TestMain.c – e128 main loop code

**initialization:**

**--------------------------------------------------------------------------------------------------------------------------------**

**global function prototypes:**

void init\_sound\_timer(void);

**module function prototypes:**

static void init\_sci(void);

static void init\_pins(void);

static void receive\_message(void);

static void setup\_handshake\_packet(void);

static void setup\_command\_packet(void);

static void finalize\_packet(void);

static unsigned char find\_checksum(void);

static void decode\_packet(void);

static void reset\_timeout(unsigned char message\_received);

**module level variables:**

// buffers

static char transmit\_buffer[max\_buffer\_length];

static unsigned char receive\_buffer[max\_buffer\_length];

// tx/rx counters

static unsigned char transmit\_count;

static unsigned char receive\_count;

static unsigned char transmit\_length;

static unsigned char receive\_length;

static unsigned char transmit\_checksum;

static unsigned char receive\_checksum;

// state machine variables

static unsigned char transmit\_in\_progress;

static unsigned char ready\_to\_transmit;

static unsigned char ready\_to\_package;

static unsigned char receive\_in\_progress;

static unsigned char ready\_to\_decode;

static unsigned char next\_transmit\_ready;

static unsigned char last\_message\_received;

static unsigned char last\_message\_received;

static unsigned char xbee\_connected;

// controller data

static char data\_speed;

static char data\_direction;

static char data\_action;

// destination address

static unsigned char source\_addr\_msb;

static unsigned char source\_addr\_lsb;

static unsigned char boat\_address;

// timer data

static unsigned char num\_overflows

static unsigned char sound\_overflows;

static unsigned char controller\_timeout;

**--------------------------------------------------------------------------------------------------------------------------------**

**pseudo code:**

**--------------------------------------------------------------------------------------------------------------------------------**

// main loop

void main (void) {

init pins

init sci

init timers

init variables

while (1)

{

if message incoming

read in message

set ready to decode

end if

if ready to decode

decode packet

reset ready to decode

end if

if timeout occurred

turn off leds

turn off audio

update luc state machine, pass hzv timeout event

reset last message received flag

end if

if handshake requested

update luc state machine, pass handshake requested event

end if

if transmit is in progress

transmit next byte

update variables

end if

if ready to transmit

reset transmit count

set transmit in progress

end if

if ready to package

finalize packet

set ready to transmit

end if

if last message was successfully received

if next transmit is ready

if handshake is requested

get current boat address

if boat address is valid

set up handshake packet

set ready to package

end if

else if luc is connected

set up command packet

set ready to package

else

// keep waiting

end if

end if

end if

}

}

**--------------------------------------------------------------------------------------------------------------------------------**

**--------------------------------------------------------------------------------------------------------------------------------**

// initialize the sci

static void init\_sci(void) {

set baud rate to 9600

enable transmit and receive

}

**--------------------------------------------------------------------------------------------------------------------------------**

**--------------------------------------------------------------------------------------------------------------------------------**

// initialize the e128 pins

static void init\_pins(void) {

initialize port AD 0,1,2 as inputs

initialize port T0-4 as outputs, T5-7 as inputs

initialize port U as outputs

initialize port S as inputs

turn led dislay off

}

**--------------------------------------------------------------------------------------------------------------------------------**

**--------------------------------------------------------------------------------------------------------------------------------**

// init timer

static void init\_timer(void){

turn the timer system on

set pre-scale to /128 = 0.1875MHz timer clk

Set up TIM0 OC4 to time updates

set cap/comp 4 to output compare rest are inputs

no pin connected for OC4

schedule first rise

clear OC4 flag

enable OC4 interrupt

Set up TIM0 OC5 to time updates

set cap/comp 5 to output compare rest are inputs

no pin connected for OC53

schedule first rise

clear OC5 flag

enable OC5 interrupt

Set up TIM0 OC6 to time updates

set cap/comp 6 to output compare rest are inputs

no pin connected for OC63

schedule first rise

clear OC6 flag

enable OC6 interrupt

default next transmit ready to true

clear all overflow and timeout flags

enable interrupts

}

**--------------------------------------------------------------------------------------------------------------------------------**

**--------------------------------------------------------------------------------------------------------------------------------**

