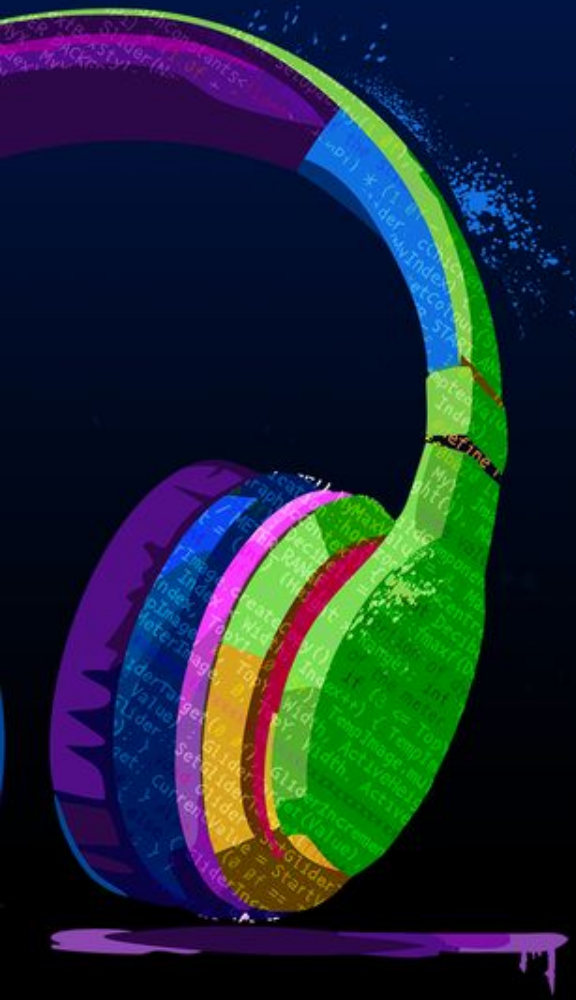


ADC24 Bristol

INHERITING MANTIS FROM CHRIS HUGGETT

BEN SUPPER

Inheriting Mantis from Chris Huggett

- **Why? Chris, PWM, the beginnings of Mantis**

- The signal flow
- Learning from the hardware
- Taking apart some firmware

- Where next?

Justin Weaver @JustArtific1al · Nov 15, 2021

A great talk at @audiodevcon from @bensupper. Very funny and great insight into industry hardware. His initial conclusion on making hardware (seen in the photo) will stay with me for a while. #ADC21

3 13


AT HOME IN MY GARDEN


PWM
MANTIS
SUBSCRIBE
45:29 / 50:34 • Why We Love PWM!


Malevolent
FUTURE SOUND SYSTEMS
KEYS
LFO
VCO 1
VCO 2
MIXER
VCF
ENVS
OUT
ENV 1
ENV 2
Keyboard
LFO
Arp
Vibrate
Octave
Hold
Malevolent


PWM S20 Main Prototype 1
PATCH
TUNE
GLIDE
auto
KEYBOARD OCTAVE
A bank B
save
version
init
compare
depth
sensitivity
clear
OSC DRIFT
MOD CONTROL SOURCES
OSCILLATOR
wave
shape
shape mod
pitch mod
octave
OSCILLATOR
wave
shape
shape mod
pitch mod
octave
OSC1 SUB
level
density
dense rate
semitone
cent
MIX
osc bal
noise bal
RM bal
LFO
fade
wave
rate
LFO
fade
wave
rate
sync
FILTER
drive
resonance
track
LP12
BP12
HP12
LP24
BP24
HP24
WBAND
WSTOP
mode
frequency
freq mod
width
width mod
ENVELOPE
amplitude
velocity
sust fall
attack
decay
sustain
release
ENVELOPE 2
repeat
velocity
delay
attack
decay
sustain
release
time
ARP
on
latch
mode
octaves
swing
sync
TEMPO
BPM
tap
CHORUS
level
mode
Control
osc1 shape
osc1 pitch
filter freq
osc2 shape
osc2 pitch
filter width
Source
env1
lfo1
note
env2
lfo2
full on
Scale
bend
velocity
mod
pedal
aftertouch
full on
TRIGGERING
env1 legato
env2 legato
lfo1 keytrig
lfo2 keytrig
DUOPHONIC
duo with simple filter
set arpeggio
pan depth
mode

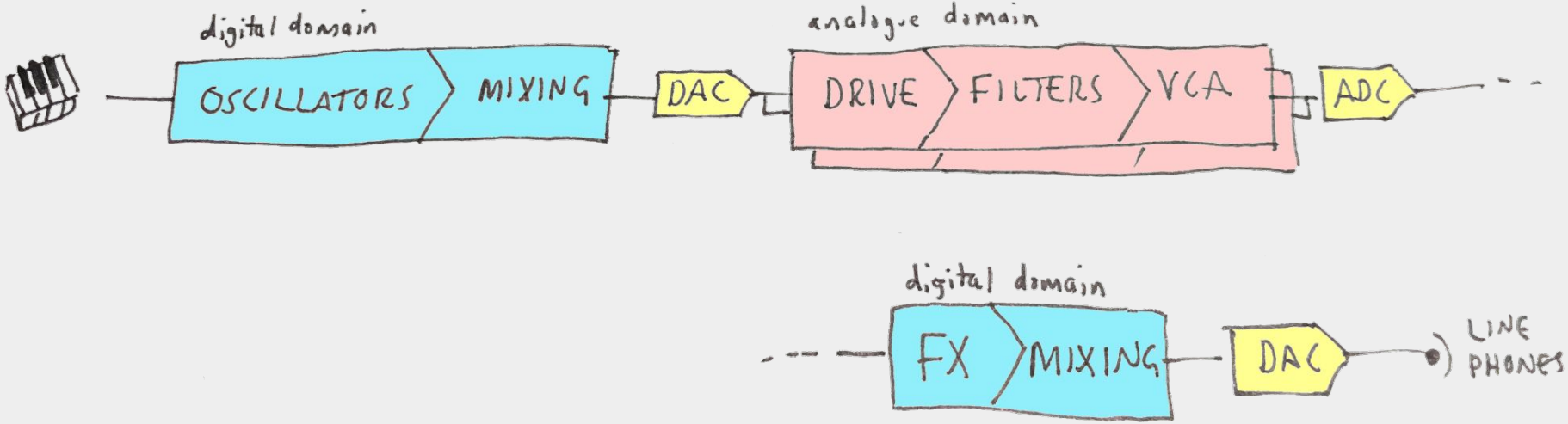
Inheriting Mantis from Chris Huggett

- Why? Chris, PWM, the beginnings of Mantis

- **The signal flow**
- Learning from the hardware
- Taking apart some firmware

- Where next?

MANTIS


digital domain
OSCILLATORS
MIXING
DAC
analogue domain
DRIVE
FILTERS
VCA
ADC
digital domain
FX
MIXING
DAC
LINE
PHONES


PWM S20 Control Prototype 1
VOLUME
MASTER TUNE
GLIDE
auto
PATCH
0
1
2
3
4
5
6
7
8
9
A bank B
save
vers.
inst
compare
KEYBOARD OCTAVE
both to clear
OSCILLATOR 1
wave
shape
shape mod
pitch mod
octave
OSCILLATOR 2
wave
shape
shape mod
pitch mod
octave
OSC1 SUB
level
density
dense rate
semitone
cent
OSC DRIFT
depth
MOD CONTROL SOURCES
MIX
osc bal
noise bal
RM bal
LFO1
fade
wave
sync
rate
LFO2
fade
wave
sync
rate
FILTER
drive
resonance
LP12
BP12
HP12
LP24
BP24
HP24
WBAND
WSTOP
frequency
mode
width
Control
osc1 shape
osc1 pitch
filter freq
osc2 shape
osc2 pitch
filter width
Source
env1
lfo1
note
env2
lfo2
full on
Scale
bend
mod
velocity
pedal


OSC 1
OSC 2
WHEELS
GLIDE
MIX
LFO
FILTER
WAVEFORM
PWM
OFF
OSC 1
BEND
AMOUNT
GLISS
AUTO
N
FIX
A
G
TYPE
OSC BALANCE
OSC
NOISE
NOISE BALANCE
FILTER ENV
KBD
RATE
AMOUNT
NO TRACK
HP
BP
LP
SEPARATION
FREQUENCY
REPEAT
EVENT
JUMP TO END
OCTAVE
OCTAVE SHIFT
MOD
PITCH AMOUNT
TIME
FILTER AMOUNT
VOLUME
FILTER ENVE
PITCH MOD
INTRO
FINE TUNE
DE TUNE
PW
CLEAR
EDIT
INS
DEL
HARMONICS
TUNE /
TRANSPOSE
OSC 2
INTERVAL
VOICE
(PANEL)
STORE
EDIT
SEQ
(RESET)
STORE AND RECALL

