```
PROGRAM  BMP085
VAR
    //BMP Data Arrays
    BMP_DATAIN1  :  ARRAY [ 0 .. 40 ]  OF  BYTE ;
    BMP_DATAOUT1  :  ARRAY [ 0 .. 40 ]  OF  BYTE ;
    BMP_DATAOUT_old1  :  ARRAY [ 0 .. 40 ]  OF  BYTE ;

    //Cal Table arrays
    BMP_DATAIN2  :  ARRAY [ 0 .. 40 ]  OF  BYTE ;
    BMP_DATAOUT2  :  ARRAY [ 0 .. 40 ]  OF  BYTE ;

    BMPCal_State :  INT ;
    BMP_Cal_Scan_Delay :  Ton ;
    BMP_Cal_Scan_Start :  BOOL ;
    BMP_Cal_Scan_Done :  BOOL ;
    BMP_Cal_Scan_Elapsed :  TIME ;
    BMP_Cal_Scan_Time :  TIME  :=  t#100ms ;
    BMP_TP_Scan_Delay :  TON ;
    BMP_TP_Scan_Start :  BOOL ;
    BMP_TP_Scan_Time :  TIME  :=  t#100ms ;
    BMP_TP_Scan_Done :  BOOL ;
    BMP_TP_Scan_Elapsed :  TIME ;
    BMP_TP_State :  INT ;

    BMP_AC1 :  INT ;
    BMP_AC2 :  INT ;
    BMP_AC3 :  INT ;
    BMP_AC4 :  UINT ;
    BMP_AC5 :  UINT ;
    BMP_AC6 :  UINT ;
    BMP_B1 :  INT ;
    BMP_B2 :  INT ;
    BMP_MB :  INT ;
    BMP_MC :  INT ;
    BMP_MD :  INT ;
    BMP_temp :  DINT ;
    BMP_X1 :  DINT ;
    BMP_X2 :  DINT ;
    BMP_B5 :  DINT ;
    BMP_T :  DINT ;
    BMP_Calc_Done :  BOOL ;
    BMP_Scanning :  BOOL := FALSE ;
    BMP_ftemp :  REAL ;
    BMP_pressure :  DINT ;
    BMP_P_B6 :  DINT ;
    BMP_P_X1 :  DINT ;
    BMP_P_X2 :  DINT ;
    BMP_P_X3 :  DINT ;
    BMP_P_B3 :  DINT ;
    BMP_P_X1_2 :  DINT ;
    BMP_P_X2_2 :  DINT ;
    BMP_P_X3_2 :  DINT ;
```

```
        BMP_P_B4 :  DINT ;
        BMP_P_B7 :  LINT ;
        BMP_P_P :  LINT ;
        BMP_P_X1_3 :  LINT ;
        BMP_P_X1_4 :  LINT ;
        BMP_P_X2_3 :  LINT ;
        BMP_P_pfinal :  LINT ;
        Defaults_Enable :  BOOL := 0 ;
        BMP_P_inHG :  REAL ;
        BMP_altitude :  REAL ;
        Pressure_Filter :  FILTER_MAV_DW ;
        BMP_alt_filtered :  Dint ;
        BMP_pascals :  DINT ;
        BMP_alt_filt :  DINT ;
        Alt_Filt_Buffer :  UINT  :=  30 ;
    END_VAR

```

```
    //Setup Number of Registers IN and OUT for I2C device (this is the BMP085 sensor
     on the IMU)
    i2c_single_1 . REG_IN_START := 246 ;                              //Start of Data IN
    i2c_single_1 . REG_OUT_START := 244 ;                         //Start of Control Word
    i2c_single_1 . REGNUM_OUT := 1 ;                                  //1 control word
    i2c_single_1 . REGNUM_IN := 8 ;                               //All 8 bytes being read

    //Setup Number of Registers IN and OUT for I2C device (this is the BMP085 sensor
     on the IMU, looking at the cal table)
    i2c_single_2 . REGNUM_OUT := 0 ;                                  //No control word needed
    i2c_single_2 . REGNUM_IN := 8 ;                               //All 8 bytes being read

    //These are timer delays for the temp sampling and pressure sampling of the
    BMP085
    BMP_Cal_Scan_Delay ( IN := BMP_Cal_Scan_Start  ,  PT := BMP_Cal_Scan_Time  ,  Q =>
    BMP_Cal_Scan_Done  ,  ET => BMP_Cal_Scan_Elapsed  ) ;
    BMP_TP_Scan_Delay ( IN := BMP_TP_Scan_Start  ,  PT := BMP_TP_Scan_Time  ,  Q =>
    BMP_TP_Scan_Done  ,  ET => BMP_TP_Scan_Elapsed  ) ;

    //State Machine to read Calibration table in BMP085. It reads once.
