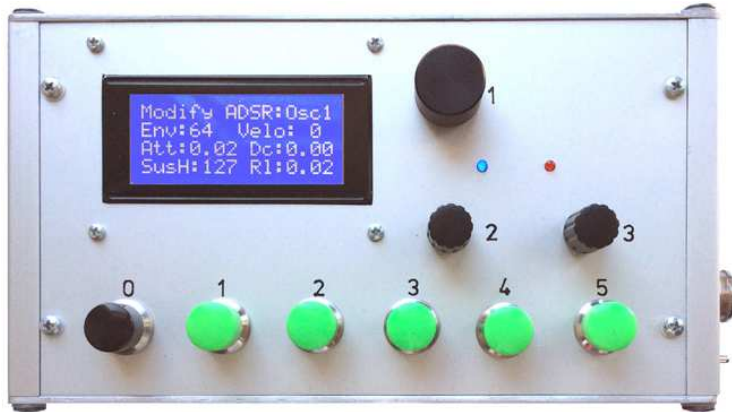


NucleoSynth Operating Manual

Status 11 Aug 2019

The NucleoSynth is a digital 10 voices polyphonic Audio Synthesizer built around the STM32 Nucleo-F446RE evaluation board. The synthesizer is controlled completely digitally.

Another alternative firmware version with a modified filter arrangement and a special firmware for minimal hardware is available. See Appendix of this manual for differing details of operation.



Digital sound data are transformed to an analog output signal using the internal DAC of the STM32F446 processor and finally amplified to headphone impedance by an audio amp. IC.

This has some advantages concerning the hardware complexity, but disadvantages too.

Due to the low operation voltage and 12 bit resolution of the DAC, the "headroom" of the sound signal is quite restricted, especially when played heavily polyphonic and filters have high resonance.

There are many hooks for volume setting and the user/instrument player is demanded to find her/his optimum balance between loudness/noise margin and possible distortion. With default volume setting (-6dB), an S/N ratio of ca.60 dB is achieved at the analog output. Up to 10 dB better if not connected with USB data (USB power bank driven e.g.).

Near the audio out connector, there is an array of pinheads (J1,J2 - see hardware schematic). Normally the 2 pins next to the PCB board edge must be jumpered (Ground to Sleeve). This is ok for output to a mono 6.35 mm plug. If a stereo TRS plug is inserted, the other 2 pins are jumpered, too. If only a jumper is placed onto the 2 middle pins and a stereo plug is used, both headphone transducers are connected serially and the phase of one is inverted. Funny idea I found somewhere in the internet, but I am not fully convinced. Just now it is there.

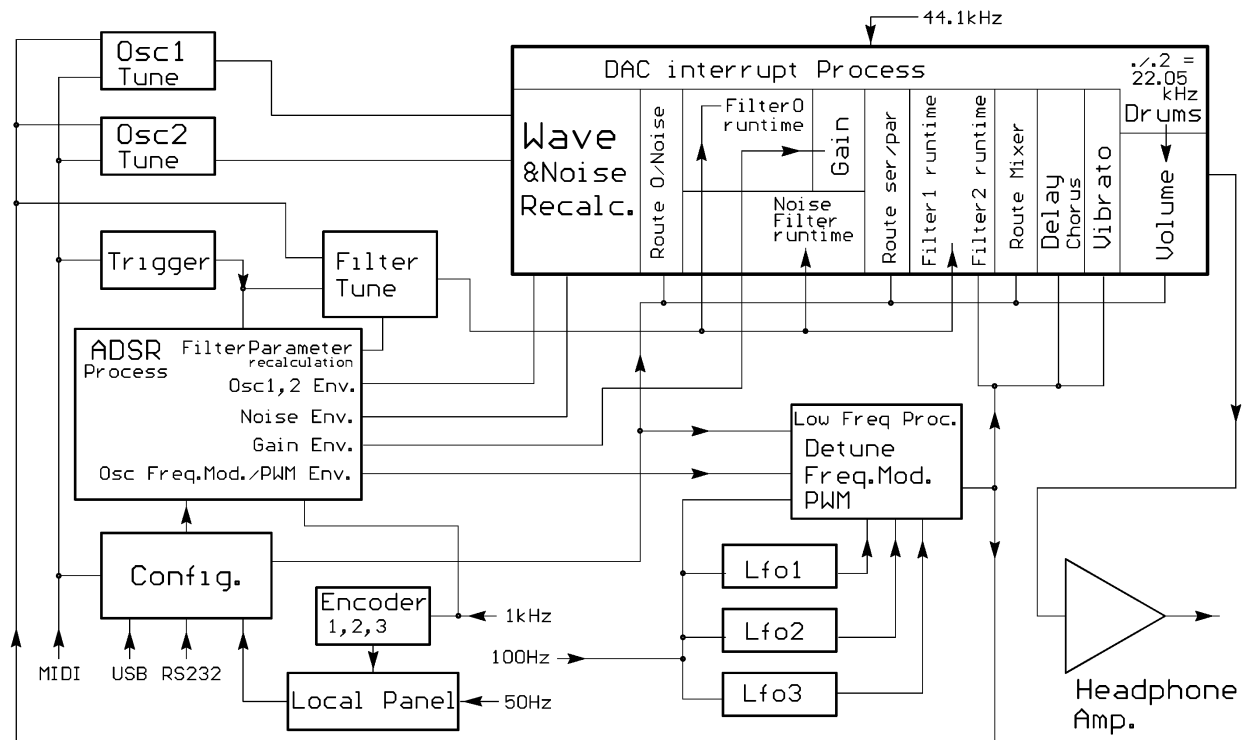
The **program flow** has to pass two critical bottle necks: one is the 44.1 kHz interrupt routine of the DAC (33.075 kHz alternative hardware), where up to 24 waves have to be read and their pointers updated simultaneously, noise is recalculated, the actual signal filtering of up to 4 filters has to be recalculated in realtime and the digital delay must be handled. The other bottle neck is the 1kHz main level routine for ADSR processing. Here, depending on dynamic frequency shift of any filter, essentially the recalculation of the actual digital filter parameters consumes CPU time.

As long as all tasks fit into the DAC interrupt time slot, the system is stable. If the DAC interrupt takes more than ca.80% of its time slot, the ADSR processing gets increasingly slowed down up to 2x, but it does not cause a crash or reset by watchdog. This has been tested carefully, but unexpected situations may exist.

Without use of an oscilloscope, distorted sound levels are recognized by a harsh sound. Extreme modulation of filters with high resonance setting (e.g. by high Keytrack or Square Wave Lfo) may cause audible sound clicks, which is a result of filter behaviour as such. Zero settings of Attack, Decay and Release or heavy Chorus may cause audible sound clicks, too, which is a physically necessary result of a sudden "jump" in the output volume.

Real musical instruments cannot play such harsh level changes (strings e.g) or are designed for it (percussion e.g). Please take this into account and adjust filters or ADSR times in case.

Block Diagram



Communication features

Communication for playing the instrument and for configuration is possible **via USB and USART**.

Different modes of communication are selected by a dual contact DIP switch:

- left switch OFF: USB works as USB CDC device (→virtual COM port)
- left switch ON: USB works as MIDI interface on the PC
- right switch OFF: configuration via RS232 and reduced possibilities to play the instrument (for test or by hardware multimedia controllers like Crestron or AMX).
- right switch ON: USART works as conventional MIDI interface (with selectable Baud Rate, n.b.)

A Local Operator Panel is provided. This is not absolutely necessary for tests or "embedded" solutions driven by other software, but will complete the NucleoSynth as a real user playable instrument.

Local Operation with Encoder, Pushbuttons("Key") and LCD Display

Generally **Encoder1** (top) controls the **first line** of the display, **Encoder2** (left lower) and **Encoder3** (right) control the **lower 3 lines** of the display.

In addition to the initial **"OFF" state (no LED on)**, there exists a **"SHIFT" state** (toggled with the hardwired pushbuttons of **Encoder1 and 2**, signalled with a **blue LED**) and an additional **"ALT" state** (toggled with the pushbutton of **Encoder3**, signalled with a **red LED**). LEDs and buttons share one line with multiplexed I/O pin states. So the corresponding LED is always on when the button is pushed, independent of the actual logic state.

Panic function: if the buttons of Encoder2 and Encoder3 are pressed simultaneously, all notes and ADSR envelopes are **switched off** immediately in any state of operation

During Led OFF state, Encoder1 controls the most left item of the first display line, during SHIFT state it controls the middle item, during ALT state it controls the most right item (if implemented).

During OFF state, Encoder2 controls the left item of the 2nd display line, Encoder3 controls the right item of the 2nd display line.

During SHIFT state, Encoder2 controls the left item of the 3rd display line, Encoder3 controls the right item of the 3rd display line.

During ALT state, Encoder2 controls the left item of the 4th display line, Encoder3 controls the right item of the 4th display line.

There are very few exceptions from this rule which are explained where relevant.

The **6 pushbuttons(keys) 0 ... 5** are primarily used to change sound sets quickly or to select the different "themes" of configuration.

BASIC

From other themes, this is selected with **Key0**.



```
B: 0- 3 Snd:0 P   CONFIG. Snd:0 P
Vol:-6 Vibra:0   Vol:-6 Vibra:0
Prta:0 Gain:0    Prta:0 Gain:0
Drm:All Lev: 0   Drm:All Lev: 0
```

After power on, NucleoSynth starts with the BASIC theme. This is preferably intended to "play" the instrument when no parameters have to be changed. In this special case, the keys 0 ... 3 are used to **switch quickly between 4 Sound Sets**, whereas key 4 and 5 toggle up and down banks of each 4 Sound Sets (60 SoundSets available, numbers 0 ... 59).

Alternatively, in SHIFT state, the active Sound Set is selected with Encoder1.

In ALT state Encoder1 selects Polyphonic (right display "P") or Monophonic "M" mode. **When Encoder1 is turned during OFF state, the first line of the display turns into "CONFIG.", "SYSTEM CONFIG." or "SOUND<->MEMORY"**. These sub-themes are described below.

Vol:

sets the final output volume between -46dB (i.e. output is off) and max. +6 dB. Keep in mind, that the output level has a limited "headroom". Especially when played heavily polyphonic or filters are highly resonant, it may be advised to use lower Volume. For this reason, -6dB is chosen as default volume.

