This page is part of the mios_c_simulator_-_debugger

ACSim_console.h
ACSim_console.c

ACSim_mios.h
ACSim_mios.c

ACSim_toolbox.h
ACSim_toolbox.c
ACMidiDefines.h

This code is for viewing only and may not be up to date. You can download the files in a .zip file here. If you make any updates, please send them to stryd_one and he will update the zip for you.

```
/*
 *  ACSim_console.c
 *  v 0.0.7
 *
 *  see header for version info and details
 */



// import standard I/O headers for console in- and output (printf and scanf)
#import <stdio.h>


#ifdef _DEBUG_OS_MAC
    #import <carbon/carbon.h>
#endif


// ACSim_mios.c contains all MIOS_FUNCTIONS() that are used throughout the
program
// eg MIOS_MIDI_AIN_PinGet(0) returns some random numbers in a range 0..127
// open up ACSim_mios.c to tweak around some values...
// normally there should be no need to change the content of these files:
#import "ACSim_mios.h"

// toolbox, handy functions like random number generation, hex-output...
#import "ACSim_toolbox.h"


// import c files that are included in "main.h"
//#include "../ACMidi.c" // <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< ADD
ADDITIONAL C.-FILES HERE!!!!

// import main c source file
#include "../main.c"
```

```
// ***** RUNLOOP *****

void runloop(int loopcount) {
    _Bool continueRunloop = TRUE;
    char str[64] = "";
    char choice = 'q';
    unsigned char c = 0;
    unsigned int i;
    unsigned int pin = 0;
    unsigned int value = 0;
    unsigned int value2 = 0;
    // poll Timer()
    if(debug_user_timer.TIMER_ENABLED) {
        Timer();
    }
    // poll Tick()
    Tick();
    // poll LCD_Display
    DISPLAY_Tick();
    // debug info
    printf("\n\nMIOS.ACSim(%i) >> ", loopcount);
    // read input
    fgets(str, 32, stdin);    // MAX chars = 32; this is NOT protected
against buffer overflows!!!
    choice = str[0];
    switch (choice) {

        case 'q': // quit
        case 'x':
            continueRunloop = FALSE;
            break;

        case ' ': // OK
            debug_din_value[DEBUG_BUTTON_OK] = 0;
            DIN_NotifyToggle(DEBUG_BUTTON_OK,
debug_din_value[DEBUG_BUTTON_OK]);
            break;

        case '+': // ENC++
            for(i=0;i<32;i++) { if(str[i]=='+') { c++; } }
                ENC_NotifyChange(DEBUG_ENCODER, c);
            break;
        case '-': // ENC --
            for(i=0;i<32;i++) { if(str[i]=='-') { c--; } }
            ENC_NotifyChange(DEBUG_ENCODER, c);
            break;
```

```
        case 'a': // AIN...
            sscanf(str, "a%i,%i", &pin, &value);
            if((pin > DEBUG_AIN_NUM) || (value > 1023)) {
                printf("! max_pin=9, max_value=1023 !");
                pin = 0; value = 0;
            } else {
                printf("scanning pin %i: %i", pin, value);
                // "set" pin values:
                debug_ain_value[pin] = value;
                AIN_NotifyChange(pin, value);
            }
            break;

        case 'b': // System Realtime BYTE
            debug_MIDI_byteNum = 0;    // sending single byte, maybe without
MIDI_START/_STOP
            sscanf(str, "b%i", &pin);
            MPROC_NotifyReceivedByte(pin);
            debug_MIDI_byteNum = 0;
            break;

        case 'd': // DIN...
            sscanf(str, "d%i,%i", &pin, &value);
            // invert value for easier rememberance
            value = value ^ 0x1;
            debug_din_value[pin] = value;
            DIN_NotifyToggle(pin, value);
            break;

        case 'e': // ENC # ++/--
            sscanf(str, "e%i", &pin);
            if(pin > DEBUG_ENC_NUM) {
                printf("! max_enc = %i", (DEBUG_ENC_NUM - 1));
                return;
            }
            for(i=3;i<32;i++) {
                if(str[i]=='+') {
                    c++;
                } else if(str[i]=='-') {
                    c--;
                }
            }
            ENC_NotifyChange(pin, c);
            break;

        case 'j': // set jumper pin
            sscanf(str, "j%i", &pin);
            switch(pin) {
                case 10:
                    PORTCbits.RC5 = 1;
                    Timer();
```

```
                    PORTCbits.RC5 = 0;
                    Timer();
                    break;
                case 14:
                    PORTDbits.RD4 = 1;
                    Timer();
                    PORTDbits.RD4 = 0;
                    Timer();
                    break;
                default:
                    printf("pin %i not (yet?) supported", pin);
                    break;
            }
            break;

        case 'm': // MIDI Receive (m msgType, argA, argB)
            sscanf(str, "m%i,%i,%i", &pin, &value, &value2);
            MPROC_NotifyReceivedEvnt(pin,value,value2);
            break;

        case 'n': // send note_on (n noteValue, volume, {channel})
            sscanf(str, "n%i,%i,%i", &pin, &value, &value2);
            if((value2 < 1) || (value2 > 16)) { value2 = 1; }
            MPROC_NotifyReceivedEvnt((143+value2),pin,value);
            break;

        case 'r': // random
            pin = ACRandomPin();
            value = ACRandomInt();
            printf("scanning pin %i: %i", pin, value);
            // "set" pin values:
            debug_ain_value[pin] = value;
            AIN_NotifyChange(pin, value);
            break;

        case 't': // test function
            sscanf(str, "t%i", &value);
//            for example:
//            IIC_SPEAKJET_Transmit14bit(value);
//            printf("\nDEC: %i \tPARAM3: %X \tPARAM2: %X \tPARAM1: %X",
value, MIOS_PARAMETER3, MIOS_PARAMETER2, MIOS_PARAMETER1);
            break;

        default:
            // nothing...
            break;
    }

    if(continueRunloop) {
        runloop(++loopcount);
    }
```

```
}



// ***** MAIN *****
int main(int argc, char **argv) {
    int exit_code = 0;

    // manual:
    printf("\n++++++ MIOS-DEBUG-CONSOLE ++++++\n");
    printf("(r)and (a)(pin),(value) (d)(pin),(state) \n");
    printf("(e{opt.})(++)(-) (j)umper(pin) \n");
    printf("(m)idi(msg),(argA),(argB) \n");
    printf("(n)ote_on(value),(velocity),{channel} \n");
    printf("(SPACE)OK e(x)it + [ENTER] \n\n");

    // init debug
    srandomdev(); // set the random seed

    // init MIOS
    MIOS_BOX_STAT.BS_AVAILABLE = 1;
    Init();
    DISPLAY_Init();

    printf("\n");

    // runloop
    runloop(0);

    return exit_code;
}
```

From:
https://wiki.midibox.org/ - **MIDIbox**

Permanent link:
https://wiki.midibox.org/doku.php?id=acsim_console_c&rev=1191673144

Last update: **2007/11/17 16:43**