    CASE  BMPCal_State  OF

        0 :
            BMP_Table_Loaded := 0 ;
            i2c_single_2 . REG_IN_START := 170 ;

            //read Data from the BMP085 sensor cal table
            BMP_DATAIN2 [ 0 ] := i2c_single_2 . DATAIN [ 0 ] ;
            BMP_DATAIN2 [ 1 ] := i2c_single_2 . DATAIN [ 1 ] ;

            BMP_Cal_Scan_Start := 1 ;
            //After delay
            IF  BMP_Cal_Scan_Done  THEN
                BMPCal_State := 10 ;
                BMP_Cal_Scan_Start := 0 ;
            END_IF

        10 :
            i2c_single_2 . REG_IN_START := 172 ;
            //Data from the BMP085 sensor cal table
            BMP_DATAIN2 [ 2 ] := i2c_single_2 . DATAIN [ 0 ] ;
            BMP_DATAIN2 [ 3 ] := i2c_single_2 . DATAIN [ 1 ] ;

            BMP_Cal_Scan_Start := 1 ;
            //After delay
            IF  BMP_Cal_Scan_Done  THEN
                BMPCal_State := 20 ;
                BMP_Cal_Scan_Start := 0 ;
            END_IF

        20 :
            i2c_single_2 . REG_IN_START := 174 ;
            //Data from the BMP085 sensor cal table
            BMP_DATAIN2 [ 4 ] := i2c_single_2 . DATAIN [ 0 ] ;
            BMP_DATAIN2 [ 5 ] := i2c_single_2 . DATAIN [ 1 ] ;

            BMP_Cal_Scan_Start := 1 ;
            //After delay
            IF  BMP_Cal_Scan_Done  THEN
                BMPCal_State := 30 ;
                BMP_Cal_Scan_Start := 0 ;
            END_IF

        30 :
            i2c_single_2 . REG_IN_START := 176 ;
            //Data from the BMP085 sensor cal table
            BMP_DATAIN2 [ 6 ] := i2c_single_2 . DATAIN [ 0 ] ;
            BMP_DATAIN2 [ 7 ] := i2c_single_2 . DATAIN [ 1 ] ;

            BMP_Cal_Scan_Start := 1 ;
            //After delay
            IF  BMP_Cal_Scan_Done  THEN
                BMPCal_State := 40 ;
                BMP_Cal_Scan_Start := 0 ;
            END_IF

        40 :
            i2c_single_2 . REG_IN_START := 178 ;
            //Data from the BMP085 sensor cal table
            BMP_DATAIN2 [ 8 ] := i2c_single_2 . DATAIN [ 0 ] ;
            BMP_DATAIN2 [ 9 ] := i2c_single_2 . DATAIN [ 1 ] ;
            BMP_Cal_Scan_Start := 1 ;

            //After delay
            IF BMP_Cal_Scan_Done  THEN
                BMPCal_State := 60 ;
                BMP_Cal_Scan_Start := 0 ;
            END_IF

        60 :
            i2c_single_2 . REG_IN_START := 180 ;
            //Data from the BMP085 sensor cal table
            BMP_DATAIN2 [ 10 ] := i2c_single_2 . DATAIN [ 0 ] ;
            BMP_DATAIN2 [ 11 ] := i2c_single_2 . DATAIN [ 1 ] ;

            BMP_Cal_Scan_Start := 1 ;