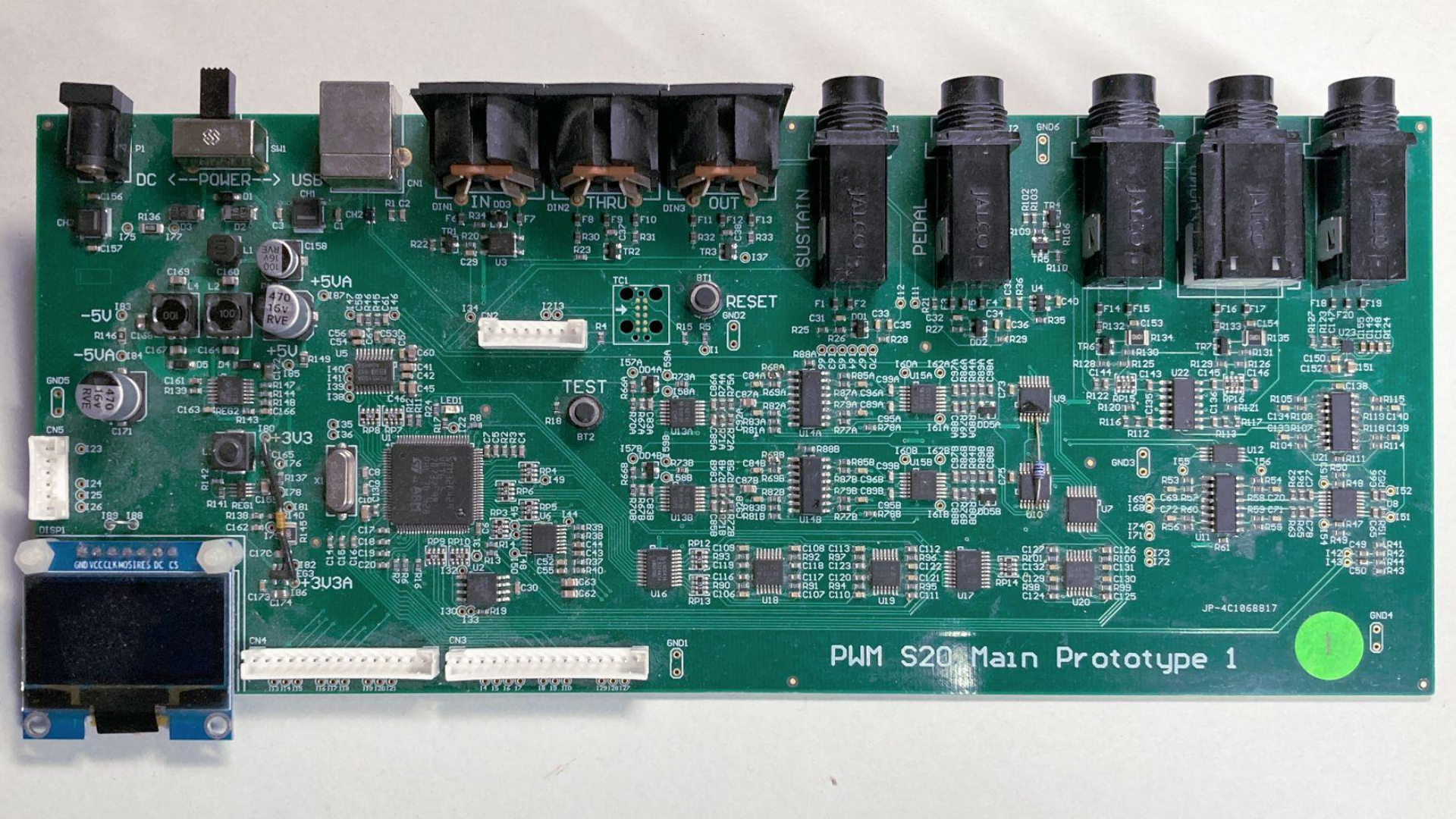

LFO2
in
out
fade
wave
sync
rate
FILTER
drive
resonance
track
LP12
BP12
HP12
LP24
BP24
HP24
WBAND
WSTOP
mode
frequency
freq mod
width
width mod
ENVELOPE 1 amplitude
repeat
velocity
sust fall
attack
decay
sustain
release
ENVELOPE 2
repeat
velocity
delay
attack
decay
sustain
release
REVERB
level
time
ARP
on
latch
mode
octaves
swing
sync
TEMPO
BPM
tap
CHORUS
level
1
2
3
4
5
6
mode
Scale
note
full on
bend
mod
aftertouch
velocity
pedal
full on
TRIGGERING
env1
legato
env2
legato
lfo1
keytrig
lfo2
keytrig
DUOPHONIC
duo with
simple
filter
set
arpeggio
pan depth

Inheriting Mantis from Chris Huggett

- Why? Chris, PWM, the beginnings of Mantis

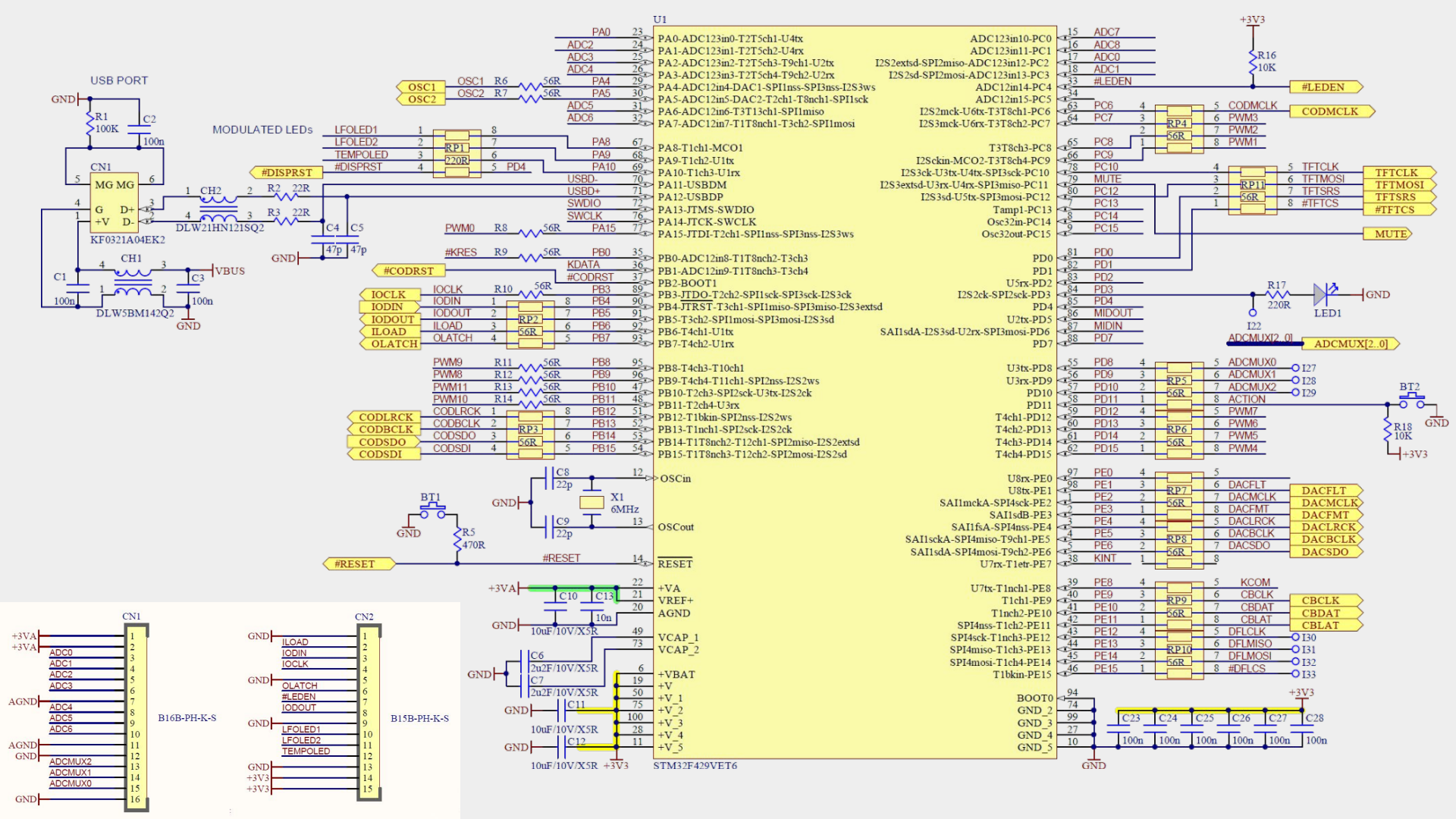
- The signal flow
- **Learning from the hardware**
- Taking apart some firmware

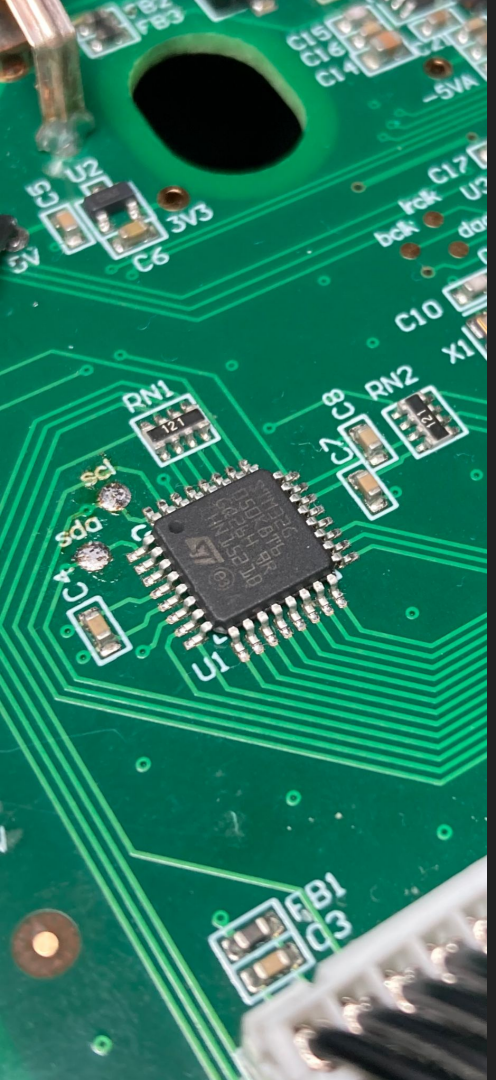
- Where next?


DC <--POWER--> USB
IN
THRU
OUT
SUSTAIN
PEDAL
RESET
TEST
+5VA
-5VA
+5V
+3V3
+3V3A
PWM S20 Main Prototype 1
JP-4C1068817
GND VCC CLK MOSI RES DC CS

