//MAIN PROGRAM

main(){

 Read\_ConfigurationFile() //Determines type and number of hardware elements

 Initialize\_Hardware() //Prepare GUI according hardware that is present

 Process\_Events() //Execute commands triggered by controller or interface

 //as in an infinite loop

 Close() //Free allocated resources

}

//INITIALIZATION PROCEDURE

initialize\_hardware(){

 //Some hardware requires a connection first

 Connect\_Joystick() //Connect to controller and start state polling loop

 Connect\_Manipulators() //Connect to manipulators and start position readout loop

 Connect\_PressureController() //Connect to pressure controller and start readout loop

 Connect\_Amplifiers() //Connect to amplifiers

 Connect\_VideoAcquisition() //Connect to video acquisition card

 //Some hardware requires setting initial values properly

 Initialize\_PressureController() //Adjust settings for pressure controller operation

 Initialize\_Amplifiers() //Set amplifiers to initial configuration VC, no holding

 Initialize\_Oscilloscopes() //Adjust oscilloscope signal scale and position

 Initialize\_VideoAcquisition() //Start video passthru operation with overlay loop

}

//PRESSURE CONTROL LOOP

PressuretTimer\_Tick(){ //Executed at regular intervals

 if( pressureSystemConnected AND pressureControlEnabled){

 if( positiveBufferPressure < reference\_value){

 CLOSE allValves

 OPEN positivePumpValve

 WAIT 200ms

 CLOSE allValves

 }

 if( negativeBufferPressure > -reference\_value){

 CLOSE allValves

 OPEN negativePumpValve

 WAIT 200ms

 CLOSE allValves

 }

 for ( i = 1 : numberOfChannels ){

 READ pressureValue

 if ( channel[i].pressure < reference\_value ){

 CLOSE allValves

 OPEN positiveBufferValve

 OPEN channel[i].valve

 WAIT 200ms

 CLOSE allValves

 }

 }

 }

}

//VIDEO DISPLAY LOOP

VideoTimer\_tick(){

 if( videoConnected AND passthruStarted ){

 Create\_graphics()

 if(passthrumode & 128) DrawText()

 if(passthrumode & 64) DrawTip()

 if(passthrumode & 32) DrawCoordinates()

 if(passthrumode & 8) DrawImage()

 if(passthrumode & 4) DrawTraces()

 if(passthrumode & 2) DrawCells()

 if(passthrumode & 1) DrawScale()

 Overlay\_graphics() //draw elements overlaid on video

 }

}

//JOYSTICK COMMANDS LOOP

ControllerTimer\_Tick(){

 joystickPreviousState = joystickState

 Acquire\_JoystickState()

 buttonflag = 0

 if( ButtonPressed(BUTTON\_A)){

 buttonflag = 1

 if( ButtonRaised(BUTTON\_R1)){

 Microscope.Recede()

//Go to position [0,0,7000] in Microscope Coordinates

 }

 if( ButtonRaised(BUTTON\_R2)){

 Microscope.GoToSearchPosition()

//Go to position [0,0,2000] in Microscope Coordinates

 }

 if( ButtonRaised(BUTTON\_L2)){

 Pipette[selected].GoToSearchPosition()

//Go to position [0,0,2000] in Microscope Coordinates

 }

 if( ButtonRaised(BUTTON\_L1)){

 Pipette[selected].GoToSafePosition()

//Go to position [0,0,0] in Microscope Coordinates receding x-axis 6mm

 }

 }

 if( ButtonPressed(BUTTON\_C)){

 buttonflag = 3

 if( ButtonRaised(BUTTON\_Z)){

//buttons C + Z

 Pipette[selected].QueryPosition()

 Pipette[selected].SaveFoundPosition()

 Pipette[selected].SaveTipImage()

 Pipette[selected].AdjustOffset()

 }

 if( ButtonRaised(BUTTON\_Y)){

//buttons C + Y

 Pipette[selected].SaveAssignedCellPosition()

 }

 if(ButtonRaised(BUTTON\_L1)){

//Move pipette to the centre of the screen

 Pipette[selected].MoveToCentreOfScreen()

 }

 if(ButtonRaised(BUTTON\_L2)){

//Approach Pipette towards cell , microscope follows

 Pipette[selected].ApproachCell()

 }

 if(ButtonRaised(BUTTON\_R2)){

//Move microscope to active cell

 Microscope.MoveToCell()

 }

 if(ButtonRaised(BUTTON\_R1)){

//Move microscope to active pipette

 Microscope.MoveToPipette(selected)

 }

 if(ButtonRaised(BUTTON\_B)){

//Buttons C + B ->Locate tip on screen

 Pipette[selected].LocateOnScreen()

 }

 if(ButtonRaised(BUTTON\_B)){

 buttonflag = 2

 for(i = 1:numberOfPipettes){

 Pipettes[i].Stop();

 }

 }

 if(ButtonPressed(BUTTON\_Z)){

 buttonflag = 6;

 if(ButtonRaised(BUTTON\_START)){

 Pipette[selected].TogglePressureControl();

 }

 if(ButtonRaised(BUTTON\_X)){

 Pipette[selected].ReleasePressure();

 }

 if(ButtonRaised(BUTTON\_Y)){

 Pipette[selected].ApplyNegativePressure();

