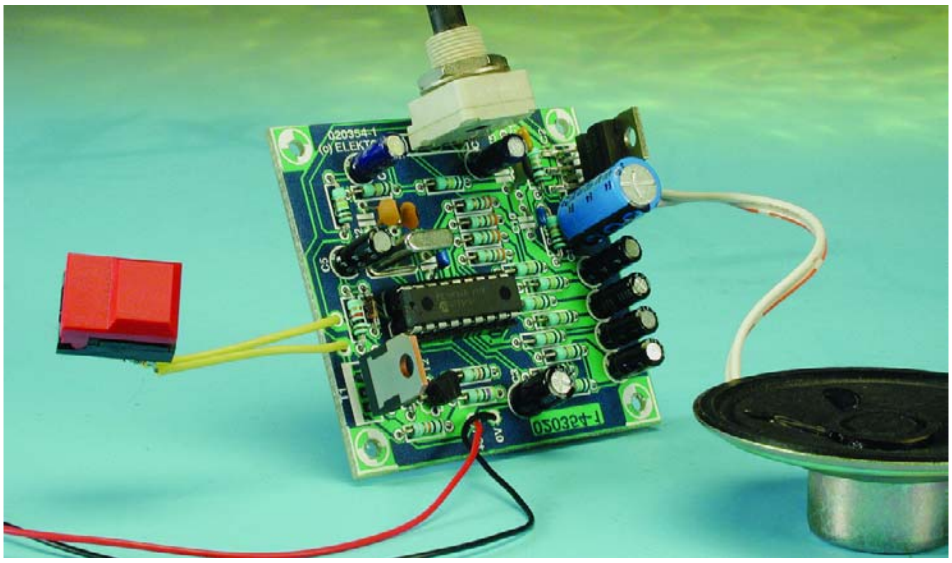
Polyphonic Doorbell

Several tunes from a PIC 16F84

Design by D. Manier

Music has been defined as ‘the art of combining sounds in a manner that is agreeable to the ear’. We thought would be a nice idea to entrust this delicate task to the best-known microcontroller in the Microchip stable, the PIC 16F84.



This circuit, which can serve as a musical doorbell or music generator, is built around a microcontroller, and it can be configured by the user. You’ll be amazed by its virtuosity and the quality of the (polyphonic) sound it produces.

As a doorbell, this circuit has several attractive features:
– very good sound
– eight different tunes
– zero current when quiescent
– customisable tunes
– reasonable construction cost

The concept

Our electronic musician must perform the following tasks:

1. Switch on the power when the doorbell is pressed, and reduce the current consumption to zero at the end of the tune.
2. Generate musical notes (up to four at the same time), generate envelopes for these notes and mix them together, all based on a short tune stored in memory.
3. Store the number of the most recently played tune in memory and then switch to the next tune. The tune number is stored in EEPROM, so the information is retained even in the absence of power.
4. Amplify the signal for output to a loudspeaker.

The first three tasks (power management, music generation and data storage) are entrusted to the PIC 16F84. For amplification, we use a special audio IC, the TDA2006. The volume of the sound can be set using a potentiometer. The impedance of the connected loudspeaker may lie between 4 and 8 ohms.

Some basic terms

First we need to describe a few basic musical terms.

When a musician plays a piece of music, he first reads the score. The score describes the notes to be played and their respective durations.

The location of a note on a musical staff (the well known set of five parallel lines) determines the name and the pitch of the note. The four notes shown in **Figure 1** are (from left to right) an A, a G, a C and an F.

The form of the note (open or filled, with or without a tail, with or without a flag) determines the duration of the note. In Figure 1, the notes are, in succession, a quarter note, an eight note, a full note and a half note. A full note lasts twice as long as a half note, a half note lasts twice as long as a quarter note, and a quarter note lasts twice as long as an eighth note.

