Εργαστηρια 2-3 - Μικροεπεξεργαστες & Περιφερειακα

Σακελλαρίου Βασίλειος - AEM:9400
Φίλης Χάρης - AEM:9449

June 2021

0.1 Lab 2 — Blinky Led - PushButton Events — HelloWorldC

0.1.1 Implementation

Αρχείο — Application.c

Στο συγκεκριμένο αρχείο υλοποιήσαμε το *callback* - ***onPushButtonChanged-Cbk()*** όπου παίρνουμε το *input* από το κουμπί στο *redBlocks* μέσω του *abstraction Layer* που έχει οριστεί(*Platform.h*) και ανάλογα με αυτό ανάβουμε (*Platform_Led_setValue(ledValue)*),σβήνουμε το led ή ενεργοποιούμε το *blinking mode* μεταβάλλοντας τα κατάλληλα *flags*.

Επίσης εδώ υλοποιούμε και την συνάρτηση ***onSystemTick()*** που καθορίζει τι κάνει το σύστημα σε κάθε "χτύπο" του εικονικού ρολογιού του *redBlocks*. Αρχικά θέτει το *msTicks = 0* και αν έχουμε *blink mode* τότε κάνουμε εναλλαγή της εξόδου του *led(ledValue = !ledValue)*. Εδώ γίνεται *count* των *Tick* του συστήματος σε περίπτωση που θέλουμε να σταματάει από μόνη της η διαδικασία του *blinking* και τότε αφαιρούμε την εντολή "' msTicks = 0 ; "' μέσα στην

```
....
    if(msTicks%150 == 0) {
        msTicks = 0; //to make sure that the led is blinking for
    ever!!! (until we hit the button;
        .....
    }
....
```

Ωστόσο εμείς έχουμε κάνει και μια άλλη υλοποίηση με *delay function* που κάνει χρήση της εντολής *__nop();*. Βέβαια αυτή θέλει έναν πολύ μεγάλο αριθμό *delayTicks* όπως ξεκαθαρίσαμε στο εργαστήριο.

Αρχείο — main.c

Εδώ υπάρχει η συνάρτηση *SysTick_Handler()* που ουσιαστικά ελέγχει το *tick* μέσω συνάρτησης που διαχειρίζεται το εικονικό ρολόι του *redBlocks*.

Εμείς υλοποιήσαμε την συνάρτηση ***SysTick_control(u8 disable)*** που δέχεται σαν όρισμα ένα uint8_t σήμα και μέσω των συναρτήσεων του *Nested Vector Interrupt Controller* διαχειρίζεται τα device specific interrupts του συστήματος ανάλογα με αυτό το σήμα.

Αλλάξαμε διάφορες μεταβλητές όπως το SYSTEM_TICKS_PER_SEC και παραμέτρους του *rb_sim_uvision_init* με αποτέλεσμα να αλλάξουμε τον χρόνο στην προσομοίωση και το *led* να αναβοσβήνει στο $\frac{1}{3}$ του δευτερολέπτου.

0.1.2 Problems

- Δυσκολευτήκαμε να καταλάβουμε αρχικά πολλά πράγματα λόγω ελλιπούς *documentation* όσον αφορά το redBlocks.
- Είχαμε το πρόβλημα με την υλοποίηση της συνάρτησης με το delay που αναφέραμε και παραπάνω (__nop();)

0.1.3 Testing

Τρέξαμε προσομοιώσεις και αλλάξαμε και τους χρόνου ρολογιού.

0.2 Lab 3 — Vending Machine - Change Return — VendingMachine

0.2.1 Implementation

Το συγκεκριμένο ήταν πάλι έτοιμο *project* του *redBlocks* το οποίο έπρεπε να το εμπλουτίσουμε και να το παραμετροποιήσουμε.

Εφόσον δεν ξέραμε πώς να προσπελάσουμε αντικείμενα του *Cpp Abstraction Layer* που επικοινωνεί με το *redBlocks* σε C , χρησιμοποιήσαμε συναρτήσεις έτοιμες του *LowLevelPlatform.h*, αλλά και κάναμε επέμβαση στην κλάση **cashBox**.Εκεί προσθέσαμε την συνάρτηση που επιστρέφει τα ρέστα. Μικρές αλλαγές έγιναν και σε άλλα αρχεία κλάσεων και βιβλιοθηκών που θα αναπτυχθούν παρακάτω (φυσικά για ότι αρχείο υλοποίησης κλάσης έχουμε αλλάξει έχουμε ορίσει κατάλληλα και την συνάρτηση στο *header file (hpp)*).

Αρχικά έπρεπε να προσθέσουμε στον *rb simulator* το κουμπί ενεργοποίησης και απενεργοποίησης του συστήματος. Εκεί μετά την προσθήκη του, το *code generator tab* μας έδωσε ορισμένες εντολές τις οποίες βάλαμε στα αρχεία LowLevelPlatform.h και PlatformCallback.h.