// code to transmit byte

static void transmit(void) {

if last byte

if transmit buffer is empty

if receive is not already in progress

write final byte to transmit register

reset transmit in progress flag

end if

end if

else

if transmit buffer is empty

if receive not already in progress

write next byte to transmit register

increment transmit count

end if

end if

end if

}

**--------------------------------------------------------------------------------------------------------------------------------**

**--------------------------------------------------------------------------------------------------------------------------------**

// code to receive incoming message

static void receive\_message(void) {

read in receive data to buffer

if first byte in receive packet

if receive byte is equal to start delimiter

reset receive checksum

increment receive byte count

end if

else if second byte in receive packet

if receive byte is equal to 0x00

increment counter

else

reset receive byte count

end if

else if third byte in receive packet

store receive length

increment receive byte count

else if not last byte in receive packet

update receive checksum

increment receive byte count

else if last byte in receive packet

update receive checksum

if checksum is correct

set ready to decode

end if

reset receive byte count

end if

}

**--------------------------------------------------------------------------------------------------------------------------------**

**--------------------------------------------------------------------------------------------------------------------------------**

// set up handshake packet

static void setup\_handshake\_packet(void) {

fill in transmit buffer

set transmit\_buffer[3] as transmit\_request

set transmit\_buffer[4] as send\_ack

set transmit\_buffer[5] as destination address msb

set transmit\_buffer[6] as boat address

set transmit\_buffer[7] as packet options

set transmit\_buffer[8] as handshake packet

set transmit\_buffer[9] as handshake message

set transmit\_length to 7

calculate transmit checksum

}

**--------------------------------------------------------------------------------------------------------------------------------**

**--------------------------------------------------------------------------------------------------------------------------------**

// set up command packet

static void setup\_command\_packet(void) {

read in current speed request

read in current direction request

read in current shoot request

fill in transmit buffer

set transmit\_buffer[3] as transmit request

set transmit\_buffer[4] as don’t send ack

set transmit\_buffer[5] as destination address msb

set transmit\_buffer[6] as boat address

set transmit\_buffer[7] as packet options

set transmit\_buffer[8] as command packet

set transmit\_buffer[9] as current speed request

set transmit\_buffer[10] as current direction request

set transmit\_buffer[11] as current shoot request

set transmit\_buffer[12] as custom data 1

set transmit\_buffer[13] as custom data 2

set transmit\_buffer[14] as custom data 3

set transmit length to 12

calculate transmit checksum

}

**--------------------------------------------------------------------------------------------------------------------------------**

**--------------------------------------------------------------------------------------------------------------------------------**

// finalize packet

static void finalize\_packet(void) {

fill in standard packet info

set transmit\_buffer[0] as start delimiter

set transmit\_buffer[1] as 0x00

set transmit\_buffer[2] as transmit length

set transmit\_buffer[transmit\_length+3] as transmit checksum

}

**--------------------------------------------------------------------------------------------------------------------------------**

**--------------------------------------------------------------------------------------------------------------------------------**

// calculate transmit checksum

static unsigned char find\_checksum(void)

{

clear total

for (i=3; i<transmit\_length+3; i++) {

add transmit\_buffer[i] to total

}

total is equal to 0xFF - total;

return total;

}

**--------------------------------------------------------------------------------------------------------------------------------**

**--------------------------------------------------------------------------------------------------------------------------------**

// decode packet

static void decode\_packet(void) {

read in api identifier

determine what type of packet was received

if packet is a transmit status

if “no ack” received

set last message received

else if packet is a receive packet

set last message received

read in source address msb

read in source address lsb

read in message identifier

if message is a handshake packet

if source address is correct

if correct handshake message was sent

reset timeout timer

update luc state machine, pass handshake received event

turn on leds

turn on audio

end if

end if

else if message is a hzv ack packet

if source address is correct

reset timeout timer

end if

end if

}

**--------------------------------------------------------------------------------------------------------------------------------**

**--------------------------------------------------------------------------------------------------------------------------------**

// reset timeout

static void reset\_timeout(unsigned char message\_received) {

program next output compare

reset overflow counter

}

**--------------------------------------------------------------------------------------------------------------------------------**

**--------------------------------------------------------------------------------------------------------------------------------**

