



Picotracker:



the fully open groovebox is finally
here!



Agenda

1. Setup the build environment
2. Introduction
3. Assemble hardware
4. Build new firmware feature
5. Flash our firmware
6. Compose a track!
7. 🎉 🎵 🎵 🎵 🎉

Building the firmware: PREP-1

Clone the git repo:

```
git clone https://github.com/democloid/picoTracker
```

```
cd picoTracker
```

```
picoTracker % git submodule update --init --recursive
```

Building the firmware: PREP-2

If on Ubuntu or Debian:

- Install prerequisite OS packages:

```
sudo apt-get update
```

```
sudo apt-get install cmake gcc-arm-none-eabi libnewlib-arm-none-eabi  
build-essential
```

Building the firmware: PREP-2

If on MacOS:

- Install prerequisite OS packages:

```
brew install cmake gcc-arm-embedded
```

Building the firmware: PREP-2

If on any other OS use Dev container:

- Install **Docker**, **VSCode**, Devcontainers extension
- Open local picoTracker directory in VSCode
- then:



```
>
```

Dev Containers: Rebuild and Reopen in Container

Last year at E023...

The ML-2 project:

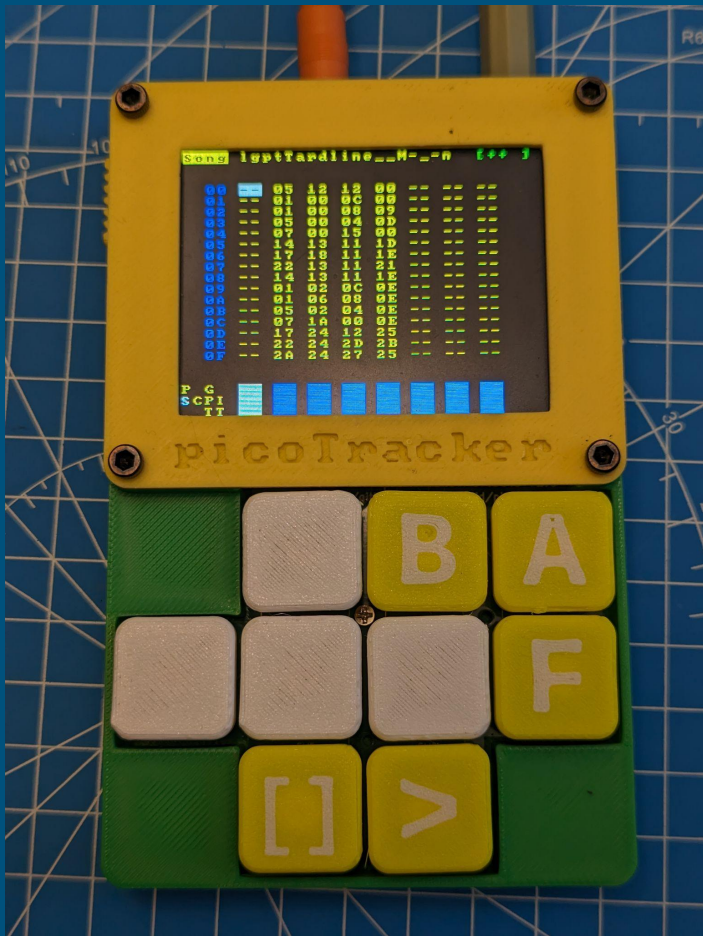
- Akai Fire Midi controller +
- RPI2 +
- libSunvox +
- Dart



This year...

The Picotracker!

