

KodeKLIX for PUP

Sounds and Music



Sounds and Music

- Sound capabilities of the PICAXE
- How to use the SFX Designer
- How to add sounds and effects to your apps
 - Foreground sounds
 - Background sounds



Sounds and Music - PICAXE

- The very first PICAXE chips only had the SOUND command
 - SOUND is a foreground command – everything stops whilst the sound is played
 - SOUND allows for 255 sounds
 - 0-127 are notes (ascending scale)
 - 128-255 are pseudo noise effects (ascending frequency)
 - SOUND allows for duration control

```
'=====
' play SOUND
'=====
{
let b0=64
let b1=250
Sound, _Speaker, (b0,b1)
}
```



Sounds and Music - PICAXE

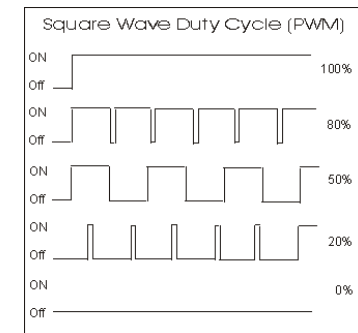
- The M series onwards have introduced PLAY and TUNE commands
 - Both PLAY and TUNE are also foreground commands – everything stops whilst the sound is played
 - PLAY outputs a predefined tune stored in the PICAXE chip
 - TUNE allows notes to be played
 - However, by playing notes one at a time using a coding loop the TUNE can be played whilst other code gets its change to execute



Sounds and Music - PICAXE

- Some PICAXE chips also support PWM

- PWM = Pulse Width Modulation
- PWM = square wave output

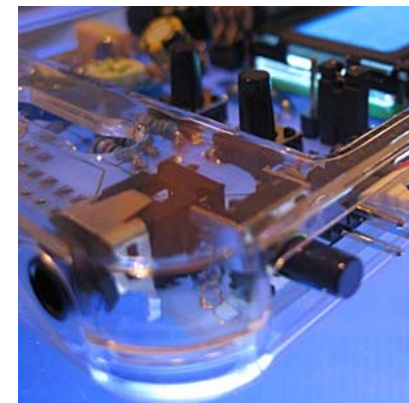


- When PWM is connected to sound device, like a peizo speaker, sound is heard
 - PWM can be done in the background if the PICAXE hardware supports it
 - This means that sounds can continue play whilst your code gets on with other tasks



Sounds and Music - PUP

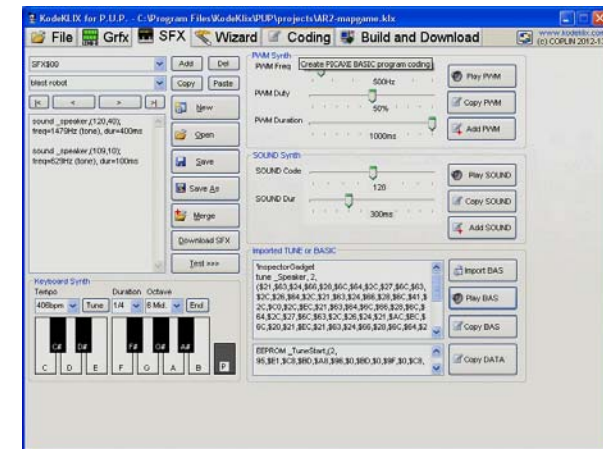
- The PUP hardware supports up to two sound channels
 - One foreground and one background (PICAXE 20M2 and 20X2)
 - One foreground channel with PICAXE 20M
 - The two sound channels are combined by the PUP circuit hardware and output as one signal
 - The output of one channel can affect the signal from the other





SFX Designer and Manager

- Create your own sound effects (SFX)
- Create your own TUNES
- Import TUNES from the PICAXE library
- Convert SFX and Tunes to PUP code
- New PWM based SFX and TUNE formats
- Manage sound creation libraries
 - Share these with other KodeKLIX users





SFX – CloseUP Guide

The screenshot shows the KodeKLIX software interface with several callout boxes pointing to specific features:

- SFX ID**: Points to the SFX ID field (SFX\$00).
- Descriptive Name**: Points to the descriptive name field (blast robot).
- SFX code Window**: Points to the SFX code window showing sound code (sound_speaker, (120,40); freq=1479Hz (tone), dur=400ms).
- Test the SFX Code**: Points to the Test >>> button.
- Keyboard Synthesizer**: Points to the Keyboard Synth section with a piano keyboard.
- Close**: Points to the Close button in the top right corner.
- PWM Sounds**: Points to the PWM Synth section with sliders for PWM Freq (500Hz), PWM Duty (50%), and PWM Duration (1000ms).
- PICAXE Sounds**: Points to the SOUND Synth section with sliders for SOUND Code (128) and SOUND Dur (300ms).
- Import TUNES**: Points to the Imported TUNE or BASIC section with buttons for Import BAS and Play BAS.
- TUNE Format**: Points to the TUNE code window showing hex values.
- EEPROM Format**: Points to the EEPROM code window showing hex values.



SFX – PWM Synthesizer

Frequency

PWM Duty Cycle

Duration (test only)

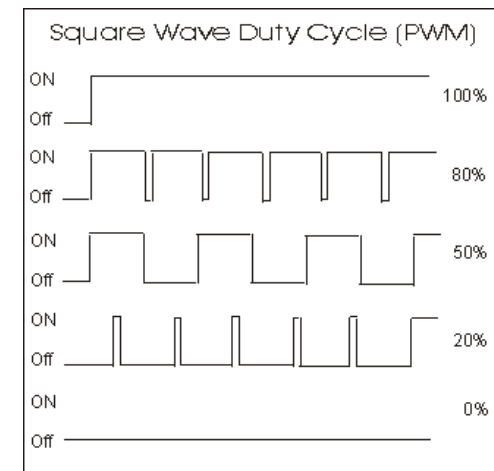
PWM Synth
PWM Freq 500Hz
PWM Duty 50%
PWM Duration 1000ms
Play PWM
Copy PWM
Add PWM

Hear Sound

Copy PWM code

Append PWM code

- PWM = Pulse Width Modulation
- Sound is not square waves, but it's the best the standard PICAXE can do...
- PWM is hardware controlled, and will continue playing whilst program runs
- Duty cycle adjusts volume
 - 50% is max volume
 - <50% or >50% is quieter
 - 0% or 100% is "off"





SFX – Sound Synthesizer

The screenshot shows the 'SOUND Synth' interface with the following elements and annotations:

- Sound Code:** A label pointing to the 'SOUND Code' slider, which is currently set to 128.
- Sound Duration:** A label pointing to the 'SOUND Dur' slider, which is currently set to 300ms.
- Hear Sound:** A label pointing to the 'Play SOUND' button.
- Copy Sound code:** A label pointing to the 'Copy SOUND' button.
- Append Sound code:** A label pointing to the 'Add SOUND' button.

- PICAXE Chips produce sounds
 - Sounds are played in the foreground, so the PICAXE pauses whilst the sound is played
- Codes 0-127 are “notes” or square wave tones/notes
- Codes 128-255 are “noises” or sound effects
 - Randomised square waves for game effects such as explosions, shots, etc
 - KodeKLIX includes full playback database



SFX – Keyboard Synthesizer

The screenshot shows the 'Keyboard Synth' interface with the following controls and callouts:

- Tempo Setting:** A dropdown menu showing '203bpm'.
- Start New TUNE:** A button labeled 'Tune'.
- Keyboard (octave):** A dropdown menu showing '6 Mid.'.
- Note Details:** A dropdown menu showing '1/4' for Duration and 'End' for Octave.
- End current TUNE:** A button labeled 'End'.
- Pause / Rest Interval:** A button labeled 'P'.

The keyboard layout includes keys for C, D, E, F, G, A, B, and C# (labeled as C#), D# (labeled as D#), F# (labeled as F#), G# (labeled as G#), and A# (labeled as A#).