Naturally, there are notes with other durations and other elements of musical notation, but this condensed summary is all we need for

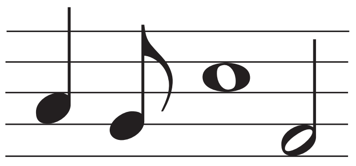


Figure 1. A staff with the four most commonly used types of notes.

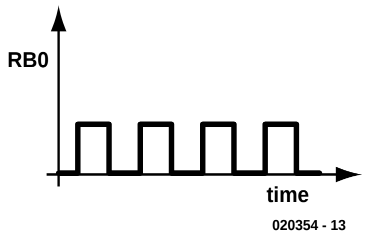


Figure 2. A square-wave signal as found on ports RB0–RB3.

the purposes of this project.

A chromatic scale consists of twelve notes (*C*, *C*$_\#$, *D*, *D*$_\#$, *E*, *E*$_\#$, *F*, *F*$_\#$, *G*, *G*$_\#$, *A*, *A*$_\#$ and *B*). The frequencies of these notes are generated as a mathematical series based on the twelfth root of 2 (or 1.05962…).

The notes for the various melodies are stored in the PIC IC in the form of tables holding the pitch and associated duration of each note.

To generate these notes, the program uses down-counters that are preloaded with the values corresponding to the notes we wish to hear. Each time a counter reaches zero, the state of an output port (RB0–RB3 for the four tones) is inverted, thereby generating a square wave as illustrated in **Figure 2.**

The circuit

The block diagram of the doorbell is shown in **Figure 3**. The circuit consists of four subassemblies: the power supply, the 'brains' and memory (in the form of the PIC 16F84, which provides all of the active functions), a mixer and an audio amplifier to drive the speaker that reproduces the tune.

Power supply

Our 'musician' takes his cue from the doorbell pushbutton S1 (refer to the schematic diagram in **Figure 6**). When it is pressed, transistor T2 starts to conduct, thus driving T1 into conduction. When T1 is conducting, a voltage of 5.1 V is present across Zener diode D1. This voltage, filtered by C4 and C5, powers the PIC 16F84. The voltage on the collector of transistor T1 also goes to the audio amplifier (IC2).

When the PIC starts up, output RA3 goes high and a voltage of 5 V is applied to the base of T2 via resistor R6. Once this happens, the pushbutton can be released. At the end of the tune, the PIC sets output RA3 low again, causing T1 and T2 to cut off and switch off power to the circuit (if the pushbutton has already been released).

Note generator

The program in the PIC can generate four notes at the same time, which is called **polyphony**. In order to obtain the maximum possible speed, the notes are generated in the main program routine.

The PIC timer sets the tempo for playing the tune. For each note to be played, a counter loaded with a value corresponding to the duration of a note is decremented each time an interrupt occurs. When this counter reaches zero, the program continues with the following note and its associated duration.

The signals produced in this manner are square waves, which are rich in harmonics. This increases their sonic range, but in order to make the sound of our doorbell more pleasing to the ear, we have added a function called 'envelope generation'.

Envelope generator

The envelope generator works according to the principle shown in **Figure 4**. At the start of each note, a capacitor (C6–C9) is very quickly charged via a low-value resistor (330 Ω) connected to a port configured as an output (RB4–RB7). After the capacitor has been charged, the port is immediately reconfigured as an input, which causes it to have a high impedance, so the capacitor cannot discharge via the port.

The note generators (RB0–BB3) are connected to the envelope generators via relatively high-value pull-up resistors (100 kΩ). In order to obtain the curve shown in Figure 4, ports RB0–RB3 would have to be open-collector outputs (type RA4), but this is not the case. In order to obtain the desired mode of operation, ports RB0–RB3 are configured as follows:

– as outputs for a '0' value
– as inputs for a '1' value

A port configured as an output and having a '0' value is essentially equivalent to an open-collector output generating a low level ('0'). The same port configured as an input has a high impedance, and it is pulled high ('1' value) by the pull-up resistor.

