/\* Analog and Digital Input and Output Server for MATLAB \*/

/\* Giampiero Campa, Copyright 2013 The MathWorks, Inc \*/

/\* This file is meant to be used with the MATLAB arduino IO

package, however, it can be used from the IDE environment

(or any other serial terminal) by typing commands like:

0e0 : assigns digital pin #4 (e) as input

0f1 : assigns digital pin #5 (f) as output

0n1 : assigns digital pin #13 (n) as output

1c : reads digital pin #2 (c)

1e : reads digital pin #4 (e)

2n0 : sets digital pin #13 (n) low

2n1 : sets digital pin #13 (n) high

2f1 : sets digital pin #5 (f) high

2f0 : sets digital pin #5 (f) low

4j2 : sets digital pin #9 (j) to 50=ascii(2) over 255

4jz : sets digital pin #9 (j) to 122=ascii(z) over 255

3a : reads analog pin #0 (a)

3f : reads analog pin #5 (f)

E0cd : attaches encoder #0 (0) on pins 2 (c) and 3 (d)

E1st : attaches encoder #1 on pins 18 (s) and 19 (t)

E2vu : attaches encoder #2 on pins 21 (v) and 20 (u)

G0 : gets 0 position of encoder #0

I0u : sets debounce delay to 20 (2ms) for encoder #0

H1 : resets position of encoder #1

F2 : detaches encoder #2

R0 : sets analog reference to DEFAULT

R1 : sets analog reference to INTERNAL

R2 : sets analog reference to EXTERNAL

X3 : roundtrip example case returning the input (ascii(3))

99 : returns script type (0 adio.pde ... 3 motor.pde ) \*/

/\* define internal for the MEGA as 1.1V (as as for the 328) \*/

#if defined(\_\_AVR\_ATmega1280\_\_) || defined(\_\_AVR\_ATmega2560\_\_)

#define INTERNAL INTERNAL1V1

#endif

/\* define encoder structure \*/

typedef struct { int pinA; int pinB; int pos; int del;} Encoder;

volatile Encoder Enc[3] = {{0,0,0,0}, {0,0,0,0}, {0,0,0,0}};

void setup() {

/\* initialize serial \*/

Serial.begin(115200);

}

void loop() {

/\* variables declaration and initialization \*/

static int s = -1; /\* state \*/

static int pin = 13; /\* generic pin number \*/

static int enc = 0; /\* generic encoder number \*/

int val = 0; /\* generic value read from serial \*/

int agv = 0; /\* generic analog value \*/

int dgv = 0; /\* generic digital value \*/

/\* The following instruction constantly checks if anything

is available on the serial port. Nothing gets executed in

the loop if nothing is available to be read, but as soon

as anything becomes available, then the part coded after

the if statement (that is the real stuff) gets executed \*/

if (Serial.available() >0) {

/\* whatever is available from the serial is read here \*/

val = Serial.read();

/\* This part basically implements a state machine that

reads the serial port and makes just one transition

to a new state, depending on both the previous state

and the command that is read from the serial port.

Some commands need additional inputs from the serial

port, so they need 2 or 3 state transitions (each one

happening as soon as anything new is available from

the serial port) to be fully executed. After a command

is fully executed the state returns to its initial

value s=-1 \*/

switch (s) {

/\* s=-1 means NOTHING RECEIVED YET \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

case -1:

/\* calculate next state \*/

if (val>47 && val<90) {

/\* the first received value indicates the mode

49 is ascii for 1, ... 90 is ascii for Z

s=0 is change-pin mode;

s=10 is DI; s=20 is DO; s=30 is AI; s=40 is AO;

s=90 is query script type (1 basic, 2 motor);

s=210 is encoder attach; s=220 is encoder detach;

s=230 is get encoder position; s=240 is encoder reset;

s=250 is set encoder debounce delay;

s=340 is change analog reference;

s=400 example echo returning the input argument;

\*/

s=10\*(val-48);

}

/\* the following statements are needed to handle

unexpected first values coming from the serial (if

the value is unrecognized then it defaults to s=-1) \*/

if ((s>40 && s<210 && s!=90) || (s>250 && s!=340 && s!=400)) {

s=-1;