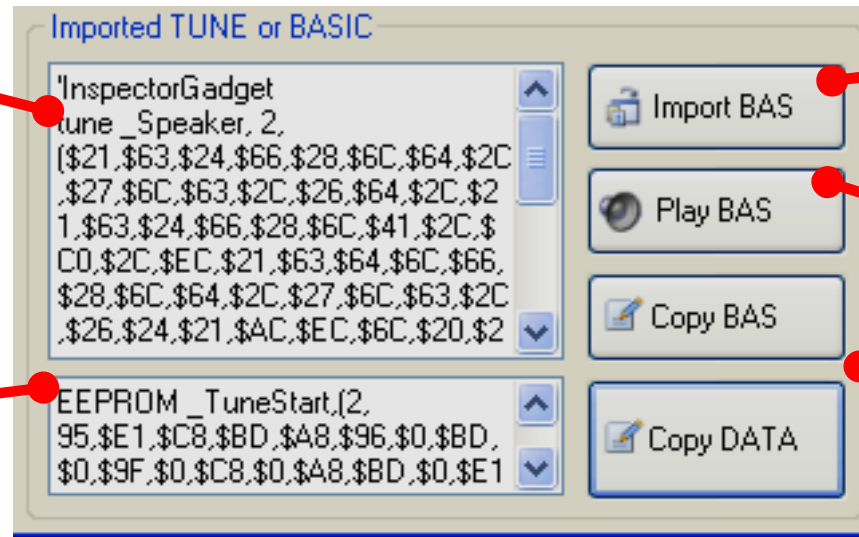
- Some PICAXE support the TUNE command
 - TUNES are played in the foreground, so the PICAXE pauses whilst the notes are played
- KodeKLIX introduces new ways to play TUNES in the background by translating the routines to EEPROM stored codes
 - Background TUNES still pause for each note played
 - Background PWM tunes do not pause for each note played (note: some limitations on tune complexity apply)



SFX – TUNE Importing

Imported
BASIC

EEPROM
Formatted



Import
PICAXE

Playback
of TUNE

Copy to
Library

- Import pre-existing TUNEs from the PICAXE suite
 - Add to the KodeKLIX library or convert to EEPROM format
- Built-in playback and convert feature
- Copy either BASIC code or just DATA values to the library

Note: Keyboard Synthesizer output is also written to this window



Adding Sounds and Music

- Sounds and notes can be hardcoded by simply using the numeric values for notes and durations – this is PICAXE BASIC
- With KodeKLIX you can also specify a sound or tune from your SFX library using the SFX\$xx label where xx is the ID number, for example SFX\$03
 - SFX\$xx could be a code or tune snippet
- Your KLX project file saves the name of the SFX library used during creation



Example Code – controls

```
'=====
' Multiplexed Button B
'=====
{ ' play foreground sound
if _btnADC > _MuxB and _btnADC < _MuxC then
    sound _Speaker, (64,5)
endif
}
```

```
'=====
' Multiplexed Button C
'=====
{ ' suspend background playing
if _btnADC > _MuxC and _btnADC < _MuxD then
    suspend 1
endif
}
```

```
'=====
' Multiplexed Button D
'=====
{ ' resume background playing
if _btnADC > _MuxD and _btnADC < _MuxTR then
    resume 1
endif
}
```



Example Code – background

```
' =====  
' Background Music Player using PWM method  
' Use of EEPROM based data storage allows  
' tunes to be loaded from other devices.  
' =====  
' either use EEPROM, or simply the #SFXxx macro  
EEPROM _TuneStart, (...tune goes here...)  
' PWMDIV16 codes, 1st byte tempo (bpm) index, 2nd sets note count  
read _TunePos, _TuneNote  
  
_TuneDuty=_TuneNote +1  
_TuneDuty=_TuneDuty *2 max 255  
PWMOU PWM DIV16, _SpeakerPWM, _TuneNote, _TuneDuty  
' convert BPM to a pause duration,  
' PWM (note) keeps going during pause!  
' index to BPM table * 74ms gives pause duration  
read _TuneBPM, _TuneNote  
_TunePause=_TuneNote *74 - 20 ' -20 corresponds to delay in software loop  
pause _TunePause  
  
inc _TunePos  
read _TuneEnd, _TuneEndPos  
if _TunePos > _TuneEndPos then  
    _TunePos = _TuneStart +1  
endif
```



Tutorial: 3.7

- Open tutorial 3.7
- Study code snippets for:
 - Button B, Button C, and Button D
 - Task 1
- Connect PUP and Download program
- What happens when you press?
 - A _____
 - B _____
 - C or D _____