PICAXE Forum Post: 30/9/2024

Improved Debugging

In a recent project I made a module called [ICODE]Debugger.basinc[/ICODE]. I found it quite useful so it's here to share!

It's very simple and doesn't consume any extra memory or delays compared to using sertxd for debugging. You can also turn it on and off with a compiler directive.

The main benefit to me was program readability. It was nice to read code without needing extra comments, particularly when text was used in macros. Inside the macro you could debug out the text, using a lot of program space; or disable the debugging either by commenting out a line or turning the whole debugging off. Whatever you choose the PICAXE Editor keeps on displaying the bright red text as part of your program.

I couldn't find anything similar on the forum, so I hope someone finds it useful.

There's a lot help information in the file, just rename the extension from .bas to .basinc, or download it from here [URL] https://github.com/alnhnt/PICAXE/blob/main/Library/Debugger.basinc[/URL].

PICAXE Forum Post: 4/10/2024

Delving into and supporting Timing with Clock Rates (module/library)

The attached module was originally used for a game where the clock rate hadn't been finalised and the audio interface required consistent timings to be near minimum levels. Modifying more than a few pause commands by hand is unwieldy and error prone, particularly because arithmetic like "Pause 1 * ClockMultiplier" is not allowed with the pause commands. Ultimately, a short list of logarithmic timings was easy to remember and they provided all the accuracy needed on most projects. It became a bit of fun to delve deeper into PICAXE timing and record some timing on an oscilloscope and this module emerged. This module's list of pause statements doesn't consume any extra program memory, and the module has these features:

[LIST]

[*] A list of pause statements that provide timing delays regardless of the PICAXE clock rate: Pause1mS, Pause3mS, Pause10mS, Pause30mS, Pause100mS, Pause300mS, Pause1S, Pause3S and Pause10S.

[*] The macro ``Pause_mSec(t)`` allows you to specify specific pause times regardless PICAXE clock rate.

[*] The PICAXE clock rate is set using a kHz value: 31, 250, 500, 1000, 2000, 4000, 8000, 16000, 32000, or 64000 (Dependent on chip).

[*] The debug terminal baud rate is automatically set to the appropriate value for the PICAXE clock rate.

[*] For X2 chips, the "timer" variable is set to increment at a configured rate if ``TIMER_IN_HUNDRETHS`` is defined in your program and given a value. This proved useful to measure lap time in a game.

[*] A reference list of actual pause timings is shown in the Programming Guide notes below, these are valid if you use this module, or just use regular pause commands and do integer math based on the ratio of actual clock rate to default clock. From this some subjective recommendations are made for minimum clock rates with certain durations. Just don't expect a 1mS delay to be added when you use ``pause 1`` :)

[/LIST]

The attached file should be renamed to [ICODE]Timing.basinc[/ICODE] with the basinc extension. There's a lot of help information in the file and you can also download the master file with the basinc extension from GitHub [URL=https://github.com/alnhnt/PICAXE/blob/main/Library/Timing.basinc] here [/URL].

PICAXE Forum Post: 16/10/2024

Speech, Effects and Music with DF Player Mini (module/library)

Back in the 80's the most exciting outputs I could muster usually involved LEDs and 7 segment displays. Returning to electronics I'm finding that generating audio is now easy and can be far more informative and fun. The attached module provides the ability for your PICAXE program to speak numbers and sentences by stringing together words or phrases, it can also play music, sound effects and adverts. The adverts I found useful in interrupting idle music to encourage game play of my Buzz Wire Race Game at the village fete, also they provided crash sound effects during background music when the game was played.

With websites like www.ttsmp3.com it's easy to create and download MP3 tracks for words or entire paragraphs. I've put together a foundation set of tracks for some projects in addition to the one's used in my game. Together with this "DF Player_Mini.basinc" module, the audio tracks make it easy to:

[LIST]

[*] Speak numbers up to 65,535 using the macro AudioSpeakNumber(val).

[*] Speak decimals assuming a word value represents 10's, 100's, or 1000's, such as AudioSpeakThousandths(4023), which speaks four-point-zero-two-three.

[*] Speak numbers with ordinal suffixes like 18^{th}, which has its own audio track; or 28^{th} which would be formed by joining the tracks for twenty and eighth.

[*] Speak modifiable sentences by stringing together a sequence of words, phrases and numbers.

[*] Recite speech, or play any sound, effects or music, by just pre-recording MP3s that a PICAXE either starts immediately using PlayNow(136, “You finished!”); or starts when the preceding track is finished using PlayMP3(137, “Your time was”).

