Juan Juvera

CS350

**Milestone 3 Code**

**#include** <stdint.h>

**#include** <stddef.h>

/\* Driver Header files \*/

**#include** <ti/drivers/GPIO.h>

/\* Driver configuration \*/

**#include** "ti\_drivers\_config.h"

**#include** <ti/drivers/Timer.h>

/\*setting TimerFlag to 0\*/

**volatile** **unsigned** **char** TimerFlag = 0;

/\*When called, it will increase the TimerFlag\*/

**void** **timerCallback**(Timer\_Handle myHandle, int\_fast16\_t status)

{

TimerFlag = 1;

}

**void** **initTimer**(**void**)

{

Timer\_Handle timer0;

Timer\_Params params;

**Timer\_init**();

**Timer\_Params\_init**(&params);

/\*Changed the period to 500000 500 ms = 500000 us\*/

params.period = 500000;

params.periodUnits = *Timer\_PERIOD\_US*;

params.timerMode = *Timer\_CONTINUOUS\_CALLBACK*;

params.timerCallback = timerCallback;

timer0 = **Timer\_open**(CONFIG\_TIMER\_0, &params);

**if** (timer0 == NULL) {

/\* Failed to initialized timer \*/

**while** (1) {}

}

**if** (**Timer\_start**(timer0) == Timer\_STATUS\_ERROR) {

/\* Failed to start timer \*/

**while** (1) {}

}

}

/\* State Machine \*/

**enum** SM\_STATES {

*SM\_Start*,

*SM\_1*,

*SM\_O*,

*SM\_2*,

*SM\_K*,

*SM\_Pause*,

} SM\_STATE;

/\* setting the ButtonClick variable \*/

**unsigned** **char** ButtonClick = 0;

/\* setting the OkOrMorse value to default to Morse \*/

**unsigned** **char** OkOrMorse = 0;

**void** **Morse\_Tick**(){

**unsigned** **char** i;

**switch**(SM\_STATE) {

**case** *SM\_Start*:

i = 0;

SM\_STATE = *SM\_1*;

**break**;

/\*Will change the state after 8 \*/

**case** *SM\_1*:

**if** (!(i < 8)){

i = 0;

SM\_STATE = *SM\_O*;

}

**break**;

/\*Will change the state after 14 \*/

**case** *SM\_O*:

**if** (!(i <14)){

i = 0;

**if** (OkOrMorse){

SM\_STATE = *SM\_K*;

}

**else**{

SM\_STATE = *SM\_2*;

}

}

**break**;

/\*Will change the state after 5 \*/

**case** *SM\_2*:

**if** (!(i <5)){

i = 0;

SM\_STATE = *SM\_Pause*;

}

**break**;

/\*Will change the state after 9 \*/

**case** *SM\_K*:

**if** (!(i <9)){

i = 0;

SM\_STATE = *SM\_Pause*;

}

**break**;

/\*Will change the state after 7 \*/

**case** *SM\_Pause*:

**if** (!(i < 7)){

i = 0;

**if** (OkOrMorse){

SM\_STATE = *SM\_O*;

}

**else** {

SM\_STATE = *SM\_1*;

}

}

**break**;

**default**:

SM\_STATE = *SM\_Start*;

**break**;

}

**switch**(SM\_STATE){

**case** *SM\_Start*:

**break**;

**case** *SM\_1*:

**if** (i == 0 || i == 2 || i ==4){

**GPIO\_write**(CONFIG\_GPIO\_LED\_0, CONFIG\_GPIO\_LED\_ON);

}

**else** {

**GPIO\_write**(CONFIG\_GPIO\_LED\_0, CONFIG\_GPIO\_LED\_OFF);

}

i++;

**break**;

**case** *SM\_O*:

**if** ((i < 3) || (i > 3 && i < 7) || (i > 7 && i < 11)) {

**GPIO\_write**(CONFIG\_GPIO\_LED\_1, CONFIG\_GPIO\_LED\_ON);

} **else** {

**GPIO\_write**(CONFIG\_GPIO\_LED\_1, CONFIG\_GPIO\_LED\_OFF);

}

i++;

**break**;

**case** *SM\_2*:

**if** (i == 0 || i == 2 || i == 4) {

**GPIO\_write**(CONFIG\_GPIO\_LED\_0, CONFIG\_GPIO\_LED\_ON);

} **else** {

**GPIO\_write**(CONFIG\_GPIO\_LED\_0, CONFIG\_GPIO\_LED\_OFF);

}

i++;

