Wake On Proximity

MSP430

CapTIvate

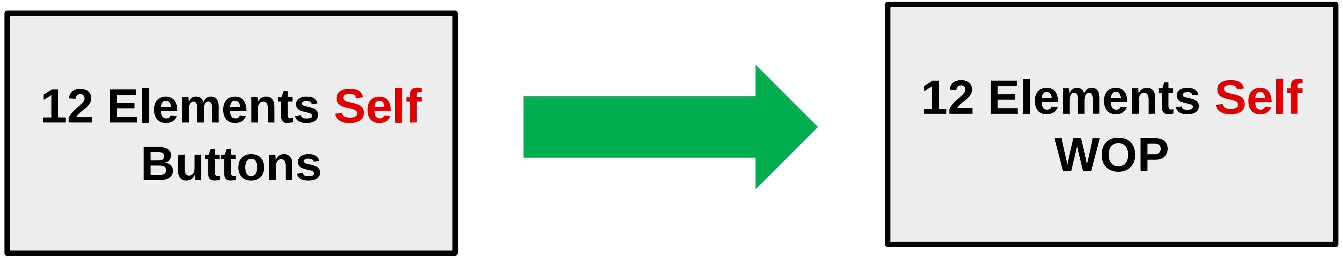
March, 2019

CapTIvate Library WOP Limitation

- With the current library, user can only use WOP function with one time cycle.
- The FSM can scan all 4 blocks without waking up the CPU.
- Changing time cycle will require waking up the CPU.

Use Cases 1

- **Self mode elements group together as one self WOP sensor**



12 Elements Self Buttons

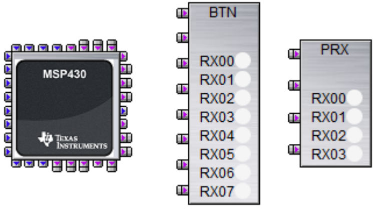
12 Elements Self WOP

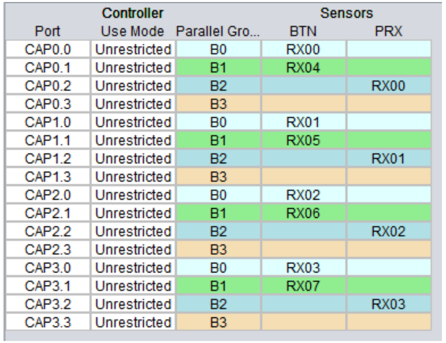
- **Example Code**

Try_WOP_SelfMode_allSelf.zip

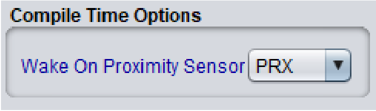
Implementation Example

- **Hardware:** CAPTIVATE-BSWP board
- **Goal:** wake up from all 8 buttons.
- **Step 1:** Create a 8 buttons sensor (BTN) and an virtual sensor (PRX) as wake-on-proximity sensor
- **Step 2:** Since the BTN sensor uses all 4 blocks, make sure the PRX sensor also has 4 elements use 4 blocks.
  It is important to make sure the PRX sensor uses the same blocks as the BTN sensor for calibration purpose. If the BTN sensor only uses 3 blocks then the PRX sensor only needs 3 elements for 3 blocks.





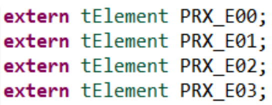
	Controller		Sensors	
Port	Use Mode	Parallel Gro...	BTN	PRX
CAP0.0	Unrestricted	B0	RX00	
CAP0.1	Unrestricted	B1	RX04	
CAP0.2	Unrestricted	B2		RX00
CAP0.3	Unrestricted	B3		
CAP1.0	Unrestricted	B0	RX01	
CAP1.1	Unrestricted	B1	RX05	
CAP1.2	Unrestricted	B2		RX01
CAP1.3	Unrestricted	B3		
CAP2.0	Unrestricted	B0	RX02	
CAP2.1	Unrestricted	B1	RX06	
CAP2.2	Unrestricted	B2		RX02
CAP2.3	Unrestricted	B3		
CAP3.0	Unrestricted	B0	RX03	
CAP3.1	Unrestricted	B1	RX07	
CAP3.2	Unrestricted	B2		RX03
CAP3.3	Unrestricted	B3		



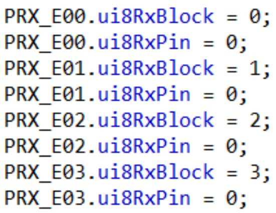
Compile Time Options

Wake On Proximity Sensor: PRX ▼

- **Step 3:** In main.c, change all the PRX sensor CAP IO assignment to match with the BTN sensor.
  PRX sensor is a virtual sensor for BTN sensor so we need to make sure PRX sensor only assign elements to BTN sensor CAP IOs. Currently CDC does not allow you to assign 2 RX to the same CAP IO in self mode, we will need to add this feature.



