Μικροεπεξεργαστές και Περιφερειακά

Εργαστήριο 2ο & 3ο - Ομάδα 1

Ονοματεπώνυμο	ΑΕΜ	E-mail
Καλαντζής Γεώργιος	**8818**	**gkalantz@ece.auth.gr**
Κοσέογλου Σωκράτης	**8837**	**sokrkose@ece.auth.gr**

1. Σκοπός 2ης Εργασίας

Σκοπός του 2ου εργαστηρίου είναι να αναβοσβήσουμε ένα led πατώντας ένα button στο περιβάλλον RedBlocks που συνδέεται με το keil uVision 5, όπου και γίνεται η υλοποίηση του κώδικα.

2. Υλοποίηση 2ης Εργασίας

Αρχικά, η **main()** αφού κάνει κάποιες αρχικοποιήσεις, το ρολόι του συστήματος καθώς και κάθε πότε καλείτε ο *timer* που θα χρησιμοποιήσει, τρέχει την συνάρτηση **Application_run()** που περιέχει την υλοποίηση της εργασίας. Μέσα στην Application_run(), υπάρχουν τέσσερεις συναρτήσεις. Η Platform_Led_setValue(led) αρχικοποιεί το led ώστε να είναι σβηστό. Έπειτα, η **Platform_PushButton_setIsrCallback(onPushButtonChangedCbk)** δηλώνει ως *Interrupt Service Routine(ISR)* του button την ρουτίνα *onPushButtonChangedCbk()*, δηλαδή μόλις πατηθεί το κουμπί, θα τρέξει αυτή η ρουτίνα. Ακόμη, η **Platform_PushButton_enableInterrupt()** ενεργοποιεί την χρήση interrupt, ενώ τέλος η **Platform_registerSystickCallback(onSystemTick)** δηλώνει ότι κάθε φορά που κάνει expire ο timer θα εκτελείτε η ρουτίνα *onSystemTick()*. Στην συνέχεια θα δείξουμε πιο συγκεκριμένα πως υλοποιήσαμε τον κώδικα σε αυτές τις δύο ρουτίνες. Αρχικά, στην *onPushButtonChangedCbk()*, διαβάζουμε την τιμή του button, δηλαδή αν είναι πατημένο ή όχι. Σε περίπτωση που είναι πατημένο, εάν η μεταβλητή **blink** είναι διάφορη του μηδενός (δηλαδή αυτή την στιγμή το led αναβοσβήνει), τότε την κάνουμε 0 και κάνουμε 1 την μεταβλητή **led**, έτσι ώστε να κρατήσουμε αναμμένο το led. Διαφορετικά, εάν η μεταβλητή **blink** είναι 0, τότε ελέγχουμε εάν και η μεταβλητή **led** είναι διάφορη του 0. Αν ναι, τότε πηγαίνουμε στην 3η κατάσταση που είναι να είναι σβηστό το led, κάνοντας την μεταβλητή led ίση με το μηδέν. Διαφορετικά, εάν το blink είναι 0 αλλά και το led είναι 0 τότε κάνουμε το led = 1 έτσι ώστε να το ανάψουμε. Μετά την if, καλούμαι την συνάρτηση **Platform_Led_setValue(led)** για να ανάψει ή να σβήσει το led ανάλογα με την τιμή του ορίσματος. Η ρουτίνα *onSystemTick()* χρησιμοποιείται για να μπορέσουμε να αναβοσβήσουμε το led. Ελέγχει εάν η μεταβλητή **blink** είναι 1 (δηλαδή πρέπει να αναβοσβήσει το led). Αν ναι, τότε μέσω ενός δείκτη **i** εναλλάσσει την τάση στο pin που είναι συνδεδεμένο το led 500 φορές. Διαφορετικά, δεν κάνει τίποτα. Οι κώδικες αυτοί φαίνονται στις παρακάτω εικόνες.

```
static void onPushButtonChangedCbk(){
  if (Platform_PushButton_getValue() != 0){
    if (blink != 0){
      led = true;
      blink = false;
    }else{
      if (led != 0){
        led = false;
      }else{
        blink = true;
      }
    }
    Platform_Led_setValue( led );
  }
}
```

```
static void onSystemTick(){
  static unsigned int i = 0;
  if (blink != 0){
    if (0 == i % 500){
      i = 0;
      led = !led;
      Platform_Led_setValue( led );
    }
    ++i;
  }else{
    i = 0;
  }
}
```

