The Becoming of the "Apparat 20"

("Apparat" is an old-fashioned german word for a telephone or also for the extension number of a phone.)

As I just started with my next MidiBox project, I thought this time I'll have to contribute a build blog. So here we go:

Concept

The idea is to build a synthesizer together with a vocoder into an old telephone. This brings up a few considerations:

Telephone

I really wanted a "Plug'n'Go" thing for stage use, so everything must fit into the phone. This means (of course) the phone needs to offer some space inside and has to look cool. I chose this phone - it's an 80ies phone built by SEL from the "design" line of the Deutsche Telekom. It emerged from a phone for disabled people, hence the really BIG dial buttons.



Vocoder

The vocoder has to be as small as possible. Mostly, vocoding is found in (at least) 9.5" FX units. Standalone vocoders (like the MAM VF11 or DIY versions) are normally analogue and even bigger units. Then I found the Alesis Metavox - it's the smallest (14 x 8 cm) and also cheapest vocoder existing (got it for 30 EUR new). Most of its functions are not really useful for music, all around the internal oscillator/LFO is more like a DJ toy. But the sound quality in external mode is quite good.



Synthesizer

The carrier signal has to be polyphonic and should be quite brilliant to achieve good vocoder results. At first, I thought about using a MBSID but quickly rejected this idea. It's only polyphonic with (mostly boring) single oscillator sounds, and the higher frequencies are not really present. Even more, The vocoder would have to be used with the normal filter switched off, so a great part of the SID character would be gone anyway. So the choice is the MB FM.

"User Interface"/Features

My central objective is that the finished Apparat 20 should just look as much as possible like an ordinary phone. So there won't be a display or any additional CS elements for the MBFM. Up to 20 patches will be selected via the normal dial keys (1..0, and you can select two "banks" with # and *). I can imagine some of you saying "But you can't edit anything and wouldn't it be cool to have this or that parameter accessible in live performance?". Of course it would, but I'm building a big MBFM for home use anyway, and there's always JSynthLib...

The vocoder has only one single parameter to modify when external mode is selected (Siblance, i.e. how much high frequency of the original signal will bleed thru, i.e. consonants). The corresponding pot will be mounted hidden beneath the pickup. The pickup will switch on and off the vocoder bypass and will get an electret capsule and a simple mic pre.

MIDIbox - http://www.midibox.org/dokuwiki/

Build Blog

2025/08/11 18:50

the phone

This is what the phone looks like inside - you can see there's quite some room in there.



After throwing everything out, I examined the keyboard to find out how to use it.

Last update: 2006/10/15 apparat_20_construction_blog http://www.midibox.org/dokuwiki/doku.php?id=apparat_20_construction_blog&rev=1153511352 09:35



Under the control board there is a simple 4×3 button matrix (without diodes - but that's no problem, it won't be necessary to press more than one button at the same time).



Because the contacts for the # and * buttons were not connected, I had to solder a few thin wires and I rewired the connector:



There's some space under the keypad, so I decided to mount the Core directly onto the backside of the keypad - the OPL3 module will still fit beneath it.

I built a simple Core module with most of the connectors and the PSU section left out - the phone will be powered from the nice +5V, +/-12V PSU I bought cheaply from Pollin last year, so there's no need for rectifier, regulator etc.. That way there was enough unused space on the PCB that I could cut a few tracks and add the BankStick directly to the core PCB.

The keypad matrix is connected to J5. There are 10k PullDown resistors where the three columns are connected. The OPL3 board wires were a bit tricky - the pin order is swapped because of the board being mounted heads-down.



2025/08/11 18:50



Software changes for the matrix keyboard

So now I had a keyboard with integrated core - how to scan the key matrix? There are the sm_examples from Thorsten, but the FM has hardly any RAM left and it's also preferable not to use too much processing power. Debouncing is not necessary, because every button press is a single key action and it doesn't hurt if the same Program Change is executed a few times. So I chose to do my own simple matrix scan.