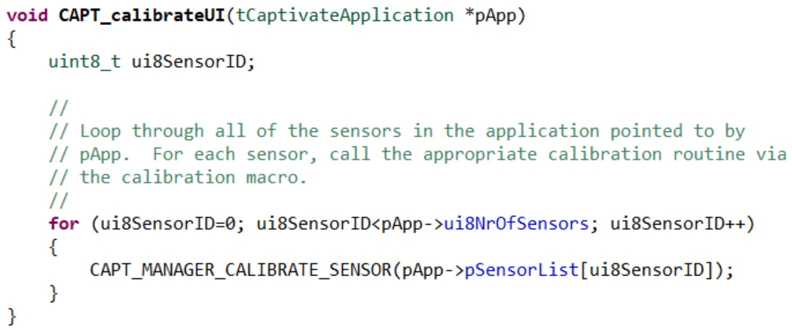
```
extern tElement PRX_E00;
extern tElement PRX_E01;
extern tElement PRX_E02;
extern tElement PRX_E03;
```



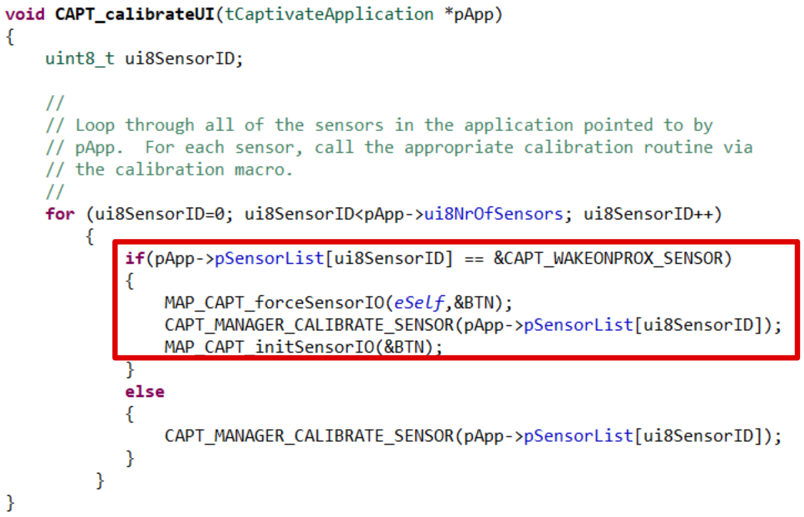
```
PRX_E00.ui8RxBlock = 0;
PRX_E00.ui8RxPin = 0;
PRX_E01.ui8RxBlock = 1;
PRX_E01.ui8RxPin = 0;
PRX_E02.ui8RxBlock = 2;
PRX_E02.ui8RxPin = 0;
PRX_E03.ui8RxBlock = 3;
PRX_E03.ui8RxPin = 0;
```

Implementation Example

- **Step 4:** Make sure to enable and activate all the element for BTN sensor when calibrating the PRX sensor since we are using PRX sensor as a virtual sensor for BTN sensor. In CAPT_Manager.c file CAPT_calibrateUI function.



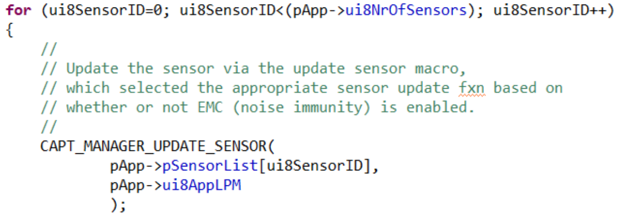
```
void CAPT_calibrateUI(tCaptivateApplication *pApp)
{
    uint8_t ui8SensorID;

    //
    // Loop through all of the sensors in the application pointed to by
    // pApp.  For each sensor, call the appropriate calibration routine via
    // the calibration macro.
    //
    for (ui8SensorID=0; ui8SensorID<pApp->ui8NrOfSensors; ui8SensorID++)
    {
        CAPT_MANAGER_CALIBRATE_SENSOR(pApp->pSensorList[ui8SensorID]);
    }
}
```