Vibra: or VibL1: or VibL2:

sets the Vibrato depth between 0 and 100%. If Vibrato is off, "Vibra" is displayed. If Encoder3 is turned clockwise, "VibL2" is displayed. Then Vibrato is controlled by Lfo2. If Encoder3 turned **counter clockwise**, "VibL1" is displayed and Vibrato is controlled by Lfo1.

Prta:

controls the speed of gliding tone pitch (Portamento). This is mostly used only in Monophonic mode, Polyphonic may sound strange. Setting 0 means immediate change of pitch (default).

Gain

sets the amplification factor of the Gain stage between Filter0 and Filter1, no direct influence on filters. **With setting "0" the Gain stage is passed by, ie. amplification = 1**, but no extra effects (saves CPU time). With setting "1", and higher, the sound is amplified by the Gain factor, a **soft level clip** works behind the amplification, the sound is distorted softly towards a square wave and the **Gain ADSR** works behind the soft clip stage.

A more or less smooth transition may be achieved with a proper combination of the Oscillator(s) ADSR setting and the Gain factor. Because the clipped sound is always very loud, this is regulated with the Envelope and Sustain of the Gain Adsr, preferably with low Attack. **This configuration is sensitive** and takes care for good results! The effect sounds harsh and should not be overstressed. Could be used as "Heavy Metal" guitar simulation.

Filter0 prefilters the sound to be distorted. In many cases distortion works best with low bass and slightly enhanced treble. Filters1 and 2 shape the distorted sound. Noise bypasses the Gain stage in any case. Undistorted bass may be enhanced by appropriate route setting.

Drm and Lev

Encoder2 selects the drum type, Encoder3 sets the sound level of the selected drum.

Following drum types are implemented: Kick Drum, High Tom, Snare, Rimshot, Clap, HiHat, Cymbal and Cowbell.

Drums are triggered by a Note On message with the corresponding note value of the "General MIDI Drum Kit". In addition to the explicitly named drum types, these drums are triggered if MIDI messages for similar drum types are received.

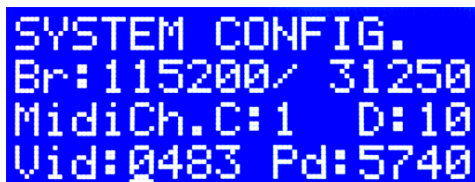
By default the MIDI channel for Drums is 10 - but can be changed to any. If the MIDI channel for chromatic notes is the same, music has priority. Drums will cause hanging notes then.

"All" modifies the level of the complete drum set, else the level of the selected drum type is modified. Drum level can be changed between -40dB (=Drum Off) and +12dB. It has to be noted, that the dB values are relative, because each drum level is modified by "All" and global output volume "Vol" too.

CONFIG.

same as above, **but the function of key 0...5 is changed: when pressed, now the associated "theme" is selected.** This is the preferred mode during sound setup and test.

SYSTEM CONFIG.



```
SYSTEM CONFIG.  
Br:115200/ 31250  
MidiCh.C:1 D:10  
Vid:0483 Pd:5740
```

OFF state:

Encoder2 selects the baud rate in RS-232 mode (active when Dipsw2 is OFF)

Encoder3 selects the baud rate in MIDI mode (active when Dipsw2 is OFF)

SHIFT state:

Encoder2 selects the MIDI channel for chromatic Note On/Off and setup commands

Encoder3 selects the MIDI channel for drums: If both are the same, drums have lower priority and drum trigger messages may result in hanging notes.

ALT state:

Encoder2 shifts the underline cursor to the position of the Vid/Pid display to be changed

Encoder3 changes the hex character at the selected position.

Please note: the default Vid/Pid is the one of the ST-LINK provided with the Nucleo64 board. In this context, its use is only allowed for test and setup. For any (public) use else, you are responsible for the rights of the Vid/Pid you are using !!

Be careful when changing Vid/Pid not to shoot your USB access. In many cases, different drivers have to be installed

SOUND<->MEMORY

```
SOUND <-> MEMORY
select with Enc2
FactSnd#Enc3: 00
confirm w. Key1
```

Select with Encoder2: **Check what you are doing before pressing Key1.**

Flash>AllSounds

AllSounds>Flash

All user configurable SoundSets are stored as well as the microcontroller SRAM as well as in the microcontroller Flash segment#7.

If loaded from new Flash (or after Flash segment#7 was erased) into SRAM, all SoundSet entries are interpreted as Default settings.

All user settings are stored in SRAM only before they are explicitly transferred into Flash by command. Else new settings are lost after reset or power OFF.

FactSnd#Enc3

Encoder3 selects which "FactoryPreset" shall be copied into the actual SoundSet. All previous settings of this SoundSet are overwritten then.

The "factory presets" are still under construction, but are documented for programmers at the bottom of this manual. Reprogramming with the free Embitz 1.1 or Atollic True Studio environment is easy even without deep knowledge of C-programming

DumpSnd#Enc3

Make a Dump is another possibility to manage more sounds than directly supported.

Encoder3 selects which SoundSet shall be dumped as MIDI SysEx message via USB or legacy serial MIDI OUT. "All" releases a dump sequence of all SoundSets.

Intentionally, the SysEx dump has no checksum. It is documented below (page20) and has the same data structure as the "Factory Presets" (except communication setup, which is not affected by "Factory Presets").

OSC

this theme is selected with **Key1**.

```
01:Sine    2:asTri  Osc1+2Detune:+9
Tran: 0    xT2:+12 Tran: 0    xT2:+12
Mod1:0     1M2:0    PWM1:12    3M2:0
Lf3: 0.0   xD2: 0    Lf3: 1.8   xD2: 0
```

In OFF state, **Encoder1** selects the **waveform of Osc1**, in SHIFT state it selects the **waveform of Osc2**. In ALT state the display of the first line changes and Encoder1 **detunes** the complete instrument up/down (linear scale, display 50 = ca.1 semitone).

The **first 5 elementary waveforms and waveform no.16** are stored in program flash and loaded to SRAM during waveform selection for slightly faster execution and possibilities to modify them mathematically.

The **square** waves are calculated during runtime to enable pulse width modulation.

The **User waveforms** can be uploaded as MIDI SysEx files (see below).

If not uploaded, they are provisionally substituted by elementary waveforms as dummies

The other waveforms are combined mathematically from the elementary waveforms while the waveform gets selected. Some of the waveforms have heavy overtone components, other are repeated as cyclic sequences of different waveforms, creating subtle components of sub-octaves.

The waveform types are shortened as follows (Present state, this feature is still under development.):

"Sine" = Sinewave	"User0" = Userwave no.0
"Trian" = Triangle	"User1" = Userwave no.2
"asTri" = Sawtooth with flattened shape	"User2" = Userwave no.2
"Saw" = Sawtooth	"User3" = Userwave no.3
"Trpez" = Trapez,square w. flattened shapes	"Si~2S" = sequence of sine + 2 waves sine freq*2
"Sq50%" = 50% symmetric square wave	"Si~Tp" = sequence of sine wave and trapez
"Sq33%" = 33% square wave	"E5S~S" = sequence of 1 Exp5S and 1 Sine wave
"Sq25%" = 25% square wave	"Us0~1" = sequence of Userwaves 0 and 1
"Sq12%" = 12.5% square wave	"Us0~2" = sequence of Userwaves 0 and 2
"T3T23" = tri halfwave freq*3+halfw.freq2/3	"Us0~3" = sequence of Userwaves 0 and 3
"SiSw2" = sinus + 1x sawtooth double freq	"Us1~2" = sequence of Userwaves 1 and 2
"Si+4S-" = sinus + sinus halfwaves freq*4	"Us1~3" = sequence of Userwaves 1 and 3
"Z+2S2" = silence + 2 half waves double freq	"Us2~3" = sequence of Userwaves 2 and 3
"Z+Si2" = silence +1 sinewave double freq	"UsAl~" = sequence of all 4 Userwaves
"Z+ST3" = silence +3x freq Saw+Square	"All~" = sequence of 4 elementary waves
"Strat" = simulation of Strat humbucker B string	"S~Mix" = sequence of silence, sine and double freq sine
"Exp5S" = 5th harmonic w. exponential decay	"Sw~3S" = sequence of 1sawtooth and 3 sine waves

Tran: and xT2:

Encoder 2 transposes the complete instrument by semitones as displayed. Encoder3 puts an extra transpose on Osc2.

Mod1: or PWM1:

For satisfactory results, MOD1 and MOD2 should have the same amount. Splitted setting due to PWM option. Frequency Modulation or Pulse Width Modulation of Osc1 by Lfo1. Turned clockwise, Frequency Modulation is selected ("Mod1"). Exclusively if the waveform of Osc1 is one of the square waves (50, 33,25 or 12.5%), PWM is selected when Encoder2 is turned **counter clockwise** ("PWM1"). After the waveform of Osc1 has changed from Square to another waveform or vice versa, the setting of Mod1 or PWM1 possibly has to be readjusted.

1M2 or 1P2: or 3M2 or 3P2:

Frequency Modulation or Pulse Width Modulation of Osc2 by Lfo1 or Lfo3. Turned clockwise, Frequency Modulation is selected ("1M2 or 3M2"). Exclusively if the waveform of Osc2 is one of the square waves (50, 30, 20 or 10%), PWM is selected when Encoder3 is turned **counter clockwise** ("1P2 or 3P2"). **If Lfo3 is OFF (freq.=0), the Lfo frequency and waveform is controlled by Lfo1("1P2 or 1M2"), else it is controlled by Lfo3("3M2 or 3P2").**

After the waveform of Osc2 has changed from Square to another waveform or vice versa, this setting possibly has to be readjusted.

Lf3:

adjusts the frequency of Lfo3. Lfo3 exclusively serves Frequency Modulation or PWM of Osc2. **If the frequency is not 0.0, Lfo3 replaces Lfo1 at Osc2.** Else Lfo1 does this job. Detailed setup of Lfo's 1, 2, 3 see Delay theme.

xD2:

extra detune of Osc2 in addition to the total detune.

ADSR

this theme is selected with **Key2**.

```
Modify ADSR:Osc1 ADSR:Free->FNois
Env:64 Velo: 0 Env:-28 Velo: 0
Att:0.02 Dc:0.00 Att:0.02 Dc:0.20
SusH:127 R1:0.02 Sus1:50 R1:1.00
```

Instead of selecting ADSR resources in a routing matrix, a bigger number of ADSR structures for selected features is implemented. **Each ADSR is triggered by a MIDI Note On message. The ADSR is switched off, if Env=0 and Velo=0.** This saves CPU runtime if the specific ADSR is not needed for sound shaping.