3. Σκοπός 3ης Εργασίας

Στόχος της 3ης εργασίας ήταν η επεξεργασία ενός ήδη υπάρχοντος project του RedBlock το οποίο αφορά έναν αυτόματο πωλητή (Vending Machine). Πιο συγκεκριμένα, στόχος ήταν να προσθέσουμε ένα κουμπί το οποίο θα ενεργοποιεί τον αυτόματο πωλητή και να προσθέσουμε την λειτουργία υπολογισμού και εμφάνισης των λεφτών που θα πρέπει να επιστραφούν στον άνθρωπο που αγόρασε κάποιο αναψυκτικό.

4. Υλοποίηση 3ης Εργασίας

Αρχικά, όσον αφορά την υλοποίηση του κουμπιού ενεργοποίησης του **Vending Machine,** προστέθηκε ένα κομμάτι κώδικα στο αρχείο **main.cpp**. Αρχικά μέσα στην **main()** γίνονται κάποιες αρχικοποιήσεις και έπειτα μπαίνει σε μία while() λούπα, όπου γίνεται έλεγχος μέσω της συνάρτησης **checkButtonState()** εάν έχει πατηθεί ή όχι το κουμπί ενεργοποίησης. Η συνάρτηση αυτή υλοποιείται σε **C** μέσα στην **extern "C"**. Ένα το κουμπί έχει πατηθεί, τότε η μεταβλητή **powerOnButtonValue** γίνεται 1 και μπαίνουμε μέσα την **if** όπου και εκτελείται κανονικά ο υπόλοιπος κώδικας. Διαφορετικά, δεν μπαίνουμε ποτέ μέσα στην if και έτσι το Vending Machine δεν ενεργοποιείται ακόμα. Για να διαβάσουμε την τιμή του **button** κάναμε χρήση της συνάρτησης getValue() με την μορφή **Platform::DicustomButton::getValue()**, όπου **DicustomButton** είναι το button που βάλαμε για την ενεργοποίηση του Vending Machine. Για να εισάγουμε αυτό το button έπρεπε να το αρχικοποιήσουμε και στα αρχεία **PlatformCallbacks.h** και **LowLevelPlatform.h**.

```
77   while(1){
78     powerOnButtonValue = checkButtonState();
79     if (powerOnButtonValue != 0){
80       // Application initialization.
81       Application app;
82       // Prevent dynamic memory allocation after this point.
83       HeapManager::disableAllocation();
84       RB_LOG_DEBUG( "Application initialized, starting event processing" );
85       RB_LOG_DEBUG( "Heap memory used:" << HeapManager::getUsed() );
86       app.run();
87     }
88   }
```

```
26  extern "C"
27  {
28    int checkButtonState( void ){
29      return powerOnButtonValue = Platform::DicustomButton::getValue();
30    }
```

