

```
1  PROGRAM BMP085
2  VAR
3      //BMP Data Arrays
4      BMP_DATAIN1 : ARRAY [ 0 .. 40 ] OF BYTE ;
5      BMP_DATAOUT1 : ARRAY [ 0 .. 40 ] OF BYTE ;
6      BMP_DATAOUT_old1 : ARRAY [ 0 .. 40 ] OF BYTE ;
7
8      //Cal Table arrays
9      BMP_DATAIN2 : ARRAY [ 0 .. 40 ] OF BYTE ;
10     BMP_DATAOUT2 : ARRAY [ 0 .. 40 ] OF BYTE ;
11
12     BMP_Cal_State : INT ;
13     BMP_Cal_Scan_Delay : Ton ;
14     BMP_Cal_Scan_Start : BOOL ;
15     BMP_Cal_Scan_Done : BOOL ;
16     BMP_Cal_Scan_Elapsed : TIME ;
17     BMP_Cal_Scan_Time : TIME := t#100ms ;
18     BMP_TP_Scan_Delay : TON ;
19     BMP_TP_Scan_Start : BOOL ;
20     BMP_TP_Scan_Time : TIME := t#100ms ;
21     BMP_TP_Scan_Done : BOOL ;
22     BMP_TP_Scan_Elapsed : TIME ;
23     BMP_TP_State : INT ;
24
25     BMP_AC1 : INT ;
26     BMP_AC2 : INT ;
27     BMP_AC3 : INT ;
28     BMP_AC4 : UINT ;
29     BMP_AC5 : UINT ;
30     BMP_AC6 : UINT ;
31     BMP_B1 : INT ;
32     BMP_B2 : INT ;
33     BMP_MB : INT ;
34     BMP_MC : INT ;
35     BMP_MD : INT ;
36     BMP_temp : DINT ;
37     BMP_X1 : DINT ;
38     BMP_X2 : DINT ;
39     BMP_B5 : DINT ;
40     BMP_T : DINT ;
41     BMP_Calc_Done : BOOL ;
42     BMP_Scanning : BOOL := FALSE ;
43     BMP_ftemp : REAL ;
44     BMP_pressure : DINT ;
45     BMP_P_B6 : DINT ;
46     BMP_P_X1 : DINT ;
47     BMP_P_X2 : DINT ;
48     BMP_P_X3 : DINT ;
49     BMP_P_B3 : DINT ;
50     BMP_P_X1_2 : DINT ;
51     BMP_P_X2_2 : DINT ;
52     BMP_P_X3_2 : DINT ;
```

```
53     BMP_P_B4 : DINT ;
54     BMP_P_B7 : LINT ;
55     BMP_P_P : LINT ;
56     BMP_P_X1_3 : LINT ;
57     BMP_P_X1_4 : LINT ;
58     BMP_P_X2_3 : LINT ;
59     BMP_P_pfinal : LINT ;
60     Defaults_Enable : BOOL := 0 ;
61     BMP_P_inHG : REAL ;
62     BMP_altitude : REAL ;
63     Pressure_Filter : FILTER_MAV_DW ;
64     BMP_alt_filtered : Dint ;
65     BMP_pascals : DINT ;
66     BMP_alt_filt : DINT ;
67     Alt_Filt_Buffer : UINT := 30 ;
68 END_VAR
69
```

```
1  //Setup Number of Registers IN and OUT for I2C device (this is the BMP085 sensor
   on the IMU)
2  i2c_single_1 . REG_IN_START := 246 ;           //Start of Data IN
3  i2c_single_1 . REG_OUT_START := 244 ;          //Start of Control Word
4  i2c_single_1 . REGNUM_OUT := 1 ;               //1 control word
5  i2c_single_1 . REGNUM_IN := 8 ;               //All 8 bytes being read
6
7  //Setup Number of Registers IN and OUT for I2C device (this is the BMP085 sensor
   on the IMU, looking at the cal table)
8  i2c_single_2 . REGNUM_OUT := 0 ;               //No control word needed
9  i2c_single_2 . REGNUM_IN := 8 ;               //All 8 bytes being read
10
11 //These are timer delays for the temp sampling and pressure sampling of the
   BMP085
12 BMP_Cal_Scan_Delay ( IN := BMP_Cal_Scan_Start , PT := BMP_Cal_Scan_Time , Q =>
   BMP_Cal_Scan_Done , ET => BMP_Cal_Scan_Elapsed ) ;
13 BMP_TP_Scan_Delay ( IN := BMP_TP_Scan_Start , PT := BMP_TP_Scan_Time , Q =>
   BMP_TP_Scan_Done , ET => BMP_TP_Scan_Elapsed ) ;
14
15 //State Machine to read Calibration table in BMP085. It reads once.