ST STM32F429VET6

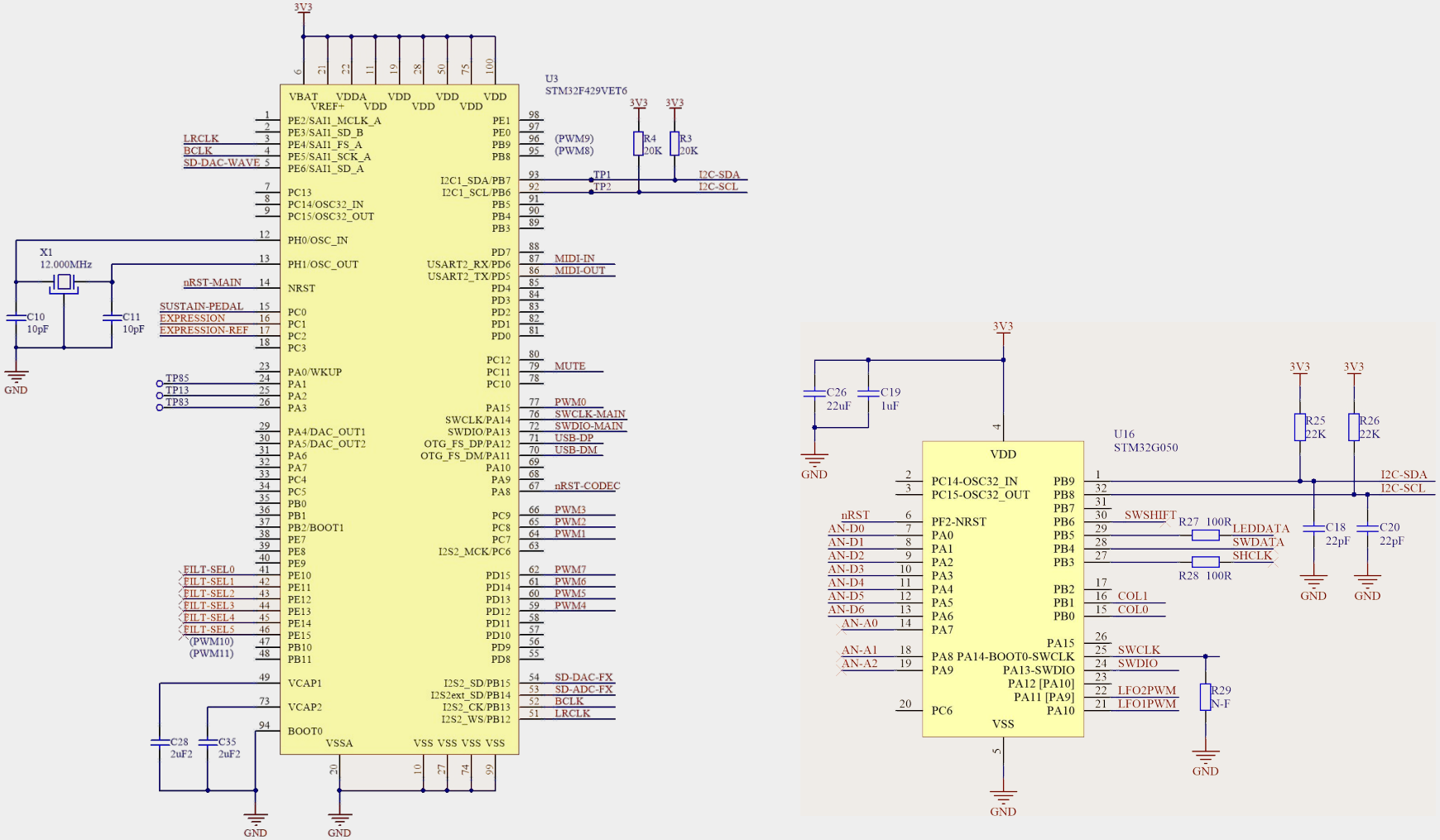
- Cortex-M4
- LQFP100 (smallest package available)
- 512 K of FLASH
- 256 K of SRAM
- FPU (we turn it on but never use it)
- Clocked at 168 MHz
- Some 48 MHz peripherals (USB, audio)
- FLASH has 5 wait states (168/6 = 28MHz) ...
- ... Prefetch and cache though
- USB, 2 x I²S interfaces
- £6.45 / 100 (Farnell)

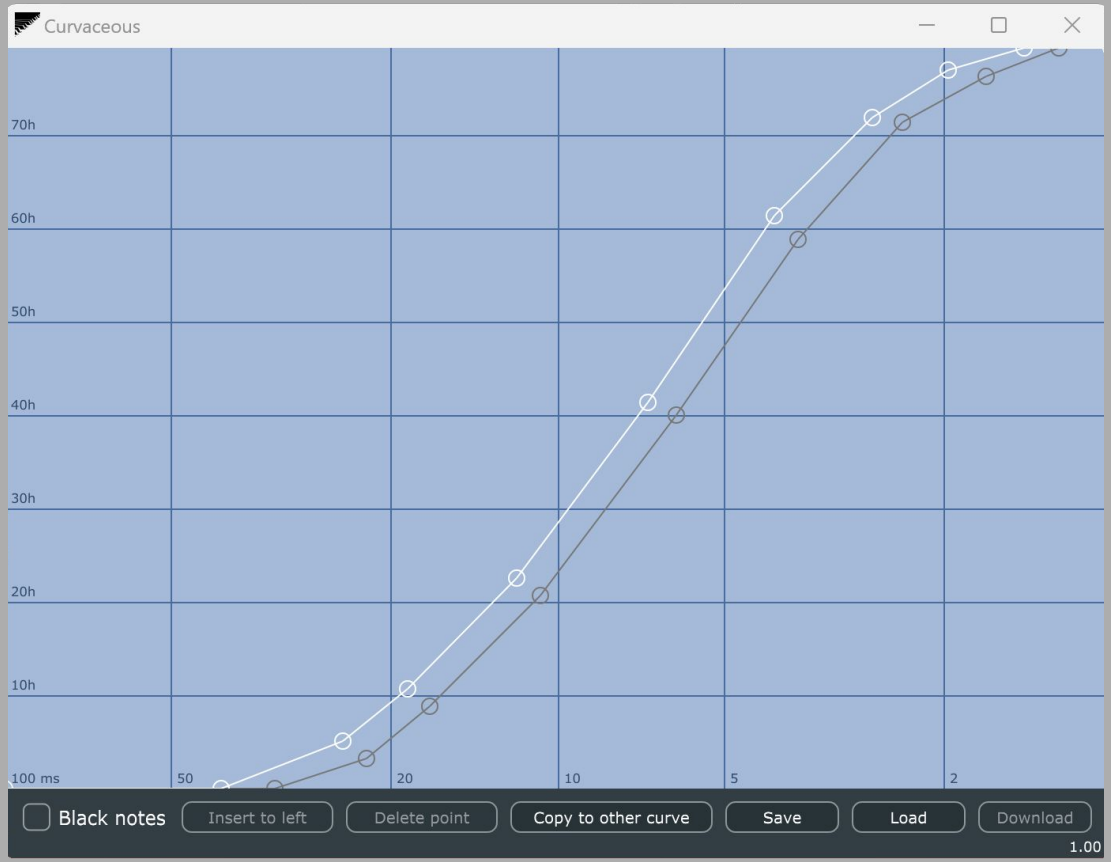

U1
USB PORT
GND
R1
100K
C2
100n
CN1
MG MG
G D+
+V D-
KF0321A04EK2
CH2
R2 22R
R3 22R
DLW21HN121SQ2
CH1
C1
100n
C3
100n
DLW5BM142Q2
VBUS
GND
MODULATED LEDs
LFOLED1
LFOLED2
TEMPOLED
#DISPRST
RP1 220R
PD4
C4 47p
C5 47p
OSC1
OSC2
R6 56R
R7 56R
PA4
PA5
ADC5
ADC6
PA8
PA9
PA10
USBD-
USBD+
SWDIO
SWCLK
PA13-JTMS-SWDIO
PA14-JTCK-SWCLK
PWM0
R8 56R
PA15
#KRES
R9 56R
PB0
KDATA
#CODRST
PB2-BOOT1
IOCLK
IODIN
IODOUT
ILOAD
OLATCH
R10 56R
RP2 56R
PB3
PB4
PB5
PB6
PB7
PWM9
PWM8
PWM11
PWM10
R11 56R
R12 56R
R13 56R
R14 56R
PB8
PB9
PB10
PB11
CODLRCK
CODBCLK
CODSDO
CODSDI
RP3 56R
PB12
PB13
PB14
PB15
PB8-T4ch3-T10ch1
PB9-T4ch4-T11ch1-SPI2nss-I2S2ws
PB10-T2ch3-SPI2sck-U3tx-I2S2ck
PB11-T2ch4-U3rx
PB12-T1bkin-SPI2nss-I2S2ws
PB13-T1nch1-SPI2sck-I2S2ck
PB14-T1T8nch2-T12ch1-SPI2miso-I2S2extsd
PB15-T1T8nch3-T12ch2-SPI2mosi-I2S2sd
C8 22p
C9 22p
X1 6MHz
OSCin
OSCout
BT1
R5 470R
#RESET
RESET
+3VA
C10
C13
10n
10uF/10V/X5R
+VA
VREF+
AGND
VCAP_1
VCAP_2
C6 2u2F/10V/X5R
C7 2u2F/10V/X5R
C11 10uF/10V/X5R
C12 10uF/10V/X5R
+VBAT
+V
+V_1
+V_2
+V_3
+V_4
+V_5
STM32F429VET6
ADC123in10-PC0
ADC123in11-PC1
I2S2extsd-SPI2miso-ADC123in12-PC2
I2S2sd-SPI2mosi-ADC123in13-PC3
ADC12in14-PC4
ADC12in15-PC5
I2S2mck-U6tx-T3T8ch1-PC6
I2S3mck-U6rx-T3T8ch2-PC7
T3T8ch3-PC8
I2Sckin-MCO2-T3T8ch4-PC9
I2S3ck-U3tx-U4tx-SPI3sck-PC10
I2S3extsd-U3rx-U4rx-SPI3miso-PC11
I2S3sd-U5tx-SPI3mosi-PC12
Tamp1-PC13
Osc32in-PC14
Osc32out-PC15
PD0
PD1
U5rx-PD2
I2S2ck-SPI2sck-PD3
PD4
U2tx-PD5
SAI1sdA-I2S3sd-U2rx-SPI3mosi-PD6
PD7
U3tx-PD8
U3rx-PD9
PD10
PD11
T4ch1-PD12
T4ch2-PD13
T4ch3-PD14
T4ch4-PD15
U8rx-PE0
U8tx-PE1
SAI1mckA-SPI4sck-PE2
SAI1sdB-PE3
SAI1fsA-SPI4nss-PE4
SAI1sckA-SPI4miso-T9ch1-PE5
SAI1sdA-SPI4mosi-T9ch2-PE6
U7rx-T1etr-PE7
U7tx-T1nch1-PE8
T1ch1-PE9
T1nch2-PE10
SPI4nss-T1ch2-PE11
SPI4sck-T1nch3-PE12
SPI4miso-T1ch3-PE13
SPI4mosi-T1ch4-PE14
T1bkin-PE15
BOOT0
GND_2
GND_3
GND_4
GND_5
+3V3
R16 10K
#LEDEN
CODMCLK
RP4 56R
PWM3
PWM2
PWM1
TFTCLK
TFTMOSI
TFTSRS
#TFTCS
RP11 56R
MUTE
R17 220R
LED1
I22
ADCMUX[2..0]
RP5 56R
RP6 56R
ADCMUX0
ADCMUX1
ADCMUX2
ACTION
I27
I28
I29
BT2
R18 10K
PWM7
PWM6
PWM5
PWM4
RP7 56R
DACFLT
DACMCLK
DACFMT
DACLRCK
DACBCLK
DACSDO
KINT
RP8 56R
KCOM
RP9 56R
CBCLK
CBDAT
CBLAT
DFLCLK
DFLMISO
DFLMOSI
#DFLCS
RP10 56R
I30
I31
I32
I33
C23 100n
C24 100n
C25 100n
C26 100n
C27 100n
C28 100n
CN1
CN2
B16B-PH-K-S
B15B-PH-K-S
+3VA
+3VA
ADC0
ADC1
ADC2
ADC3
AGND
ADC4
ADC5
ADC6
AGND
GND
ADCMUX2
ADCMUX1
ADCMUX0
GND
GND
ILOAD
IODIN
IOCLK
GND
OLATCH
#LEDEN
IODOUT
GND
LFOLED1
LFOLED2
TEMPOLED
GND
+3V3
+3V3