Care has to be taken, if an ADSR is activated with Attack > zero and Decay = zero and Sustain < 127. Though internal mechanisms for cross zero switching are implemented, this may cause a click, which is a natural result of this configuration. A similar effect may be experienced when Sustain is high and Release is = zero.

In **any** state, **Encoder1 selects the ADSR** which shall be edited. **If turned fully clockwise, the "Free ADSR" is selected.** It can be assigned to one effect at time, which is selected with **Encoder1 in Shift** state.

The ADSRs for Osc1 and Osc2 envelope are cloned for every played voice (up to 10).

All ADSRs else have only one instance for all voices.

For filter ADSRs special trigger features are available, see item **KeytrackMode** in the Route theme, Alt state.

The Noise envelope and Gain ADSRs should be used only when it makes a special sense.

ADSR "Fm12" sets the frequency modulation of Osc1&2 commonly.

"FmO2" triggers additional Osc2 frequency modulation. Fm12 and FmO2 work in addition to constant detune and modulation by Lfo.

Env: or **FMi:**

Env is the maximum level this ADSR can reach in subjective units.

When Oscillators **work in FM Synthesis mode** (see **options of R0N** in the Routing theme), **for Osc2 the modulation index FMi is selected and displayed** (subjective scale - not identical with "academic" FM index.) Oscillator ADSR settings have great influence on FM sound. Only simple Osc1 sound if FMi = 0 or ADSR is totally faded down !!

For filter ADSR's: the max frequency shift can be set to about +1 or - 1/2 octave.

Velo:

adds an extra positive or smaller negative maximum level to the envelope according to the velocity the triggering note is played with. Scale is subjective.

Att: , Dc: RI:

configures Attack, Decay and Release in the appropriate state (SHIFT or ALT) and appropriate Encoder2 or 3. The transition time is displayed in seconds and parts of seconds. For times up to one second, the raster is 20 milliseconds. Above the raster is 50 milliseconds, providing a max effect time of 4.85 seconds (based on the internal range 0..127).

SusH or **Sus1:**

Sustain level as part of the sum of Env and Velo. The scaling is subjective, 127 means 100%. If Sus is **followed by H**, the the envelope is of **"hold" type**, which means that the sound is held at Sustain level as long as the key is pressed. When Encoder3 is turned **counter clockwise**, the "H" turns into "1". Then the envelope is a **"one shoot" type**. This means that the level decays to Sustain level and then immediately is faded to zero during the Release phase, even if the key is pressed longer. Key release starts the Release phase immediately.

FILTER

this theme is selected with **Key3**.

Filter handling has substantially changed with version 1.5

Deviating from the general rule, **by default this theme starts with ALT state** (red Led ON).

3 Band Filter	Route F0N12 ser	Route F011 F2mix
F12:Hi 0 Lo 0	>F0:+16 >out:-16	>F0:+20 >F2:-20
F0Freq: 932 Typ7	F0Freq:1046 Typ5	F0Freq:2959 Typ3
MidPeak:+1.50	Bandwidth: 523	Resonance:0.50

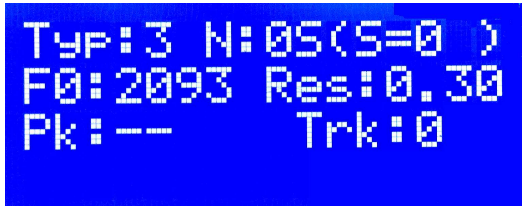
Three different arrangements of the filter cascade are provided. The first display line shows the actual routing arrangement of filters1 and 2. (Details of the filter arrangement are managed in the ROUTE theme, see page11).

- in each arrangement **selected fractions of Osc1 and Osc2 are fed to filter0**, this output is processed by the Gain stage and next **mixed with the actual level of Noise** (if selected for routing). This selection is managed with item R0N in the Route theme (p11). Further handling depends on the active filter arrangement:
- **3 Band filter: The R0N output is fed into Filters 1 and 2, which are always configured parallel.** By default filter1 is a 2nd order Highpass and filter2 is a 2nd order Lowpass. **Their relative strength is balanced** or weighted with Encoder1 (display line 2). The combined result is a simple bass/treble tone control, known from traditional guitar stomp boxes as "Big Muff" filter. When the Balance control is in neutral position, there is a slight notch around 1 kHz due to the different phase shift of high and low pass - sounds "thin". This is compensated by **filter 0**, which is configured by default as "parametric" type: the center frequency is controlled with Encoder2 and the peak is controlled with Encoder3. The overall result is a 3 band tone control similar to traditional guitar tube amps like Marshall or Fender. To get a fat sound at middle frequencies, raise the Peak of filter0.
- **straight serial: all filters 0(+Gain), 3(Noise),1,2 are cascaded serially.** The R0N output is fed to the serial chain of filter1 and 2. Selected fractions of the unfiltered signals Osc1, Osc2 and Noise (see item R12 in the Route theme) are mixed with the output of the filter cascade. **This mixture is balanced** or weighted with Encoder1 (display line 2)
- **mixed serial/parallel:** Filter0, Gain, Noise, filter1 are cascaded serially and **Filter 2 is completely arranged parallel with this cascade.** Selected fractions of Osc1, Osc2 and Noise (see item R2 in the Route theme, p.12) are **fed into filter2** (which is passed parallel with the cascade of filter0, Gain,Noise and filter1). Finally both outputs are added. **The relationship is balanced** with Encoder1.
- If Filter2 is OFF, the mixed arrangement is identical with straight serial.
- In both configurations, by default filter0 is configured as second order Lowpass with moderate Resonance. Filters1 and 2 are OFF by default. The border frequency F0 of filter0 is set with Encoder2, the resonance (or bandwidth if filter0 is configured correspondingly) is set with Encoder3. This may be changed here or in the Route theme to boost or damp middle/treble frequencies. At the end of line3 the actual type of filter0 is shown. Parallel or serial arrangement of filters modifies their overall characteristic significantly. For this reason, each SoundSet stores **three independent parameter sets for the 4 filters**. To provide an easy change between filter configurations, three independent instances for the **Balance** (modified with Encoder1 and shown in line 2 of the display) are provided and stored too. Only one set is actually active - depending on the selected filter routing arrangement. But if the user switches between serial/parallel routing, each preconfigured characteristic follows.

F0Freq: because Filter0 is the most relevant, it's Resonance or Border frequency can be changed here directly. The actual filter type is shown right hand (for information only)
Reso: or **MidPeak:** (depending on the actual filter type) can be changed here directly.

Different filter behaviour can be configured individually. In detail, **any filter shape can be rearranged arbitrarily in OFF or SHIFT state:**

If any encoder pushbutton is pressed during ALT state, the blue LED gets ON and **the SHIFT state is selected. Now the individual setup of all filter parameters is available.**



```
Typ:3 N:05(S=0 )
F0:2093 Res:0.30
Pk:--      Trk:0
```

Starting with SHIFT state instead OFF state implies: **Encoder1 first will select the filter Number (N:)**. So, in OFF state you can select the most essential filter parameters without state change. The middle position of line1 shows the selected filter number followed by "S", "M" or "P", which signals the actual arrangement of filters1and 2. The display of the active Sound Set at right position in line1 is for information only and cannot be changed here.

Filter shapes can be tested independent of keyboard input: see ROUTE,ALT state (p.12)

Typ:

0= filter OFF, does not modify the sound, saves CPU time.

1= **first order (6dB) Lowpass.** Only F0 can be changed, filter has no parameters else.

2= **first order (6dB) Highpass.** Only F0 can be changed, filter has no parameters else.

3= **2nd order (12dB) Lowpass.** "Res" sets the Resonance (a little bit unconventionally) in "zeta" = $1/2Q$ units. I.e. **lower values increase the Resonance peak and sharpness!** Self-oscillation of filters is not supported. The default value is 0.71. This describes a "Bessel" filter, which is regarded as the optimum stopband transition without peak. Higher values make the transition between passband and stopband more smooth, but provide 12dB/octave far away from F0. Less recommendable in most situations. Peak is not available here.

4= **2nd order (12dB) Highpass.** Resonance and Peak same as above.

5= **Bandpass.** Encoder3 sets the bandwidth. With Peak the resonance amplitude can be increased by ca.0 .. 15dB.

6= **Notch Filter.** Encoder3 sets the bandwidth. No Peak parameter.

7= **Parametric Filter.** Encoder3 sets the bandwidth. Peak sets the filter characteristic: with negative values it works as adjustable notch filter. With positive values it works similarly like a bandpass – but the level outside the bandwidth region is not suppressed to zero, instead the filter input level is kept. So, the peak is added to the input level.

8,9 = **LowShelf and High Shelf Filter.** These filter types are combined Highpass and Lowpass filter with Resonance setting. Their Highpass/Lowpass characteristic is adjustable with Peak in a similar way like the Parametric Filter. The Low Shelf filter modifies levels below F0, the High Shelf Filter modifies levels above F0.

F0: , Res: or Bwi:

Setting of **Resonance/Border frequency** respectively **Resonance or Bandwidth.** Internally, the corresponding frequency is stored as MIDI note value. For this reason exclusively specific

frequencies (which conform with semitones) are selectable. The F0 setting is limited to about 8kHz, the Bandwidth is limited to about 5kHz

Pk:

Peak setting, different meaning for different filter types. See above.

Trk:

Keytrack setting. The resonance of the selected filter follows the note value of the triggering key by the Keytrack value (approx. in percent). If F0=440Hz and Keytrack=100%, F0 follows exactly the played note.

Because due to CPU limitations there is only one filter set instance (incl Filter ADSRs) for all voices, an inherent problem arises: the total sound changes with the note value of the triggering key. So a **compromise with keytrack and sharp filter resonance has to be made** when played polyphonic. On the other hand, with special playing techniques the sound is affected considerably. See item **KeytrackMode** in the ROUTE theme, ALT state, too.

To workaround this problem, an alternative firmware was developed (see page 23).

DELAY

this theme is selected with **Key4**.

```
Del: 30.0 Cho:2
Lev:+32 Fb: 0
LDry:68 Lfo#:2
Freq: 1.0 Wv:T50
```

Del:

sets **DelayTime** in 1/10ms units up to 199.9ms (alternative firmware:279.9ms). In OFF state Encoder1 changes the time coarsely in 10ms steps, in SHIFT state in 0.1ms steps. This way, rather elementary comb filter effects are possible.