// This timer starts the sound\_overflow counter

void init\_sound\_timer(void) {

reset sound overflows counter

}

**--------------------------------------------------------------------------------------------------------------------------------**

**--------------------------------------------------------------------------------------------------------------------------------**

// 200 ms message rate interrupt

void interrupt \_Vec\_tim0ch4 timer\_0\_ch\_4(void){

clear OC4 flag

program next output compare

set next transmit ready

increment sound timer overflows

if sound timer overflows is equal to 6

set Port U Bit 1

end if

}

// 1 sec timeout interrupt

void interrupt \_Vec\_tim0ch5 timer\_0\_ch\_5(void){

clear OC5 flag

program next output compare

if timeout is enabled

increment num overflows

else

reset num overflows

end if

}

// 20 ms analog sensor update

void interrupt \_Vec\_tim0ch6 timer\_0\_ch\_6(void){

clear OC6 flag

program next output compare

}

**--------------------------------------------------------------------------------------------------------------------------------**

luc\_state\_machine.c – e128 state machine code

**initialization:**

**--------------------------------------------------------------------------------------------------------------------------------**

**global function prototypes:**

void call\_luc\_sm(unsigned char cur\_event)

unsigned char is\_handshake\_requested(void)

unsigned char is\_luc\_connected(void)

unsigned char is\_timeout\_enabled(void)

**module level variables:**

static unsigned char handshake\_request\_status = 0

static unsigned char luc\_connect\_status = 0

static unsigned char timeout\_status = 0

**--------------------------------------------------------------------------------------------------------------------------------**

**pseudo code:**

**--------------------------------------------------------------------------------------------------------------------------------**

// state machine

void call\_luc\_sm(unsigned char cur\_event) {

if current state is waiting to contact

if current event is handshake requested

set current state to waiting for ack

enable handshake requests

enable timeout

end if

else if current state is waiting for ack

if current event is handshake received

set current state to sending commands

disable handshake requests

enable sending commands

else if current event is hzv timeout

set current state to waiting to contact

disable sending

disable timeout

end if

else if current state is sending commands

if current event is hzv timeout

set current state to waiting to contact

disable sending

disable timeout

end if

end if

}

**--------------------------------------------------------------------------------------------------------------------------------**

**--------------------------------------------------------------------------------------------------------------------------------**

// check if handshake is requested

unsigned char is\_handshake\_requested(void) {

return handshake\_request\_status;

}

**--------------------------------------------------------------------------------------------------------------------------------**

**--------------------------------------------------------------------------------------------------------------------------------**

// check if luc is connected

unsigned char is\_luc\_connected(void) {

return luc\_connect\_status;

}

**--------------------------------------------------------------------------------------------------------------------------------**

**--------------------------------------------------------------------------------------------------------------------------------**

// check if timeout is enabled

unsigned char is\_timeout\_enabled(void) {

return timeout\_status;

}

**--------------------------------------------------------------------------------------------------------------------------------**

controller\_sensors.c – e128 sensor code

**initialization:**

**--------------------------------------------------------------------------------------------------------------------------------**

**global function prototypes:**

char get\_throttle(void);

char get\_direction(void);

char get\_shoot(void);

unsigned char isButtonPressed(void);

unsigned char get\_boat\_address(void);

void setLEDDisplay(DisplayChar\_t charToDisplay);

**--------------------------------------------------------------------------------------------------------------------------------**

**pseudo code:**

**--------------------------------------------------------------------------------------------------------------------------------**

// get throttle (flex sensor)

char get\_throttle(void) {

read in speed request from Port AD Pin 0

convert from short to char

if value is less than 0, add 128

else

subtract 128

end if

return signed\_throttle

}

**--------------------------------------------------------------------------------------------------------------------------------**

**--------------------------------------------------------------------------------------------------------------------------------**

// get direction (accelerometer)

char get\_direction(void) {

read in direction request from Port AD Pin 1

convert from short to char

if value is less than 0, add 128

else

subtract 128

end if

return signed\_direction

}

**--------------------------------------------------------------------------------------------------------------------------------**

**--------------------------------------------------------------------------------------------------------------------------------**

// get shoot (button)

char get\_shoot(void) {

read in shoot request from Port T Bit 5

return shoot

}

**--------------------------------------------------------------------------------------------------------------------------------**