16 CASE BMPCal_State OF
17
18     0 :
19         BMP_Table_Loaded := 0 ;
20         i2c_single_2 . REG_IN_START := 170 ;
21
22         //read Data from the BMP085 sensor cal table
23         BMP_DATAIN2 [ 0 ] := i2c_single_2 . DATAIN [ 0 ] ;
24         BMP_DATAIN2 [ 1 ] := i2c_single_2 . DATAIN [ 1 ] ;
25
26         BMP_Cal_Scan_Start := 1 ;
27         //After delay
```

```
28         IF BMP_Cal_Scan_Done THEN
29             BMPCal_State := 10 ;
30             BMP_Cal_Scan_Start := 0 ;
31         END_IF
32
33     10 :
34         i2c_single_2 . REG_IN_START := 172 ;
35         //Data from the BMP085 sensor cal table
36         BMP_DATAIN2 [ 2 ] := i2c_single_2 . DATAIN [ 0 ] ;
37         BMP_DATAIN2 [ 3 ] := i2c_single_2 . DATAIN [ 1 ] ;
38
39         BMP_Cal_Scan_Start := 1 ;
40         //After delay
41         IF BMP_Cal_Scan_Done THEN
42             BMPCal_State := 20 ;
43             BMP_Cal_Scan_Start := 0 ;
44         END_IF
45
46     20 :
47         i2c_single_2 . REG_IN_START := 174 ;
48         //Data from the BMP085 sensor cal table
49         BMP_DATAIN2 [ 4 ] := i2c_single_2 . DATAIN [ 0 ] ;
50         BMP_DATAIN2 [ 5 ] := i2c_single_2 . DATAIN [ 1 ] ;
51
52         BMP_Cal_Scan_Start := 1 ;
53         //After delay
54         IF BMP_Cal_Scan_Done THEN
55             BMPCal_State := 30 ;
56             BMP_Cal_Scan_Start := 0 ;
57         END_IF
58
59     30 :
60         i2c_single_2 . REG_IN_START := 176 ;
61         //Data from the BMP085 sensor cal table
62         BMP_DATAIN2 [ 6 ] := i2c_single_2 . DATAIN [ 0 ] ;
63         BMP_DATAIN2 [ 7 ] := i2c_single_2 . DATAIN [ 1 ] ;
64
65         BMP_Cal_Scan_Start := 1 ;
66         //After delay
67         IF BMP_Cal_Scan_Done THEN
68             BMPCal_State := 40 ;
69             BMP_Cal_Scan_Start := 0 ;
70         END_IF
71
72     40 :
73         i2c_single_2 . REG_IN_START := 178 ;
74         //Data from the BMP085 sensor cal table
75         BMP_DATAIN2 [ 8 ] := i2c_single_2 . DATAIN [ 0 ] ;
76         BMP_DATAIN2 [ 9 ] := i2c_single_2 . DATAIN [ 1 ] ;
77         BMP_Cal_Scan_Start := 1 ;
78
```

```
79      //After delay
80      IF BMP_Cal_Scan_Done THEN
81          BMPCal_State := 60 ;
82          BMP_Cal_Scan_Start := 0 ;
83      END_IF
84
85  60 :
86      i2c_single_2 . REG_IN_START := 180 ;
87      //Data from the BMP085 sensor cal table
88      BMP_DATAIN2 [ 10 ] := i2c_single_2 . DATAIN [ 0 ] ;
