AC Sensorizer

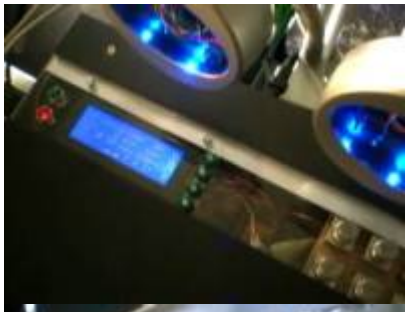
Description

AC Sensorizer sensorizes up to 8 sensors and interpolates its AIN-readings. The main target of this application are sensoric devices delivering not exactly 0 - 5 V, like pressure-, distance-, resistor-based sensors or softPots.

This is a preliminary beta-release.
Use and build at your own risk!
There will be major improvements in the GUI/HUI within the release of the next version. Though the code basis (ACSensorizer.c) won't probably change that much... – *audiocommander*











Picture Gallery:
http://www.audiocommander.de/picBrowser/picBrowser.php?go=galleries/060827_sensorizer&thumbs=1&thumbsize=XL





Demo-Video 1:
http://www.audiocommander.de/picBrowser/galleries/060827_sensorizer/,m5_smp01_billig-short.mov
Demo-Video 2:
http://www.audiocommander.de/picBrowser/galleries/060827_sensorizer/,m5_smp02_iwnh-short.mov

audicommander's YouTube Video Page: http://www.youtube.com/profile?user=audiocommander

Note:
The Sensorizer is attached to a Computer which converts the Controller Messages – produced by the Sensorizer – to harmonized and clock-synchronized Note-Signals. I'm using Absynth Sounds ...but due to the buggy versions and ugly customer support from NI, I discourage everyone using NI-products :(As soon as I have enough money, I'll switch to Virus :)

Features

- supports up to 8 sensors, code can be adapted to use more (eg with PIC18F4620)
- enable/disable single AINs
- slowdown: slows down the signal and increases the gaps between generated values
- assignable CH and Controller-Number
- sense-min: 10bit value, ignores every signal below
- sense-max: 10bit value, ignores every signal above
- sense-factor: used for signal interpolation... uses fast bitshifting or complex division depending on value
- AUTO-sense feature: auto-calibration of sensor, detect MIN/MAX by sensing automatically adapts sense-factor!
- scale from and scale to: scales the output value from/to
- invert signal
- pedal modes:
  - filter ⇒ only forward if pedal down;
  - panic ⇒ send panic on release pedal;
  - combinations of all pedal mode options are possible
- detect release: send 0-value if signal drops below sense-min
- bankstick support: 1 connected bankstick provides 2 banks with 127 patches each
- midi configurable: full configuration possible with NRPN-messages

    - NRPN-MSB CC99 for sensorSelect / sysEx mode
    - NRPN-LSB CC98 for controlType
    - DataEntry MSB CC6 and LSB CC38 for controlValue
- LCD-Output (2×16 or -recommended- 4×20)
- HUI-Input to control up to 4 sensors with 8 switch-encoders, pedal, store- & panic-button in 2 modes (PLAY/SETUP)
- ACSim Console Debugger: code integrated and ready to use configured for XCode2
    - select “ACSim” as target and test the application via command-line
    - inspect variables with a (graphical) debugger (GDB support within XCode2)
    - visit http://www.midibox.org → there's a tutorial how to use Code::Blocks

Required hardware

MBHP Modules:

- one MBHP_CORE module
- one MBHP_LCD module (5×20 optimized, 2×16 available)
- one MBHP_DIN module for
    - 8 Encoders (preferrably with):
    - 8 pushButtons
    - 1 mode-select switch
    - 1 pedal (optional)
    - 1 PANIC button (optional)
    - 1 STORE button (optional)
    - 1 UP & 1 DOWN button (optional)

Sensors:

- up to 8 (default: 4) sensors connected to unmuxed AIN (J5 of MBHP_CORE). Note that sensors 5 to 8 can only be configured via MIDI by sending NRPN-messages

Application Software

- m5 sensorizer v 0.2.4 (348 kB), download from http://www.audiocommander.de
- readme.txt …you should really read this, even if you're not recompiling the software!