Cho: (may be assigned to "ADSR Free")

Chorus modulates the DelayTime in a subjective scale driven by Lfo2 or by the "free" ADSR. With "Cho:0" the Delay Time is not modulated.

Because the Chorus effect modulates the length of the delay buffer, the delayed signal has an abrupt phase change between consecutive samples. Beyond a certain level, this may cause an audible click, depending on several factors: Chorus width, Lfo frequency, delay level, audio signal frequency. For this reason, the Chorus effect should be used carefully.

Lev: (may be assigned to "ADSR Free")

Sets the **level of the delayed signal** when it is mixed with the undelayed signal (as set by LDry) in %. The actually played signal is fed together with the scaled Feedback into the digital delay chain (long circular buffer clocked with 44.1 kHz, alternative firmware clocked with 33.075 kHz).

When Encoder2 is turned **counter clockwise**, the displayed level gets a negative sign. In this case, the delayed signal (output of the delay chain) is mixed with **180 degree inverted phase**, but the amount is the same in %.

Fb: (may be assigned to "ADSR Free")

Sets the portion of the delayed signal (in %, before the delayed signal is scaled by the setting of "Lev"), which is **fed back into the delay** chain together with the actually played signal. But the signal read at the end of the delay chain (delay plus feedback) is controlled by "Lev".

When Encoder3 is turned **counter clockwise**, the displayed level gets a negative sign. Then the feedback is inserted with **180 degree inverted phase**, but the amount is the same in %.

LDry: (only relevant, if ((Del && Lev) != 0), else Dry Level always 100%. May be assigned to "ADSR Free")
Portion of the undelayed signal which is mixed with the delayed signal (as set by Lev and Fb). This separate setting is useful for fine tuning of delay effects.

Lfo#:

Selects LFO 1, 2, 3 for modification by items Freq and Wv.

Lfo1 is assigned to the frequency modulation of Osc1(permanently) and Osc2 (may be changed to Lfo3). Furthermore Lfo1 can be selected as **Vibrato frequency** and **Filter2 F0 Modulation** frequency

Lfo2 is permanently assigned to the Delay Time modulation (Chorus), but can be selected as **Vibrato frequency** and **Filter2 F0 Modulation** frequency, too

Exclusively if the frequency of Lfo3 is not = 0 it is assigned to **Oscillator 2 frequency modulation**. Else Osc2 is modulated by Lfo1. For this reason, the frequency of Lfo3 additionally can be changed in the Osc theme. This is the only function of Lfo3.

Freq:

Frequency adjustment of the addressed Lfo up to 12.7 Hz in 0.1 Hz raster.

Setting "0" turns the addressed Lfo OFF (saves CPU runtime).

Wv:

modifies the waveform of the actually selected Lfo. If **Encoder3** is turned **clockwise**, **"T"** is displayed and the Lfo generates a **triangle/sawtooth wave**. The slope can be set between soft Sawtooth (10% =starting with fast rising edge), symmetric Triangle (50% setting) and soft Sawtooth (90% =starting with slowly rising edge)

If **Encoder3** is turned **counter clockwise**, **"S"** is displayed and the Lfo generates a **square wave**. The pulse width can be set between 10% and 90%. To improve the audible effect, the edges are provided with softened slopes

ROUTE

this theme is selected with **Key5**.

In OFF and SHIFT state, following parameters are modified:

3 Band Filter	Route F0N12 ser	Route F01 F2mix
F12:Hi 0 Lo 0	>F0:+20 >out:-20	>F0:+2 >F2:-2
R0N:12N ->F12par	R0N:12 Rout:12	R0N1:FM R2:FM
7F0: 932 P:+1.50	3F0:2959 R:0.50	3F0: 830 R:0.25

Encoder1 controls the mixture relationship as described below

The most essential routing features are selected with Encoders2 and 3 in OFF state:

R0N or R0N1: (Encoder2, OFF state)

The display shows, which signal (Osc1, Osc2, or both) is fed into Filter0 followed by the Gain stage. Noise is added to the output of the Gain stage. So the R0N setting is a **combination of the following sound fractions: 1=Osc1, 2=Osc2, N = Noise** (the corresponding ADSR must be ON!). **Selection "FM" starts a simple FM sound** of Osc1 modulated by OSC2 instead of linear addition of both oscillator waves. This reduces the options which are selectable: FM (Osc1 FM modulated), FMN (+ Noise), N@M is noise exclusively. Latter case is important, because it implies the FM sound organisation of oscillators to be used as "unfiltered signal".

R12 or R2: (Encoder3, OFF state)

The first letter shows the filter arrangement: "**par**" selects the 3 Band filter arrangement, which is described more detailed in the Filter theme (p.8). Encoder1 balances the bass/treble tone. The first letter "**s**" means straight serial mode, where the unfiltered signal is added after the filters, "**m**" means a mixed serial/parallel cascade: The selection of R0N is fed into the cascade of filter0,Noise,filter1 and the selection of F2 is fed into filter 2. Outputs of filters 1 and 2 are added finally. Encoder1 balances the mixture. The following combination of letters has the same meaning as described above at F0N. "par" has no further options.


xF0: (Encoder2, SHIFT state)

sets the **resonance frequency or border frequency of filter0**, depending on filter type. The leftmost number describes the actual filter type (for information only).

R,B or P: (Encoder3, SHIFT state)

sets **Resonance or Bandwidth** of filter0 (serial routing) or **Peak** of filter0 (parallel routing) More individual filter settings are made in the filter theme.

In ALT state some special parameters are adjusted, which do not fit into the other themes:



```
etc. Features:  
F2Mod by Lfo1:--  
KeytrackMode:1st  
FilterTest: OFF
```

F2Mod by Lfo:

sets the amount of **F0 modulation of Filter2 by Lfo** in addition to the modulation by ADSR. If Encoder3 is turned clockwise, L2 is displayed and the modulation is controlled by Lfo2. If Encoder3 is **turned counter clockwise**, L1 is displayed, modulation is controlled by Lfo1.

KeytrackMode: (not available with alternative firmware, blank line is shown then)

Selects the order of keys to trigger Keytrack and Filter ADSRs. (Alternative Firmware triggers always on "new" key)

1st: trigger and Keytrack only on the first hit key (legato trigger mode)

All settings else trigger timed on the latest hit key, but:

New: Keytrack to the note value of each hit key (i.e.note value of the latest one),

Low: Keytrack to the lowest note value of all actually hit keys,

Avr: Keytrack to the average note value of all actually hit keys,

Hi: Keytrack to the highest note value of all actually hit keys

FilterTest:

Usually the FilterTest is OFF. If **Encoder3 is turned clockwise, FilterTest is activated** with adjustable (relative) level, starting from -22dB up to 0dB. With level 0 dB,global volume -6 dB and moderate filter peak/resonance, the maximum noise pulses just start being clipped by the DAC. For highly resonant filters, FilterTest should be set to a lower level.

During FilterTest, **the input to filters0 and 3(noise) (and additionally routed unfiltered signal) is taken from the noise generator** with constant level (no envelope ADSR) instead of from the keyboard. Frequency shift of filters caused by Lfo or keyboard input is monitored however.

The filter shape can be viewed and adjusted with a spectrum analyzer. (I am using "DL4YHF Spectrum Lab" (free download, see Google) with "Scarlett 2i2" audio interface)

The **Noise signal is rather LOUD and awful !!** Take the headphone off or turn the speaker down!

Sequencer, Arpeggiator

are not implemented and will not be because I feel that is impossible to implement a good tool in this restricted environment.

As sequencer, I use my "Rhythmic" software <www.midi-and-more.de/rhythmic.htm> on a small "Netbook".

Short Reference of ASCII commands

Letters can be entered upper or lower case. 'i' and 'o' are written lower case here for better readability

- !** (exclamation mark) **Panic OFF**, everything audible is terminated immediately
- ? or H** Lists all parameters of the actually loaded/modified Sound Set
- S** select and activate **Sound Set** (0...59)
- &** switch:0=polyphonic 10 voices (default) 1=monophonic
- V** set global volume (final output stage) -46dB (= switched Off) ... +6dB). Default = -6dB.
- W** **Waveform Osc1** (0=Sinewave,1=Triangle,2=asym.Tri,3=Saw,4=Trapez,5=Sq50%,6=Sq33% ,7=Sq25%, 8=Sq12%, and more are displayed. Detailed description see page 6)
- X** **Waveform Osc2** (0=Sinewave,1=Triangle,2=asym.Tri,3=Saw,4=Trapez,5=Sq50%,6=Sq33%,7=Sq25%,8=Sq12%, and more are displayed. Detailed description see page 6))
- T** **Transpose of Osc1 and Osc2** commonly (-36...+36 in semitone steps)
- U** additional **Transpose of Osc2** (-36.. +36 in semitone steps)
- Y** **global-Detune** (of Osc1 and 2) (-64...+63 ca. linear)
- Z** additional **Detune of Osc2** (-64...+63 ca. linear)
- P** set **Glide(Portamento)** speed of Osc1 und Osc2 commonly. 0=OFF
- L** init or readjust **Lfo1,2,3**. Max frequency 12.7 Hz in 0.1 Hz steps.
Enter frequency with decimal dot. Waveform is asked too.
- i** **Frequency Modulation of Osc1** or PWM (square wave only) by Lfo1
(0..127 ca.+100/-50% symmetric linear frequency shift)
- J** **Frequency Modulation of Osc2** or PWM (square wave only) by Lfo1
(or automatically by **Lf03** if **Lfo3 frequency not = 0**)
(0..127 ca.+100/-50% symmetric linear frequency shift)
- A** configure **ADSR** (Osc1, Osc2, Noise, Gain, FModOsc1, FModOsc2, Filters 0,1,2,Free)
- =** assignment to "**Free**" **ADSR** (Noise Filt.,Lfo1,2,Vibrato,Chorus,Balance,Delay,Dry,Del<->Dry,Feedb)
- G** set **Gain** (0 ... 50) **0 means amplification = 1** - but ADSR and clip/limiter is **not** active
Gain amplification works always after filter0 but ahead of the "soft clip" limiter. Finally the sound is shaped by the Gain ADSR (% of limiter output)
- F** **(re)configure Filter** in the active Sound Set. Filter3=Noise
- {** **Filter Test** : 1-22 = Filter Test is active (confirm w. upper case Y),
0 = back to standard filter operation
In Filter Test mode, the noise signal is fed to filter 0 and filter3(noise). Gain is always off in this case. The input level of the noise is regulated with the parameter: 1 means about max. undistorted level, 22 means ca.22dB below max. level. To test filters with high resonance, it is recommended to use lower level. Level is independent of any key pressed, but the filter modulation reacts on keys. Depending on routing, fractions of the unfiltered noise signal are added before or after filters1&2. The total spectrum is available at OUT. **(Be careful, the output is LOUD and ugly !!)**
- K** set filter **Keytrack** (0..127) in % relative to A3(440Hz)
- M** **Frequency Modulation of filter2** by Lfo(max ca.+/-1 octave).
Asks too if controlled by Lfo1 or2
- +** **Keytrack/Freq.Shift trigger priority** of filter0,1,2 (Alternative Firmware triggers on "new" key)