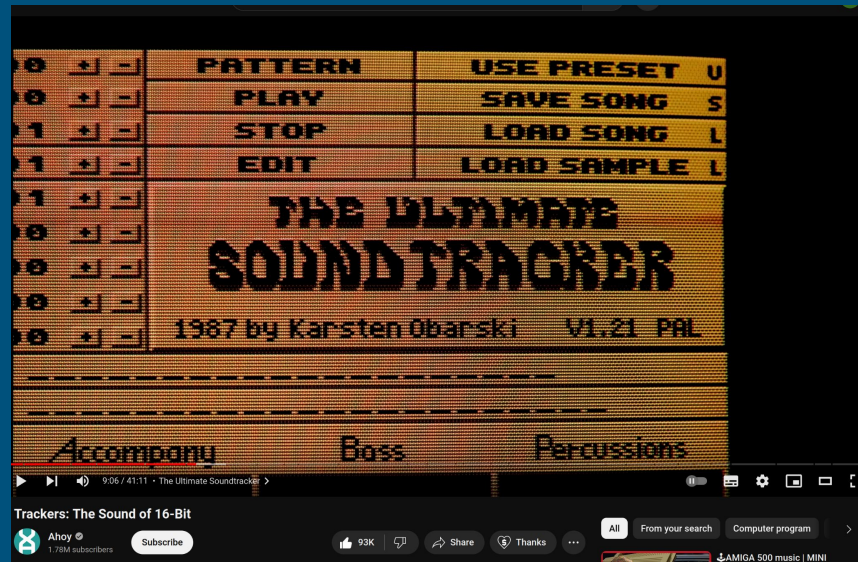
- Fully *Open* Hardware!
- Fully *Open* Source firmware!
- Fully *open* 3D printable parts!
- Running on openly doc'd RP2040!
- **Open** and friendly *community*



Some History (repeating...)

Or what is a Tracker?

- Started in 1987 with OG:
The Ultimate Sound Tracker by Carsten Obarski for Amiga 1000
- Gained following for crack screen “demos”
- [Great video of Tracker history by @Ahoy on Youtube](#)

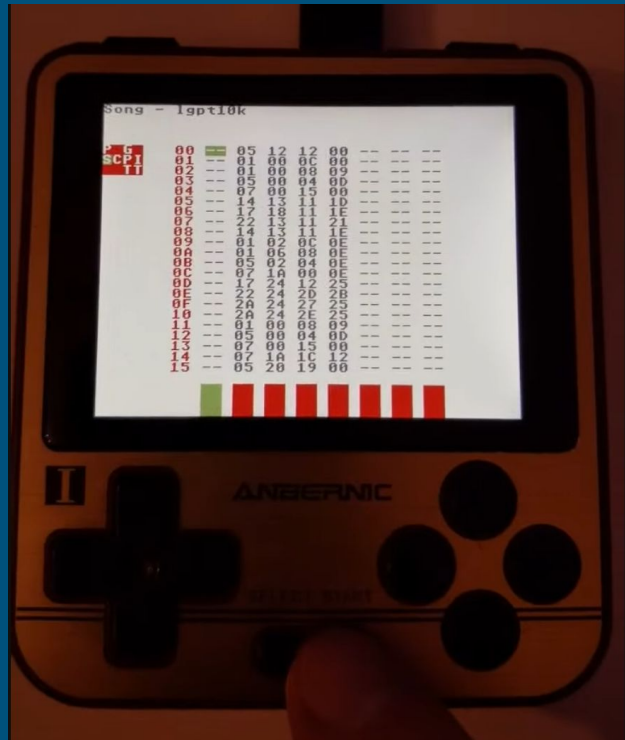


Some (not so recent) History

Or what is a **LGPT**?

- Little GP Tracker
- By “Marc Nostromo”: 1st public commit **2014**
- Inspired by LSDJ - Tracker app for a Gameboy
- Intended to run on Linux based retro game emulator handhelds
- Still on Github:

<https://github.com/Mdashdotdashn/LittleGPTracker>



Some recent History

Or what is a Picotracker?

- Started in 2023 by **@democloid**
- Uses modified version of LGPT as its firmware
- Runs on a Pi Foundation RP2040 u-controller
- Original “desktop prototype”
- Current “portable” versions with custom PCBs