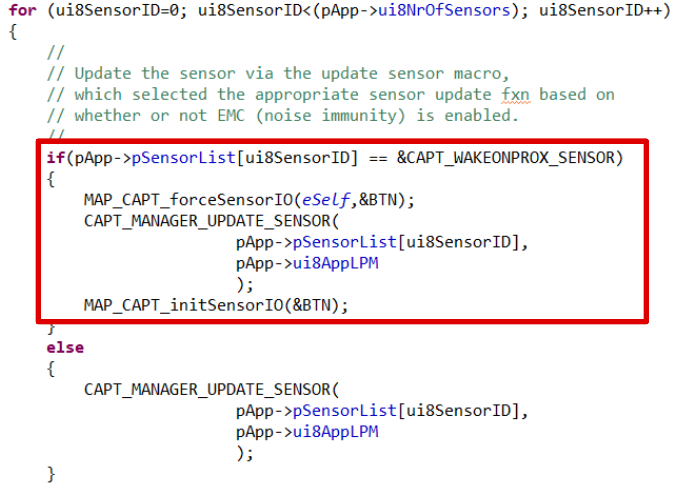
```
void CAPT_calibrateUI(tCaptivateApplication *pApp)
{
    uint8_t ui8SensorID;

    //
    // Loop through all of the sensors in the application pointed to by
    // pApp.  For each sensor, call the appropriate calibration routine via
    // the calibration macro.
    //
    for (ui8SensorID=0; ui8SensorID<pApp->ui8NrOfSensors; ui8SensorID++)
        {
          if(pApp->pSensorList[ui8SensorID] == &CAPT_WAKEONPROX_SENSOR)
          {
              MAP_CAPT_forceSensorIO(eSelf,&BTN);
              CAPT_MANAGER_CALIBRATE_SENSOR(pApp->pSensorList[ui8SensorID]);
              MAP_CAPT_initSensorIO(&BTN);
          }
          else
          {
              CAPT_MANAGER_CALIBRATE_SENSOR(pApp->pSensorList[ui8SensorID]);
          }
        }
}
```

Implementation Example

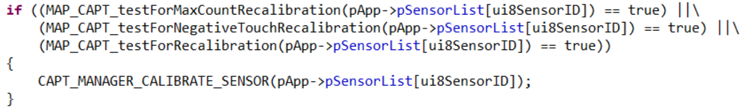
- **Step 5:** Make sure to enable and activate all the element for BTN sensor when update the PRX sensor since we are using PRX sensor as a virtual sensor for BTN sensor. In CAPT_Manager.c file CAPT_updateUI function.



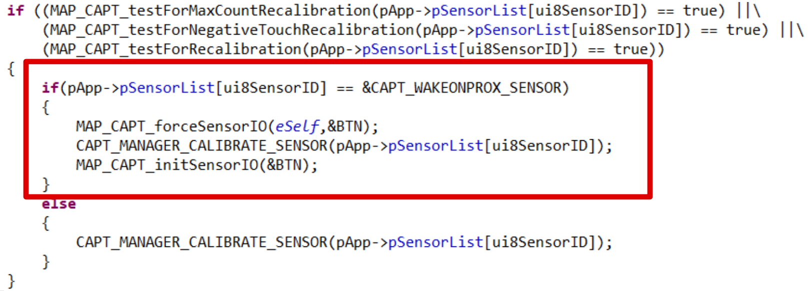
```
for (ui8SensorID=0; ui8SensorID<(pApp->ui8NrOfSensors); ui8SensorID++)
{
    //
    // Update the sensor via the update sensor macro,
    // which selected the appropriate sensor update fxn based on
    // whether or not EMC (noise immunity) is enabled.
    //
    CAPT_MANAGER_UPDATE_SENSOR(
            pApp->pSensorList[ui8SensorID],
            pApp->ui8AppLPM
            );
```



```
for (ui8SensorID=0; ui8SensorID<(pApp->ui8NrOfSensors); ui8SensorID++)
{
    //
    // Update the sensor via the update sensor macro,
    // which selected the appropriate sensor update fxn based on
    // whether or not EMC (noise immunity) is enabled.
    //
    if(pApp->pSensorList[ui8SensorID] == &CAPT_WAKEONPROX_SENSOR)
    {
        MAP_CAPT_forceSensorIO(eSelf,&BTN);
        CAPT_MANAGER_UPDATE_SENSOR(
                pApp->pSensorList[ui8SensorID],
                pApp->ui8AppLPM
                );
        MAP_CAPT_initSensorIO(&BTN);
    }
    else
    {
        CAPT_MANAGER_UPDATE_SENSOR(
                pApp->pSensorList[ui8SensorID],
                pApp->ui8AppLPM
                );
    }
```



```
if ((MAP_CAPT_testForMaxCountRecalibration(pApp->pSensorList[ui8SensorID]) == true) ||\
    (MAP_CAPT_testForNegativeTouchRecalibration(pApp->pSensorList[ui8SensorID]) == true) ||\
    (MAP_CAPT_testForRecalibration(pApp->pSensorList[ui8SensorID]) == true))
{
    CAPT_MANAGER_CALIBRATE_SENSOR(pApp->pSensorList[ui8SensorID]);
}
```