}

/\* the break statements gets out of the switch-case, so

/\* we go back and wait for new serial data \*/

break; /\* s=-1 (initial state) taken care of \*/

/\* s=0 or 1 means CHANGE PIN MODE \*/

case 0:

/\* the second received value indicates the pin

from abs('c')=99, pin 2, to abs('¦')=166, pin 69 \*/

if (val>98 && val<167) {

pin=val-97; /\* calculate pin \*/

s=1; /\* next we will need to get 0 or 1 from serial \*/

}

else {

s=-1; /\* if value is not a pin then return to -1 \*/

}

break; /\* s=0 taken care of \*/

case 1:

/\* the third received value indicates the value 0 or 1 \*/

if (val>47 && val<50) {

/\* set pin mode \*/

if (val==48) {

pinMode(pin,INPUT);

}

else {

pinMode(pin,OUTPUT);

}

}

s=-1; /\* we are done with CHANGE PIN so go to -1 \*/

break; /\* s=1 taken care of \*/

/\* s=10 means DIGITAL INPUT \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

case 10:

/\* the second received value indicates the pin

from abs('c')=99, pin 2, to abs('¦')=166, pin 69 \*/

if (val>98 && val<167) {

pin=val-97; /\* calculate pin \*/

dgv=digitalRead(pin); /\* perform Digital Input \*/

Serial.println(dgv); /\* send value via serial \*/

}

s=-1; /\* we are done with DI so next state is -1 \*/

break; /\* s=10 taken care of \*/

/\* s=20 or 21 means DIGITAL OUTPUT \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

case 20:

/\* the second received value indicates the pin

from abs('c')=99, pin 2, to abs('¦')=166, pin 69 \*/

if (val>98 && val<167) {

pin=val-97; /\* calculate pin \*/

s=21; /\* next we will need to get 0 or 1 from serial \*/

}

else {

s=-1; /\* if value is not a pin then return to -1 \*/

}

break; /\* s=20 taken care of \*/

case 21:

/\* the third received value indicates the value 0 or 1 \*/

if (val>47 && val<50) {

dgv=val-48; /\* calculate value \*/

digitalWrite(pin,dgv); /\* perform Digital Output \*/

}

s=-1; /\* we are done with DO so next state is -1 \*/

break; /\* s=21 taken care of \*/

/\* s=30 means ANALOG INPUT \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

case 30:

/\* the second received value indicates the pin

from abs('a')=97, pin 0, to abs('p')=112, pin 15 \*/

if (val>96 && val<113) {

pin=val-97; /\* calculate pin \*/

agv=analogRead(pin); /\* perform Analog Input \*/

Serial.println(agv); /\* send value via serial \*/

}

s=-1; /\* we are done with AI so next state is -1 \*/

break; /\* s=30 taken care of \*/

/\* s=40 or 41 means ANALOG OUTPUT \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

case 40:

/\* the second received value indicates the pin

from abs('c')=99, pin 2, to abs('¦')=166, pin 69 \*/

if (val>98 && val<167) {

pin=val-97; /\* calculate pin \*/

s=41; /\* next we will need to get value from serial \*/

}

else {

s=-1; /\* if value is not a pin then return to -1 \*/

}

break; /\* s=40 taken care of \*/

case 41:

/\* the third received value indicates the analog value \*/

analogWrite(pin,val); /\* perform Analog Output \*/

s=-1; /\* we are done with AO so next state is -1 \*/

break; /\* s=41 taken care of \*/

/\* s=90 means Query Script Type:

(0 adio, 1 adioenc, 2 adiosrv, 3 motor) \*/

case 90:

if (val==57) {

/\* if string sent is 99 send script type via serial \*/

Serial.println(1);

}

s=-1; /\* we are done with this so next state is -1 \*/

break; /\* s=90 taken care of \*/

/\* s=210 to 212 means ENCODER ATTACH \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

case 210:

/\* the second value indicates the encoder number:

either 0, 1 or 2 \*/

if (val>47 && val<51) {

enc=val-48; /\* calculate encoder number \*/

s=211; /\* next we need the first attachment pin \*/

}

else {

s=-1; /\* if value is not an encoder then return to -1\*/

}

break; /\* s=210 taken care of \*/

case 211:

/\* the third received value indicates the first pin

from abs('c')=99, pin 2, to abs('¦')=166, pin 69 \*/

if (val>98 && val<167) {

pin=val-97; /\* calculate pin \*/

Enc[enc].pinA=pin; /\* set pin A \*/

s=212; /\* next we need the second attachment pin \*/

}

else {

s=-1; /\* if value is not a servo then return to -1 \*/

}

break; /\* s=211 taken care of \*/

case 212:

/\* the fourth received value indicates the second pin

from abs('c')=99, pin 2, to abs('¦')=166, pin 69 \*/

if (val>98 && val<167) {

pin=val-97; /\* calculate pin \*/

Enc[enc].pinB=pin; /\* set pin B \*/

/\* set encoder pins as inputs \*/

pinMode(Enc[enc].pinA, INPUT);

pinMode(Enc[enc].pinB, INPUT);

/\* turn on pullup resistors \*/

digitalWrite(Enc[enc].pinA, HIGH);

digitalWrite(Enc[enc].pinB, HIGH);

/\* attach interrupts \*/

switch(enc) {

case 0:

attachInterrupt(getIntNum(Enc[0].pinA), isrPinAEn0, CHANGE);

attachInterrupt(getIntNum(Enc[0].pinB), isrPinBEn0, CHANGE);

break;

case 1:

attachInterrupt(getIntNum(Enc[1].pinA), isrPinAEn1, CHANGE);

attachInterrupt(getIntNum(Enc[1].pinB), isrPinBEn1, CHANGE);

break;

case 2:

attachInterrupt(getIntNum(Enc[2].pinA), isrPinAEn2, CHANGE);

attachInterrupt(getIntNum(Enc[2].pinB), isrPinBEn2, CHANGE);

break;

}

}

s=-1; /\* we are done with encoder attach so -1 \*/

break; /\* s=212 taken care of \*/

/\* s=220 means ENCODER DETACH \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

case 220:

/\* the second value indicates the encoder number:

either 0, 1 or 2 \*/

if (val>47 && val<51) {

enc=val-48; /\* calculate encoder number \*/

/\* detach interrupts \*/

detachInterrupt(getIntNum(Enc[enc].pinA));

detachInterrupt(getIntNum(Enc[enc].pinB));

}

s=-1; /\* we are done with encoder detach so -1 \*/

break; /\* s=220 taken care of \*/

/\* s=230 means GET ENCODER POSITION \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

case 230:

/\* the second value indicates the encoder number:

either 0, 1 or 2 \*/

if (val>47 && val<51) {

enc=val-48; /\* calculate encoder number \*/

/\* send the value back \*/

Serial.println(Enc[enc].pos);

}

s=-1; /\* we are done with encoder detach so -1 \*/

break; /\* s=230 taken care of \*/

/\* s=240 means RESET ENCODER POSITION \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

case 240:

/\* the second value indicates the encoder number:

either 0, 1 or 2 \*/

if (val>47 && val<51) {

enc=val-48; /\* calculate encoder number \*/

/\* reset position \*/

Enc[enc].pos=0;

}

s=-1; /\* we are done with encoder detach so -1 \*/

break; /\* s=240 taken care of \*/

/\* s=250 and 251 mean SET ENCODER DEBOUNCE DELAY \*\*\*\*\* \*/

case 250:

/\* the second value indicates the encoder number:

either 0, 1 or 2 \*/

if (val>47 && val<51) {

enc=val-48; /\* calculate encoder number \*/

s=251; /\* next we need the first attachment pin \*/

}

else {

s=-1; /\* if value is not an encoder then return to -1\*/

}

break; /\* s=250 taken care of \*/

case 251:

/\* the third received value indicates the debounce

delay value in units of approximately 0.1 ms each

from abs('a')=97, 0 units, to abs('¦')=166, 69 units\*/

if (val>96 && val<167) {

Enc[enc].del=val-97; /\* set debounce delay \*/

}

s=-1; /\* we are done with this so next state is -1 \*/

break; /\* s=251 taken care of \*/

/\* s=340 or 341 means ANALOG REFERENCE \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

case 340:

/\* the second received value indicates the reference,

which is encoded as is 0,1,2 for DEFAULT, INTERNAL

and EXTERNAL, respectively. Note that this function

is ignored for boards not featuring AVR or PIC32 \*/

#if defined(\_\_AVR\_\_) || defined(\_\_PIC32MX\_\_)

switch (val) {

case 48:

analogReference(DEFAULT);

break;

case 49:

analogReference(INTERNAL);

break;

case 50:

analogReference(EXTERNAL);

break;

default: /\* unrecognized, no action \*/

break;

}

#endif

s=-1; /\* we are done with this so next state is -1 \*/

break; /\* s=341 taken care of \*/

/\* s=400 roundtrip example function (returns the input)\*/

case 400:

/\* the second value (val) can really be anything here \*/

/\* This is an auxiliary function that returns the ASCII

value of its first argument. It is provided as an

example for people that want to add their own code \*/

/\* your own code goes here instead of the serial print \*/

Serial.println(val);

s=-1; /\* we are done with the aux function so -1 \*/

break; /\* s=400 taken care of \*/

/\* \*\*\*\*\*\*\* UNRECOGNIZED STATE, go back to s=-1 \*\*\*\*\*\*\* \*/

default:

/\* we should never get here but if we do it means we

are in an unexpected state so whatever is the second

received value we get out of here and back to s=-1 \*/

s=-1; /\* go back to the initial state, break unneeded \*/

} /\* end switch on state s \*/

} /\* end if serial available \*/

} /\* end loop statement \*/

/\* auxiliary function to handle encoder attachment \*/

int getIntNum(int pin) {

/\* returns the interrupt number for a given interrupt pin

see http://arduino.cc/it/Reference/AttachInterrupt \*/

switch(pin) {

case 2:

return 0;

case 3:

return 1;

case 21:

return 2;

case 20:

return 3;

case 19:

return 4;

case 18:

return 5;

default:

return -1;

}

}

/\* auxiliary debouncing function \*/

void debounce(int del) {

int k;

for (k=0;k<del;k++) {

/\* can't use delay in the ISR so need to waste some time

perfoming operations, this uses roughly 0.1ms on uno \*/

k = k +0.0 +0.0 -0.0 +3.0 -3.0;

}

}

/\* Interrupt Service Routine: change on pin A for Encoder 0 \*/

void isrPinAEn0(){

/\* read pin B right away \*/

int drB = digitalRead(Enc[0].pinB);

/\* possibly wait before reading pin A, then read it \*/

debounce(Enc[0].del);

int drA = digitalRead(Enc[0].pinA);

/\* this updates the counter \*/

if (drA == HIGH) { /\* low->high on A? \*/

if (drB == LOW) { /\* check pin B \*/

Enc[0].pos++; /\* going clockwise: increment \*/

} else {

Enc[0].pos--; /\* going counterclockwise: decrement \*/

}

} else { /\* must be high to low on A \*/

if (drB == HIGH) { /\* check pin B \*/

Enc[0].pos++; /\* going clockwise: increment \*/

} else {

Enc[0].pos--; /\* going counterclockwise: decrement \*/

}

} /\* end counter update \*/

} /\* end ISR pin A Encoder 0 \*/

/\* Interrupt Service Routine: change on pin B for Encoder 0 \*/

void isrPinBEn0(){

/\* read pin A right away \*/

int drA = digitalRead(Enc[0].pinA);

/\* possibly wait before reading pin B, then read it \*/

debounce(Enc[0].del);

int drB = digitalRead(Enc[0].pinB);