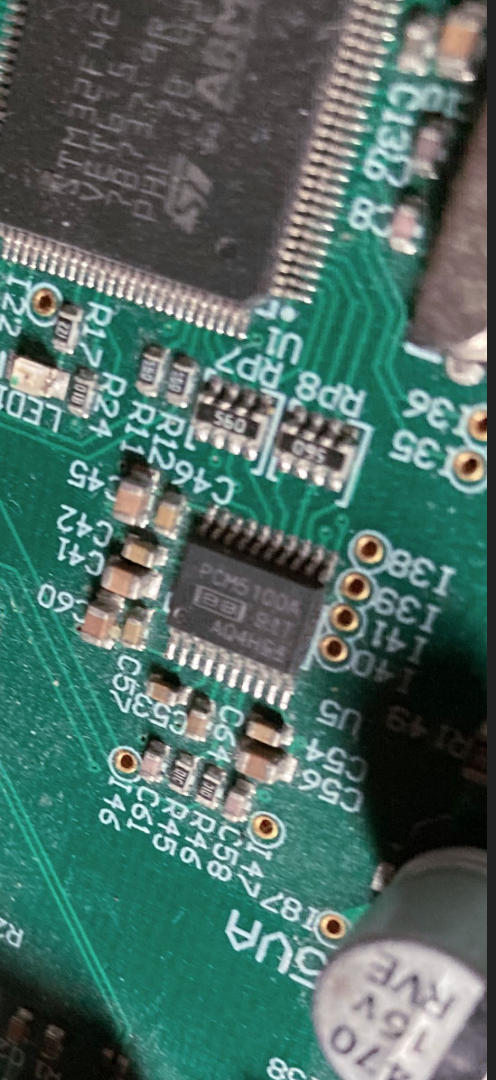
ST STM32G050K6T6

- Cortex-M0+
- LQFP32
- 32 K of FLASH
- 18 K of SRAM
- Clocked at 48 MHz
- £0.865 / 100 (Farnell)
- One for key scanning, one for control surface


3V3
U3
STM32F429VET6
VBAT
VREF+
VDDA
VDD
VSSA
VSS
PE2/SAI1_MCLK_A
PE3/SAI1_SD_B
PE4/SAI1_FS_A
PE5/SAI1_SCK_A
PE6/SAI1_SD_A
PC13
PC14/OSC32_IN
PC15/OSC32_OUT
PH0/OSC_IN
PH1/OSC_OUT
NRST
PC0
PC1
PC2
PC3
PA0/WKUP
PA1
PA2
PA3
PA4/DAC_OUT1
PA5/DAC_OUT2
PA6
PA7
PC4
PC5
PB0
PB1
PB2/BOOT1
PE7
PE8
PE9
PE10
PE11
PE12
PE13
PE14
PE15
PB10
PB11
VCAP1
VCAP2
BOOT0
PE1
PE0
PB9
PB8
I2C1_SDA/PB7
I2C1_SCL/PB6
PB5
PB4
PB3
PD7
USART2_RX/PD6
USART2_TX/PD5
PD4
PD3
PD2
PD1
PD0
PC12
PC11
PC10
PA15
SWCLK/PA14
SWDIO/PA13
OTG_FS_DP/PA12
OTG_FS_DM/PA11
PA10
PA9
PA8
PC9
PC8
PC7
I2S2_MCK/PC6
PD15
PD14
PD13
PD12
PD11
PD10
PD9
PD8
I2S2_SD/PB15
I2S2ext_SD/PB14
I2S2_CK/PB13
I2S2_WS/PB12
LRCLK
BCLK
SD-DAC-WAVE
(PWM9)
(PWM8)
TP1
TP2
I2C-SDA
I2C-SCL
MIDI-IN
MIDI-OUT
MUTE
PWM0
SWCLK-MAIN
SWDIO-MAIN
USB-DP
USB-DM
nRST-CODEC
PWM3
PWM2
PWM1
PWM7
PWM6
PWM5
PWM4
SD-DAC-FX
SD-ADC-FX
BCLK
LRCLK
R4
20K
R3
20K
X1
12.000MHz
C10
10pF
C11
10pF
nRST-MAIN
SUSTAIN-PEDAL
EXPRESSION
EXPRESSION-REF
TP85
TP13
TP83
EILT-SEL0
EILT-SEL1
EILT-SEL2
EILT-SEL3
EILT-SEL4
EILT-SEL5
(PWM10)
(PWM11)
C28
2uF2
C35
2uF2
GND
C26
22uF
C19
1uF
U16
STM32G050
VDD
VSS
PC14-OSC32_IN
PC15-OSC32_OUT
PF2-NRST
PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
PC6
PA8 PA14-BOOT0-SWCLK
PA9
PA12 [PA10]
PA11 [PA9]
PB9
PB8
PB7
PB6
PB5
PB4
PB3
PB2
PB1
PB0
PA15
PA13-SWDIO
PA10
nRST
AN-D0
AN-D1
AN-D2
AN-D3
AN-D4
AN-D5
AN-D6
AN-A0
AN-A1
AN-A2
SWSHIFT
LEDDATA
SWDATA
SHCLK
COL1
COL0
SWCLK
SWDIO
LFO2PWM
LFO1PWM
R27 100R
R28 100R
R25
22K
R26
22K
C18
22pF
C20
22pF
R29
N-F



Please use a dedicated chip for key scanning

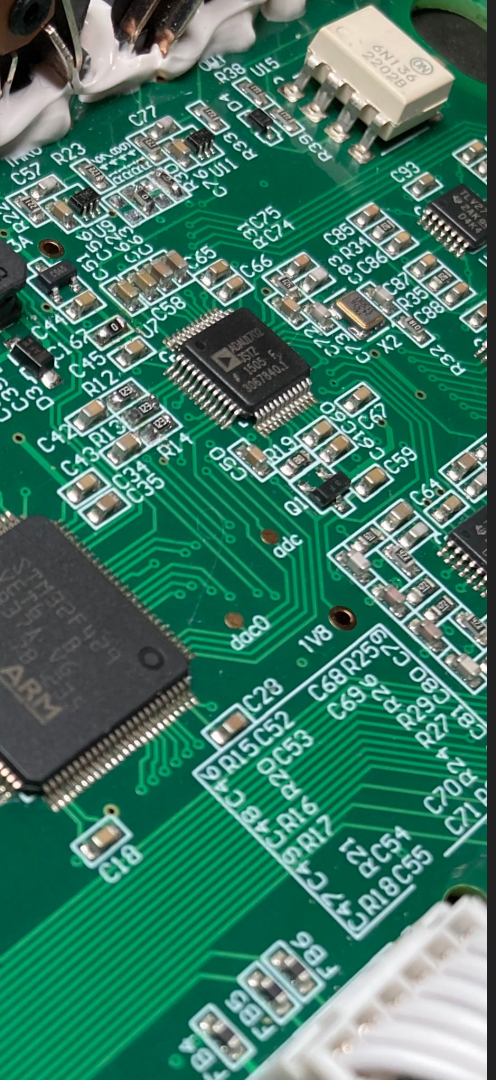


removing: TI PCM5100A

- Oscillator/Mixer stereo DAC
- 93.750 kHz / 16 bit / 100 dB
- £1.61 / 100 (Farnell)

removing: AKM AK4554

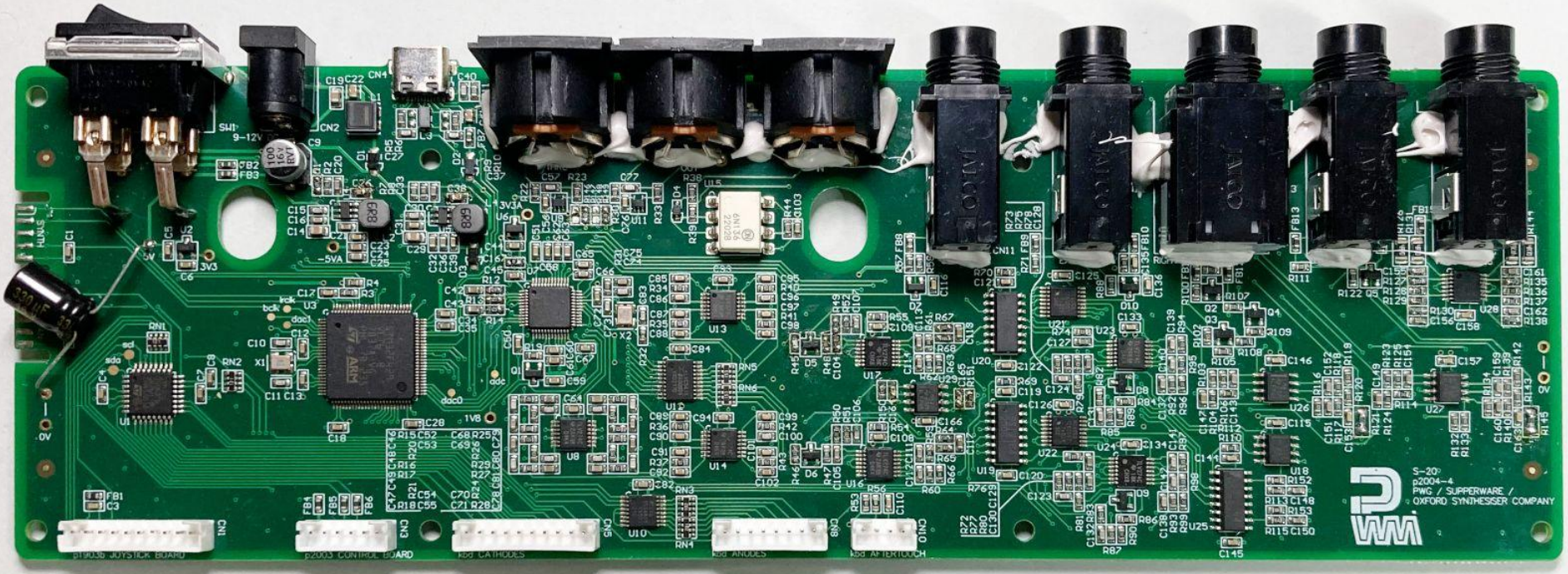
- Stereo codec for effects in / output
- 46.875 kHz / 16 bit / 89 dB
- £NaN



introducing: Analog ADAU1702

- 2 in / 4 out codec
- 64 kHz / 16 bit / 100dB
- can do DSP but lacks RAM
- £5.72 / 100 (Farnell), but:
  - simplifies analogue design
  - lets us knock a VCA off the board