            //After delay
            IF BMP_Cal_Scan_Done  THEN
                BMPCal_State := 70 ;
                BMP_Cal_Scan_Start := 0 ;
            END_IF

        70 :
            i2c_single_2 . REG_IN_START := 182 ;
            //Data from the BMP085 sensor cal table
            BMP_DATAIN2 [ 12 ] := i2c_single_2 . DATAIN [ 0 ] ;
            BMP_DATAIN2 [ 13 ] := i2c_single_2 . DATAIN [ 1 ] ;
            BMP_Cal_Scan_Start := 1 ;

            //After delay
            IF BMP_Cal_Scan_Done  THEN
                BMPCal_State := 80 ;
                BMP_Cal_Scan_Start := 0 ;
            END_IF

        80 :
            i2c_single_2 . REG_IN_START := 184 ;
            //Data from the BMP085 sensor cal table
            BMP_DATAIN2 [ 14 ] := i2c_single_2 . DATAIN [ 0 ] ;
            BMP_DATAIN2 [ 15 ] := i2c_single_2 . DATAIN [ 1 ] ;
            BMP_Cal_Scan_Start := 1 ;

            //After delay
            IF BMP_Cal_Scan_Done  THEN
                BMPCal_State := 90 ;
                BMP_Cal_Scan_Start := 0 ;
            END_IF

        90 :
            i2c_single_2 . REG_IN_START := 186 ;
            //Data from the BMP085 sensor cal table
            BMP_DATAIN2 [ 16 ] := i2c_single_2 . DATAIN [ 0 ] ;
            BMP_DATAIN2 [ 17 ] := i2c_single_2 . DATAIN [ 1 ] ;

            BMP_Cal_Scan_Start := 1 ;
            //After delay
            IF  BMP_Cal_Scan_Done  THEN
                BMPCal_State := 100 ;
                BMP_Cal_Scan_Start := 0 ;
            END_IF

        100 :
            i2c_single_2 . REG_IN_START := 188 ;
            //Data from the BMP085 sensor cal table
            BMP_DATAIN2 [ 18 ] := i2c_single_2 . DATAIN [ 0 ] ;
            BMP_DATAIN2 [ 19 ] := i2c_single_2 . DATAIN [ 1 ] ;

            BMP_Cal_Scan_Start := 1 ;
            //After delay
            IF  BMP_Cal_Scan_Done  THEN
                BMPCal_State := 110 ;
                BMP_Cal_Scan_Start := 0 ;
            END_IF

        110 :
            i2c_single_2 . REG_IN_START := 190 ;
            //Data from the BMP085 sensor cal table
            BMP_DATAIN2 [ 20 ] := i2c_single_2 . DATAIN [ 0 ] ;
            BMP_DATAIN2 [ 21 ] := i2c_single_2 . DATAIN [ 1 ] ;
            BMP_Cal_Scan_Start := 1 ;

            //After delay
            IF  BMP_Cal_Scan_Done  THEN
                BMPCal_State := 99 ;     //only 1 pass
                BMP_Cal_Scan_Start := 0 ;
                BMP_Table_Loaded := 1 ;
            END_IF

    END_CASE

    //State machine to read temp, then pressure with delay between.