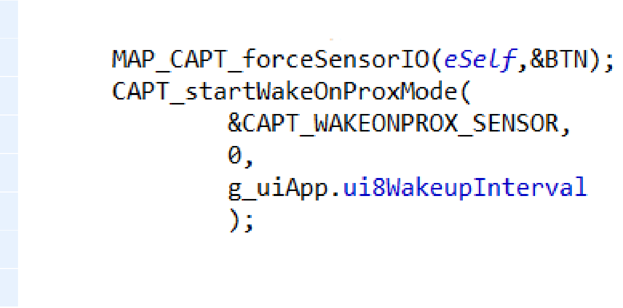


```
if ((MAP_CAPT_testForMaxCountRecalibration(pApp->pSensorList[ui8SensorID]) == true) ||\
    (MAP_CAPT_testForNegativeTouchRecalibration(pApp->pSensorList[ui8SensorID]) == true) ||\
    (MAP_CAPT_testForRecalibration(pApp->pSensorList[ui8SensorID]) == true))
{
    if(pApp->pSensorList[ui8SensorID] == &CAPT_WAKEONPROX_SENSOR)
    {
        MAP_CAPT_forceSensorIO(eSelf,&BTN);
        CAPT_MANAGER_CALIBRATE_SENSOR(pApp->pSensorList[ui8SensorID]);
        MAP_CAPT_initSensorIO(&BTN);
    }
    else
    {
        CAPT_MANAGER_CALIBRATE_SENSOR(pApp->pSensorList[ui8SensorID]);
    }
}
```

Implementation Example

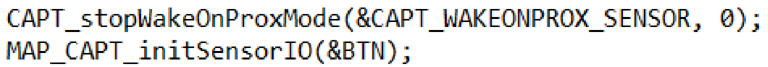
- **Step 6:** In CAPT_App.c, make sure to enable and activate all the BTN sensor elements before starting wake on proximity mode and deactivate the BTN sensor after stopping the wake on proximity mode.

  To enable and activate all the BTN sensor elements: `MAP_CAPT_forceSensorIO(eSelf,&BTN);`



```
MAP_CAPT_forceSensorIO(eSelf,&BTN);
CAPT_startWakeOnProxMode(
        &CAPT_WAKEONPROX_SENSOR,
        0,
        g_uiApp.ui8WakeupInterval
        );
```

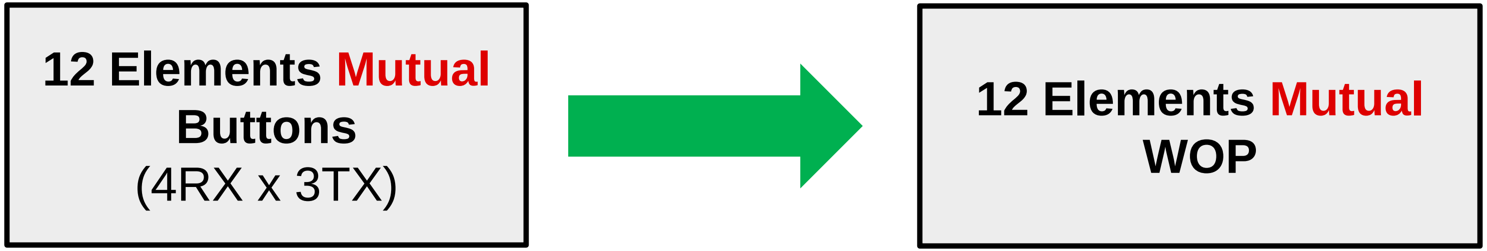
  To deactivate the BTN sensor: `MAP_CAPT_initSensorIO(&BTN);`



```
CAPT_stopWakeOnProxMode(&CAPT_WAKEONPROX_SENSOR, 0);
MAP_CAPT_initSensorIO(&BTN);
```