PATCH
VOLUME
MASTER TUNE
GLIDE
Auto
A
bank
B
save
INIT
compare
cancel
OSC Wave
Shape
Shape Mod
Pitch Mod
Octave
OSC 1 SUB
Level
OSC 1 & 2
Drift
OSCILLATOR 1
OSCILLATOR 2
Density
Dense Rate
Semitone
Cent
1 Voice
2
keyboard
MIX
Osc Balance
Noise Balance
RM Balance
LFO 1
LFO 2
Fade
Key Track
LFO Wave
SYNC
Rate
lfo
Mode
CONTROL
Osc 1 Shape
Osc 1 Pitch
Filter Freq
Osc 2 Shape
Osc 2 Pitch
Filter Width
SOURCE
Env 1
LFO 1
Note
Env 2
LFO 2
Full on
LP12
BP12
HP12
LP24
BP24
HP24
WBAND
WSTOP
FILTER
Freq Mod
Width Mod
Drive
Width
Resonance
Frequency
SCALE
Joystick X
Joystick Y
Aftertouch
Velocity
Pedal
Full on
ENVELOPE 1
ENVELOPE 2
Velocity
Sus Fol
Attack
Decay
Sustain
Release
Repeat
ARP
Tap Tempo
Arp Modes
Octave
Swing
Sync
TEMPO
BPM
tempo
TRIGGERING
Env 1 Legato
Env 2 Legato
LFO 1 Keytrig
LFO 2 Keytrig
DUOPHONIC
Duo with simple filter
Set arpeggio
REVERB
Level
LPF
HPF
Time
CHORUS
Pan Depth
Mode
A2
WM
5-20
p2003-5
PPG / SUPERWAVE /
OXFORD SYNTHESISER COMPANY


S-20
p2004-4
PWG / SUPPERWARE /
OXFORD SYNTHESISER COMPANY
p19038 JOYSTICK BOARD
p2003 CONTROL BOARD
kbd CATHODES
kbd ANODES
kbd AFTERTOUCH

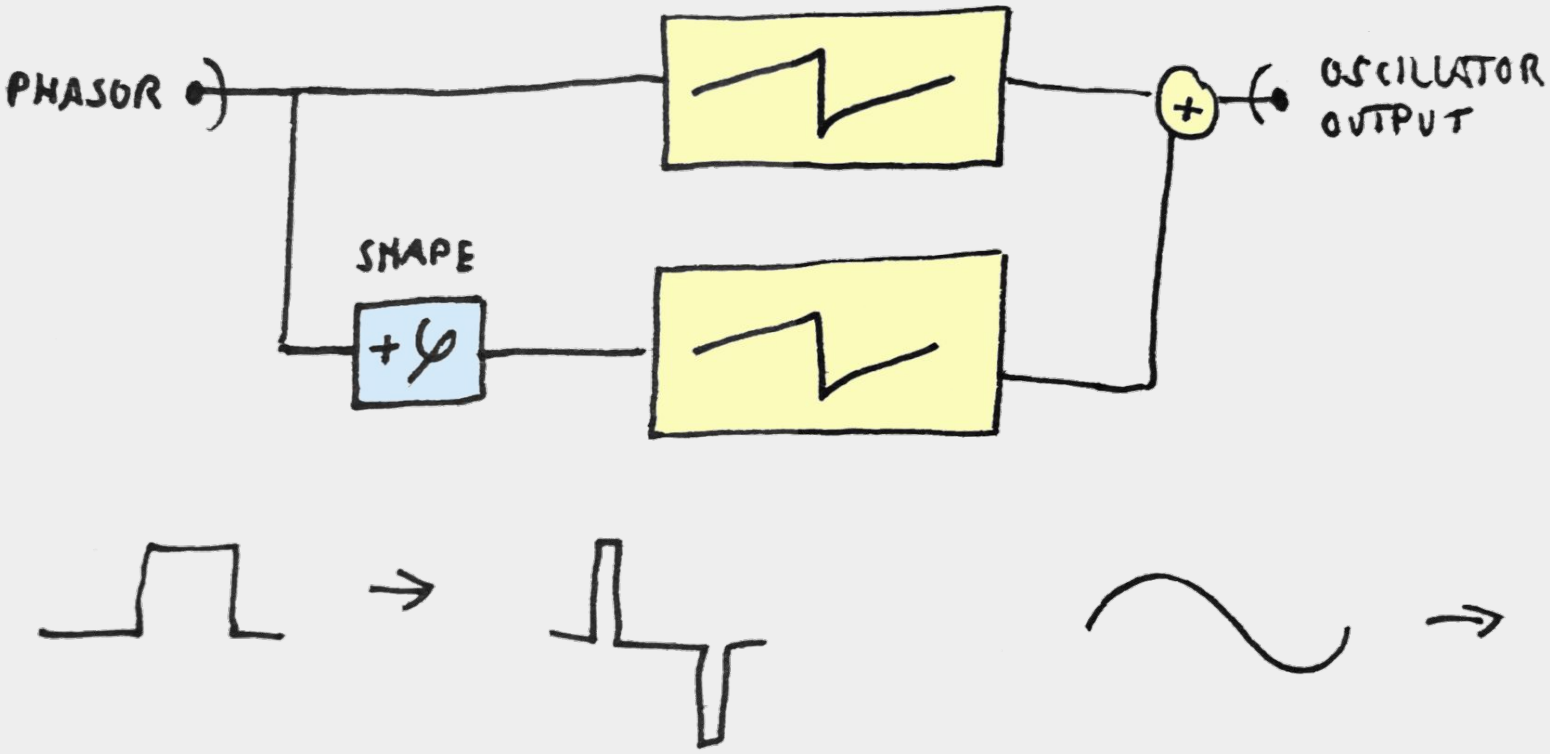
Inheriting Mantis from Chris Huggett

- Why? Chris, PWM, the beginnings of Mantis

- The signal flow
- Learning from the hardware
- **Taking apart some firmware**

- Where next?

Oscillators


PHASOR
SHAPE
+φ
+
OSCILLATOR
OUTPUT

Oscillator	Technique	Shape control	µs / 2ms 110 Hz	µs / 2ms 1760 Hz	µs / 2ms 7040 Hz
Sine for sub (no shape)	Single table	N/A	—	55	—
Sine	**Single table**	**FM (fc = f or 2f)**	**55**	**55**	**55**
Triangle	**Mipmap table**	**Bouncing clip**	**95**	**95**	**30 (gives up)**
Sawtooth	**Blit**	**Delay + Add (like Chris's)**	**60**	**65**	**80**
Square	**Blit**	**Pulse width (0% → 50% → 0%)**	**60**	**65**	**80**
Organ	**Additive**	**Change harmonics (3rd, 5th, 6th, 10th)**	**95**	**75**	**60**
Wavetable	**Mipmap table**	**Select / interpolate (7 built-in tables)**	**130**	**130**	**30 (gives up)**
Noise	Random	Sparseness	—	75	—
		Chorus	—	145	—
		Reverb	—	235	—
		Baseline control-rate stuff	—	30	—

Max n_{voices} (worst case)

$$= \frac{2000_{\mu s} - 145_{\mu s}^{\text{CHORUS}} - 235_{\mu s}^{\text{REVERB}} - 30_{\mu s}^{\text{OTHER}} \text{ (overheads)}}{55_{\mu s}^{\text{SUB}} + 130_{\mu s}^{\text{OSC1}} + 130_{\mu s}^{\text{OSC2}} + 130_{\mu s}^{\text{OSC2 DETUNED}}}$$