The envelope generator capacitors discharge via these relatively high-value pull-up resistors. This yields a time constant of approximately 0.3 s. The resulting waveforms on RB0–RB3 this have the shape shown in **Figure 5**. This waveform, with a brusque attack and an exponential decay, gives the generated notes a timbre resembling that of a piano (with apologies to purists!).

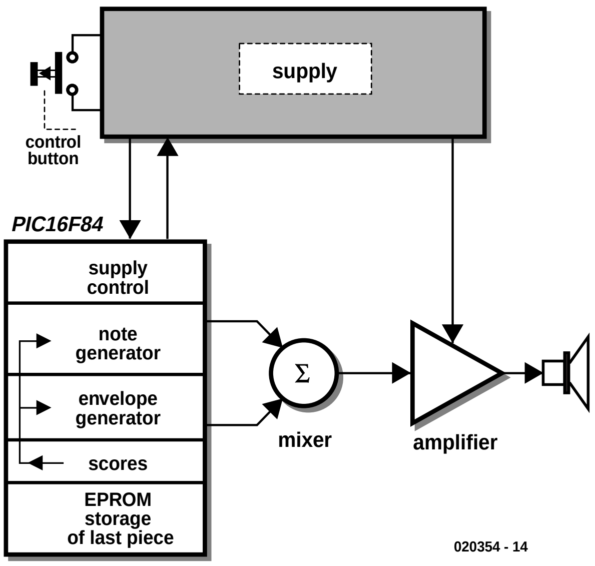


Figure 3. Block diagram of the polyphonic doorbell.

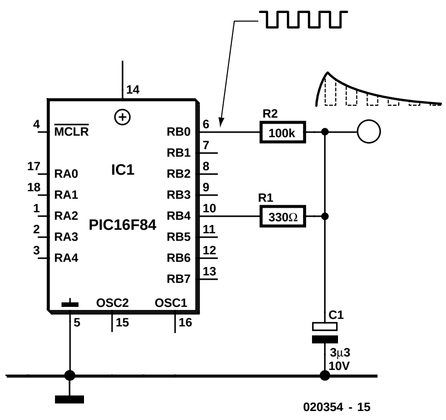


Figure 4. Working principle of the envelope generator.

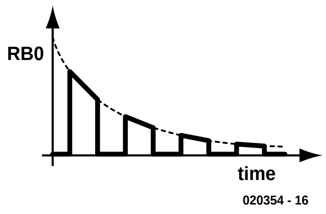


Figure 5. The final signal before the four notes are mixed.

The signals at the four note outputs (RB0–RB3) are mixed using resistors R11–R14. The resulting signal is taken from resistor R19. This combined signal is passed to the audio amplifier via capacitor C10. The gain of the amplifier can be adjusted using potentiometer P1. P1 should be to provide maximum non-distorted output power. To increase the audio power level, the supply voltage for this stage can be increased to as much as 13.5 V (three 4.5-V batteries in series).

If for some reason you want to be able to stop a tune before it has finished playing, you can connect a pushbutton switch across R3.

Construction

If you use the printed circuit board shown in Figure 7, building the circuit is child's play. Start with the lowest components, as usual. After this, fit the 'taller' components, such as T1, IC2 and P1. If you use a socket for IC1, you can later add new tunes if you wish. If you have no intention of ever changing the repertoire of the doorbell, simply solder IC1 to the circuit board.

Software

Now that you've learned something about the theoretical and practical aspects of the doorbell circuit, it may be interesting to look at the software, particularly if you can program PICs.

The software is available on diskette from Readers Services (order number **020354-11**), or you can download it free of charge from the *Elektor Electronics* website. It was generated using Microchip's MPLAB environment. The main routine of the program, carillon.asm, uses three subroutines: an initialisation routine, a 'play' routine and an interrupt routine, which calls the morceau ('piece') routine.