Use Cases 2

- **Mutual mode elements group together as one mutual WOP sensor**



12 Elements Mutual Buttons
(4RX x 3TX)

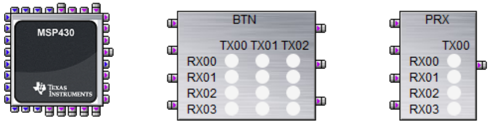
12 Elements Mutual WOP

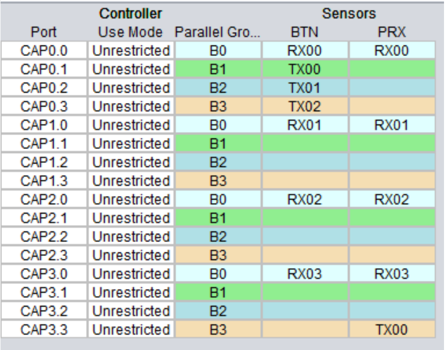
- **Example Code**

Try_WOP_MutualMode_allMutual.zip

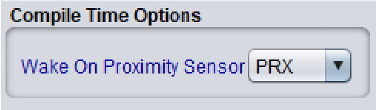
Implementation Example

- **Hardware:** CAPTIVATE-PHONE board
- **Goal:** wake up from all 12 buttons.
- **Step 1:** Create a 12 buttons sensor (BTN) and an virtual sensor (PRX) as wake-on-proximity sensor
- **Step 2:** Since the BTN sensor uses all 4 blocks, make sure the PRX sensor also has 4 elements use 4 blocks.
  It is important to make sure the PRX sensor uses the same blocks as the BTN sensor for calibration purpose. If the BTN sensor only uses 3 blocks then the PRX sensor only needs 3 elements for 3 blocks.





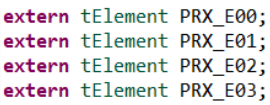
	Controller		Sensors	
Port	Use Mode	Parallel Gro...	BTN	PRX
CAP0.0	Unrestricted	B0	RX00	RX00
CAP0.1	Unrestricted	B1	TX00	
CAP0.2	Unrestricted	B2	TX01	
CAP0.3	Unrestricted	B3	TX02	
CAP1.0	Unrestricted	B0	RX01	RX01
CAP1.1	Unrestricted	B1		
CAP1.2	Unrestricted	B2		
CAP1.3	Unrestricted	B3		
CAP2.0	Unrestricted	B0	RX02	RX02
CAP2.1	Unrestricted	B1		
CAP2.2	Unrestricted	B2		
CAP2.3	Unrestricted	B3		
CAP3.0	Unrestricted	B0	RX03	RX03
CAP3.1	Unrestricted	B1		
CAP3.2	Unrestricted	B2		
CAP3.3	Unrestricted	B3		TX00



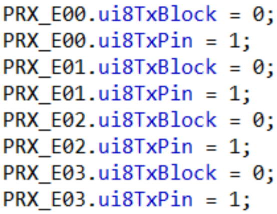
Compile Time Options

Wake On Proximity Sensor: PRX ▼

- **Step 3:** In main.c, change all the PRX sensor CAP IO assignment to match with the BTN sensor.
  PRX sensor is a virtual sensor for BTN sensor so we need to make sure PRX sensor only assign elements to BTN sensor CAP IOs. Currently CDC does not allow you to assign 2 TX to the same CAP IO in mutual mode, we will need to add this feature.