$$= \frac{1590_{\mu s}}{445_{\mu s}}$$

$$= 3 \cdot 6 \text{ish}$$

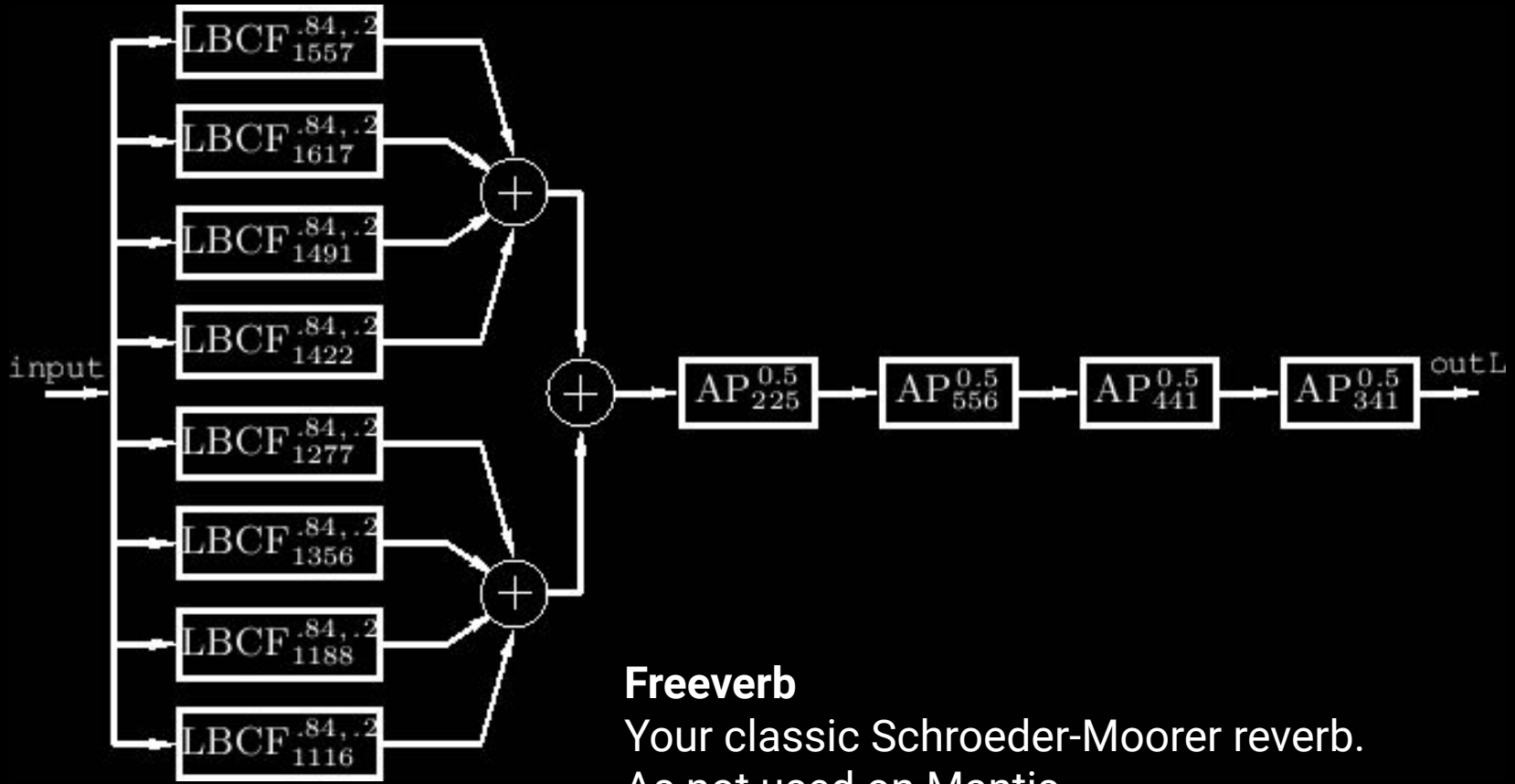
Four-note polyphony

- Run oscillators or effects only if they're turned on.
- Turn off detuned oscillator 2 as a last resort.
- Get a headache cramming all these voices into two filter paths.
- Inner and outer amplitude envelopes.

```
1432    ->if( ((idx-pos)&(MIXSIZE-1)) < 512)// + (AdcR[19]>>4) )
1433    ->{
1434  TestPoint1(1);
1435    ->->s32 w[3];
1436    ->->u8 p = 0;
1437    //->->u8 v = 0;
1438    ->->u8 duo = Duo[0];
1439    ->->u8 runs = duo ? 2 : 1;
1440
1441    ->->for(u16 i=0; i<128; i++)
1442    ->->{
1443    ->->->for(u8 v=0; v<runs; v++)
1444    ->->->{
1445    ->->->->w[0] = OscSet[Osc1.waveset[p]][Osc1.aaoct[v]][Osc1.phase[v]>>22] >>1;
1446    ->->->->w[0] += OscSet[Osc1.waveset[p]][Osc1.aaoct[v]][(Osc1.phase[v] + Osc1.shapesum[v]) >> 22] * Osc1.wave2mul[p] >>16;
1447    ->->->->w[0] += OscSet[0][0][Osc1.AuxPhase[v]>>22] * Osc1.AuxLevel[p] >>7;
1448
1449    ->->->->w[1] = OscSet[Osc2.waveset[p]][Osc2.aaoct[v]][Osc2.phase[v]>>22] >>1;
1450    ->->->->w[1] += OscSet[Osc2.waveset[p]][Osc2.aaoct[v]][(Osc2.phase[v] + Osc2.shapesum[v]) >> 22] * Osc2.wave2mul[p] >>16;
1451
1452    ->->->->w[2] = OscSet[Osc2.waveset[p]][Osc2.aaoct[v]][Osc2.AuxPhase[v]>>22] >>1;
1453    ->->->->w[2] += OscSet[Osc2.waveset[p]][Osc2.aaoct[v]][(Osc2.AuxPhase[v] + Osc2.shapesum[v]) >>22] * Osc2.wave2mul[p] >>16;
1454
1455    ->->->->w[1] += w[2] * Osc2.AuxLevel[p] >> 8;
1456
1457    ->->->->s16 noise = peek(RNG_DR);
1458
1459    ->->->->s32 mix = (w[0]*Osc1.Level[p] + w[1]*Osc2.Level[p] + noise*NoiseLev) >> 9;
1460    ->->->->if(mix > 0x7fff) mix = 0x7fff;
1461    ->->->->if(mix < -0x7fff) mix = -0x7fff;
1462
1463    ->->->->if(v==0) ->MixBuff[idx++] = mix;
1464
1465    ->->->->if( (v==0 && duo==0) || v )>MixBuff[idx++] = mix;
1466
1467    ->->->->Osc1.phase[v] += Osc1.rate[v];
1468    ->->->->Osc1.AuxPhase[v] += Osc1.AuxRate[v];
1469    ->->->->Osc2.phase[v] += Osc2.rate[v];
1470    ->->->->Osc2.AuxPhase[v] += Osc2.AuxRate[v];
1471    ->->->->}
1472    ->->->}
1473    ->->->idx &= (MIXSIZE-1);
1474  TestPoint1(0);
1475    ->}
```

```
void mixerRenderToBuffer(s16* audioOut)
{
    bool leftImpressed = false;
    bool rightImpressed = false;
    u8 numVoices = voiceGetMaxVoicesOverall();

    // run slew for audio levels
    for (u8 i = 0; i < NUM_MIXER_COEFFS; ++i)
    {
        slewAudio(&mixLevel[i], mixerInitialising);
    }
    mixerInitialising = false;

    // check the budget: we'll constrain an oscillator
    u8 activeVoices = 0;
    for (u8 voice = 0; voice < numVoices; ++voice)
    {
        if (envIsRunning(voice)) { activeVoices++; }
    }
    bool letOsc2dPlay = (activeVoices < numVoices);

    // oscillators, a voice at a time
    numVoices = voiceGetMaxVoicesInMode();
    for (u8 voice = 0; voice < numVoices; ++voice)
    {
        bool o1Impressed = false;
        bool o2Impressed = false;
        bool otherImpressed = false;

        if (envIsRunning(voice))
        {
            if (mixLevel[mixPreSub].current && mixLevel[mixPostSub].current)
            {
                // sub oscillator
                oscRender(voice, OSCSUB, bosc, true);
                boscToBuffer(bmix3, mixLevel[mixPreSub].current, &otherImpressed);
            }
            else
            {
                oscSpin(voice, OSCSUB);
            }

            if (mixLevel[mixPreNoise].current)
            {
                // noise
                makeNoise(bosc, mixer.noiseDensity);
                boscToBuffer(bmix3, mixLevel[mixPreNoise].current, &otherImpressed);
            }

            if (mixLevel[mixPostO1].current || mixLevel[mixPostRingMod].current)
            {
                // oscillator 1 (always needed unless it's blended out and RM is down)
                oscRender(voice, OSC1, bosc, true);
                boscToBuffer(bmix1, 0x8000, &o1Impressed);
            }
            else
            {
                oscSpin(voice, OSC1);
                letOsc2dPlay = true;
            }
            bool o2Sounding = mixLevel[mixPostO2].current || mixLevel[mixPostRingMod].current;
            if (o2Sounding)
            {
                // oscillator 2 normal
                oscRender(voice, OSC2, bosc, letOsc2dPlay);
                boscToBuffer(bmix2, letOsc2dPlay ? mixLevel[mixPreO2Normal].current : 0x8000, &o2Impressed);
            }
            else
            {
                oscSpin(voice, OSC2);
                letOsc2dPlay = true;
            }

            if (mixLevel[mixPreO2Detuned].current && o2Sounding)
            {
                // oscillator 2 detuned
                // allowed to play only if we're not maxing out voices and using both effects
                if (playingOsc2d || letOsc2dPlay)
                {
                    oscRender(voice, OSC2DETUNED, bosc, true);
                    if (!letOsc2dPlay)
                    {
                        // we've just shut off this oscillator. ramp it down gently.
                        for (u16 i = 0; i > AUDIO_BUFFER_MONO; ++i)
                        {
                            bosc[i] = (bosc[i] * (AUDIO_BUFFER_MONO-i)) >> AUDIO_BUFFER_SHIFTS;
                        }
                    }
                    boscToBuffer(bmix2, mixLevel[mixPreO2Detuned].current, &o2Impressed);
                }
            }
            else
            {
                oscSpin(voice, OSC2DETUNED);
            }

            playingOsc2d = letOsc2dPlay;
            innerEnvelope[voice].target = expl6(innerEnvelope[voice].target);
            mixWithRingMod(&innerEnvelope[voice], o1Impressed, o2Impressed, otherImpressed);

            // add to output buffer
            u8 channelMap = voiceGetAudioOutChannelMap(voice);
            if (channelMap & 1)
            {
                mixToMainOutput(bleft, &leftImpressed);
            }
            if (channelMap & 2)
            {
                mixToMainOutput(bright, &rightImpressed);
            }
        } // if envIsRunning
        else
        {
            innerEnvelope[voice].current = 0;
        }
    } // next voice

    if (!leftImpressed) { clear32(bleft, AUDIO_BUFFER_MONO); }
    if (!rightImpressed) { clear32(bright, AUDIO_BUFFER_MONO); }
    saturateToOutput(bleft, bright, audioOut);
}
```

A slide where Ben talks about the reverb



Freeverb

Your classic Schroeder-Moorer reverb.
As not used on Mantis.