The def84.asm file contains the Special File Registers (SFRs) and declarations, while the macro.asm file contains macros that allow a 'pseudo-Pascal' program structure to be used (IF … REPEAT … BEGIN … END). Finally, the morceau.asm file contains the musical pieces (scores) and the down-counter values for the note generators.

Figure 8 shows a high-level flow chart that describes the operation of the program. When pushbutton switch S1 is pressed, power is applied to the circuit. The initialisation routine first configures the various ports as inputs or outputs, configures the various timer parameters related to interrupts, and then sets

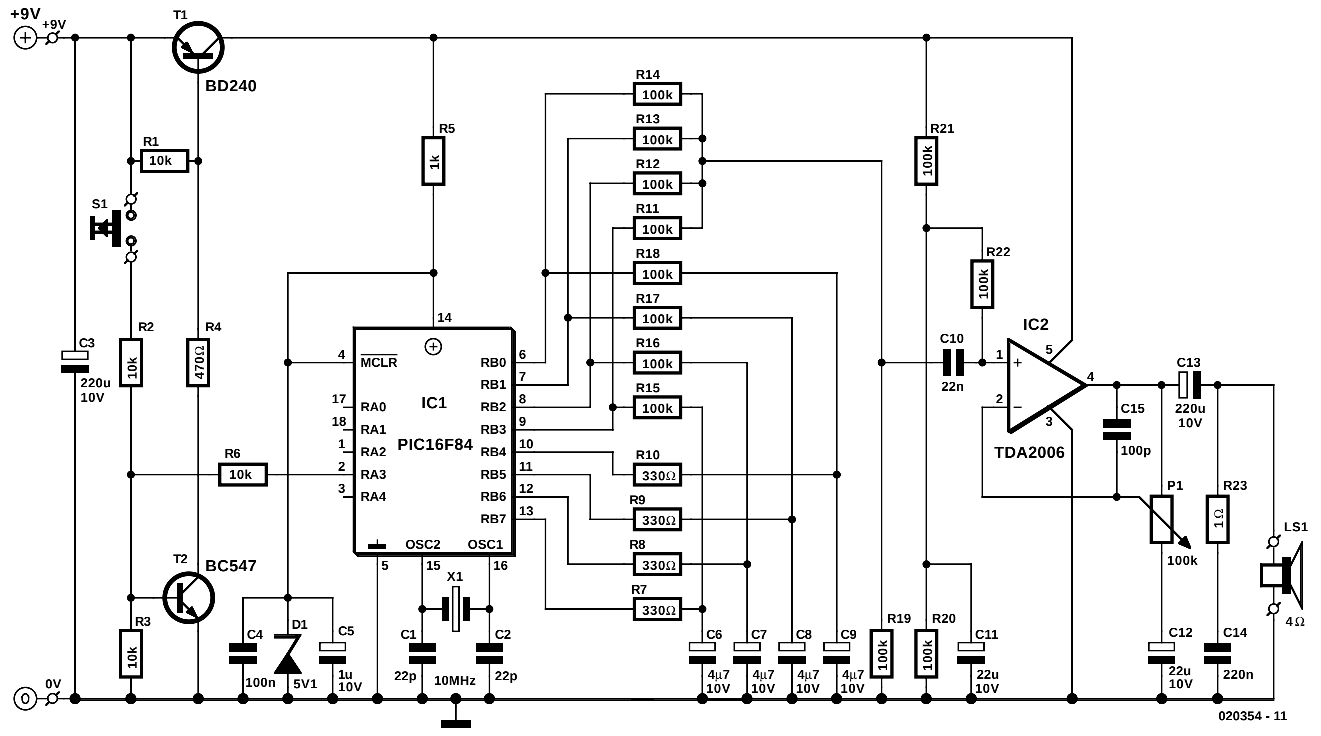


Figure 6. The electronics consists of only two ICs surrounded by a handful of passive components.