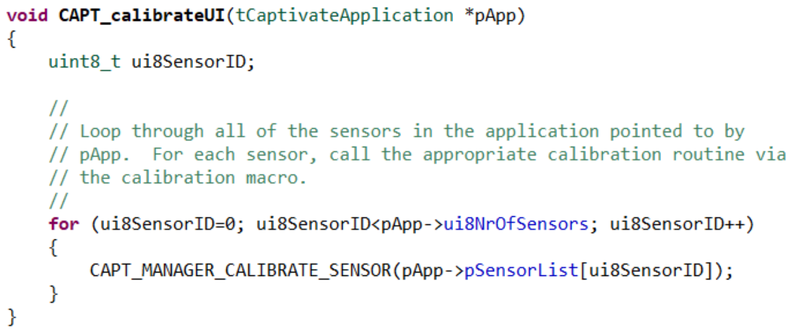
```
extern tElement PRX_E00;
extern tElement PRX_E01;
extern tElement PRX_E02;
extern tElement PRX_E03;
```



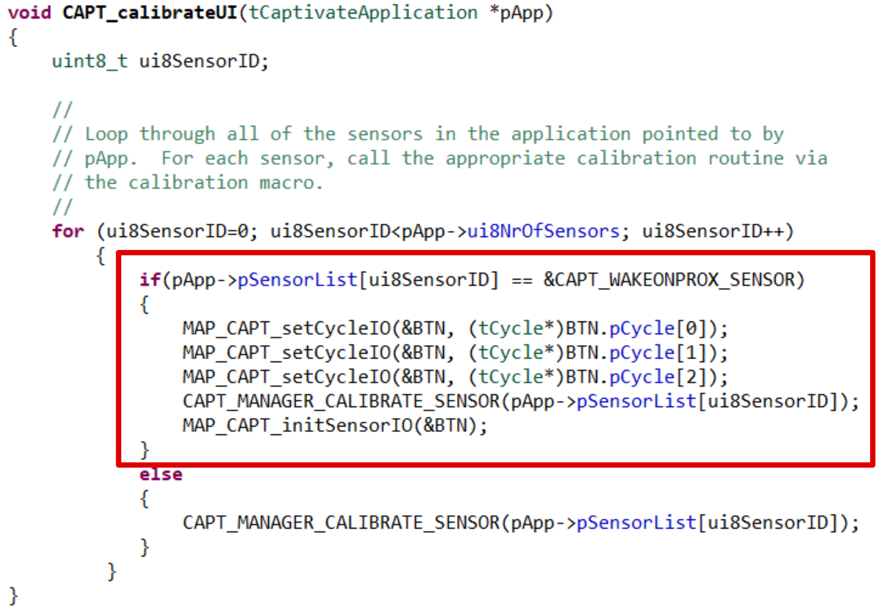
```
PRX_E00.ui8TxBlock = 0;
PRX_E00.ui8TxPin = 1;
PRX_E01.ui8TxBlock = 0;
PRX_E01.ui8TxPin = 1;
PRX_E02.ui8TxBlock = 0;
PRX_E02.ui8TxPin = 1;
PRX_E03.ui8TxBlock = 0;
PRX_E03.ui8TxPin = 1;
```

Implementation Example

- **Step 3:** Make sure to enable and activate all the element for BTN sensor when calibrating the PRX sensor since we are using PRX sensor as a virtual sensor for BTN sensor. In CAPT_Manager.c file CAPT_calibrateUI function.



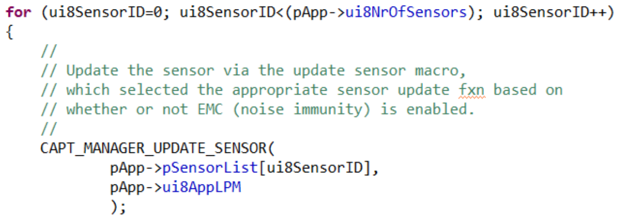
```
void CAPT_calibrateUI(tCaptivateApplication *pApp)
{
    uint8_t ui8SensorID;

    //
    // Loop through all of the sensors in the application pointed to by
    // pApp.  For each sensor, call the appropriate calibration routine via
    // the calibration macro.
    //
    for (ui8SensorID=0; ui8SensorID<pApp->ui8NrOfSensors; ui8SensorID++)
    {
        CAPT_MANAGER_CALIBRATE_SENSOR(pApp->pSensorList[ui8SensorID]);
    }
}
```