 }

 if(ButtonRaised(BUTTON\_X)){

 buttonflag = 4

 if(Pipette[selected].GetMode() != VC){

 Pipette[selected].SetMode(VC)

Pipette[selected].SetOscilloscopeScale(0.5);

 if(Pipette[selected].GetHoldingEnable() == TRUE){

 Pipette[selected].SetOscilloscopeCoupling(AC)

 }else{

 Pipette[selected].SetOscilloscopeCoupling(DC)

 }

 }

 }

 if(ButtonPressed(BUTTON\_X)){

 buttonflag = 4

 if(ButtonRaised(BUTTON\_L1)){

 Pipette[selected].ToggleSealTest()

 }

 if(ButtonRaised(BUTTON\_L2)){

 Pipette[selected].ToggleHoldingPotential()

 }

 if(ButtonRaised(BUTTON\_R1)){

 Pipette[selected].DisableSealTest()

 Pipette[selected].AutoPipetteOffset();

 Pipette[selected].SetOsciloscopeCoupling(DC)

 }

 if(ButtonRaised(BUTTON\_R2)){

 Pipette[selected].Zap();

 }

 if(POVDownPressed()){

 Pipette[selected].DecreaseOffset();

 }

 if(POVUpPressed()){

 Pipette[selected].IncreaseOffset();

 }

 }

 if(ButtonRaised(BUTTON\_Y)){

 buttonflag = 5

 if(Pipette[selected].GetMode() != IC){

 Pipette[selected].SetMode(IC)

 Pipette[selected].SetHoldingEnable(false);

 Pipette[selected].SetHolding(0);

 Pipette[selected].SetOscilloscopeCoupling(AC);

 Pipette[selected].SetOscilloscopeScale(0.02);

 }

 }

 if(ButtonPressed(BUTTON\_Y)){

 buttonflag = 5

 if(POVDownPressed()){

 Pipette[selected].DecreaseHoldingCurrent();

 }

 if(POVUpPressed()){

 Pipette[selected].IncreaseHoldingCurrent();

 }

 if(POVRightPressed()){

 Pipette[selected].IncreaseBridgeBalResist();

 }

 if(POVLeftPressed()){

 Pipette[selected].DecreaseBridgeBalResist();

 }

 if(ButtonRaised(BUTTON\_L1)){

 Pipette[selected].ToggleSetSealTest();

 }

 if(ButtonRaised(BUTTON\_L2)){

 Pipette[selected].ToggleSlowCurrentInjEnable();

 }

 if(ButtonRaised(BUTTON\_R1)){

 Pipette[selected].AutoPipetteOffset();

 }

 }

 if (buttonflag != 0) return

 if( LeftStickLeftRaised()){

 DecreaseSpeed

 }

 if( LeftStickRightRaised()){

 IncreaseSpeed

 }

 if( LeftStickUpRaised() ){

 selected = Selected + 1

 If( selected > NumberOfPipettes){

 selected = 1

 }

 }

 if( LeftStickDownRaised() ){

 selected = Selected - 1

 If( selected < 1){

 selected = NumberOfPipettes

 }

 }

 if( RightStickLeftRaised()){

 Microscope.MoveLeft()

 }

 if( RightStickRightRaised()){

 Microscope.MoveRight

 }

 if( RightStickUpRaised() ){

 Microscope.MoveUp()

 }

 if( RightStickDownRaised() ){

 Microscope.MovveDown()

 }

 if( RightStickLeftReleased()){

 Microscope.Stop()

 }

 if( RightStickRightReleased()){

 Microscope.Stop()

 }

 if( RightStickUpReleased() ){

 Microscope.Stop()

 }

 if( RightStickDownReleased() ){

 Microscope.Stop()

 }

 if(ButtonReleased(BUTTON\_R1)) Microscope.Stop();

 if(ButtonReleased(BUTTON\_R2)) Microscope.Stop();

 if(ButtonReleased(BUTTON\_L1)) Pipette[selected].Stop();

 if(ButtonReleased(BUTTON\_L2)) Pipette[selected].Stop();

 if(ButtonRaised(BUTTON\_L2)){

 Pipette[selected].MoveUpZ();

 }

 if(ButtonRaised(BUTTON\_L1)){

 Pipette[selected].MoveDownZ();

 }

 if(ButtonRaised(BUTTON\_R2)){

 Microscope.MoveUpZ();

 }

 if(ButtonRaised(BUTTON\_R1)){

 Microscope.MoveDownZ();

 }

 switch(joystickState.POV){

 case -1:

 if(joystickPreviousState.POV != -1){

 Pipette[selected].Stop();

 }

 break;

 case POV\_RIGHT:

 if(joystickState.POV != POV\_RIGHT){

 Pipette[selected].MoveRight();

 }

 break;

 case POV\_LEFT:

 if(joystickState.POV != POV\_LEFT){

 Pipette[selected].MoveLeft();

 }

 break;

 case POV\_UP:

 if(joystickState.POV != POV\_UP){

 Pipette[selected].MoveUp();

 }

 break;

 case POV\_DOWN:

 if(joystickState.POV != POV\_DOWN){

 Pipette[selected].MoveDown();

 }

 break;

 }

 }

}