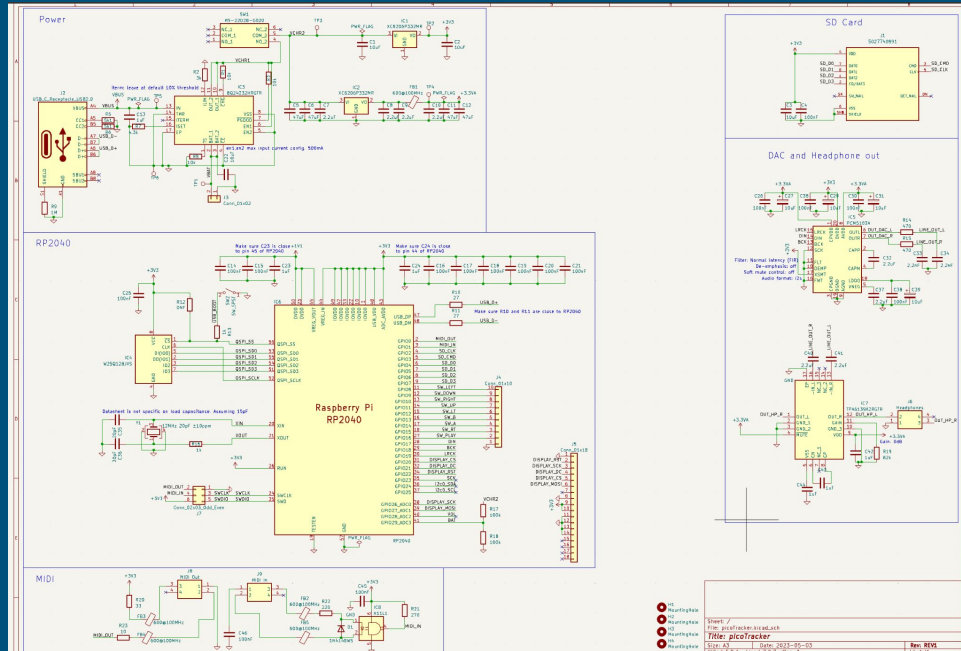
Building the hardware

Thanks to OH&S here at EO venue, no need to solder, only screwdriver required! 🙌

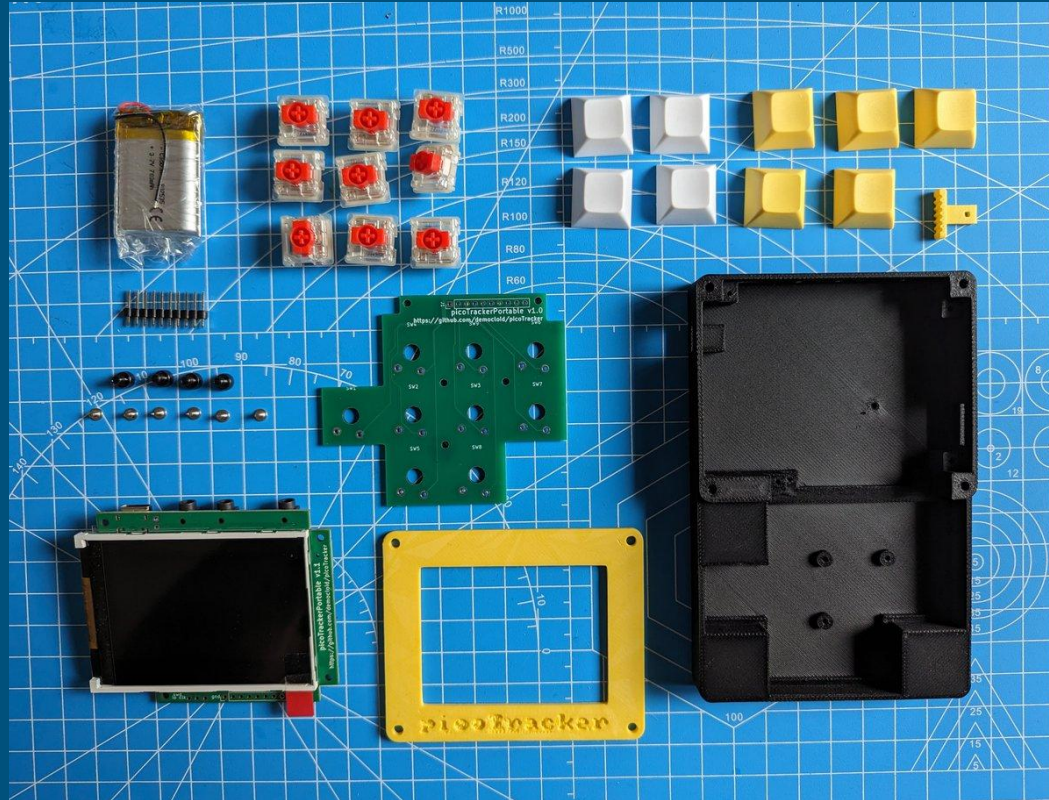
What's inside a picoTracker?