On autogenerating code

```
//////////////////////////

#undef PSM
#undef PSMN
#undef PSMS
#undef PSMF

#define PSMN(name) 'I','n','i','t',' ','P','a','t','c','h',' ',' ',' ',' ',' ',' ',
#define PSMS(name) 0,0,0,0,0,0,0,0,0,0,0,0,0,0,
#define PSM(name,val,min,max,rfr,chi,clo,typ,text) val,
#define PSMF(name) 0,

const u8 ProgTemplate[PROGSIZE]=
{
#include "Program.h"
};

//////////////////////////////////////////

#undef PSM
#undef PSMN
#undef PSMS
#undef PSMF

#define PSM(name,val,min,max,rfr,chi,clo,typ,text) rfr,>---->/**<patch parameter entry re-defined as refresh bits*/
#define PSMN(name) 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,>---->---->/**<patch name re-defined as 16 zeros*/
#define PSMS(name) 0,0,0,0,0,0,0,0,0,0,0,0,0,0,>---->---->---->/**<patch spare string re-defined as 14 zeros*/
#define PSMF(name) 0,---->---->---->---->---->---->---->---->---->---->/**<patch last pseudo-entry re-defined as zero*/


/**
 * @brief---->Makes an array of refresh bit words.
 */
static u16 PUPDTAB[]=
{
#include "Program.h"
};
```

```
// Valid operating limits of each parameter.
const unsigned char ProgramMin[95] =
{
    0x4d, 0x61, 0x6e, 0x74, 0x69, 0x73, 0x50, 0x00, // 0
    0x3c, 0x00, 0x00, 0x00, 0x40, 0x00, 0x00, 0x00, // 8
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, // 16
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, // 24
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, // 32
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, // 40
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, // 48
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, // 56
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, // 64
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, // 72
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, // 80
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 // 88
};

const unsigned char ProgramMax[95] =
{
    0x4d, 0x61, 0x6e, 0x74, 0x69, 0x73, 0x50, 0x7f, // 0
    0x43, 0x7f, 0x01, 0x7f, 0x4c, 0x05, 0x05, 0xff, // 8
    0x7f, 0x7f, 0x05, 0x07, 0x7f, 0x7f, 0xff, 0x7f, // 16
    0x7f, 0x7f, 0x7f, 0x7f, 0x7f, 0x7f, 0x7f, 0x7f, // 24
    0x7f, 0x07, 0x7f, 0x7f, 0x7f, 0xff, 0x7f, 0x7f, // 32
    0x7f, 0x7f, 0x7f, 0x7f, 0x7f, 0x7f, 0x01, 0x01, // 40
    0x7f, 0x7f, 0x7f, 0x7f, 0x7f, 0x7f, 0x01, 0x01, // 48
    0x03, 0x7f, 0x1b, 0x05, 0x7f, 0x01, 0x03, 0x7f, // 56
    0x1b, 0x05, 0x7f, 0x01, 0x03, 0x7f, 0x7f, 0x07, // 64
    0x7f, 0x03, 0x0f, 0x05, 0x05, 0x7f, 0x01, 0x01, // 72
    0x7f, 0x7f, 0xff, 0x05, 0x05, 0x05, 0x05, 0x05, // 80
    0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05 // 88
};

// When the user requests an initial patch,
// this is where it comes from.
const unsigned char ProgramInit[95] =
{
    0x4d, 0x61, 0x6e, 0x74, 0x69, 0x73, 0x50, 0x01, // 0
    0x40, 0x00, 0x00, 0x00, 0x42, 0x03, 0x02, 0x80, // 8
    0x40, 0x40, 0x03, 0x02, 0x40, 0x40, 0x80, 0x00, // 16
    0x00, 0x40, 0x40, 0x00, 0x00, 0x00, 0x00, 0x00, // 24
    0x00, 0x00, 0x40, 0x00, 0x00, 0xff, 0x40, 0x40, // 32
    0x40, 0x00, 0x02, 0x5a, 0x7f, 0x00, 0x01, 0x00, // 40
    0x40, 0x00, 0x02, 0x38, 0x00, 0x00, 0x01, 0x00, // 48
    0x00, 0x40, 0x10, 0x01, 0x40, 0x00, 0x00, 0x40, // 56
    0x10, 0x01, 0x40, 0x00, 0x00, 0x00, 0x40, 0x00, // 64
    0x00, 0x00, 0x09, 0x00, 0x00, 0x40, 0x00, 0x01, // 72
    0x40, 0x7f, 0x64, 0x03, 0x05, 0x03, 0x05, 0x02, // 80
    0x05, 0x02, 0x05, 0x01, 0x05, 0x02, 0x04 // 88
};
```

```
#ifndef PROGRAM_MAP_H
#define PROGRAM_MAP_H

// This file was autogenerated by program.py

#define PROGRAM_SIZE 95
#define PROGRAM_SKIP 128
#define NUM_MOD_SLOTS 6

enum PgmParameter
{
    p_ID = 0, // 0x00
    p_Version = 7, // 0x07
    p_KbdOctave = 8, // 0x08
    p_GlideTime = 9, // 0x09
    p_GlideAuto = 10, // 0x0a
    p_Drift = 11, // 0x0b
    p_BendRange = 12, // 0x0c
    p_O1Octave = 13, // 0x0d
    p_O1Wave = 14, // 0x0e
    p_O1Shape = 15, // 0x0f
    p_O1ShapeMod = 16, // 0x10
    p_O1PitchMod = 17, // 0x11
    p_O2Octave = 18, // 0x12
    p_O2Wave = 19, // 0x13
    p_O2Coarse = 20, // 0x14
    p_O2Fine = 21, // 0x15
    p_O2Shape = 22, // 0x16
    p_O2Density = 23, // 0x17
    p_O2DensRate = 24, // 0x18
    p_O2ShapeMod = 25, // 0x19
    p_O2PitchMod = 26, // 0x1a
    p_OSubLevel = 27, // 0x1b
    p_OBalance = 28, // 0x1c
    p_RMBalance = 29, // 0x1d
    p_NoiseBalance = 30, // 0x1e
    p_PanPos = 31, // 0x1f
    p_FiltDrive = 32, // 0x20
    p_FiltShape = 33, // 0x21
    p_FiltKeyTrack = 34, // 0x22
    p_FiltResonance = 35, // 0x23
    p_FiltWidth = 36, // 0x24
    p_FiltFreq = 37, // 0x25
    p_FiltFreqMod = 38, // 0x26
    p_FiltWidthMod = 39, // 0x27
    p_E1Velocity = 40, // 0x28
    p_E1SusFall = 41, // 0x29
    p_E1Attack = 42, // 0x2a
    p_E1Decay = 43, // 0x2b
    p_E1Sustain = 44, // 0x2c
    p_E1Release = 45, // 0x2d
    p_E1Legato = 46, // 0x2e
    p_E1Repeat = 47, // 0x2f
```

```
// Used to encode outgoing MIDI messages.
const unsigned short ProgramToMIDI[95] =
{
    0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, // 0
    0x8005, 0x0005, 0x0023, 0x800a, 0x8014, 0x8018, 0x800e, 0x000c, // 8
    0x8015, 0x8016, 0x8020, 0x8017, 0x801c, 0x801d, 0x0013, 0x801a, // 16
    0x801b, 0x801e, 0x801f, 0x0018, 0x0017, 0x001a, 0x001b, 0x8033, // 24
    0x0050, 0x802e, 0x8030, 0x0047, 0x004b, 0x001d, 0x8031, 0x8032, // 32
    0x8037, 0x8039, 0x0052, 0x0053, 0x0054, 0x0055, 0x8038, 0x801b, // 40
    0x803c, 0x803e, 0x0056, 0x0057, 0x0058, 0x0059, 0x803d, 0x801c, // 48
    0x8044, 0x8046, 0x8047, 0x8045, 0x8043, 0x804a, 0x804d, 0x804f, // 56
    0x8050, 0x804e, 0x804c, 0x8053, 0x8044, 0x005b, 0x8067, 0x8066, // 64
    0x005d, 0x806f, 0x8074, 0x8075, 0x8077, 0x8078, 0x8079, 0xffff, // 72
    0x0007, 0x807b, 0xffff, 0x8100, 0x8101, 0x8200, 0x8201, 0x8300, // 80
    0x8301, 0x8400, 0x8401, 0x8500, 0x8501, 0x8600, 0x8601 // 88
};

// Used to decode incoming MIDI messages.
// The first half of this table is CC; the second half is NRPN0.
// Other NRPNs have to be discovered using the forward look-up table.
const unsigned short MIDIToProgram[0x100] =
{
    0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0x0009, 0xffff, 0x0050, // 0
    0xffff, 0xffff, 0xffff, 0xffff, 0x000f, 0xffff, 0xffff, 0xffff, // 8
    0xffff, 0xffff, 0xffff, 0x0016, 0xffff, 0xffff, 0xffff, 0x001c, // 16
    0x001b, 0xffff, 0xffff, 0x001d, 0x001e, 0xffff, 0x0025, 0xffff, // 24
    0xffff, 0xffff, 0xffff, 0x000a, 0xffff, 0xffff, 0xffff, 0xffff, // 32
    0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, // 40
    0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, // 48
    0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, // 56
    0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0x0023, // 64
    0xffff, 0xffff, 0xffff, 0x0024, 0xffff, 0xffff, 0xffff, 0xffff, // 72
    0x0020, 0xffff, 0x002a, 0x002b, 0x002c, 0x002d, 0x0032, 0x0033, // 80
    0x0034, 0x0035, 0xffff, 0x0045, 0xffff, 0x0048, 0xffff, 0xffff, // 88
    0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, // 96
    0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, // 104
    0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, // 112
    0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, // 120
    0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0x0008, 0xffff, 0xffff, // 128
    0xffff, 0xffff, 0x000b, 0xffff, 0xffff, 0xffff, 0x000e, 0xffff, // 136
    0xffff, 0xffff, 0xffff, 0xffff, 0x000c, 0x0010, 0x0011, 0x0013, // 144
    0x000d, 0xffff, 0x0017, 0x002f, 0x0037, 0x0015, 0x0019, 0x001a, // 152
    0x0012, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, // 160
    0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0x0021, 0xffff, // 168
    0x0022, 0x0026, 0x0027, 0x001f, 0xffff, 0xffff, 0xffff, 0x0028, // 176
    0x002e, 0x0029, 0xffff, 0xffff, 0x0030, 0x0036, 0x0031, 0xffff, // 184
    0xffff, 0xffff, 0xffff, 0x003c, 0x0044, 0x003b, 0x0039, 0x003a, // 192
    0xffff, 0xffff, 0x003d, 0xffff, 0x0042, 0x003e, 0x0041, 0x003f, // 200
    0x0040, 0xffff, 0xffff, 0x0043, 0xffff, 0xffff, 0xffff, 0xffff, // 208
    0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, // 216
    0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0x0047, 0x0046, // 224
    0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0xffff, 0x0049, // 232
```

```
# ------------------------------------------------------------------------
# volume

table = [
    # adapted from ALPS datasheet
    # https://www.farnell.com/datasheets/317969.pdf
    [0,minf],
    [0.001,onef],
    [0.08,-60],
    [0.30,-22],
    [0.60,-12],
    [1,0]
]

s += c_autogenerated(False)
s += gen_table("Volume", 128, table)
s += "\n" + c_ul()
```

```
// This table was autogenerated by s_pot_maps.py
const u16 Volume[0x80] =
{
    0x0000, 0x0001, 0x0002, 0x0003, 0x0004, 0x0005, 0x0008, 0x000b,
    0x000f, 0x0016, 0x001f, 0x0025, 0x002c, 0x0033, 0x003c, 0x0046,
    0x0052, 0x0060, 0x0070, 0x0083, 0x0099, 0x00b3, 0x00d1, 0x00f5,
    0x011e, 0x014f, 0x0187, 0x01ca, 0x0217, 0x0272, 0x02dc, 0x0358,
    0x03e9, 0x0493, 0x055a, 0x0642, 0x0751, 0x088f, 0x0a02, 0x0a73,
    0x0ac5, 0x0b19, 0x0b70, 0x0bca, 0x0c27, 0x0c86, 0x0ce9, 0x0d4e,
    0x0db7, 0x0e22, 0x0e91, 0x0f04, 0x0f7a, 0x0ff3, 0x1070, 0x10f1,
    0x1177, 0x1200, 0x128d, 0x131f, 0x13b5, 0x1450, 0x14ef, 0x1594,
    0x163d, 0x16ec, 0x17a0, 0x1859, 0x1918, 0x19de, 0x1aa9, 0x1b7a,
    0x1c52, 0x1d30, 0x1e16, 0x1f02, 0x1ff5, 0x20dc, 0x21c4, 0x22b2,
    0x23a7, 0x24a3, 0x25a5, 0x26af, 0x27c0, 0x28d9, 0x29f9, 0x2b21,
    0x2c51, 0x2d8a, 0x2ecc, 0x3016, 0x3169, 0x32c6, 0x342c, 0x359d,
    0x3717, 0x389c, 0x3a2b, 0x3bc6, 0x3d6c, 0x3f1d, 0x40db, 0x42a4,
    0x447b, 0x465e, 0x484f, 0x4a4d, 0x4c59, 0x4e74, 0x509e, 0x52d7,
    0x551f, 0x5778, 0x59e1, 0x5c5c, 0x5ee8, 0x6185, 0x6436, 0x66f9,
    0x69d0, 0x6cbb, 0x6fba, 0x72ce, 0x75f9, 0x7939, 0x7c91, 0x8000
};
```