We need a few preparations: First, I defined the two variables MATRIX_BUTTON and MATRIX_LAYER. Then at the end of USER_Init (main.asm) I inserted this:

```
;; set initial state
   bcf LATA, 0
   bcf LATA, 1
   bcf LATA, 2
   bcf LATA, 3
   movlw 0 \times 00
   movwf MATRIX_BUTTON
   movwf MATRIX LAYER
    ----- end modification -----
;;
#if CS_ENABLED
   ;; reset the control surface
         CS_MENU_Reset
   goto
#else
   return
#endif
```

The scanning is done in cs_menu_timer.inc. The following code is inserted at the beginning of the CS_MENU_TIMER function:

CS_MENU_TIMER

```
;; ----- scan button matrix -----
_ _ _ _ _
   ;; rows on PORTE (with 10k PullDown resistors)
   ;; cols on PORTA (0..3)
   movlw 0x00
   bsf LATA, 3
   IFSET PORTE, 0, movlw 0x01
   IFSET PORTE, 1, movlw 0x02
   IFSET PORTE, 2, movlw 0x03
   bcf LATA, 3
   bsf LATA, 2
   IFSET PORTE, 0, movlw 0x04
   IFSET PORTE, 1, movlw 0x05
   IFSET PORTE, 2, movlw 0x06
   bcf LATA, 2
   bsf LATA, 1
   IFSET PORTE, 0, movlw 0x07
   IFSET PORTE, 1, movlw 0x08
   IFSET PORTE, 2, movlw 0x09
   bcf LATA, 1
   bsf LATA, 0
```

IFSET PORTE, 0, movlw 0x11 IFSET PORTE, 1, movlw 0x0A IFSET PORTE, 2, movlw 0x10 bcf LATA, 0 movwf MATRIX_BUTTON ;; ----- end modification -----decfsz CS_MENU_TIMER_CTR, F; since this routine is called every 1 ms, but the cursor

;; ...

Here MATRIX_BUTTON is set to a value greater 0 while a key is pressed. The result is then processed at the end of each USER_TICK in the main.asm:

----- Program Change on key press ------;; movlw 0x00 ;; Button pressed? cpfsgt MATRIX BUTTON return movlw 0x0f ;; Yes -> * or # ? cpfsgt MATRIX_BUTTON goto PATCH SEL movlw 0×10 ;; Yes -> select layer (1..10/11..20) cpfsgt MATRIX BUTTON goto PATCH 10 ;; 1..10 movlw 0x00 movwf MATRIX LAYER return ;; 11..20 PATCH 10 movlw 0x0A movwf MATRIX_LAYER return ;; Normal patch selection PATCH SEL movf MATRIX BUTTON, W addlw (-1) movwf MIOS_PARAMETER2 ;; Add layer movf MATRIX LAYER, W addwf MIOS_PARAMETER2 movlw 0xC0 ;; Program Change Ch 1 movwf MIOS PARAMETER1 call MBFM MIDI NotifyReceivedEvent

;; ----- end modification ------ return

finishing the MBFM part

There will be some more coding necessary for the vocoder (the MB has to toggle one vocoder button 4 times on startup to select external mode, and the telephone hook has to toggle the bypass "pedal" (on/off) for every change), but the key scanning already works. So let's finish the MBFM. First, I milled away most of the mounting posts in the phone:



All jacks and the OPL3 board were installed and the wiring for power and MIDI was done:



As the keypad descends towards the front, I laid all larger parts (mainly the electrolytics) flat on the boards. The clearance between the boards is quite narrow but sufficient not to fear any shorts etc.



11/11



From: http://www.midibox.org/dokuwiki/ - MIDIbox

Permanent link: http://www.midibox.org/dokuwiki/doku.php?id=apparat_20_construction_blog&rev=1153511352



Last update: 2006/10/15 09:35