89      BMP_DATAIN2 [ 11 ] := i2c_single_2 . DATAIN [ 1 ] ;
90
91      BMP_Cal_Scan_Start := 1 ;
92      //After delay
93      IF BMP_Cal_Scan_Done THEN
94          BMPCal_State := 70 ;
95          BMP_Cal_Scan_Start := 0 ;
96      END_IF
97
98  70 :
99      i2c_single_2 . REG_IN_START := 182 ;
100     //Data from the BMP085 sensor cal table
101     BMP_DATAIN2 [ 12 ] := i2c_single_2 . DATAIN [ 0 ] ;
102     BMP_DATAIN2 [ 13 ] := i2c_single_2 . DATAIN [ 1 ] ;
103     BMP_Cal_Scan_Start := 1 ;
104
105     //After delay
106     IF BMP_Cal_Scan_Done THEN
107         BMPCal_State := 80 ;
108         BMP_Cal_Scan_Start := 0 ;
109     END_IF
110
111  80 :
112     i2c_single_2 . REG_IN_START := 184 ;
113     //Data from the BMP085 sensor cal table
114     BMP_DATAIN2 [ 14 ] := i2c_single_2 . DATAIN [ 0 ] ;
115     BMP_DATAIN2 [ 15 ] := i2c_single_2 . DATAIN [ 1 ] ;
116     BMP_Cal_Scan_Start := 1 ;
117
118     //After delay
119     IF BMP_Cal_Scan_Done THEN
120         BMPCal_State := 90 ;
121         BMP_Cal_Scan_Start := 0 ;
122     END_IF
123
124  90 :
125     i2c_single_2 . REG_IN_START := 186 ;
126     //Data from the BMP085 sensor cal table
127     BMP_DATAIN2 [ 16 ] := i2c_single_2 . DATAIN [ 0 ] ;
128     BMP_DATAIN2 [ 17 ] := i2c_single_2 . DATAIN [ 1 ] ;
129
```

```
130     BMP_Cal_Scan_Start := 1 ;
131     //After delay
132     IF BMP_Cal_Scan_Done THEN
133         BMPCal_State := 100 ;
134         BMP_Cal_Scan_Start := 0 ;
135     END_IF
136
137 100 :
138     i2c_single_2 . REG_IN_START := 188 ;
139     //Data from the BMP085 sensor cal table
140     BMP_DATAIN2 [ 18 ] := i2c_single_2 . DATAIN [ 0 ] ;
141     BMP_DATAIN2 [ 19 ] := i2c_single_2 . DATAIN [ 1 ] ;
142
143     BMP_Cal_Scan_Start := 1 ;
144     //After delay
145     IF BMP_Cal_Scan_Done THEN
146         BMPCal_State := 110 ;
147         BMP_Cal_Scan_Start := 0 ;
148     END_IF
149
150 110 :
151     i2c_single_2 . REG_IN_START := 190 ;
152     //Data from the BMP085 sensor cal table
153     BMP_DATAIN2 [ 20 ] := i2c_single_2 . DATAIN [ 0 ] ;
154     BMP_DATAIN2 [ 21 ] := i2c_single_2 . DATAIN [ 1 ] ;
155     BMP_Cal_Scan_Start := 1 ;
156
157     //After delay
158     IF BMP_Cal_Scan_Done THEN
159         BMPCal_State := 99 ;      //only 1 pass
160         BMP_Cal_Scan_Start := 0 ;
161         BMP_Table_Loaded := 1 ;
162     END_IF
163
164 END_CASE
165
166 //State machine to read temp, then pressure with delay between.