[/LIST]

With the foundation tracks you make spoken reports like the demo program below:

[CODE]

```
'Compiler Directives
   #picaxe 20X2
   #no_data
   #no_table
   '@Timing.basinc directive
   #define CLOCK_SET_KHZ 8000

'Pins
   symbol AUDIO_TX = B.0            '@DF_Player_Mini.basinc: PICAXE Serial Tx to DF Player via 1K resistor.
   symbol AUDIO_STATUS = pinB.2     '@DF_Player_Mini.basinc: Busy indication from DF Player is active low after about 230mS.

'Variables
   '@DF_Player_Mini.basinc Variables
   symbol _Audio_Digit = b43        'Assign symbol to any spare Byte.
   symbol _Audio_Integer = w22      'Assign symbol to any spare Word.
   symbol _Audio_Fraction = w23     'Assign symbol to any spare Word.
   symbol _Audio_Track = b48        'Assign symbol to any spare Byte.
   symbol _Audio_Byte = b49         'Assign symbol to any spare Byte.

'Constants
   '@DF_Player_Mini.basinc Constants
```

```
   symbol AUDIO_VOLUME = 15          '0 to 31(Maximum).  You can assign this symbol to a variable if you don't want fixed volume.

   symbol AUDIO_BAUD = T9600_8       'Tx logic is idle high (T) and 9600 bps, the last characters you adjust to your clock speed.

   symbol AUDIO_ACTIVE = 0           'The AUDIO_STATUS pin is active low, indicating audio playing.

   symbol AUDIO_IDLE = 1             'AUDIO_STATUS high indicates the module is idle or seeking a track to play on the disk.


'Modules

   #include "Timing.basinc"

   #include "DF_Player_Mini.basinc"


'Initialisation

   gosub AudioInitialise             'Prepares PICAXE output pin, Resets the DF Player, selects SD input and sets volume.


Main:

   PlayNow(0063, "The system has restarted.")

   PlayMP3(0070, "The cell voltage levels are")

   AudioSpeakThousandths(3423)       'This speaks "three-point-four-two-three".

   PlayMP3(0112, "volts")

   PlayMP3(0030, "and")

   AudioSpeakNumber(3375)            'This speaks "three-thousand-three-hundred-and-seventy-five".

   PlayMP3(0111, "millivolts.")

   PlayMP3(0068, "Warning, the battery is below 20%.")

   PlayMP3(0103, "Internal")

   PlayMP3(0121, "temperature")

   PlayMP3(0105, "is")

   AudioSpeakTenths(354)             'This speaks "thirty-five-point-four".

   PlayMP3(0122, "degrees C.")

   PlayMP3(0104, "External")

   PlayMP3(0121, "temperature")

   PlayMP3(0105, "is")

   AudioSpeakTenths(176)
```

```
  PlayMP3(0122, "degrees C.")
  PlayMP3(0106, "Total")
  PlayMP3(0101, "solar")
  PlayMP3(0116, "power")
  PlayMP3(0105, "is")
  AudioSpeakHundreths(1263)        'This speaks "Twelve-point-six-three".
  PlayMP3(0093, "kilo")
  PlayMP3(0117, "watt")
  PlayMP3(0087, "hours")
  PlayMP3(0108, "over the last")
  AudioSpeakNumber(7)
  PlayMP3(0089, "days.")

  Pause3S
  goto Main
```
[/CODE]

The attached zip file contains a recording that lets you hear what the demo was like. The staccato of individual words is fine and clear for personal use but to save program space and greatly improve presentation it's best to join phrases rather than individual words. For my game I also edited tracks to speed up the pace a little and occasionally added a cartoon effect and changed pitch.

The DF Player Mini is referred to as a PICAXE SPE033, which isn't in the shop; or as a SPE035 Project Board which includes a tiny yet powerful speaker, just note the speaker's pins are too chunky for breadboard. The DF Player Mini is also widely and cheaply available, just make sure you're obtaining from a quality source. The PICAXE only needs to monitor the DF Player Busy Signal and send serial data to play or stop MP3 or WAV tracks.

The attached "DF_PLAYER_MINI" file contains lots of useful information and it should be renamed with a ".BASINC" extension, the master file is [URL=https://github.com/alnhnt/PICAXE/blob/main/Library/DF_Player_Mini.basinc] here [/URL].

Please note the audio tracks are not for commercial use, although the majority are from ttsmp3.com using the AI "Alloy" voice and their website claims they don't have any restrictions. NCH WavePad was used to touch-up the tracks, mainly this was normalising based on peak loudness (RMS) and trimming the start and end to remove small amounts of quiet time.