- (0=first key, 1=new, 2= lowest, 3=average,4= highest). See page 12 for more details.
- R** configure **Filter Routing**
asks first routing to filter0,
next filter arrangement "s,m,p"=serial, mixed or parallel routing. Details see page12.
 For Osc1 enter "1", for Osc2 enter "2". Instead of "N" enter "3"; instead of "Frequency Modulation" enter "4" in rising order. Specially enter "**5**" **at R0G1** to send exclusively Noise to Filter3, but this changes the listing for FM at R2 (e.g. "123" is Osc1+Osc2+Noise, or "34" is FM+Noise)
- %** **Balance** or **Weight** between different signal paths, depending on configuration:
parallel: 20=only filter1; 0=equivalent ; -20=only filter2
straight serial: 20=only filtered signal, 0=equivalent; -20=only unfiltered
mixed serial: 20=only signal input into filters0 and 3(noise); 0=equivalent ;
 -20=only signal input into filter2.
- D** set **Delay Time** (0-199.9 ms in steps of 0,1 milliseconds). Alternative firmware 0-279.9 ms.
 Enter tenths with dot.
- E** **Delay Level** (127 ... -127, negative values = 180 degree phase inversion)
- o** **DryLevel** (0-127) Level of undelayed signal in contrast to Delay. **Always 100% if Delay=0**
- N** **Delay Feedback** Level (127 ... -127, negative values = 180 degree phase inversion.)
- C** configure **Chorus** depth (0..99, about max. delay shift in ms). 0 means Chorus off)
- Q** **Vibrato** Depth (0...100%). Asks too if controlled by Lfo1 or 2
- \$** **Drum Volume**:
 A = all drums equally in addition to individual setting of drum type
 0 .. 7 = drum type as listed below
- 0..7** **Trigger Drum**: (for test or by special hard/software with serial ASCII interface)
 0=Kick, 1=HiTom, 2=Snare, 3=Rim, 4=Clap, 5=HiHat, 6=Cymbal, 7=Cowbell
- 8** **simulate Note Off**. (for test or by special hard/software with serial ASCII interface)
 followed by the MIDI note value, entered as hex number (2 places) in ASCII text
- 9** **simulate Note ON**. (for test or by special hard/software with serial ASCII interface)
 followed by the MIDI note value, entered as hex number (2 places) in ASCII text.
 Next the velocity is entered is entered as hex number (2 places) in ASCII text
 Input is echoed. E.g. enter text sequence 9450F to send note "A"(440Hz) with velocity=15
- /** send **MIDI messages** after local parameter change: 1= ON, 0= OFF
- *** **Dump Sound Set** (#0 ... 59, A = all) as MIDI SysEx, confirm with upper case Y
firmware for minimal hardware: enter without parameters, always dump of all Sound Sets !!
- [** change **USB Vid/Pid** (driver on PC must be changed too!)
- #** select **MIDI channel**:
first for chromatic instruments Note, PgnCh, CtrlCh, **next** for drums (Note On)
- B** select **Baudrate**, **first** for RS232(used when Dipsw2 OFF), **next** for MIDI(Dipsw2 ON)
- ** **load** all Sound Sets from Flash into SRAM (new Flash->default), confirm w. upper case Y
- >n** **copy** actual Soundset into Soundset no.n , confirm with upper case Y
- @n** load "Factory Preset" n into the actual Soundset (into SRAM – not in Flash), n=0..19,
 confirm with upper case Y
- |** (vertical line) **store** all actual Sound Sets in Flash, confirm with upper case Y
- ~** (tilde) **delete all user data in Flash** = "Factory Reset", confirm w. upper case Y, next with – (minus)
Undo is impossible! , so handle with care. Make a Dump before.

Protocol of MIDI Messages

Exclusively Status Bytes with the selected "chromatic" or "drums" MIDI channel are evaluated.

A USART based MIDI-Scanner for a "legacy" MIDI IN and a separate USB based-MIDI-Scanner is implemented. **Both merge MIDI channel messages correctly**, so in fact two independent MIDI inputs are available, which can be played simultaneously. The first completely received Note, Pgm Change or CtrlChange is executed first. SysEx messages (dump or userwaves) are not merged correctly. Other MIDI messages are not supported Note Values below decimal 18(ca. 23 Hz) and above 108(ca. 4200Hz) are ignored. However, Control Change messages for filter configuration accept Note Values up to 121(ca.8300 Hz).

Transmitted MIDI-Control Change messages as response on local parameter changes have the same format as the corresponding MIDI Control Change message which changes the same parameter.

Program Change messages

Parameters are data bytes sent. Many controllers/software ask to enter 1+ to send this byte value !!

program no (0...127 !!).	Effect
0 ... 59 (0 ... 0x3B)	Select Sound Set 0 .. 59
80 .. 82 (0x50 ... 0x53)	select Filter Number 0, 1, 2, 3
84 .. 94 (0x54 ... 0x5E)	select assignment of the "ADSR Free" feature
96 .. 107 (0x60 ... 0x6B)	set Filter Type 0 ..9 + special modes 10 and 11
explicitly:	
96 (0x60)	delete, neutralize filter
97 (0x61)	1st order (6dB) Low Pass
98 (0x62)	1st order (6dB) High Pass
99 (0x63)	2nd order (12dB) Low Pass
100 (0x64)	2nd order (12dB) High Pass
101 (0x65)	Bandpass
102 (0x66)	Notch
103 (0x67)	Parametric Bandpass/Notch
104 (0x68)	2nd order Low Shelf
105 (0x659)	2nd order High Shelf
112 (0x70)	10 voices Polyphonic Mode (default)
113 (0x71)	Monophonic Mode
114 (0x72)	MIDI messages OFF (default)
115 (0x73)	MIDI Messages ON (sent after local parameter change)
126 (0x7E)	requestSoundDump (of the active SoundSet)
127 (0x7F)	request Dump All Soundsets

Control Change messages:

Ctrl number	Ctrl value	Effect
0	0 ... 8	FilterRouting cascade Filter0,Gain, Filter3(Noise)
1	0 ... 20	FilterRouting Filter1&2
2	0 ... 50	Gain (additionally shaped by Gain ADSR)
3	0 ... 40	Balance of routing (20= neutral)
4	0 ... 99	set Vibrato depth controlled by Lfo1 (%)
5	0 ... 99	set Vibrato depth controlled by Lfo2 (%)
6	0 ... 127	Dry level (exclusively to be mixed with Delay level)
7	0 ... 52	Global Out Volume (0 = -46dB, 40= -6dB(default), 52 = +6dB)
8	0 ... 34	Osc1 Waveform (0=sin, 1= triangle, 2=asymmetric triangle, 3=saw, 4=trapez 5=50% square, 6=33% square, 7=25% square, 8=12.5% square, ... etc)
9	0 ... 72	Osc1+2 Transpose, max 3 octaves (36=neutral, internal scale -36 ..+36)
10 (0x0A)	0 ... 127	Osc1+2 global Detune (64=neutral, internal scale -64 .. +63)
11 (0x0B)	0 ... 30	Osc2 Waveform (0=sin, 1= triangle, 2=asymmetric triangle, 3=saw, 4=trapez 5=50% square, 6=33% square, 7=25% square, 8=12.5% square, ... etc)
12 (0x0C)	0 ... 72	Osc2 add Transpose, max 3 octaves (36=neutral, internal scale -36 .. +36)
13 (0x0D)	0 ... 127	Osc2 add Detune (64=neutral, internal scale -64 .. +63)
14 (0x0E)	0 ... 127	Glide(Portamento)
15 (0x0F)	5 ... 104	Lfo1 Waveform (Bit6=0:Square,=1:Triangle/Sawtooth bits 0..5=pulse width/slope) Ctrl value2 format: (shape or pulse width)/2 + 64 for Triangle/Sawtooth ie. 5...45 for Square(25=50%) or 69...109 for Triangle/Saw (89=symm Triangle)
16 (0x10)	0 ... 127	Lfo1 Freq (scale = 1/10 Hz)
17 (0x11)	5 ... 104	Lfo2 Waveform (Bit6=0:Square,=1:Triangle/Sawtooth bits 0..5=pulse width/slope) Ctrl value2 format: (shape or pulse width)/2 + 64 for Triangle/Sawtooth ie. 5...45 for Square(25=50%) or 69...109 for Triangle/Saw (89=symm Triangle)
18 (0x12)	0 ... 127	Lfo2 Freq (scale = 1/10 Hz)
19 (0x13)	5 ... 104	Lfo3 Waveform (Bit6=0:Square,=1:Triangle/Sawtooth bits 0..5=pulse width/slope) Ctrl value2 format: (shape or pulse width)/2 + 64 for Triangle/Sawtooth ie. 5...45 for Square(25=50%) or 69...109 for Triangle/Saw (89=symm Triangle)
20 (0x14)	0 ... 127	Lfo3 Freq (scale = 1/10 Hz)
21 (0x15)	0 ... 127	Osc1 Frequency Modulation byLfo1 (0=Off...127)
22 (0x16)	0 ... 127	Osc1 PWM byLfo1 (0=Off...127) (works only on square wave, else FM is selected)
23 (0x17)	0 ... 127	Osc2 Frequency Modulation byLfo1 (0=Off...127) Osc2 Freq.Mod and PWM is always controlled by Lfo3 if Lf3 freq not = 0!
24 (0x18)	0 ... 127	Osc2 PWM byLfo1 (0=Off...127) (works only on square wave, else FM is selected)
25 (0x19)	0 ... 4	Filter Keytrack & ADSR Trigger mode (dummy at alternative Firmware)
26 (0x1A)	0 ... 127	Filter 0 Keytrack (%)
27 (0x1B)	0 ... 127	Filter 1 Keytrack (%)
28 (0x1C)	0 ... 127	Filter 2 Keytrack (%) (dummy at alternative Firmware)
29 (0x1D)	0 ... 127	Filter2 Modulation with Lfo1 (0 = Off, max +/- ca 1 octave)
30 (0x1E)	0 ... 127	Filter2 Modulation with Lfo2 (0 = Off, max +/- ca 1 octave)