On the custom PCB:

- RP2040
- 16MB Flash
- LCD
- PCM5100 DAC, Amp
- SD Card socket
- TRS sockets, USB socket, passives, etc.



What's inside a picoTracker?



Building the firmware

Inside local picoTracker dir:

```
picoTracker % mkdir build
```

```
picoTracker/build % cd build
```

```
picoTracker/build %
```

```
PICO_SDK_PATH=../sources/Externals/pico-sdk cmake  
../sources
```

```
picoTracker/build % make -j8
```

“Flashing” the firmware

Plug in the Picotracker to USB port.

From **build** sub-directory:

```
cp Adapters/picoTracker/main/picoTracker.uf2  
/media/maks/RPI-RP2/
```

Picotracker will reboot once finished copying the firmware across

(if using Dev Container, copy command above ***outside*** the container)

Debugging the firmware

Three options for debugging in order of difficulty:

1. Add output on the GUI in code
2. `printf()`'s to the console
(USB or UART - set vars top-level in `CMakeLists.txt`)
3. GDB + OpenOCD + SWD (eg. picoprobe)

For more details see:

<https://github.com/democloid/picoTracker/blob/master/docs/DEV.md>

Trying out the firmware

- Download the pT_workshop.zip file:
<https://bit.ly/4cZGCg8>
- Unzip into the **TOP LEVEL** directory on your SD Card

Let's build a new feature!

- Check GitHub repo issues for “*good first issue*” ...
- “Add VU meter to song screen” (#42)
- Good example of both “audio engine” & “GUI”
- What is a “VU Meter” ?



Building the VU Meter - Git FTW !!

The code we will be using is in a series of commits on a branch: **vu-meter**

Commits to **git cherry-pick**:

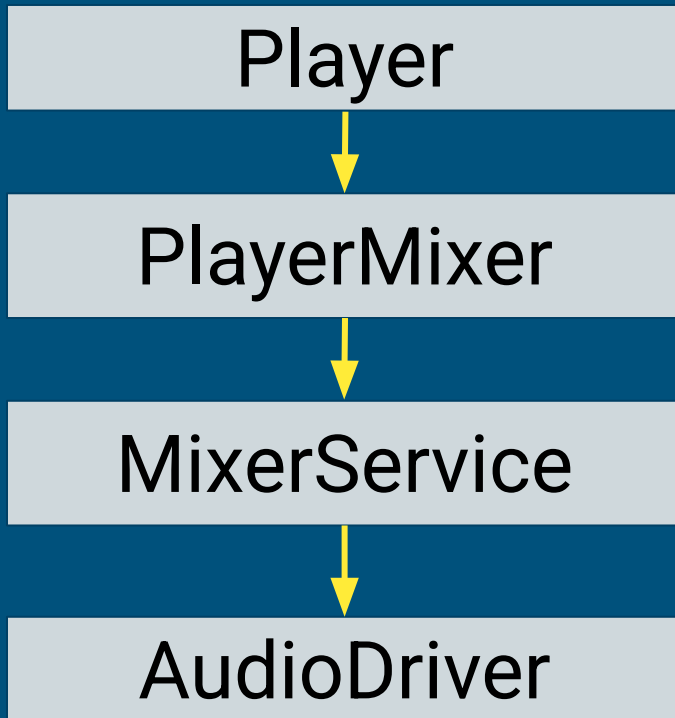
- 2211a12f
- 513b932c
- 7f4f75e8
- 99d50cef

Building the VUMeter “backend”

Need to get currently playing audio level:

- Where to access the value?
- How to expose it to the UI layer?
- The UI layer has a reference to **Player**

Building the VU Meter “backend”



Building the VU Meter UI

Need to display the value on each screen update

- How to access the current value from the “backend”?
- How to display on screen?
- How to keep display updated?