    CASE  BMP_TP_State  OF

        0 :
            BMP_DATAOUT1 [ 0 ] := 46 ;     //read temperature
            BMP_TP_Scan_Start := 1 ;

            //After delay read the data
            IF  BMP_TP_Scan_Done  THEN
                //Now read temperature data from the BMP085 sensor
                BMP_DATAIN1 [ 0 ] := i2c_single_1 . DATAIN [ 0 ] ;
                BMP_DATAIN1 [ 1 ] := i2c_single_1 . DATAIN [ 1 ] ;
                BMP_TP_State := 10 ;
                BMP_TP_Scan_Start := 0 ;
            END_IF

        10 :
            BMP_DATAOUT1 [ 0 ] := 52 ;      //read pressure
            BMP_TP_Scan_Start := 1 ;

            //After delay read the data
            IF  BMP_TP_Scan_Done  THEN
                //Now read pressure data from the BMP085 sensor
                BMP_DATAIN1 [ 2 ] := i2c_single_1 . DATAIN [ 0 ] ;
                BMP_DATAIN1 [ 3 ] := i2c_single_1 . DATAIN [ 1 ] ;
                BMP_DATAIN1 [ 4 ] := i2c_single_1 . DATAIN [ 2 ] ;
                BMP_TP_State := 0 ;        //loop back to start
                BMP_TP_Scan_Start := 0 ;
            END_IF

    END_CASE

    //Read BMP085 sensor or test calculation
    IF  NOT  Defaults_Enable  THEN
        BMP_temp :=  WORD_TO_INT ( Mem . PackBytesToWord ( BMP_DATAIN1 [ 0 ] , BMP_DATAIN1
    [ 1 ] ) ) ;
        BMP_pressure :=  DWORD_TO_DINT ( Mem . PackBytesToDWord ( 0 , 0 , BMP_DATAIN1 [ 2 ]
    , BMP_DATAIN1 [ 3 ] ) ) ;
    ELSE
        BMP_temp :=  27898 ;
        BMP_pressure :=  23843 ;
    END_IF

    //Only need to scan once after table is loaded into registers
    IF  ( BMP_Table_Loaded  AND  NOT  BMP_Scanning )  OR  Defaults_Enable  THEN

        IF  NOT  Defaults_Enable  THEN
            BMP_AC1 :=  WORD_TO_INT ( Mem . PackBytesToWord ( BMP_DATAIN2 [ 0 ] ,
    BMP_DATAIN2 [ 1 ] ) ) ;
            BMP_AC2 :=  WORD_TO_INT ( Mem . PackBytesToWord ( BMP_DATAIN2 [ 2 ] ,
    BMP_DATAIN2 [ 3 ] ) ) ;
            BMP_AC3 :=  WORD_TO_INT ( Mem . PackBytesToWord ( BMP_DATAIN2 [ 4 ] ,
    BMP_DATAIN2 [ 5 ] ) ) ;
            BMP_AC4 :=  WORD_TO_UINT ( Mem . PackBytesToWord ( BMP_DATAIN2 [ 6 ] ,
    BMP_DATAIN2 [ 7 ] ) ) ;
            BMP_AC5 :=  WORD_TO_UINT ( Mem . PackBytesToWord ( BMP_DATAIN2 [ 8 ] ,
    BMP_DATAIN2 [ 9 ] ) ) ;
            BMP_AC6 :=  WORD_TO_UINT ( Mem . PackBytesToWord ( BMP_DATAIN2 [ 10 ] ,
    BMP_DATAIN2 [ 11 ] ) ) ;
            BMP_B1 :=  WORD_TO_INT ( Mem . PackBytesToWord ( BMP_DATAIN2 [ 12 ] ,
    BMP_DATAIN2 [ 13 ] ) ) ;
            BMP_B2 :=  WORD_TO_INT ( Mem . PackBytesToWord ( BMP_DATAIN2 [ 14 ] ,
    BMP_DATAIN2 [ 15 ] ) ) ;
            BMP_MB :=  WORD_TO_INT ( Mem . PackBytesToWord ( BMP_DATAIN2 [ 16 ] ,
    BMP_DATAIN2 [ 17 ] ) ) ;
            BMP_MC :=  WORD_TO_INT ( Mem . PackBytesToWord ( BMP_DATAIN2 [ 18 ] ,
BMP_DATAIN2 [ 19 ] ) ) ;
            BMP_MD :=  WORD_TO_INT ( Mem . PackBytesToWord ( BMP_DATAIN2 [ 20 ] ,