Όσον αφορά την υλοποίηση του 2ου ερωτήματος, αλλάξαμε τα αρχεία *VendingMode.cpp* και *VendingScreen.cpp*. Πιο συγκεκριμένα, μέσα στο αρχείο *VendingMode.cpp* υλοποιείται η ρουτίνα που πραγματοποιείται όταν πατιέται κάποιο κουμπί αναψυκτικού, δηλαδή η **onProductButtonPressedEvent().** Μέσα σε αυτή την ρουτίνα, υπάρχει μια if η οποία ελέγχει εάν τα λεφτά που έβαλε ο πελάτης είναι περισσότερα από τα λεφτά που κοστίζει το αναψυκτικό, του οποίου το κουμπί μόλις πάτησε. Αν ναι, τότε καλεί την συνάρτηση **startReleaseWorkFlow()**. Στην συνέχεια επεξεργαστήκαμε την συνάρτηση αυτή έτσι ώστε να υπολογίζουμε τα ρέστα τα οποία θα πρέπει να δίνει ο αυτόματος πωλητής στον πελάτη. Πιο συγκεκριμένα, ορίσαμε την μεταβλητή **change** η οποία κρατάει το τελικό αποτέλεσμα. Έπειτα, κάνουμε την αφαίρεση, **mCashBox.getMoneyInIntermediateCash() - mProductSlots[mSelectedProduct].getPricePerItem()**, δηλαδή αφαιρούμαι από τα λεφτά που έχει μέσα ο αυτόματος πωλητής τα λεφτά που κάνει το συγκεκριμένο προϊόν. Το αποτέλεσμα αυτής της αφαίρεσης το αποθηκεύουμε στην μεταβλητή **change**. Έπειτα, για να δείξουμε στον πελάτη τα ρέστα που θα πάρει, γράψαμε την συνάρτηση **drawChangeInfo()**, στο αρχείο *VendingScreen.cpp*. Την συνάρτηση αυτή την καλούμε αμέσως μόλις κάνουμε την παραπάνω αφαίρεση, και της δίνουμε ως ορίσματα, τα ρέστα καθώς και το κόστος του προϊόντος. Η υλοποίηση της συνάρτησης αυτής φαίνεται παρακάτω. Αρχικά, εμφανίζουμε στην οθόνη τα ρέστα και το κόστος, έπειτα για να μπορέσουμε να ελέγξουμε την ορθότητα του προγράμματος μας, φτιάξαμε μια custom delay function, δηλαδή μια εμφωλευμένη for λούπα. Ενώ τέλος, εμφανίζουμε το μήνυμα «Your product is released», μέσω της συνάρτησης **drawReleaseMessage()**.

```
223  void VendingMode::Inner::startReleaseWorkflow()
224  {
225    mSlotActors[mSelectedProduct].switchLamp( false );
226    VendingScreen::drawReleaseMessage();
227
228    ul6 change = 0;
229    change = mCashBox.getMoneyInIntermediateCash() - mProductSlots[mSelectedProduct].getPricePerItem();
230    VendingScreen::drawChangeInfo( mProductSlots[mSelectedProduct].getPricePerItem(), change );
231
232    mReleaseLightBarrierEventCnt = 0;
233    mVendingState = STATE_FLAP_OPEN;
234    mReleaseStateCounter = 0;
235    mSlotActors[mSelectedProduct].switchFlap( true );
236    mTimer.start( Platform::OS::Sec<1>::value );
237  }
238
```

```
130  void VendingScreen::drawReleaseMessage()
131  {
132    drawTwoLineMessage( "Your product", "is released!" );
133  }
```

```
84   void VendingScreen::drawChangeInfo( ul6 price, ul6 change )
85   {
86     const Gui::CoordinateType left = 15;
87     const Gui::CoordinateType right = Gui::Display::getWidth() - 15;
88     const Gui::CoordinateType firstLine = 5 + ( Gui::Display::getHeight() - 10 ) / 4;
89     const Gui::CoordinateType secondLine = 5 + ( ( Gui::Display::getHeight() - 10 ) / 4 ) * 3;
90     Gui::clear( GuiColorMap::BACKGROUND );
91     Gui::paint(
92       GuiResources::getInstanceRef().fontArial20.text( "Price:" ),
93       Gui::LeftAlign( left - 5 ),
94       Gui::CenterVertical( firstLine ),
95       GuiColorMap::FOREGROUND
96     );
97     drawMoneyValue( GuiResources::getInstanceRef().fontArial20, right, firstLine, price );
98     Gui::paint(
99       GuiResources::getInstanceRef().fontArial20.text( "Change:" ),
100      Gui::LeftAlign( left - 5 ),
101      Gui::CenterVertical( secondLine ),
102      GuiColorMap::FOREGROUND
103    );
104    drawMoneyValue( GuiResources::getInstanceRef().fontArial20, right, secondLine, change );
105    ul6 k = 0;
106    for(ul6 i = 0; i < 10000; i++){
107      for(ul6 j = 0; j < 1000; j++){
108        k++;
109      }
110    }
111    Gui::update();
112    VendingScreen::drawReleaseMessage();
113  }
```

5. Αποτελέσματα