31 (0x1F)	0 ... 127	Osc1 Attack (in steps of 20(<=1sec) or 50ms, ie. range 0 ... 4.85 sec)
32 (0x20)	0 ... 127	Osc1 Decay (in steps of 20(<=1sec) or 50ms, ie. range 0 ... 4.85 sec)
33 (0x21)	0 ... 127	Osc1 Sustain (127 = full envelope) - Hold type
34 (0x22)	0 ... 127	Osc1 Sustain (127 = full envelope) – 1 Shoot type
35 (0x23)	0 ... 127	Osc1 Release (in steps of 20(<=1sec) or 50ms, ie. range 0 ... 4.85 sec)
36 (0x24)	0 ... 127	Osc1 envelope max.volume
37 (0x25)	0 ... 127	Osc1 envelope Velo (28=no effect, internal scale –28 ... +99)
38 (0x26)	0 ... 127	Osc2 Attack (in steps of 20(<=1sec) or 50ms, ie. range 0 ... 4.85 sec)
39 (0x27)	0 ... 127	Osc2 Decay (in steps of 20(<=1sec) or 50ms, ie. range 0 ... 4.85 sec)
40 (0x28)	0 ... 127	Osc2 Sustain – Hold type (127 = full envelope)
41 (0x29)	0 ... 127	Osc2 Sustain – 1 Shoot type (127 = full envelope)
42 (0x2A)	0 ... 127	Osc2 Release (in steps of 20(<=1sec) or 50ms, ie. range 0 ... 4.85 sec)
43 (0x2B)	0 ... 127	Osc2 envelope max.volume
44 (0x2C)	0 ... 127	Osc2 envelope Velo (28=no effect, internal scale –28 ... +99)
45 (0x2D)	0 ... 127t	Noise Attack (in steps of 20(<=1sec) or 50ms, ie. range 0 ... 4.85 sec)
46 (0x2E)	0 ... 127	Noise Decay (in steps of 20(<=1sec) or 50ms, ie. range 0 ... 4.85 sec)
47 (0x2F)	0 ... 127	Noise Sustain – Hold type (127 = full envelope)
48 (0x30)	0 ... 127	Noise Sustain – 1 Shoot type (127 = full envelope)
49 (0x31)	0 ... 127	Noise Release (in steps of 20(<=1sec) or 50ms, ie. range 0 ... 4.85 sec)
50 (0x32)	0 ... 127	Noise envelope max.volume
51 (0x33)	0 ... 127	Noise envelope Velo (28=no effect, internal scale –28 ... +99)
52 (0x34)	0 ... 127	Gain Attack (in steps of 20(<=1sec) or 50ms, ie. range 0 ... 4.85 sec)
53 (0x35)	0 ... 127	Gain Decay (in steps of 20(<=1sec) or 50ms, ie. range 0 ... 4.85 sec)
54 (0x36)	0 ... 127	Gain Sustain – Hold type (127 = full envelope)
55 (0x37)	0 ... 127	Gain Sustain – 1 Shoot type (127 = full envelope) – 1 Shoot type
56 (0x38)	0 ... 127	Gain Release (in steps of 20(<=1sec) or 50ms, ie. range 0 ... 4.85 sec)
57 (0x39)	0 ... 127	Gain envelope (ADSR =dynamic reduction of setting CtrlCh4 in percent)
58 (0x3A)	0 ... 127	Gain envelope Velo (28=no effect, internal scale –28 ... +99)
59 (0x3B)	0 ... 127	Osc1 Frequency Modulation Attack range 0 ...4.85 sec)
60 (0x3C)	0 ... 127	Osc1 Frequency Modulation Decay (range 0 ... 4.85 sec)
61 (0x3D)	0 ... 127	Osc1 Frequency Modulation Sustain – Hold type (127 = full envelope)
62 (0x3E)	0 ... 127	Osc1 Frequency Modulation Sustain – 1 Shoot type (127 = full envelope)
63 (0x3F)	0 ... 127	Osc1 Frequency Modulation Release (range 0 ... 4.85 sec)
64 (0x40)	0 ... 127	Osc1 Frequency Modulation Fix Shift (28=no shift, internal scale –28 ... +99)
65 (0x41)	0 ... 127	Osc1 Frequency Modulation shift Velo (28=no effect, internal scale –28 ... +99)
66 (0x42)	0 ... 127	Osc2 Frequency Modulation Attack (range 0 ...4.58 sec)
67 (0x43)	0 ... 127	Osc2 Frequency Modulation Decay (range 0 ...4.58 sec)

68 (0x44)	0 ... 127	Osc2 Frequency Modulation Sustain – Hold type (127 = full envelope)
69 (0x45)	0 ... 127	Osc2 Frequency Modulation Sustain – 1 Shoot type (127 = full envelope)
70 (0x46)	0 ... 127	Osc2 Frequency Modulation Release (range 0 ... 4.85 sec)
71 (0x47)	0 ... 127	Osc2 Frequency Modulation Fix Shift (28=no shift, internal scale –28 ... +99)
72 (0x48)	0 ... 127	Osc2 Frequency Modulation shift Velo (28=no effect, internal scale –28 ... +99)
73 (0x49)	0 ... 127	Filter 0 Attack (in steps of 20(<=1sec) or 50ms, ie. range 0 ... 4.85 sec)
74 (0x4A)	0 ... 127	Filter 0 Decay (in steps of 20(<=1sec) or 50ms, ie. range 0 ... 4.85 sec)
75 (0x4B)	0 ... 127	Filter 0 Sustain – Hold type (127 = full envelope)
76 (0x4C)	0 ... 127	Filter 0 Sustain – 1 Shoot type (127 = full envelope)
77 (0x4D)	0 ... 127	Filter 0 Release (in steps of 20(<=1sec) or 50ms, ie. range 0 ... 4.85 sec)
78 (0x4E)	0 ... 127	Filter 0 Fix Shift (28=no shift, internal scale –28 ... +99)
79 (0x4F)	0 ... 127	Filter 0 shift Velo (28=no effect, internal scale –28 ... +99)
		alternative Firmware: Filters1 and 2 don't have an ADSR
80 (0x50)	0 ... 127	Filter 1 Attack (in steps of 20(<=1sec) or 50ms, ie. range 0 ... 4.85 sec))
81 (0x51)	0 ... 127	Filter 1 Decay (in steps of 20(<=1sec) or 50ms, ie. range 0 ... 4.85 sec)
82 (0x52)	0 ... 127	Filter 1 Sustain – Hold type (127 = full envelope)
83 (0x53)	0 ... 127	Filter 1 Sustain – 1 Shoot type (127 = full envelope)
84 (0x54)	0 ... 127	Filter 1 Release (in steps of 20(<=1sec) or 50ms, ie. range 0 ... 4.85 sec)
85 (0x55)	0 ... 127	Filter 1 Fix Shift (28=no shift, internal scale –28 ... +99)
86 (0x56)	0 ... 127	Filter 1 shift Velo (28=no effect, internal scale –28 ... +99)
87 (0x57)	0 ... 127	Filter 2 Attack (in steps of 20(<=1sec) or 50ms, ie. range 0 ... 4.85 sec)
88 (0x58)	0 ... 127	Filter 2 Decay (in steps of 20(<=1sec) or 50ms, ie. range 0 ... 4.85 sec)
89 (0x59)	0 ... 127	Filter 2 Sustain – Hold type (127 = full envelope)
90 (0x5A)	0 ... 127	Filter 2 Sustain – 1 Shoot typ (127 = full envelope)e
91 (0x5B)	0 ... 127	Filter 2 Release (in steps of 20(<=1sec) or 50ms, ie. range 0 ... 4.85 sec)
92 (0x5C)	0 ... 127	Filter 2 Fix Shift (28=no shift, internal scale –28 ... +99)
93 (0x5D)	0 ... 127	Filter 2 shift Velo (28=no effect, internal scale –28 ... +99)
94 (0x5E)	0 ... 127	Free ADSR Attack (in steps of 20(<=1sec) or 50ms, ie. range 0 ... 4.85 sec)
95 (0x5F)	0 ... 127	Free ADSR Decay (in steps of 20(<=1sec) or 50ms, ie. range 0 ... 4.85 sec)
96 (0x60)	0 ... 127	Free ADSR Sustain – Hold typ (127 = full envelope)e
97 (0x61)	0 ... 127	Free ADSR Sustain – 1 Shoot type (127 = full envelope)
98 (0x62)	0 ... 127	Free ADSR Release (in steps of 20(<=1sec) or 50ms, ie. range 0 ... 4.85 sec)
99 (0x63)	0 ... 127	Free ADSR Fix Shift (28=no shift, internal scale –28 ... +99)
100 (0x64)	0 ... 127	Free ADSR shift Velo (28=no effect, internal scale –28 ... +99)
101 (0x65)	0 ... 19	DelayTime coarse (10ms steps, 0—19)
102 (0x66)	0 ... 99	DelayTime fine (0.1ms steps, 0..99)
103 (0x67)	0 ... 127	Delay Level (in %, delay phase = input phase)
104 (0x68)	0 ... 127	Delay Level (in %, delay phase = inverted input phase)

105 (0x69)	0 ... 99	Delay Feedback (in %, feedback phase = input phase)
106 (0x6A)	0 ... 99	Delay Feedback (in %, feedback phase = inverted input phase)
107 (0x6B)	0 ... 50	Chorus Width
108 (0x6C)	18 ... 121	Filter resonance frequency as MIDI Note value (internally translated into corresponding Hz)
109 (0x6D)	18 ... 111 1 ... 100	Filter types 5,6,7: Bandwidth as corresponding MIDI Note value (internally translated into corresponding Hz) Filter types 3,4,8,9: Resonance $100 * ("zeta" = 1/2Q)$. (default 71 = Bessel filter. Lower values increase resonance peak)
110 (0x6E)	0 ... 15 0 ... 120	Filter type 5: additional Peak Gain ca. in dB Filter types 7,8,9: Parametric Peak Gain. (20 = "flat", Internally divided by 20, ie. effective range = -1.00 ... +5.00 in steps of 0.05)
111 (0x6F)	0 ... 52	Drums global volume (0 = -40dB, 40= 0dB, 52 = +12dB)
112 (0x70)	0 ... 52	Kick drum volume (0 = -40dB, 40= 0dB, 52 = +12dB)
113 (0x71)	0 ... 52	HiTom volume (0 = -40dB, 40= 0dB, 52 = +12dB)
114 (0x72)	0 ... 52	Snare volume (0 = -40dB, 40= 0dB, 52 = +12dB)
115 (0x73)	0 ... 52	Rim volume (0 = -40dB, 40= 0dB, 52 = +12dB)
116 (0x74)	0 ... 52	Clap volume (0 = -40dB, 40= 0dB, 52 = +12dB)
117 (0x75)	0 ... 52	Hihat volume (0 = -40dB, 40= 0dB, 52 = +12dB)
118 (0x76)	0 ... 52	Cymbal volume (0 = -40dB, 40= 0dB, 52 = +12dB)
119 (0x77)	0 ... 52	Bell volume (0 = -40dB, 40= 0dB, 52 = +12dB)
120 (0x78)	any	All Notes Off
123 (0x7B)	any	All Notes Off
120 (0x7E)	must be = 1	activate Polyphonic mode
120 (0x7F)	unrelevant	activate 10 voices Polyphonic mode

Dump of the active SoundSet inclusive all globals, Osc, ADSR and filter configurations can be triggered from the Operator Panel or by MIDI Program Change. Upload of stored dumps and user waveforms is triggered automatically by the SysEx header when the upload starts (format see below).