The audio files and a list of all the tracks can be found [URL=https://github.com/alnhnt/PICAXE/tree/main/Buzzz_Wire_Race_Game/Audio] here [/URL].

My future projects are likely to bias more towards speech feedback in future, particularly with external projects because waterproofing and visibility becomes less of an issue. I'm glad to help if you have any issues. Please also let me know if there's any potential improvements.

Forum Post: 22/10/2024

Voltage monitoring and calibration for X2 chips (module/library)

This module is fairly simple, but maybe of use to someone. It's my penultimate write-up of a project.

The X2 operator "*/" is used to gain an accurate ADC to voltage conversion with a small amount of program code and memory. There are some test results in the notes showing that the ADC is reasonably linear and accurate but the internal FVR (Fixed Voltage Reference) is not so great as a basis to calculate the PICAXE supply voltage. To compensate for this there's an FVR calibration method included in the notes too.

There are just 3 macros:

[LIST]

[*] GetVdd updates a word variable with the power supply voltage in mV.

[*] SetVdd(Val_mV) can be used instead of GetVdd for testing purposes, or if the power supply is accurately known and stable.

[*] VoltsByVdd(Var, Channel) updates a word variable, Var, with the voltage in mV from the specified ADC Channel number.

[/LIST]

The attached "Voltage_X2" file contains its own documentation and it should be renamed with a ".BASINC" extension when used, the master file is [URL=https://github.com/alnhnt/PICAXE/blob/main/Library/Voltage_X2.basinc] here [/URL].

I'm glad to help if you have any issues. Please also let me know if there's any potential improvements.

PICAXE Forum Post: 27/10/2024

Buzz Wire Race Game

A Buzz Wire game started as a fund-raising idea for our village fete. I offered to make something based on a racetrack with timed laps, and the committee jumped at that. To make it more fun I added music, sound effects and speech with a DF Player Mini, as per a previous post. The game was engaging and fun with the audio never stopping and some of the speech speeded up for results etc.

There were advert tracks to encourage game play and to give different crash sound effects during the game. At the race start a beep-beep-beep-beeeeeep was timed to work in parallel to a led sequence. There were timers for feedback during the race. At the end of a race there was more feedback and possible fanfares along with a spoken lap time in tenths of a second. Player times were ranked against a leaderboard and the top 4 positions were spoken if required. There were independent volume controls for audio from the DF Player and for the buzz tone that was created by PWM. The volume was fairly unbearable at maximum.

Soon I ran out of program space on a 20M2, mainly because I liked to see and show the debug output. I swapped out to a 20X2 and kept having to trim back the debug as the program grew. Eventually it was 459 bytes of the 4077 used.

Unfortunately, under twelves performed far worse than expected and a couple were dejected by their lap time, which was sad to see. The adults were so competitive though! The game is a little addictive and frequently people thought they could get the fastest lap by crashing 1 less time. I think the least number of crashes all day was 2, but you must be fairly quick too.

Probably the worst decision was boldly stating I would base the layout on our local racetrack, Snetterton. I already felt committed, then I checked the look of the layout and probably expressed a few expletives based around “oh dear”! 😊 I’ve promised a different track for next year.

I’ve always hand drawn circuits and stripboard layouts in the past, knocked up programs quickly and then wondered what it all meant if I return to it later. This year I wanted to go better and probably took on a bit too many firsts for me:

[LIST]

[*] KiCad for electronics schematic and exports of a netlist and bill of materials.

[*] VeroRoute for stripboard layout after importing the netlist.

[*] InkScape for vector graphics of the laminated sheet.

[*] Wavepad for audio editing.

[*] DecalProFx for silk screening panels, which has always been a bug bare. Sadly, they're now out of business. The decal didn't peel away properly on the bare metal panel but worked perfectly for the plastic mounted volume controls.

[*] Brazing to connect the handle and wire, although a nut and bolt seemed almost as good.

[*] Getting to know GitHub, Markdown Language (with StackEdit), and Licensing for uploads.

[*] Choosing and learning how to use a table saw after cutting my last wonky bit of wood!

[*] Trying to make a full documentation set.

[/LIST]

It was fun to do and some audio tracks were amusing too. I've placed a few in the ZIP file, including the one about a PICAXE being upgraded and taking over the world!

Excluding any copyright music, I've made a full upload of remaining files to GitHub including maker details and a presentation folder [URL= https://github.com/alnhnt/PICAXE/tree/main/Buzz_Wire_Race#readme] here [/URL].