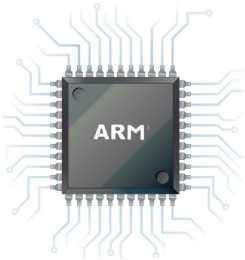


Μικροεπεξεργαστές και Περιφερειακά
-Εργαστήριο 3-

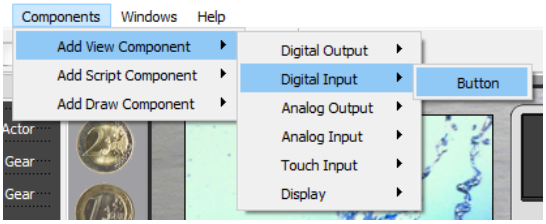
Αλέξανδρος Πετρίδης 9288
Αλέξανδρος Οικονόμου 9260
Ομάδα 35

Τελευταία ενημέρωση: 4 Ιουνίου 2021

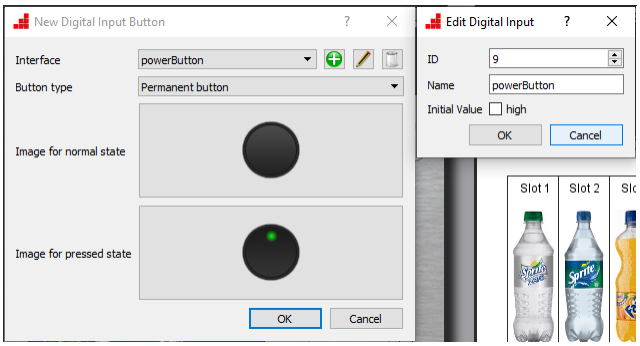
1 Κουμπί εκκίνησης του αυτόματου πωλητή

Για την υλοποίηση ενός διακόπτη ακολουθήσαμε τα παρακάτω βήματα:

1. Προσθήκη νέου κουμπιού στον RedBlocks Simulator.
2. Ορισμός του κουμπιού ως Permanent Button(Μόνιμου κουμπιού) στον Προσομοιωτή και μέσα στον κώδικα του αυτόματου πωλητή, το οποίο θα είναι με Initial Value low, όταν πατιέται θα γίνεται high και θα ακούμε στο trigger edge high, δηλαδή πότε από low γίνεται high για να δημιουργηθεί το interrupt.
3. Ορίσαμε τη συνάρτηση getStarted(), η οποία είναι η interrupt service συνάρτηση που θα καλείται όταν πατηθεί το κουμπί εκκίνησης.
4. Τέλος, έχει γίνει η προσθήκη κατάλληλου κώδικα στην main έτσι ώστε να αρχικοποιείται και να ξεκινάει ο αυτόματος πωλητής μετά το πάτημα του κουμπιού εκκίνησης. Στην ουσία η προσθήκη δηλαδή της while και της if(getStarted()) .



Σχήμα 1: Προσθήκη νέου κουμπιού



(α΄) Ορισμός του κουμπιού στον Προσομοιωτή

```
// EXTRA
typedef redBlocks::HAL::Drivers::TDigitalInputDriverStub<9> DipowerButton;
//////////////////////********************////////////////////////////////
// EXTRA
SimulationStubs::DigitalInput::add<DipowerButton>(SimulationStubs::DigitalInput::TRIGGER_EDGE_RISING);
//////////////////////********************////////////////////////////////
```

(β΄) Ορισμός του κουμπιού στον κώδικα του Προσομοιωτή στην βιβλιοθήκη LowLevelPlatform.h

Σχήμα 2: Ορισμός κουμπιού

```
//EXTRA
RB_CONNECT_ISR_CBK(Platform::DipowerButton, Platform::DipowerButton::CBK_ON_INPUT_CHANGED, getStarted() );
//////////////////////********************////////////////////////////////
```

(α΄) Ορισμός της interrupt service routine στην βιβλιοθήκη PlatformCallback.h

```
extern "C"
{
  int getStarted()
  {
    volatile int on = Platform::DipowerButton::getValue();
    if(on)
      return 1;
    return 0;
  }
}
```

(β΄) Υλοποίηση της ρουτίνας στην βιβλιοθήκη PlatformCallback.h

Σχήμα 3: Ορισμός getStarted()

```
while(1)
{
  if(getStarted())
  {
    Application app;
    // Prevent dynamic memory allocation after this point.
    HeapManager::disableAllocation();

    RB_LOG_DEBUG( "Application initialized, starting event processing" );
    RB_LOG_DEBUG( "Heap memory used:" << HeapManager::getUsed() );

    app.run();
  }
}
```