User waveforms

Up to 4 user defined waveforms ("User0,1,2,3") can be uploaded.

The upload procedure can only be triggered externally by starting the upload with an external software (tested here with "MIDITERM", see <www.midi-and-more.de/miditerm> -- push F6 or F7).

Then the LCD display of any Operator Panel "theme" changes into an upload message.

For upload, the userwave must be sent as a MIDI SysEx message (for external upload software preferably stored as .SYX file)

The mandatory upload format for one Userwave is as follows: (all byte descriptions in hex format)

first byte 0xF0

second byte 0x7F (i.e. MIDI "private" , or "experimental" 1 byte manufacturer ID)

third byte 0x40

fourth byte 0,1,2 or 3 (number of uploaded Userwave)

following **exactly 1024** integer 16 bit wide data words (with 14bit level data), each transferred as 4 subsequent MIDI data bytes containing the word split into 4 nibbles (big endian, i.e. highest nibble first). **That means: exactly 4096 MIDI data bytes must follow!**

followed by final MIDI EOX (0xF7)..

Any number of appropriate MIDI SysEx messages can be sent in a sequence.

This highly redundant data format was chosen because it is most easy for the user to transform 14 bit integer data into the nibble based format.

After reception the User waveform is stored in SRAM and can be played, but will be lost when power is turned off. To save it permanently, **all user data** (60 soundsets and 4 Userwaves) must be written into the microcontroller flash. Turn Encoder1 from the BASIC theme fully clockwise and select with Encoder2 "All Sounds>Flash". Due to the microcontroller memory organisation, a more selectable storage is impossible. The microcontroller flash is specified for about 10.000 storage processes.

Before Userwaves are uploaded, some basic waveforms are stored there as dummies.

For tests, some example User waveforms are supplied at my website. Furthermore, **the source code for a Windows console app ("waves14.c") is provided there as a template to build your own waveforms mathematically.**

Organisation of Sound Dump and "Factory Preset"

The **Sound Dump** (or vice versa Upload) is a MIDI SysEx conformant mirror of a "Factory Preset". Both have the same (7 bit MIDI conformant) byte structure, but **the dump additionally includes SysEx specific control bytes and communication parameters like USB Vid/Pid, Baud rates and MIDI channels.** Detailed data structure see below.

To trigger the dump of one or all SoundSet(s), turn Encoder1 from the BASIC theme fully clockwise, select with Encoder2 "DumpSnd#Enc3". Select the SoundSet with Encoder3.

To **save the dump on PC**, an appropriate software is necessary. If you are using MIDITERM <www.midi-and-more.de/miditerm.htm>: first push F9 there, then push Key1 at NucleoSynth. When the dump is finished, push Key12 at the PC. The dump is always saved in file MIDITERM.REC in the MIDITERM volume. No warning if the file already exists. Intentionally the dump/upload contains no checksum, so dumps can be modified with a hex editor.

The (re-)upload of a dump is triggered automatically when the adequate SysEx header is received. (MIDITERM: push F6 or F7 + Ctrl-F and enter the file name to be transmitted). Reception is signalled at the Operator Panel and there is no warning if runtime data are overwritten. Uploaded SoundSets are stored in SRAM and are lost when the power is cycled or a reset is performed. Storage from SRAM to Flash must be triggered explicitly: turn Encoder1 from the BASIC theme fully clockwise and select with Encoder2 "All Sounds>Flash". In this case, **all user data** (60 soundsets and 4 userwaves) are written into the microcontroller flash. Due to the microcontroller memory organisation, a more selectable storage is impossible.

"Factory Presets" cannot be modified at user level, but it is quite easy to do in C-programming. For this purpose, dumps can be transferred into a C-code compatible text file

for user made Factory Presets. See download of the simple Windows console app "Dump2C.exe" at my webpage <www.midi-and-more.de/nucleosynth.htm>
 At present time, "Factory Presets" in the published code are intended as raw configuration examples and templates for user optimization. Most examples only sound acceptable over a limited frequency range. Examples are a first design and still under development.

#0:only Osc1=sine. Else default,	#10:Kazoo
#1:default	#11:Blues Harp (Accordeon)
#2,3:Piano	#12,13:Flute
#4:Cembalo or Glass Vibraphone	#14,15:FM Synth examples
#5:Vibraphone	#16:Gain example
#5:Hammond organ	#17:Noise example (less well at alternative firmware)
#6:e-Bass	#18: Chain Saw
#8,9:Saxophone(Trumpet)	#19:FeedbackChaos

The **Structure of a dump of one SoundSet** is identical with a Factory Preset except first 16 bytes (= communication data) and final EOX:

```
void dumpSoundSet(uint32_t soundset) // *****
//transfer size is 181 bytes incl. Sysex Frame and ID, i.e. 177 bytes payload
// *****
{
uint32_t k,n;
uint8_t delayflags, delayLevel, feedback;
    sendM(0xF0);
    sendM(0x7D); //experimental/hobbyist MIDI Manufacturer ID
    sendM(0x20); //dump Sound Set file type ID
    sendM((uint8_t)soundset); //hex bytes:
    split16(gSysParam.Vid & 0x0000FFFF); //default = 00,04,08,03
    split16(gSysParam.Pid & 0x0000FFFF); //default = 05,07,04,00
    sendM((uint8_t)gSysParam.Channel); //default = 00
    sendM((uint8_t)gSysParam.DrumsChannel); //default = 09
    sendM((uint8_t)gSysParam.ComBaud); //default = 0B (dec.11)
    sendM((uint8_t)gSysParam.MidiBaud); //default = 1F (dec.31)
// gSysParam.MidiMsg is not stored in Dump
//16 bytes
    sendM((uint8_t)gVolumePreset[soundset]); //default = 28
    sendM((uint8_t)gaRoute0NPreset[soundset]); //default = 00
    sendM((uint8_t)gaRoute12Preset[soundset]); //default = 00
    sendM((uint8_t)gfGainPreset[soundset]); //default = 00
    sendM((uint8_t)(gfBalancePreset[soundset] & 0x000000FF)); //default = 28
    sendM((uint8_t)((gfBalancePreset[soundset] >> 8) & 0x000000FF)); //default = 28
    sendM((uint8_t)((gfBalancePreset[soundset] >> 16) & 0x000000FF)); //default = 14
    sendM((uint8_t)gsWAVEFORM_1Preset[soundset]); //default = 00
    sendM((uint8_t)gsWAVEFORM_2Preset[soundset]); //default = 02
    sendM((uint8_t)gsGLIDEPreset[soundset]); //default = 00
    sendM((uint8_t)gsTRANSPreset[soundset]); //default = 24
    sendM((uint8_t)gsTRANS_2Preset[soundset]); //default = 30
    sendM((uint8_t)gsDETUNEPreset[soundset]); //default = 40
    sendM((uint8_t)gsDETUNE_2Preset[soundset]); //default = 40
    sendM((uint8_t)(gsMOD_1Preset[soundset] & 0x7F)); //default = 00
    sendM((uint8_t)(gsMOD_2Preset[soundset] & 0x7F)); //default = 00
//32 bytes
    sendM((uint8_t)gfF2modPreset[soundset] & 0x7F); //default = 00
    sendM((uint8_t)((gsMOD_1Preset[soundset] >> 6) & 0x02) | //MODFLAGS
        (uint8_t)((gsMOD_2Preset[soundset] >> 5) & 0x04) |
        (uint8_t)((gfF2modPreset[soundset] >> 7) & 0x01) |
```

```

        (uint8_t)((gLfo1WavePreset[soundset] >> 3) & 0x10) |
        (uint8_t)((gLfo2WavePreset[soundset] >> 2) & 0x20) |
        (uint8_t)((gLfo3WavePreset[soundset] >> 1) & 0x40)); //default=77
        //(=dec.119)
    sendM((uint8_t)(gLfo1WavePreset[soundset] & 0x7F)); //default = 28
    sendM((uint8_t)gLfo1FreqPreset[soundset]); //default = 00
    sendM((uint8_t)(gLfo2WavePreset[soundset] & 0x7F)); //default = 28
    sendM((uint8_t)gLfo2FreqPreset[soundset]); //default = 0A (dec.10)
    sendM((uint8_t)(gLfo3WavePreset[soundset] & 0x7F)); //default = 28
    sendM((uint8_t)gLfo3FreqPreset[soundset]); //default = 00
//40 bytes
    for (k = 0; k < N_ASTRUCT; k++) { //alternative firmware: n=7,8 are dummies,
        n = (soundset * N_ASTRUCT) + k; //and n=9 is internally handled as n=7
        sendM((uint8_t)gsADSR[n].ATTACK); //default = 00 or 01
        sendM((uint8_t)gsADSR[n].DECAY); //default = 00
        sendM((uint8_t)gsADSR[n].SUSTAIN & 0x7F); //default = different
        sendM((uint8_t)(gsADSR[n].SUSTAIN >> 7) & 0x01) //Hold default = 01
        sendM((uint8_t)gsADSR[n].RELEASE); //default = 01
        sendM((uint8_t)gsADSR[n].FIXVOL); //default = different
        sendM((uint8_t)gsADSR[n].VELO); //default = 1C
    }
//110 bytes
    for (k = 0; k < N_FILTER; k++) {
        n = (soundset * N_FILTER) + k;
        sendM((uint8_t)gfS_IIR[n].TYPE); //default = 00
        sendM((uint8_t)gfS_IIR[n].FREQ); //default = 45
        sendM((uint8_t)gfS_IIR[n].QFACTOR); //default = 47
        sendM((uint8_t)gfS_IIR[n].PEAK); //default = 14
    }
//126 bytes
    for (k = 0; k < N_FILTER; k++) {
        n = (soundset * N_FILTER) + k;
        sendM((uint8_t)gfM_IIR[n].TYPE); //default = 00
        sendM((uint8_t)gfM_IIR[n].FREQ); //default = 45
        sendM((uint8_t)gfM_IIR[n].QFACTOR); //default = 47
        sendM((uint8_t)gfM_IIR[n].PEAK); //default = 14
    }
//142 bytes
    for (k = 0; k < N_FILTER; k++) {
        n = (soundset * N_FILTER) + k;
        sendM((uint8_t)gfP_IIR[n].TYPE); //default = 00
        sendM((uint8_t)gfP_IIR[n].FREQ); //default = 45
        sendM((uint8_t)gfP_IIR[n].QFACTOR); //default = 47
        sendM((uint8_t)gfP_IIR[n].PEAK); //default = 14
    }
//158 bytes
    sendM((uint8_t)((gfStageMask[soundset] >> 4) & 0x0000007F)); // special flags, def = 0
    sendM((uint8_t)gsFreeAdsrPreset[soundset]); //default = 00
//160 bytes
    sendM((uint8_t)gsKeytrack0Preset[soundset]); //default = 00
    sendM((uint8_t)gsKeytrack1Preset[soundset]); //default = 00
    sendM((uint8_t)gsKeytrack2Preset[soundset]); //default = 00, dummy at alternative firmware
    sendM((uint8_t)((gDelayTimePreset[soundset] >> 8) & 0x0000007F)); //def=0
    sendM((uint8_t)(gDelayTimePreset[soundset] & 0x0000007F)); //default =00
    sendM((uint8_t)gVibratoPreset[soundset] & 0x7F); //default = 00
    delayLevel = (uint8_t)gDelayLevelPreset[soundset];
    sendM(delayLevel & 0x7F); //default =00
    delayflags = 0;