Building the VU Meter UI

- “ASCII Style”
- Can use stacked “=” characters
- Colors ?

Building the VU Meter UI 🎉🎉



Remote USB UI

How am I showing you my pictoTracker display?

For now a **custom build** because USB requires extra 6kB,
so more now with CMake vars set:

```
pico_enable_stdio_usb(${PROJECT_NAME} 1)  
add_definitions(-DUSB_REMOTE_UI)
```

Making music!

1. Learning the picoTracker UI
2. Learning the “Tracker workflow”

Learning the picoTracker UI



Learning the picoTracker UI

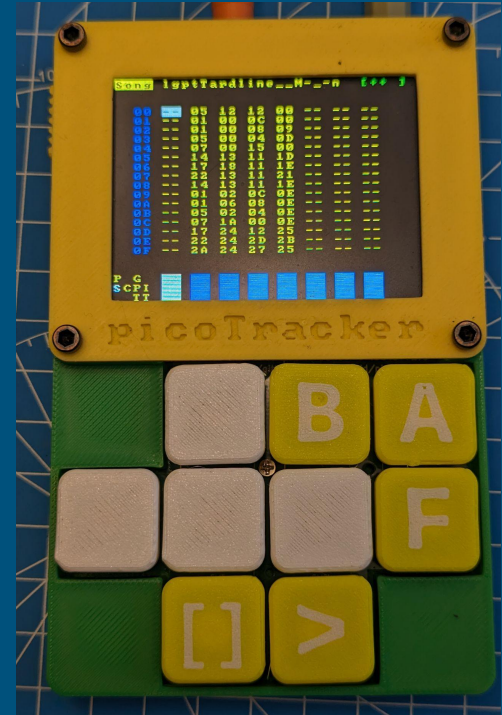
Keymap:

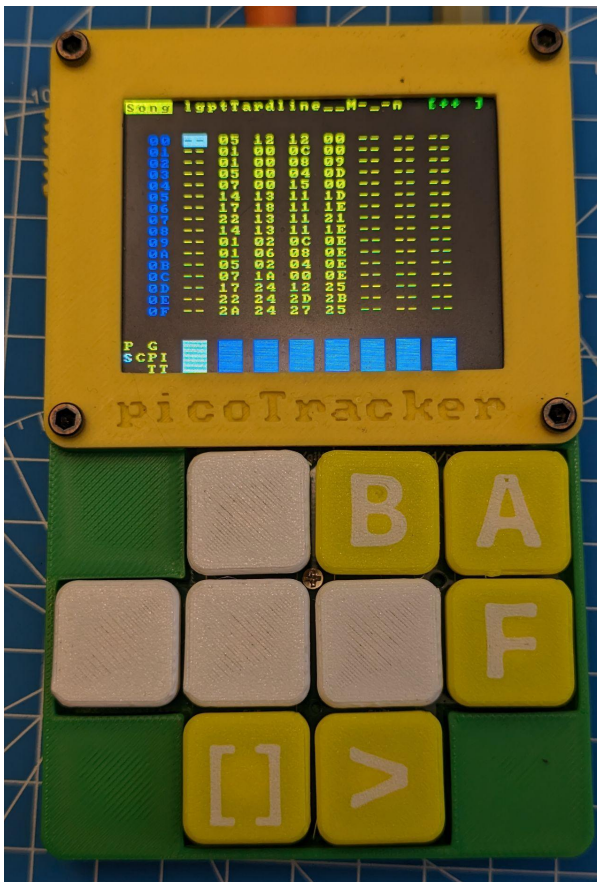
Standard or “M8”

Key names: LGPT
or “Maks style” 😊

config.xml on

SDCard !





ARROWS (White keys)

Basic Editing & Navigation

[] + ARROWS: Navigate between the Screens.

A: Insert Chain/Phrase/Note.

A,A: Insert next unused Chain/Phrase/Instrument.

F+(B,A): Clone. This will overwrite the current Highlighted Item with a copy of itself using the next unused Item available.

B+A: Cuts the current Highlighted Item .

A+ARROWS: Updates Highlighted Item value.