Tell your kids: autogenerate in a scripting language (Python/JS/whatever)

- Better than a write-only language (Excel/C preprocessor)
- See clearly from the output source code if you've been stupid
- Keep the write/compile/test cycle nice and quick (no manual component)
- Add helpful comments to the generator and/or source code
- Build more advanced stuff like reverse lookup tables

Everything is really a demo for your framework


C Source
analogue_controls.c
C Source
7.00 KB
arpeggiator.c
C Source
11.2 KB
blockfifo.c
C Source
2.13 KB
fifoq.c
C Source
2.60 KB
info_text.c
C Source
1.51 KB
info_text_editable.c
C Source
5.11 KB
keybed_scanner.c
C Source
4.73 KB
maths_pieces.c
C Source
949 bytes
midi_din_parser.c
C Source
4.59 KB
midi_flash_main.c
C Source
4.10 KB
midi_universal.c
C Source
9.25 KB
note_selector.c
C Source
12.7 KB
potentiometer_DEPRECATED.c
C Source
4.69 KB
string.c
C Source
5.45 KB
switch_debounce.c
C Source
4.88 KB
tea.c
C Source
1.60 KB
tempo.c
C Source
10.9 KB
tempo_ext.c
C Source
2.10 KB
usb_control.c
C Source
2.43 KB
usb_midi.c
C Source
4.93 KB

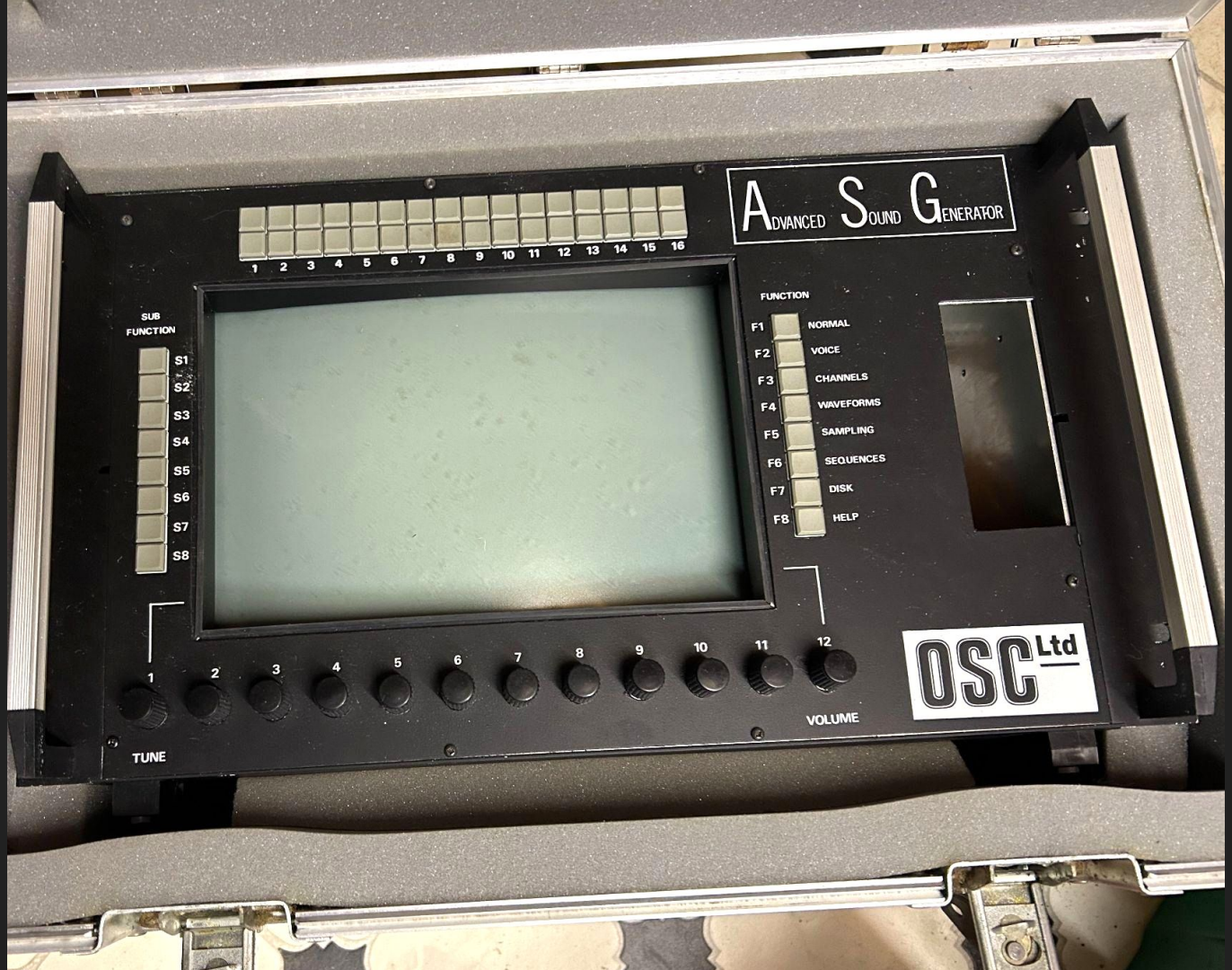
Inheriting Mantis from Chris Huggett

- Why? Chris, PWM, the beginnings of Mantis

- The signal flow
- Learning from the hardware
- Taking apart some firmware

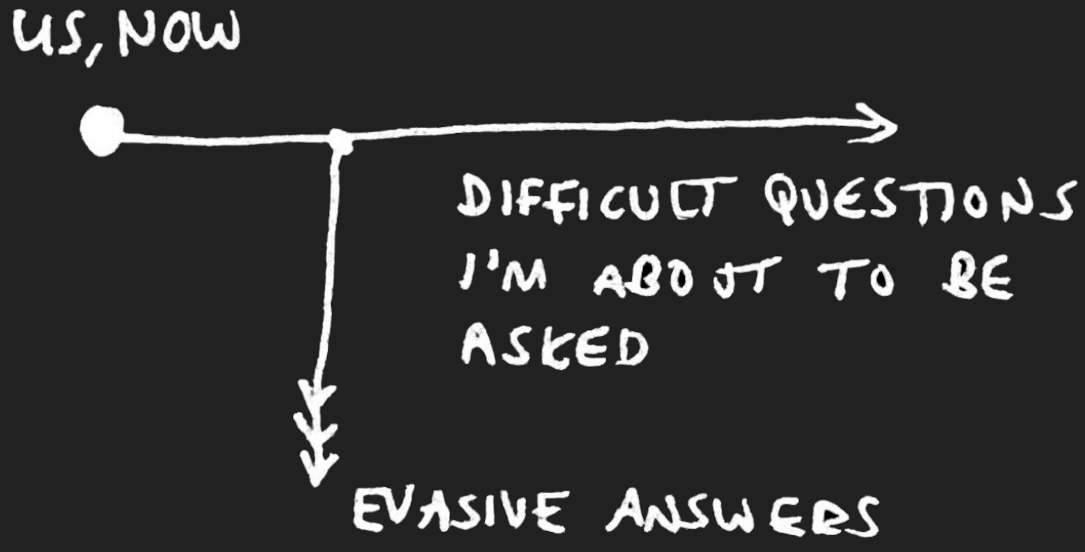
- **Where next?**


US,
NOW
WHAT BEST BUILDS OUR BRAND
WHAT WE DISCOVER BY TINKERING
WHAT LOOKS GOOD ON PAPER
WHAT OUR CUSTOMERS ASK FOR
WHERE OUR COMPETITORS ARE GOING


ADVANCED SOUND GENERATOR
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
SUB FUNCTION
S1
S2
S3
S4
S5
S6
S7
S8
FUNCTION
F1 NORMAL
F2 VOICE
F3 CHANNELS
F4 WAVEFORMS
F5 SAMPLING
F6 SEQUENCES
F7 DISK
F8 HELP
1 2 3 4 5 6 7 8 9 10 11 12
TUNE
VOLUME
OSC Ltd


OSC 1
OSC 2
WHEELS
MIX
FILTER
ENVELOPES AND TRIGGERING
FINE TUNE
DE TUNE
FILTER AMT
OSC BALANCE
FILTER DRIVE
VOLUME
PITCH MOD
INTRO
RATE
WAVEFORM
AMOUNT
SEPARATION
ATTACK
DECAY
SUSTAIN LEVEL
RELEASE
PW
PW
BEND AMT
PITCH AMT
GLIDE TIME
NOISE BAL
SET
SET
SET
FX
WAVEFORMS
behringer
POLY OSC
SHIFT
MENU
STORE
PROGRAMMER
DATA
FREQUENCY
TYPE
AMOUNT
ATTACK
DECAY
SUSTAIN LEVEL
RELEASE
ACTIVE
CLEAR
PATT
SAVE
TEMPO
SEQUENCER
OCTAVE
OCT −
OCT +
HOLD
CHORD
ARP
SEQ
BEND
MOD


LFO 1: MONO
LFO 2: POLY
SYNC
DEST
VOLUME
OVERDRIVE
PITCH
MOD
FREQ
AMOUNT
RATE
VINTAGE
PORTAMENTO
UNISON
OSCILLATORS
FREQUENCY 1
PULSE WIDTH
OSC MOD
FREQUENCY 2
X-MOD
RESON
FILTER
CUTOFF
OSC 2 DETUNE
OCTAVE
-2 -1 0 +1 +2
-2 oct
-1 oct
LOW SPLIT
HOLD
FACTORY
BANK
SYNC
OSC 1
SUB
104
TEO-5



ben@supperware.net