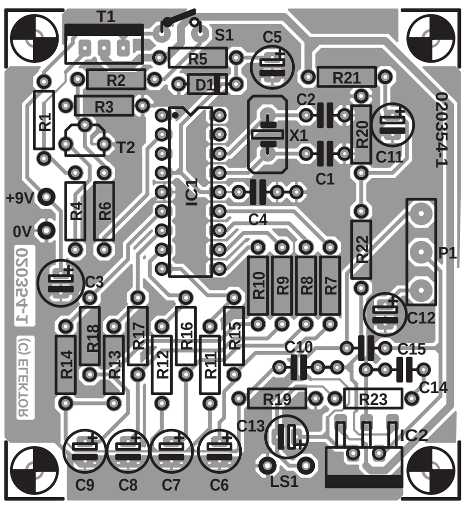
COMPONENTS LIST

Resistors:
R1,R2,R3,R6 = 10kΩ
R4 = 470Ω
R5 = 1kΩ
R7-10 = 330Ω
R11-R22= 100kΩ
R23 = 1Ω
P1= 100kΩ logarithmic potentiometer

Capacitors:
C1,C2 = 22pF
C3 = 220µF 10V radial
C4 = 100nF
C5 = 1µF 10V radial
C6-C9 = 4µF7 10V radial
C10 = 22nF
C11,C12 = 22µF 10V radial
C13 = 220µF 10V radial
C14 = 220nF
C15= 100pF

Semiconductors:
D1 = zener diode 5V1 500mW
IC1 = PIC16F84A-10/P (programmed, order code **020354-41**)
IC2 = TDA2006 (Philips)
T1 = BD240
T2 = BC547

Miscellaneous:
LS1= 4Ω loudspeaker
S1= pushbutton, 1 make contact
X1= 10MHz quartz crystal
9V battery (6F22) with clip-on leads
PCB available from the PCBShop
Disk, all project software, order code **020354-11** or Free Download



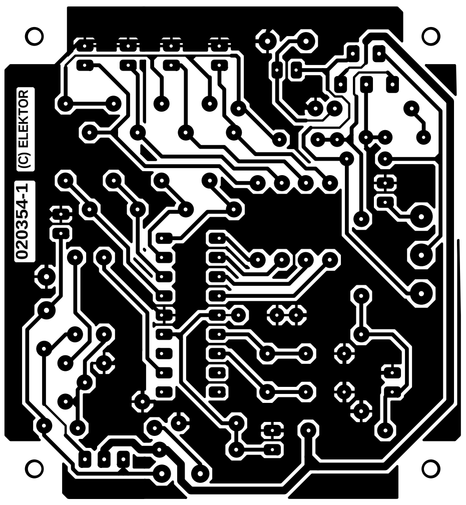


Figure 7. Copper layout and component layout of the printed circuit board for the musical doorbell.

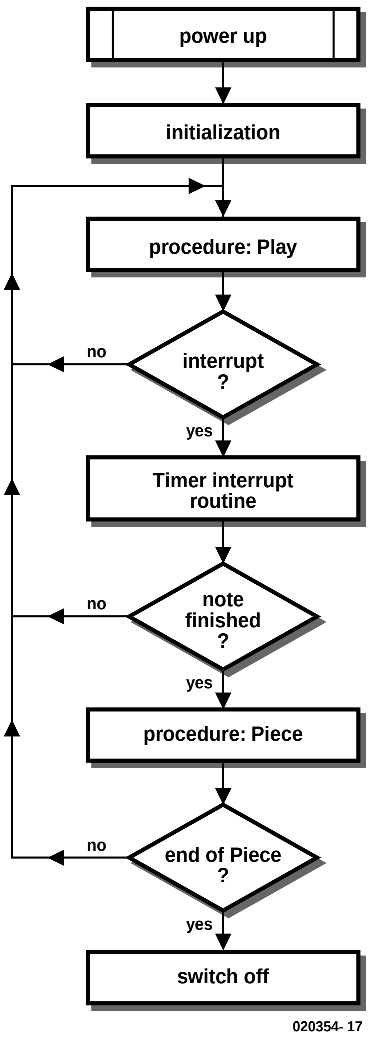


Figure 8. High-level flow chart

RA3 to '1' so that transistor T2 is held in a conducting state to provide power to the circuit.