**--------------------------------------------------------------------------------------------------------------------------------**

// check if button pressed

unsigned char isButtonPressed(void) {

return value of Port T Bit 6

}

**--------------------------------------------------------------------------------------------------------------------------------**

**--------------------------------------------------------------------------------------------------------------------------------**

// get boat address

unsigned char get\_boat\_address(void) {

set boat value to value of Port AD Pin 2

determine which range boat value falls in

call set LED display

return boat value

}

**--------------------------------------------------------------------------------------------------------------------------------**

**--------------------------------------------------------------------------------------------------------------------------------**

// set LED number display

void setLEDDisplay(DisplayChar\_t charToDisplay) {

switch charToDisplay

case 0:

turn on segment A

turn on segment B

turn on segment C

turn on segment D

turn on segment E

turn on segment F

turn off segment G

case 1:

turn off segment A

turn on segment B

turn on segment C

turn off segment D

turn off segment E

turn off segment F

turn off segment G

case 2:

turn on segment A

turn on segment B

turn off segment C

turn on segment D

turn on segment E

turn off segment F

turn on segment G

case 3:

turn on segment A

turn on segment B

turn on segment C

turn on segment D

turn off segment E

turn off segment F

turn on segment G

case 4:

turn off segment A

turn on segment B

turn on segment C

turn off segment D

turn off segment E

turn on segment F

turn on segment G

case 5:

turn on segment A

turn off segment B

turn on segment C

turn on segment D

turn off segment E

turn on segment F

turn on segment G

case 6:

turn on segment A

turn off segment B

turn on segment C

turn on segment D

turn on segment E

turn on segment F

turn on segment G

case 7:

turn on segment A

turn on segment B

turn on segment C

turn off segment D

turn off segment E

turn off segment F

turn off segment G

case 8

turn on segment A

turn on segment B

turn on segment C

turn on segment D

turn on segment E

turn on segment F

turn on segment G

case 9:

turn on segment A

turn on segment B

turn on segment C

turn on segment D

turn off segment E

turn on segment F

turn on segment G

case A:

turn on segment A

turn on segment B

turn on segment C

turn off segment D

turn on segment E

turn on segment F

turn on segment G

case B:

turn off segment A

turn off segment B

turn on segment C

turn on segment D

turn on segment E

turn on segment F

turn on segment G

case C:

turn on segment A

turn off segment B

turn off segment C

turn on segment D

turn on segment E

turn on segment F

turn off segment G

case D:

turn off segment A

turn on segment B

turn on segment C

turn on segment D

turn on segment E

turn off segment F

turn on segment G

case E:

turn on segment A

turn off segment B

turn off segment C

turn on segment D

turn on segment E

turn on segment F

turn on segment G

case F:

turn on segment A

turn off segment B

turn off segment C

turn on segment D

turn on segment E

turn on segment F

turn on segment G

end switch

}

**--------------------------------------------------------------------------------------------------------------------------------**

controller\_actuators.c – e128 actuator code

**initialization:**

**--------------------------------------------------------------------------------------------------------------------------------**

**global function prototypes:**

void turn\_on\_leds(void);

void turn\_off\_leds(void);

void turn\_on\_audio(void);

void turn\_off\_audio(void);

**module level variables:**

static char m\_ledsOn = 0;

static char m\_audioOn = 0;

**--------------------------------------------------------------------------------------------------------------------------------**

**pseudo code:**

**--------------------------------------------------------------------------------------------------------------------------------**

// turn on leds

void turn\_on\_leds(void) {

set U0 high

}

**--------------------------------------------------------------------------------------------------------------------------------**

**--------------------------------------------------------------------------------------------------------------------------------**

// turn off leds

void turn\_off\_leds(void) {

set U0 low

}

**--------------------------------------------------------------------------------------------------------------------------------**

**--------------------------------------------------------------------------------------------------------------------------------**

// turn on audio

void turn\_on\_audio(void) {

set U1 high

}

**--------------------------------------------------------------------------------------------------------------------------------**

**--------------------------------------------------------------------------------------------------------------------------------**

// turn off audio

void turn\_off\_audio (void) {

set U1 low

}

**--------------------------------------------------------------------------------------------------------------------------------**