Version History

- 0.2.2:
    - initial release
- 0.2.3:
    - removed MidiTrough and used MIOS_Merger
    - some minor bugs

    - two example miniAudicle/Chuck NRPN scripts are now included
- 0.2.4:
    - fixed severe bank select bug that prevented accessing even banks

Compiling Notes

The application can be recompiled with a variety of strictly separated #define- options. For example setting SENSORIZER_INTERFACE_HUI to 0 compiles the application without hardware input controls and therefore reduces the file- and application space. Compiling without HUI, BANKSTICK, NRPN-Config and LCD generates code with approx. 3 or 4 pages; compiling with all options will result in an application file of approx. 12 to 14 pages.

MIDI-Controllable Parameters

All Sensorizer parameters can be controlled and set by sending NRPN messages by MIDI:

1. Send NRPN MSB (Controller# 99) to select control type
2. Send NRPN LSB (Controller# 98) to set the control parameter
3. Send NRPN Data MSB (Controller# 6) and NRPN Data LSB (Controller# 38) to set the parameter value

NRPN MSB			
CC	**Value**	**Control Type**	**Note**
CC99, 0x63	0..7	Sensor 0..7	See Table B
CC99, 0x63	126	SysEx mode: ON	n/a
CC99, 0x63	127	SysEx mode: OFF	n/a

Table A: Control Types

NRPN LSB			**DATA ENTRY MSB/LSB**		
CC	**Value**	**Control Parameter**	**CC**	**Value**	**Description**
CC98, 0x62	0x00	enabled	CC38, 0x26	0/1	ON/OFF
CC98, 0x62	0x01	pedalMode	CC38, 0x26	0..7	FILTER/HOLD/PANIC/KOMBI
CC98, 0x62	0x02	autoSense	CC38, 0x26	0..2	AUTOSENSE_OFF/_MIN/_MAX
CC98, 0x62	0x03	invert	CC38, 0x26	0/1	0..127 or 127..0

CC98, 0x62	0x04	releaseDetect	CC38, 0x26	0/1	send 0 on release
CC98, 0x62	0x10	slowdown	CC38, 0x26	0..127	drop AIN notifications
CC98, 0x62	0x11	sense_min	CC38, 0x26 & CC6, 0x6	0..1023	drop below and set sense minimum
CC98, 0x62	0x12	sense_max	CC38, 0x26 & CC6, 0x6	0..1023	drop above and set sense maximum
CC98, 0x62	0x13	sense_factor	CC38, 0x26	0..64	f=(range/127)
CC98, 0x62	0x21	scale_from	CC38, 0x26	0..127	restrict and rescale output
CC98, 0x62	0x22	scale_to	CC38, 0x26	0..127	restrict and rescale output
CC98, 0x62	0x70	CH	CC38, 0x26	0..15	MIDI Channel of sensor
CC98, 0x62	0x71	CC	CC38, 0x26	0..127	MIDI Controller Change Number of sensor

Table B: Control Parameters

HUI-Controllable Parameters

The design of the Encoders is about to change with the next version. I just had one DIN at hand and the whole project had to be finished in time; with a second DIN and 4/5 more Encoders, everything will be a bit handier…

If you build the Sensorizer with enabled HUI-mode, it provides configuration possibilities for 4 sensors. However, you could change the code easily to support all 8 sensors.