167 CASE BMP_TP_State OF
168
169 0 :
170     BMP_DATAOUT1 [ 0 ] := 46 ;      //read temperature
171     BMP_TP_Scan_Start := 1 ;
172
173     //After delay read the data
174     IF BMP_TP_Scan_Done THEN
175         //Now read temperature data from the BMP085 sensor
176         BMP_DATAIN1 [ 0 ] := i2c_single_1 . DATAIN [ 0 ] ;
177         BMP_DATAIN1 [ 1 ] := i2c_single_1 . DATAIN [ 1 ] ;
178         BMP_TP_State := 10 ;
179         BMP_TP_Scan_Start := 0 ;
180     END_IF
```

```
181
182     10 :
183         BMP_DATAOUT1 [ 0 ] := 52 ;           //read pressure
184         BMP_TP_Scan_Start := 1 ;
185
186         //After delay read the data
187         IF BMP_TP_Scan_Done THEN
188             //Now read pressure data from the BMP085 sensor
189             BMP_DATAIN1 [ 2 ] := i2c_single_1 . DATAIN [ 0 ] ;
190             BMP_DATAIN1 [ 3 ] := i2c_single_1 . DATAIN [ 1 ] ;
191             BMP_DATAIN1 [ 4 ] := i2c_single_1 . DATAIN [ 2 ] ;
192             BMP_TP_State := 0 ;           //loop back to start
193             BMP_TP_Scan_Start := 0 ;
194         END_IF
195
196     END_CASE
197
198     //Read BMP085 sensor or test calculation
199     IF NOT Defaults_Enable THEN
200         BMP_temp := WORD_TO_INT ( Mem . PackBytesToWord ( BMP_DATAIN1 [ 0 ] , BMP_DATAIN1
201             [ 1 ] ) ) ;
202         BMP_pressure := DWORD_TO_DINT ( Mem . PackBytesToDWord ( 0 , 0 , BMP_DATAIN1 [ 2 ]
203             , BMP_DATAIN1 [ 3 ] ) ) ;
204     ELSE
205         BMP_temp := 27898 ;
206         BMP_pressure := 23843 ;
207     END_IF
208
209     //Only need to scan once after table is loaded into registers
210     IF ( BMP_Table_Loaded AND NOT BMP_Scanning ) OR Defaults_Enable THEN
211         IF NOT Defaults_Enable THEN
212             BMP_AC1 := WORD_TO_INT ( Mem . PackBytesToWord ( BMP_DATAIN2 [ 0 ] ,
213                 BMP_DATAIN2 [ 1 ] ) ) ;
214             BMP_AC2 := WORD_TO_INT ( Mem . PackBytesToWord ( BMP_DATAIN2 [ 2 ] ,
215                 BMP_DATAIN2 [ 3 ] ) ) ;
216             BMP_AC3 := WORD_TO_INT ( Mem . PackBytesToWord ( BMP_DATAIN2 [ 4 ] ,
217                 BMP_DATAIN2 [ 5 ] ) ) ;
218             BMP_AC4 := WORD_TO_UINT ( Mem . PackBytesToWord ( BMP_DATAIN2 [ 6 ] ,
219                 BMP_DATAIN2 [ 7 ] ) ) ;
220             BMP_AC5 := WORD_TO_UINT ( Mem . PackBytesToWord ( BMP_DATAIN2 [ 8 ] ,
221                 BMP_DATAIN2 [ 9 ] ) ) ;
222             BMP_AC6 := WORD_TO_UINT ( Mem . PackBytesToWord ( BMP_DATAIN2 [ 10 ] ,
223                 BMP_DATAIN2 [ 11 ] ) ) ;
224             BMP_B1 := WORD_TO_INT ( Mem . PackBytesToWord ( BMP_DATAIN2 [ 12 ] ,
225                 BMP_DATAIN2 [ 13 ] ) ) ;
226             BMP_B2 := WORD_TO_INT ( Mem . PackBytesToWord ( BMP_DATAIN2 [ 14 ] ,
227                 BMP_DATAIN2 [ 15 ] ) ) ;
228             BMP_MB := WORD_TO_INT ( Mem . PackBytesToWord ( BMP_DATAIN2 [ 16 ] ,
229                 BMP_DATAIN2 [ 17 ] ) ) ;