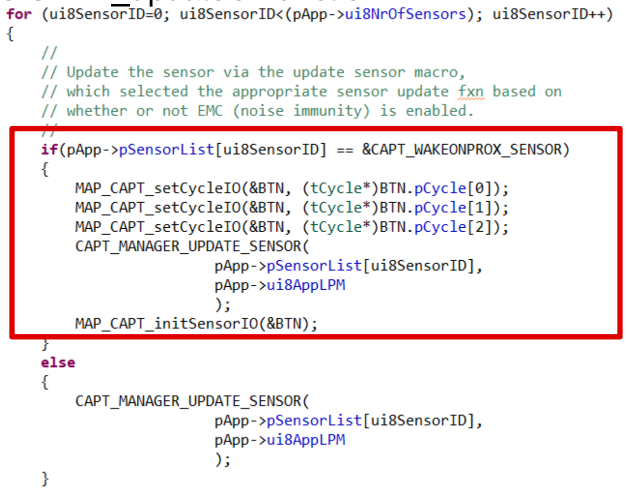
```
void CAPT_calibrateUI(tCaptivateApplication *pApp)
{
    uint8_t ui8SensorID;

    //
    // Loop through all of the sensors in the application pointed to by
    // pApp.  For each sensor, call the appropriate calibration routine via
    // the calibration macro.
    //
    for (ui8SensorID=0; ui8SensorID<pApp->ui8NrOfSensors; ui8SensorID++)
        {
            if(pApp->pSensorList[ui8SensorID] == &CAPT_WAKEONPROX_SENSOR)
            {
                MAP_CAPT_setCycleIO(&BTN, (tCycle*)BTN.pCycle[0]);
                MAP_CAPT_setCycleIO(&BTN, (tCycle*)BTN.pCycle[1]);
                MAP_CAPT_setCycleIO(&BTN, (tCycle*)BTN.pCycle[2]);
                CAPT_MANAGER_CALIBRATE_SENSOR(pApp->pSensorList[ui8SensorID]);
                MAP_CAPT_initSensorIO(&BTN);
            }
            else
            {
                CAPT_MANAGER_CALIBRATE_SENSOR(pApp->pSensorList[ui8SensorID]);
            }
        }
}
```

Implementation Example

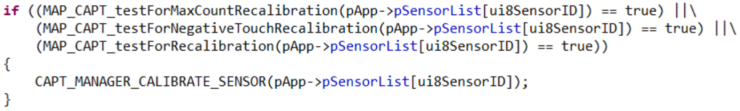
- **Step 4:** Make sure to enable and activate all the element for BTN sensor when update the PRX sensor since we are using PRX sensor as a virtual sensor for BTN sensor. In CAPT_Manager.c file CAPT_updateUI function.



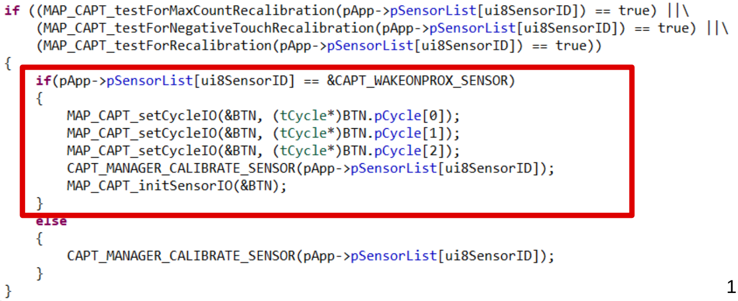
```
for (ui8SensorID=0; ui8SensorID<(pApp->ui8NrOfSensors); ui8SensorID++)
{
    //
    // Update the sensor via the update sensor macro,
    // which selected the appropriate sensor update fxn based on
    // whether or not EMC (noise immunity) is enabled.
    //
    CAPT_MANAGER_UPDATE_SENSOR(
            pApp->pSensorList[ui8SensorID],
            pApp->ui8AppLPM
            );
```



```
for (ui8SensorID=0; ui8SensorID<(pApp->ui8NrOfSensors); ui8SensorID++)
{
    //
    // Update the sensor via the update sensor macro,
    // which selected the appropriate sensor update fxn based on
    // whether or not EMC (noise immunity) is enabled.
    //
    if(pApp->pSensorList[ui8SensorID] == &CAPT_WAKEONPROX_SENSOR)
    {
        MAP_CAPT_setCycleIO(&BTN, (tCycle*)BTN.pCycle[0]);
        MAP_CAPT_setCycleIO(&BTN, (tCycle*)BTN.pCycle[1]);
        MAP_CAPT_setCycleIO(&BTN, (tCycle*)BTN.pCycle[2]);
        CAPT_MANAGER_UPDATE_SENSOR(
                pApp->pSensorList[ui8SensorID],
                pApp->ui8AppLPM
                );
        MAP_CAPT_initSensorIO(&BTN);
    }
    else
    {
        CAPT_MANAGER_UPDATE_SENSOR(
                pApp->pSensorList[ui8SensorID],
                pApp->ui8AppLPM
                );
    }
```



```
if ((MAP_CAPT_testForMaxCountRecalibration(pApp->pSensorList[ui8SensorID]) == true) ||\
    (MAP_CAPT_testForNegativeTouchRecalibration(pApp->pSensorList[ui8SensorID]) == true) ||\
    (MAP_CAPT_testForRecalibration(pApp->pSensorList[ui8SensorID]) == true))
{
    CAPT_MANAGER_CALIBRATE_SENSOR(pApp->pSensorList[ui8SensorID]);
}
```