Used Hardware:

- 1 Mode-Switch button (PLAYMODE ↔ SETUPMODE)
- 8 Encoders with Switch, two Encoders per sensor, the first row of encoders is called ENC_MIN, the lower ENC_MAX
- 4 Buttons (UP, DOWN, STORE, PANIC)

Function	**Encoder(s)**	**PLAYMODE**	**SETUP-MODE**
Turn	ENC_MIN	slowdown factor	scale from
Turn	ENC_MAX	send value	scale to
Press & Turn	ENC_MIN	CH (MIDI Channel)	senseMin (10bit)
Press & Turn	ENC_MAX	CC (MIDI Controller)	senseMax (10bit)

Function	Encoder(s)	PLAYMODE	SETUP-MODE
Press	ENC_MIN + ENC_MAX	sensor ON	AUTOSENSE_MIN
Press	ENC_MAX + ENC_MIN	sensor OFF	AUTOSENSE_MAX
Press	PANIC + PEDAL	switch PEDALMODE (Mode 0 to 7)	
Press	PANIC + ENC_MAX	switch RELEASE DETECT (ON/OFF)	
Press & Turn	PANIC + ENC_MAX	senseFactor (0-32)	-

Table C: Encoder Controls

BANKSTICK Patch Description

m5_sensorizer supports writing and reading to a connected bankstick.
Each patch consists of 2 pages ˆ 64 bytes ⇒ 128 bytes
2 banks with 128 patches each are available per 1 connected bankstick

Sending a PRG_CH message in PLAYMODE [P] loads the corresponding patch.
Sending a PRG_CH message in SETUPMODE[S] sets the corresponding patch without loading the patch data. Use this behavior to copy patches.

Choose the appropriate bank by sending a Coarse-Adjust Bankselect (CC#0).
In HUI-Mode, switching a bank on the device also sends the current Bank/PRG.

Memory-map of one patch:

Data	Size in Bytes = Sum	Page	Address
Version	01 = 1	1 @ 0x00	0 @ 0x00
PatchName	15 = 16	1	1
<reserved>	08 = 24	1	16 @ 0x10
sensor[8]	08 = 32	1	24
CH[8]	08 = 40	1	32 @ 0x20
CC[8]	08 = 48	1	40
<reserved>	16 = 64	1	48 @ 0x30
slowdown[8]	08 = 8	2 @ 0x40	0 @ 0x00
sense_factor[8]	08 = 16	2	8
sens_min[8].MSB	08 = 24	2	16 @ 0x10
sens_min[8].LSB	08 = 32	2	24
sens_max[8].MSB	08 = 40	2	32 @ 0x20
sens_max[8].LSB	08 = 48	2	40
scale_from[8]	08 = 56	2	48 @ 0x30
scale_to[8]	08 = 64	2	56

Table D: Bankstick Patch Content

Patch addresses are: (patch * 0x80)

or PIC-optimized: (unsigned int)patch « 7

Example hex-output of patch#0:

```
**MIOS_BANKSTICK_WritePage at 0x0
0:     02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
16:    00 00 00 00 00 00 00 00 80 80 80 80 40 08 00 00  ............@...
32:    00 00 00 00 04 04 04 04 14 15 16 17 00 00 00 00  ................
48:    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................

**MIOS_BANKSTICK_WritePage at 0x40
0:     08 08 08 08 00 00 00 00 04 04 04 04 00 00 00 00  ................
16:    00 00 00 00 7f 00 00 00 40 40 40 40 7f 00 00 00  ........@@@@....
32:    03 03 03 03 00 00 00 00 70 70 70 70 40 40 40 40  ........pppp@@@@
48:    00 00 00 00 00 00 00 00 7f 7f 7f 7f 00 00 00 00  ................
```

More details can be found in the readme.txt!
This site is about to change while the Sensorizer is beeing developed further!

have fun! – *audiocommander*

From:
https://www.midibox.org/dokuwiki/ - **MIDIbox**

Permanent link:
https://www.midibox.org/dokuwiki/doku.php?id=acsensorizer&rev=1160904901

Last update: **2006/11/06 11:06**