A+UP/DOWN: +/- 0x10.

A+RIGHT/LEFT: +/- 1.

B+ARROWS: Rapid Navigation.

B+UP/DOWN: Page up/down in Song Screen, Next/Previous Phrase in Current Chain in Phrase Screen. Navigation +/- 0x10 in Instrument/Table Screen.

B+LEFT/RIGHT: Next/Previous Channel in Chain/Phrase Screen.
Navigation +/- 1 in Instrument/Table Screen.
Switch between Song and Live Modes in Song Screen.

Selections

F+B: Starts selection mode with only the data at the cursor selected

F+B+B: Starts selection mode with the cursor's row selected

F+B+B+B: Starts selection mode with the entire screen selected

once a selection is started you can do a few more things:

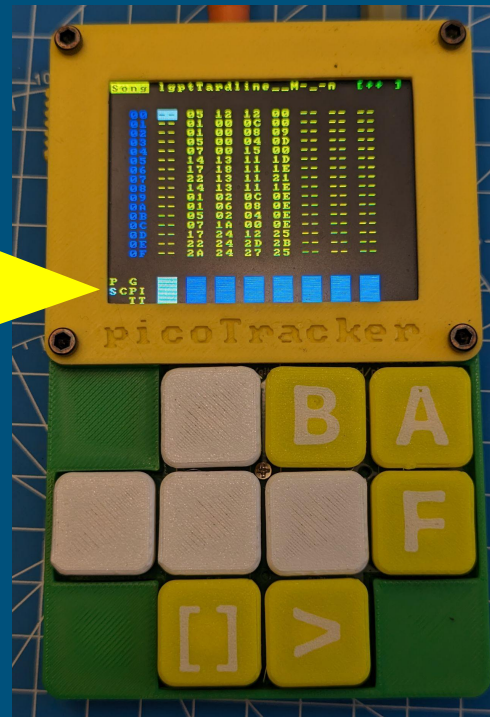
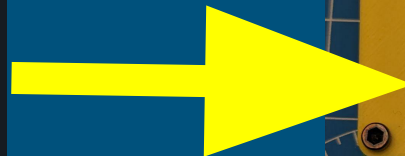
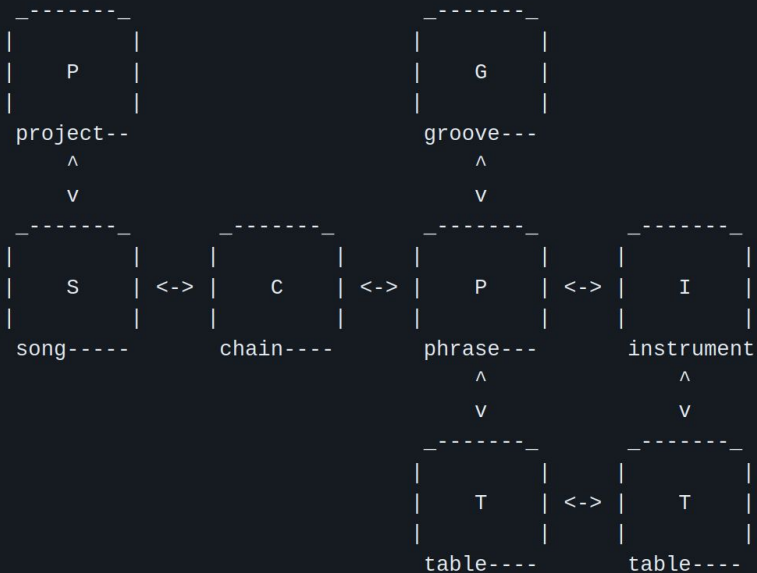
ARROWS: will make an existing selection bigger or smaller