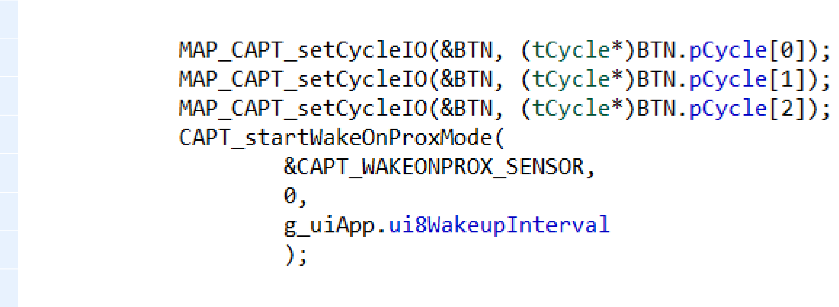


```
if ((MAP_CAPT_testForMaxCountRecalibration(pApp->pSensorList[ui8SensorID]) == true) ||\
    (MAP_CAPT_testForNegativeTouchRecalibration(pApp->pSensorList[ui8SensorID]) == true) ||\
    (MAP_CAPT_testForRecalibration(pApp->pSensorList[ui8SensorID]) == true))
{
    if(pApp->pSensorList[ui8SensorID] == &CAPT_WAKEONPROX_SENSOR)
    {
        MAP_CAPT_setCycleIO(&BTN, (tCycle*)BTN.pCycle[0]);
        MAP_CAPT_setCycleIO(&BTN, (tCycle*)BTN.pCycle[1]);
        MAP_CAPT_setCycleIO(&BTN, (tCycle*)BTN.pCycle[2]);
        CAPT_MANAGER_CALIBRATE_SENSOR(pApp->pSensorList[ui8SensorID]);
        MAP_CAPT_initSensorIO(&BTN);
    }
    else
    {
        CAPT_MANAGER_CALIBRATE_SENSOR(pApp->pSensorList[ui8SensorID]);
    }
}
```

Implementation Example

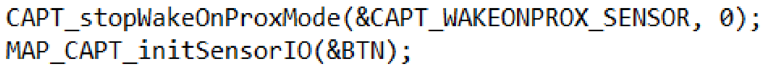
- **Step 5:** In CAPT_App.c, make sure to enable and activate all the BTN sensor elements before starting wake on proximity mode and deactivate the BTN sensor after stopping the wake on proximity mode.

  To enable and activate all the BTN sensor elements:



```
MAP_CAPT_setCycleIO(&BTN, (tCycle*)BTN.pCycle[0]);
MAP_CAPT_setCycleIO(&BTN, (tCycle*)BTN.pCycle[1]);
MAP_CAPT_setCycleIO(&BTN, (tCycle*)BTN.pCycle[2]);
CAPT_startWakeOnProxMode(
        &CAPT_WAKEONPROX_SENSOR,
        0,
        g_uiApp.ui8WakeupInterval
        );
```

  To deactivate the BTN sensor: `MAP_CAPT_initSensorIO(&BTN);`



```
CAPT_stopWakeOnProxMode(&CAPT_WAKEONPROX_SENSOR, 0);
MAP_CAPT_initSensorIO(&BTN);
```