BMP_DATAIN2 [ 21 ] ) ) ;

        //Testing only
        ELSE
            BMP_AC1 :=  408 ;
            BMP_AC2 :=  - 72 ;
            BMP_AC3 :=  - 14383 ;
            BMP_AC4 :=  32741 ;
            BMP_AC5 :=  32757 ;
            BMP_AC6 :=  23153 ;
            BMP_B1 :=  6190 ;
            BMP_B2 :=  4 ;
            BMP_MB :=  - 32767 ;
            BMP_MC :=  - 8711 ;
            BMP_MD :=  2868 ;
        END_IF

        //if these values are zero, then module is unplugged, load defaults
        IF  BMP_AC1  <>  0  OR  BMP_AC2 <>  0  THEN
            BMP_Scanning := 1 ;
            BMP_Calc_Done := 1 ;
        ELSE
            Defaults_Enable := TRUE ;
        END_IF

    END_IF

    //Only calculate if cal table is done loading
    IF  BMP_Scanning  THEN
        //Calculate temp
        BMP_X1 := ( BMP_temp - BMP_AC6 ) * BMP_AC5 / 32768 ;
        BMP_X2 := BMP_MC * 2048 / ( BMP_X1 + BMP_MD ) ;
        BMP_B5 := BMP_X1 + BMP_X2 ;
        BMP_T := ( BMP_B5 + 8 ) / 16 ;
        BMP_ftemp := ( ( ( DINT_TO_REAL ( BMP_T ) / 10.0 * 9 ) / 5 ) + 32 ) ;

        //Calculate Pressure
        BMP_P_B6 := BMP_B5 - 4000 ;
        BMP_P_X1 := ( BMP_B2 * ( BMP_P_B6 * BMP_P_B6 / 4096 ) ) / 2048 ;
        BMP_P_X2 := BMP_AC2 * BMP_P_B6 / 2048 ;
        BMP_P_X3 := BMP_P_X1 + BMP_P_X2 ;
        BMP_P_B3 := ( SHL ( ( BMP_AC1 * 4 + BMP_P_X3 ) , 0 ) + 2 ) / 4 ;
        BMP_P_X1_2 := BMP_AC3 * BMP_P_B6 / 8192 ;
        BMP_P_X2_2 := ( BMP_B1 * ( BMP_P_B6 * BMP_P_B6 / 4096 ) ) / 65536 ;
        BMP_P_X3_2 := ( ( BMP_P_X1_2 + BMP_P_X2_2 ) + 2 ) / 4 ;
        BMP_P_B4 := BMP_AC4 * ( BMP_P_X3_2 + 32768 ) / 32768 ;
        BMP_P_B7 := ( BMP_pressure - BMP_P_B3 ) * 50000 ;

        IF  BMP_P_B7  <  2147483648  THEN
            BMP_P_P := ( BMP_P_B7 * 2 ) / BMP_P_B4 ;
        ELSE
            BMP_P_P := ( BMP_P_B7 / BMP_P_B4 ) * 2 ;
        END_IF

        BMP_P_X1_3 := ( BMP_P_P / 256 ) * ( BMP_P_P / 256 ) ;
        BMP_P_X1_4 := ( BMP_P_X1_3 * 3038 ) / 65536 ;
        BMP_P_X2_3 := ( - 7357 * BMP_P_P ) / 65536 ;
        BMP_P_pfinal := BMP_P_P + ( BMP_P_X1_4 + BMP_P_X2_3 + 3791 ) / 16 ;
        BMP_pascals := REAL_TO_DINT ( LINT_TO_REAL ( BMP_P_pfinal ) ) ;

        //Moving average filter
        Pressure_Filter ( X := BMP_pascals , N := Alt_Filt_Buffer , RST := , Y =>
    BMP_alt_filtered ) ;

        BMP_P_inHG := BMP_pascals / 3386.38816 ;
        BMP_altitude := 44330.0 * ( 1 - ( EXPT ( ( ( BMP_pascals / 100 ) / ( 101325.0 /
    100.0 ) ) , 0.1903 ) ) ) ;

    END_IF

    //Ctrl data for switching BMP085 data channel 46=temp, 52=pressure
    //Write the data to the output registers
    IF BMP_DATAOUT_old1 [ 0 ] <> BMP_DATAOUT1 [ 0 ] THEN
            i2c_single_1 . DATAOUT [ 0 ] := BMP_DATAOUT1 [ 0 ] ;
            BMP_DATAOUT_old1 [ 0 ] := BMP_DATAOUT1 [ 0 ] ;
    END_IF
```