/\* this updates the counter \*/

if (drB == HIGH) { /\* low->high on B? \*/

if (drA == HIGH) { /\* check pin A \*/

Enc[0].pos++; /\* going clockwise: increment \*/

} else {

Enc[0].pos--; /\* going counterclockwise: decrement \*/

}

} else { /\* must be high to low on B \*/

if (drA == LOW) { /\* check pin A \*/

Enc[0].pos++; /\* going clockwise: increment \*/

} else {

Enc[0].pos--; /\* going counterclockwise: decrement \*/

}

} /\* end counter update \*/

} /\* end ISR pin B Encoder 0 \*/

/\* Interrupt Service Routine: change on pin A for Encoder 1 \*/

void isrPinAEn1(){

/\* read pin B right away \*/

int drB = digitalRead(Enc[1].pinB);

/\* possibly wait before reading pin A, then read it \*/

debounce(Enc[1].del);

int drA = digitalRead(Enc[1].pinA);

/\* this updates the counter \*/

if (drA == HIGH) { /\* low->high on A? \*/

if (drB == LOW) { /\* check pin B \*/

Enc[1].pos++; /\* going clockwise: increment \*/

} else {

Enc[1].pos--; /\* going counterclockwise: decrement \*/

}

} else { /\* must be high to low on A \*/

if (drB == HIGH) { /\* check pin B \*/

Enc[1].pos++; /\* going clockwise: increment \*/

} else {

Enc[1].pos--; /\* going counterclockwise: decrement \*/

}

} /\* end counter update \*/

} /\* end ISR pin A Encoder 1 \*/

/\* Interrupt Service Routine: change on pin B for Encoder 1 \*/

void isrPinBEn1(){

/\* read pin A right away \*/

int drA = digitalRead(Enc[1].pinA);

/\* possibly wait before reading pin B, then read it \*/

debounce(Enc[1].del);

int drB = digitalRead(Enc[1].pinB);

/\* this updates the counter \*/

if (drB == HIGH) { /\* low->high on B? \*/

if (drA == HIGH) { /\* check pin A \*/

Enc[1].pos++; /\* going clockwise: increment \*/

} else {

Enc[1].pos--; /\* going counterclockwise: decrement \*/

}

} else { /\* must be high to low on B \*/

if (drA == LOW) { /\* check pin A \*/

Enc[1].pos++; /\* going clockwise: increment \*/

} else {

Enc[1].pos--; /\* going counterclockwise: decrement \*/

}

} /\* end counter update \*/

} /\* end ISR pin B Encoder 1 \*/

/\* Interrupt Service Routine: change on pin A for Encoder 2 \*/

void isrPinAEn2(){

/\* read pin B right away \*/

int drB = digitalRead(Enc[2].pinB);

/\* possibly wait before reading pin A, then read it \*/

debounce(Enc[2].del);

int drA = digitalRead(Enc[2].pinA);

/\* this updates the counter \*/

if (drA == HIGH) { /\* low->high on A? \*/

if (drB == LOW) { /\* check pin B \*/

Enc[2].pos++; /\* going clockwise: increment \*/

} else {

Enc[2].pos--; /\* going counterclockwise: decrement \*/

}

} else { /\* must be high to low on A \*/

if (drB == HIGH) { /\* check pin B \*/

Enc[2].pos++; /\* going clockwise: increment \*/

} else {

Enc[2].pos--; /\* going counterclockwise: decrement \*/

}

} /\* end counter update \*/

} /\* end ISR pin A Encoder 2 \*/

/\* Interrupt Service Routine: change on pin B for Encoder 2 \*/

void isrPinBEn2(){

/\* read pin A right away \*/

int drA = digitalRead(Enc[2].pinA);

/\* possibly wait before reading pin B, then read it \*/

debounce(Enc[2].del);

int drB = digitalRead(Enc[2].pinB);

/\* this updates the counter \*/

if (drB == HIGH) { /\* low->high on B? \*/

if (drA == HIGH) { /\* check pin A \*/

Enc[2].pos++; /\* going clockwise: increment \*/

} else {

Enc[2].pos--; /\* going counterclockwise: decrement \*/

}

} else { /\* must be high to low on B \*/

if (drA == LOW) { /\* check pin A \*/

Enc[2].pos++; /\* going clockwise: increment \*/

} else {

Enc[2].pos--; /\* going counterclockwise: decrement \*/

}

} /\* end counter update \*/

} /\* end ISR pin B Encoder 2 \*/