**break**;

**case** *SM\_K*:

**if** (i < 3 || i > 5) {

**GPIO\_write**(CONFIG\_GPIO\_LED\_1, CONFIG\_GPIO\_LED\_ON);

} **else** {

**GPIO\_write**(CONFIG\_GPIO\_LED\_1, CONFIG\_GPIO\_LED\_OFF);

}

**if** (i == 4) {

**GPIO\_write**(CONFIG\_GPIO\_LED\_0, CONFIG\_GPIO\_LED\_ON);

} **else** {

**GPIO\_write**(CONFIG\_GPIO\_LED\_0, CONFIG\_GPIO\_LED\_OFF);

}

i++;

**break**;

**case** *SM\_Pause*:

**if** (i == 0) {

**GPIO\_write**(CONFIG\_GPIO\_LED\_0, CONFIG\_GPIO\_LED\_OFF);

**GPIO\_write**(CONFIG\_GPIO\_LED\_1, CONFIG\_GPIO\_LED\_OFF);

} **else** **if** (i == 6) {

/\* If a button was pressed, switch morse and reset ButtonClick \*/

**if** (ButtonClick) {

OkOrMorse = !OkOrMorse;

ButtonClick = 0;

}

}

i++;

**break**;

**default**:

**break**;

}

}

/\*

\* ======== gpioButtonFxn0 ========

\* Callback function for the GPIO interrupt on CONFIG\_GPIO\_BUTTON\_0.

\*

\* Note: GPIO interrupts are cleared prior to invoking callbacks.

\*/

**void** **gpioButtonFxn0**(uint\_least8\_t index)

{

/\* Toggle an LED \*/

**GPIO\_toggle**(CONFIG\_GPIO\_LED\_0);

}

/\*

\* ======== gpioButtonFxn1 ========

\* Callback function for the GPIO interrupt on CONFIG\_GPIO\_BUTTON\_1.

\* This may not be used for all boards.

\*

\* Note: GPIO interrupts are cleared prior to invoking callbacks.

\*/

**void** **gpioButtonFxn1**(uint\_least8\_t index)

{

/\* Toggle an LED \*/

ButtonClick = 1;

}

/\*

\* ======== mainThread ========

\*/

**void** \***mainThread**(**void** \*arg0)

{

/\* Call driver init functions \*/

**GPIO\_init**();

/\* Configure the LED and button pins \*/

**GPIO\_setConfig**(CONFIG\_GPIO\_LED\_0, GPIO\_CFG\_OUT\_STD | GPIO\_CFG\_OUT\_LOW);

**GPIO\_setConfig**(CONFIG\_GPIO\_LED\_1, GPIO\_CFG\_OUT\_STD | GPIO\_CFG\_OUT\_LOW);

**GPIO\_setConfig**(CONFIG\_GPIO\_BUTTON\_0, GPIO\_CFG\_IN\_PU | GPIO\_CFG\_IN\_INT\_FALLING);

**GPIO\_setConfig**(CONFIG\_GPIO\_BUTTON\_1, GPIO\_CFG\_IN\_PU | GPIO\_CFG\_IN\_INT\_FALLING);

/\* Turn on user LED \*/

**GPIO\_write**(CONFIG\_GPIO\_LED\_0, CONFIG\_GPIO\_LED\_ON);

/\* Install Button callback \*/

**GPIO\_setCallback**(CONFIG\_GPIO\_BUTTON\_0, gpioButtonFxn0);

**GPIO\_setCallback**(CONFIG\_GPIO\_BUTTON\_1, gpioButtonFxn1);

/\* Enable interrupts \*/

**GPIO\_enableInt**(CONFIG\_GPIO\_BUTTON\_0);

**GPIO\_enableInt**(CONFIG\_GPIO\_BUTTON\_1);

/\*

\* If more than one input pin is available for your device, interrupts

\* will be enabled on CONFIG\_GPIO\_BUTTON1.

\*/

**if** (CONFIG\_GPIO\_BUTTON\_0 != CONFIG\_GPIO\_BUTTON\_1) {

/\* Configure BUTTON1 pin \*/

**GPIO\_setConfig**(CONFIG\_GPIO\_BUTTON\_1, GPIO\_CFG\_IN\_PU | GPIO\_CFG\_IN\_INT\_FALLING);

/\* Install Button callback \*/

**GPIO\_setCallback**(CONFIG\_GPIO\_BUTTON\_1, gpioButtonFxn1);

**GPIO\_enableInt**(CONFIG\_GPIO\_BUTTON\_1);

}

**GPIO\_write**(CONFIG\_GPIO\_LED\_0, CONFIG\_GPIO\_LED\_OFF);

**GPIO\_write**(CONFIG\_GPIO\_LED\_1, CONFIG\_GPIO\_LED\_OFF);

SM\_STATE = *SM\_Start*;

initTimer();

**while**(1){

Morse\_Tick();

**while**(!TimerFlag){}

TimerFlag = 0;

}

}