B: copy selection to buffer, or

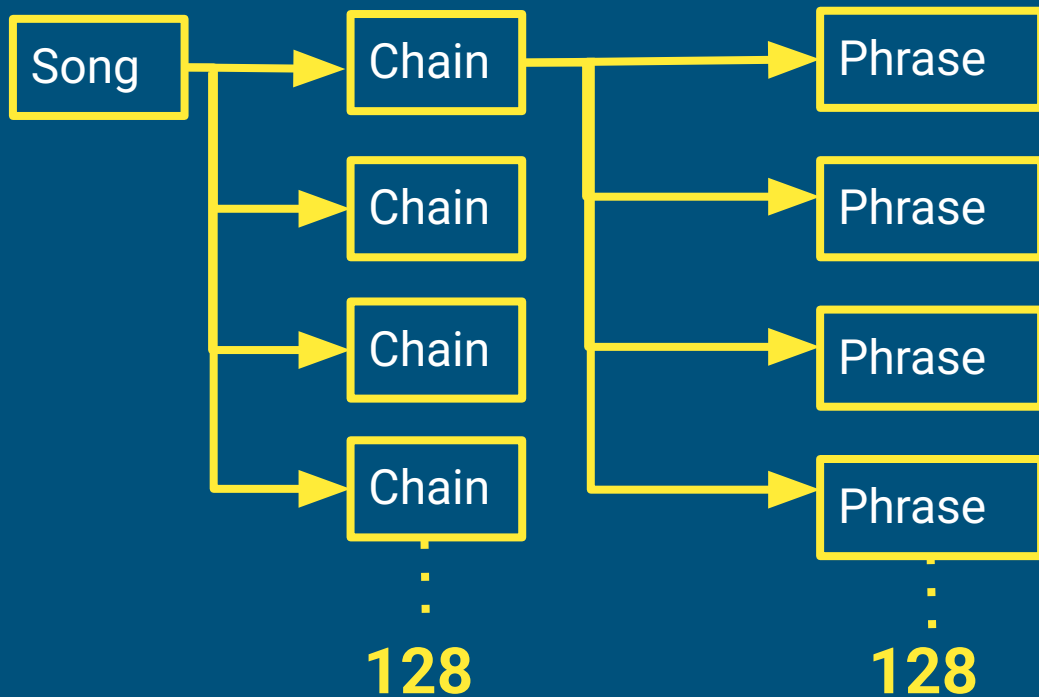
F+A: cut current selection

Navigating the picoTracker

Screen Map



Navigating the picoTracker



* Actually it's more of a **matrix** between Song, Chain & Phrases

Making music!

What can the Picotracker do?

- **8 mono** track sequencer
- Sequencer: **128** Chains with **128** Phrases, **16** Steps/Phrase
- **32** Instruments: **16** samples, **16** MIDI C-4 2
- ~**15 MB** space for samples per project
- FX Commands (Filters, Volume, etc)
- **32** “Tables”
- 8 or 16bit samples up to 44.1kHz, mono or stereo
- 16bit/44.1kHz/Stereo audio output

Making music!

What can the Picotracker do?

- **MIDI In & OUT**
- **Big feature** of picoTracker vs LGPT handhelds!
- Works great with MultiTimbral Synths! (demo)
- But won't cover it in our track making today

The tracker workflow



Making music!

Building our first track:

- Pick a tempo & scale in Project screen
- Add 00 for empty chain placeholder
- Add some percussion for a beat
- Add a baseline
- Add a chord pad

Making music!

- BPM: 125, Scale: none (Chromatic)
- Create 00
- Create Chains: 01, 02, 03, 04
- Goto into Chain 01, Create Phrase 00,
- Go into Phrase 01, Make Instrument 01, 02, ..06
- Goto browser & Import all samples for folder


Making music!

- Now let's switch to the magic spreadsheet...

The joys of being open!

- At least **3** case designs so far!
- At least **2** hardware remixes so far!
- **2** PCB remixes so far!
- **3** contributors of PRs so far!
- **~110** members in Discord server
- **3** YouTube channels with videos about the picotracker so far!

Future plans (the big todo list!)

- Synths!! (tinysynth, OPAL?, CSID?, Braids?)
- Mixer screen (per channel VU meters etc)
- More audio features: ADSR, LFOs, “DJ filter”, etc
- More commands: random, chance, delay etc
- Lots more in GitHub Issues...
- What will **YOU** build? 



@RickyTinez on YT

*Share the love,
share the knowledge,
knowledge is power,
peace!*

Thank you!



fluttercommunity.social/@maks



@maks