Αλλαγές στο αρχείο — PlatformCallback.h

```
.... Platform.h

    RB_CONNECT_ISR_CBK(Platform::DiPowerSupervision, Platform::
    DiPowerSupervision::CBK_ON_INPUT_CHANGED, /** Empty */);
...
```

όπου το *DiPowerSuppervision* είναι το αντικείμενο του κουμπιού που προσθέσαμε.

Στην συνέχεια έπρεπε να υλοποιήσουμε την έναρξη/διακοπή λειτουργίας(*start events / sleep*) του *vending machine* μέσω του *input* που έχει το πρόγραμμα από το *simulator - simulated button*.

Αλλαγές στο αρχείο — Application.cpp

```
//CHANGED THIS PART
void Application::run() const
{
  for( ;; )
  {
  if(Platform::DiPowerSupervision::getValue() == true){
     Platform::OS::processEvents();
  }else{
     Platform::enterSleepMode();

  }

  }
}
```

Η βασική συνάρτηση συνάρτηση που φτιάξαμε για την επιστροφή των χρημάτων είναι στην κλάση *cashBox* και λέγεται ***ReturnChange(u16 price)***. Πατάει στην λογική αντίστοιχης συνάρτησης που έχει το *cashBox* την *ReleaseMoney* που είναι συνάρτηση που επιστρέφει τα χρήματα σε περίπτωση που το vendingMachine δεν πετάξει το προϊόν. Απλά εδώ γίνεται ο υπολογισμός του change. Είναι αυτό που αποθηκεύεται στην μεταβλητή mMoneyInIntermediateCash (attribute της κλάσης cashBox) και είναι τα λεφτά που εν τέλη επιστρέφονται από το μηχάνημα. Επίσης τυπώνουμε και debug μήνυμα στο log για να ενημερώνεται ο χρήστης. Οι υπόλοιπες εντολές που καλούνται πριν και μετά στην συνάρτηση είναι απαραίτητες για να λειτουργήσει σωστά το platform και να μην δείξει απλά ότι επιστρέφονται χρήματα στο log αλλά να γίνει όντως αυτό στο επίπεδο προσομοίωσης*.

Αλλαγές στο αρχείο — cashBox.cpp

```
//ADDED THIS FUNCTION
void CashBox::ReturnChange(u16 price){
  if(mIsActive){
    u16 change = mMoneyInIntermediateCash - price;
    RB_LOG_DEBUG("Exchange Money is " << change << "cent");
    mMoneyInIntermediateCash = change;
    Platform::OpenCashBoxActor::setOutput(false);
    Platform::ChangeBayLamp::setOutput(true);
    mTimer.startPeriodic(Platform::OS::MilliSec<500>::value);
    mState = STATE_RELEASE_MONEY;
    //mStateCounter = 0;
  }
}
```

Αλλαγές στο αρχείο — VendingMode.cpp
Εδώ καλείται η προηγούμενη συνάρτηση σε περίπτωση που ο χρήστης δώσει περισσότερα χρήματα από ότι η τιμή του προϊόντος, κάτι που εξασφαλίζεται με κατάλληλο έλεγχο if.

```
if ( isProductValid && ( mProductSlots[mSelectedProduct].
    getPricePerItem() <= mCashBox.getMoneyInIntermediateCash() ) )
    {
      if (mProductSlots[mSelectedProduct].getPricePerItem() <
    mCashBox.getMoneyInIntermediateCash()){
        startReleaseWorkflow();
        mCashBox.ReturnChange(mProductSlots[mSelectedProduct].
    getPricePerItem()); //call of our function

      }else{
        startReleaseWorkflow();
      }
    }
    else
    {
      VendingScreen::drawPriceInfo( price, mCashBox.
    getMoneyInIntermediateCash() );
    }
  }
```

Αλλαγές στο αρχείο — VendingScreen.cpp
Εκτύπωση μηνύματος επιβεβαίωσης κατά την επιστροφή χρημάτων στην οθόνη του Vending Machine.

```
1
2 //notification message
3 void VendingScreen::drawReturningChangeMessage(){
4  drawTwoLineMessage("Your change", "is returned");
5 }
```

0.2.2 Problems

- Το project είχε ελλιπές documentation.
- Κατανοούμε ότι το redBlocks ήταν μια λύση έκτακτης ανάγκης, αλλά δημιουργήθηκαν αρκετά προβλήματα που δεν έχουν να κάνουν και με το αντικείμενο του μαθήματος.
- Δεν καταφέραμε να κάνουμε την υλοποίηση σε c όπως θέλετε εσείς αλλά δουλεύει κανονικά το πρόγραμμα.

ΤΕΛΟΣ ΑΝΑΦΟΡΑΣ