```

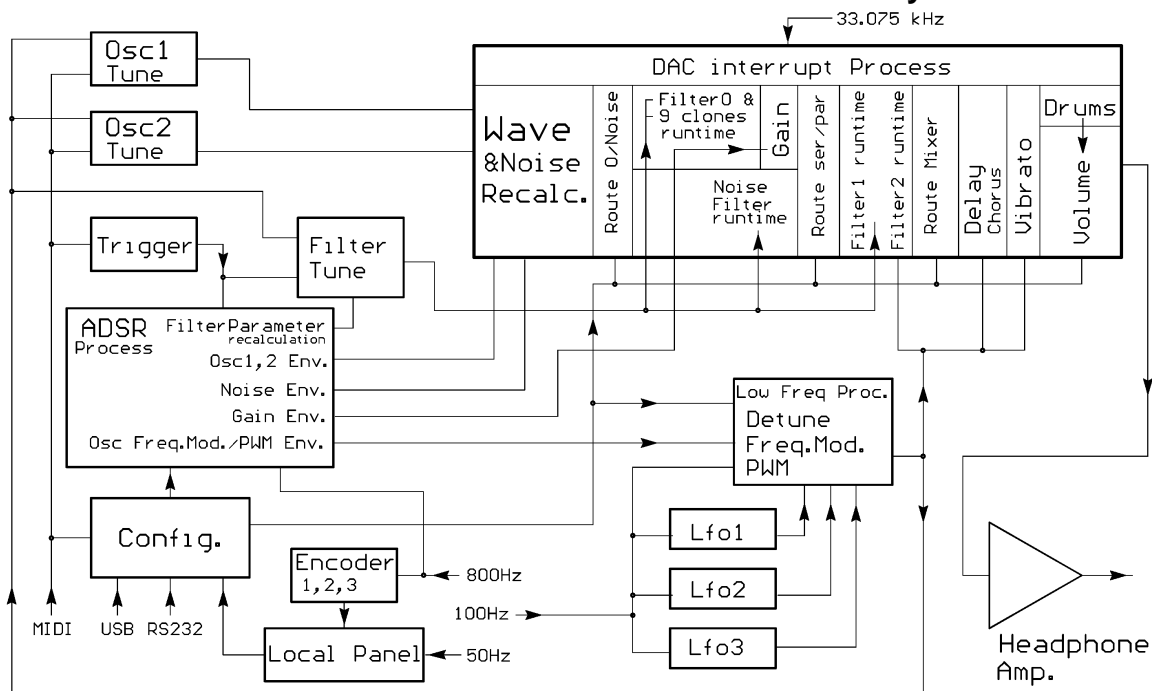
```

if (delayLevel >= 0x80) delayflags = 2; //if phase inversion
sendM((uint8_t)gDryFactorPreset[soundset]); //default = 64 (dec.100)
feedback = (uint8_t)gFeedbackPreset[soundset];
sendM(feedback & 0x7F); //default =00
if (feedback >= 128) delayflags |= 4; //if phase inversion
delayflags |= (uint8_t)((gVibratoPreset[soundset] >> 7) & 0x01); //default =01(vibrato=Lfo2)
sendM(delayflags); //default = 00
sendM((uint8_t)gChorusWidthPreset[soundset]); //default = 00
sendM((uint8_t)gdKickVol); //default = 28
sendM((uint8_t)gdHiTomVol); //default = 28
sendM((uint8_t)gdSnareVol); //default = 28
sendM((uint8_t)gdRimVol); //default = 28
sendM((uint8_t)gdClapVol); //default = 28
//176 bytes
sendM((uint8_t)gdHihatVol); //default = 28
sendM((uint8_t)gdCymbalVol); //default = 28
sendM((uint8_t)gdBellVol); //default = 28
sendM((uint8_t)gdDrumsVol); //default = 28
sendM(0xF7);
//181 bytes
}

```

Appendix: Alternative Firmware

The standard firmware has only one instance of filters for all voices, so the sound may depend on the sequence of keystroke. **To minimize this problem, Filter no.0 of the alternative firmware has a clone of its ADSR structure for every voice.**

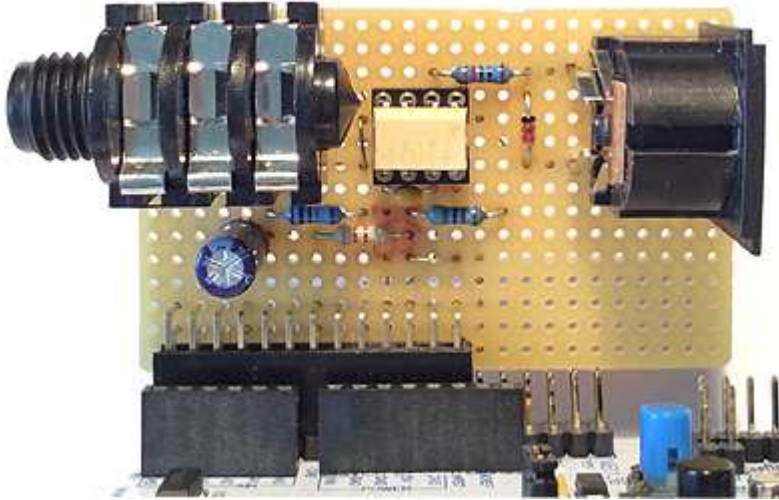


Because this is very runtime intensive, the sampling frequency was lowered to 33.075 kHz. Filters 1 and 2 don't have ADSR driven frequency shift, filter2 has no Keytrack. As a result of the lower sampling frequency, the max. delay time is 279.9ms. The drum waves are sampled now with 33.075 kHz, which gives a rather good drum engine. All features else are the same. The soundscape dump has the same structure as the standard firmware. Dumps and uploads can be interchanged between the firmware versions. Because the alternative firmware has some filter options less, these items hold place as dummies in the alternative firmware.

Firmware for minimal hardware:

This firmware can be used with an unmodified STM32F446 Nucleo and a **minimal hardware design** as described at <www.midi-and-more.de/nucleosynth/hwdiymen.pdf>

The minimal hardware may be useful for a first test before building a more complicated hardware. On the other hand, it is more complicated to configure a special sound.



The onboard USB of the F446 is disabled, instead USART2 is activated and is seen by a PC (via the **ST-LINK USB**) as **virtual COM port**. The MIDI/RS-232 interface is always configured as **legacy MIDI IN**. The green Nucleo onboard user **LED is usually off. It flashes** when data are received via USB/COM port or via legacy MIDI IN.

Configuration is possible with MIDI PgmChange and CChange commands. Dump of Sound Sets is impossible, but MIDI SysEx upload is supported.

More comfortably configuration is handled with ASCII commands via ST-LINK COM port and terminal software, e.g. Teraterm. Commands are the same as described above for the standard hardware.

A special feature of the firmware (Rev. 1.6++) for the minimal version is the **possibility to up- and download dumps in raw binary MIDI conformant SysEx format via virtual COM port**. Upload of SysEx formatted User Waves is possible this way, too.

Unfortunately most terminal software does not support raw binary file transfer. But the DTerm terminal program (available at my website <www.midi-and-more.de/more/dterm.htm>) supports this feature and was used for corresponding tests.

A complete **dump of all sounds** using DTerm can be made as follows:

Select the name of the downloaded file with menu "File>Select Record File" (or key F2). Start recording with key F9. Send command * to the NucleoSynth. Wait until the download is finished (visible), then push key F12. With a hex editor, this file can be splitted into SysEx conformant files for every sound. The data of each sound starts with the hex sequence F0 7F 20 + sound number, the end of each sound data is marked with byte F7.

Vice versa, the upload of a NucleoSynth compatible SysEx file (Dump or User Wave) can be triggered as follows:

Select the name of the file to be uploaded with menu "File>Select Send File" (or key F1). Start slow transmission with key F6 or fast transmission with key F7.

contact: wschemmert@t-online.de, <www.midi-and-more.de>

* Right of technical modifications reserved. Provided 'as is' - without any warranty. Any responsibility is excluded.

* This description is for information only. No product specification or useability is assured in juridical sense.

* Trademarks and product names cited in this text are property of their respective owners