Σχήμα 4: Τροποποίημένο μέρος της main

2 Συνάρτηση Υπολογισμού χρημάτων επιστροφής

Η παρακάτω συνάρτηση υπολογίζει τα χρήματα που πρέπει να επιστραφούν στον χρήστη.
Η συνάρτηση καλείται μέσα στην συνάρτηση startReleaseFlow(), η οποία καλείται κάθε φορά που ολοκληρώνεται μια συναλλαγή. Η τροποποίηση της συνάρτησης εμφανίζει στα logs το ποσό που επιστρέφεται. Ακόμα καλείται η συνάρτηση drawChange(), η οποία είναι μια δική μας υλοποίηση για να απεικονίζεται το ποσό που επιστρέφεται στην οθόνη του αυτόματου πωλητή. Τέλος, επιστρέφεται το ποσό που αναγράφηκε στον χρήστη μέσω της συνάρτησης releaseMoney().

```
/* EXTRA */
#include <Log.h>
extern "C"
{
  u16 calculateChange(u16 money, u16 price)
  {
    return money-price;
  }
}
////////////////////*********////////////////////
```

Σχήμα 5: Υλοποίηση Συνάρτησης στο αρχείο VendingMode.cpp

```
void VendingMode::Inner::startReleaseWorkflow()
{
  mSlotActors[mSelectedProduct].switchLamp( false );
  VendingScreen::drawReleaseMessage();

  /* EXTRA */
  RB_LOG_DEBUG("Change: " << calculateChange(mCashBox.getMoneyInIntermediateCash(),mProductSlots[mSelectedProduct].getPricePerItem()));
  VendingScreen::drawChange(calculateChange(mCashBox.getMoneyInIntermediateCash(),mProductSlots[mSelectedProduct].getPricePerItem()));
  mCashBox.releaseMoney();
  ////////////////////*********////////////////////////////
  mReleaseLightBarrierEventCnt = 0;
  mVendingState = STATE_FLAP_OPEN;
  mReleaseStateCounter = 0;
  mSlotActors[mSelectedProduct].switchFlap( true );
  mTimer.start( Platform::OS::Sec<1>::value );
}
```

Σχήμα 6: Τροποποιημένη συνάρτηση startReleaseWorkflow()

```
/* EXTRA */
static void drawChange( u16 change);
////////////////////*********////////////////////////////
```

(α′) Ορισμός στην βιβλιοθήκη VendingMode.h

```
/* EXTRA  */
void VendingScreen::drawChange(u16 change)
{
  const Gui::CoordinateType left = 15;
  const Gui::CoordinateType right = Gui::Display::getWidth() - 15;
  const Gui::CoordinateType firstLine = 5 + ( Gui::Display::getHeight() - 10 ) / 4;
  const Gui::CoordinateType secondLine = 5 + ( ( Gui::Display::getHeight() - 10 ) / 4 ) * 3;
  Gui::clear( GuiColorMap::BACKGROUND );

  Gui::paint(
    GuiResources::getInstanceRef().fontArial20.text( "Change:" ),
    Gui::LeftAlign( left - 5 ),
    Gui::CenterVertical( firstLine ),
    GuiColorMap::FOREGROUND
  );
  drawMoneyValue( GuiResources::getInstanceRef().fontArial20, right, secondLine, change );
  Gui::update();

  int delay = 0;
  for(int i=0; i<10000; i++)
    for(int j=0; j<1000; j++)
      delay++;

  VendingScreen::drawReturningMoneyMessage();
}
////////////////////*********////////////////////////////
```

(β′) Υλοποίηση της συνάρτησης στο αρχείο VendingMode.cpp

Σχήμα 7: Συνάρτηση drawChange()

3 Παρατηρήσεις Εργαστηρίου

Στην εξέταση του εργαστηρίου μας δώθηκαν περαιτέρω συμπληρωματικές διευκρινίσεις. Δεν είχαμε την ISR στο πάτημα του κουμπιού εκκίνησης. Στην $main$ μας είχαμε κατευθείαν τον έλενχο παίρνοντας την τιμή από την συνάρτηση $getValue()$ του κουμπιού.