Each time the initialisation routine is called, it reads the number of the last piece of music that was played (NUM0) from EEPROM address 0x00 and the complement of this number from address 0x01. If NUM0 and its complement do not match, something went wrong the last time these two values were stored. In this case, NUM0 is reset to '0x00' and its complement is reset to '0xFF', and these values are then stored in the EEPROM. If NUM0 and its complement do match (e.g., '0x02' and '0xFD'), the value of is NUM0 incremented (and if the result is greater than the largest possible number, it is set to zero). The new value and its complement are then stored for the next time.

The final action of the initialisation routine is to re-enable the interrupt timer.

The main routine then calls the play routine. This is the most important routine of the program, since it generates the four musical notes. It must run at the maximum possible speed, so its instructions are optimised for this purpose.

At the beginning of the play routine, interrupts are globally enabled to allow the interrupt timer to function, and the mask is set to '0'. The four down-counters are decremented each time the routine is executed. Each time a counter reaches zero, the mask is used to individually configure ports RB0–RB3 as outputs with a '0' level or inputs with a high impedance, and the counter is reloaded with the value corresponding to the note to be generated. At the end of the routine, the mask is applied to the Port B direction register using an XOR function, which causes the affected outputs to be inverted in order to generate a square-wave signal.

The interrupt routine becomes active each time the timer value changes from '0xFF' to '0x00'. In this routine, the timer is reloaded with the vitesse (tempo) value and the duration down-counter is decremented. If the duration down-counter reaches zero, the morceau routine is called. In addition, ports RB4–RB7, which correspond to the envelope generators, are configured as inputs (high impedance).

The morceau routine first configures mask1,

Free Downloads

PIC microcontroller software.
File number: 020354-11.zip
PCB layout in PDF format.
File number: 020354-1.zip
www.elektor-electronics.co.uk/ dl/dl.htm,
select month of publication.

which is used to drive outputs RB4–RB7 in order to charge the envelope generator capacitors. It then queries the morceau0 table, which represents the beginning of the musical score. Pointers ptrmorceaul and ptrmorceauh, which are calculated by the initialisation routine, allow the morceau routine to jump to the start of the desired musical piece and retrieve the first four notes to be played, along with their durations and adjuncts. An adjunct can hold a tempo value in order to control how fast the music is played (this value is used to reload the timer), or it can hold the value 'fin' (end) to mark the end of the tune. In the latter case, output RA3 is reset to '0' to switch off power to the circuit.

Conclusion

There are numerous possible applications for this circuit. Besides being used as a musical doorbell, it can also be used in a music box for children, a musical jewellery box, as an 'on hold' melody generator for the telephone, etc.

If you want to modify the tunes programmed into the PIC, you should bear in mind that you will have to redo the process of assembling and compiling the code needed to generate the hex file to be stored in the microcontroller. The development environment used by the author (Microchip MPLAB version 5.70) can be downloaded free of charge from Microchip at http://www.microchip.com/1010/pline/tools/picmicro/devenv/mplabi/mplab5x/index.htm. (note that version 6.13 requires a different project format). The procedure for editing a piece of music is simple: first load the project (taking care that all of the necessary files are present in the …\MPLAB directory), then assemble and compile the program using the 'Build All' command. This will produce a hex file that can be transferred to the PIC using a suitable programmer.

(020354-1)

Contents of diskette/file 020354-II

Assembler files:
carillon.asm
def84.asm
morceau.asm
macro84.asm

Hex file:
carillon.hex

Project file:
carillon.pjt