230             BMP_MC := WORD_TO_INT ( Mem . PackBytesToWord ( BMP_DATAIN2 [ 18 ] ,
```

```

221     BMP_DATAIN2 [ 19 ] ) ) ;
222     BMP_MD := WORD_TO_INT ( Mem . PackBytesToWord ( BMP_DATAIN2 [ 20 ] ,
223     BMP_DATAIN2 [ 21 ] ) ) ;
224
225     //Testing only
226     ELSE
227         BMP_AC1 := 408 ;
228         BMP_AC2 := - 72 ;
229         BMP_AC3 := - 14383 ;
230         BMP_AC4 := 32741 ;
231         BMP_AC5 := 32757 ;
232         BMP_AC6 := 23153 ;
233         BMP_B1 := 6190 ;
234         BMP_B2 := 4 ;
235         BMP_MB := - 32767 ;
236         BMP_MC := - 8711 ;
237         BMP_MD := 2868 ;
238     END_IF
239
240     //if these values are zero, then module is unplugged, load defaults
241     IF BMP_AC1 <> 0 OR BMP_AC2 <> 0 THEN
242         BMP_Scanning := 1 ;
243         BMP_Calc_Done := 1 ;
244     ELSE
245         Defaults_Enable := TRUE ;
246     END_IF
247
248     END_IF
249
250     //Only calculate if cal table is done loading
251     IF BMP_Scanning THEN
252         //Calculate temp
253         BMP_X1 := ( BMP_temp - BMP_AC6 ) * BMP_AC5 / 32768 ;
254         BMP_X2 := BMP_MC * 2048 / ( BMP_X1 + BMP_MD ) ;
255         BMP_B5 := BMP_X1 + BMP_X2 ;
256         BMP_T := ( BMP_B5 + 8 ) / 16 ;
257         BMP_ftemp := ( ( ( DINT_TO_REAL ( BMP_T ) / 10.0 * 9 ) / 5 ) + 32 ) ;
258
259         //Calculate Pressure
260         BMP_P_B6 := BMP_B5 - 4000 ;
261         BMP_P_X1 := ( BMP_B2 * ( BMP_P_B6 * BMP_P_B6 / 4096 ) ) / 2048 ;
262         BMP_P_X2 := BMP_AC2 * BMP_P_B6 / 2048 ;
263         BMP_P_X3 := BMP_P_X1 + BMP_P_X2 ;
264         BMP_P_B3 := ( SHL ( ( BMP_AC1 * 4 + BMP_P_X3 ) , 0 ) + 2 ) / 4 ;
265         BMP_P_X1_2 := BMP_AC3 * BMP_P_B6 / 8192 ;
266         BMP_P_X2_2 := ( BMP_B1 * ( BMP_P_B6 * BMP_P_B6 / 4096 ) ) / 65536 ;
267         BMP_P_X3_2 := ( ( BMP_P_X1_2 + BMP_P_X2_2 ) + 2 ) / 4 ;
268         BMP_P_B4 := BMP_AC4 * ( BMP_P_X3_2 + 32768 ) / 32768 ;
269         BMP_P_B7 := ( BMP_pressure - BMP_P_B3 ) * 50000 ;
270
271         IF BMP_P_B7 < 2147483648 THEN
272             BMP_P_P := ( BMP_P_B7 * 2 ) / BMP_P_B4 ;

```

```
271     ELSE
272         BMP_P_P := ( BMP_P_B7 / BMP_P_B4 ) * 2 ;
273     END_IF
274
275     BMP_P_X1_3 := ( BMP_P_P / 256 ) * ( BMP_P_P / 256 ) ;
276     BMP_P_X1_4 := ( BMP_P_X1_3 * 3038 ) / 65536 ;
277     BMP_P_X2_3 := ( - 7357 * BMP_P_P ) / 65536 ;
278     BMP_P_pfinal := BMP_P_P + ( BMP_P_X1_4 + BMP_P_X2_3 + 3791 ) / 16 ;
279     BMP_pascals := REAL_TO_DINT ( LINT_TO_REAL ( BMP_P_pfinal ) ) ;
280
281     //Moving average filter
282     Pressure_Filter ( X := BMP_pascals , N := Alt_Filt_Buffer , RST := , Y =>
BMP_alt_filtered ) ;
283
284     BMP_P_inHG := BMP_pascals / 3386.38816 ;
285     BMP_altitude := 44330.0 * ( 1 - ( EXPT ( ( ( BMP_pascals / 100 ) / ( 101325.0 /
100.0 ) ) , 0.1903 ) ) ) ;
286
287 END_IF
288
289 //Ctrl data for switching BMP085 data channel 46=temp, 52=pressure
290 //Write the data to the output registers
291 IF BMP_DATAOUT_old1 [ 0 ] <> BMP_DATAOUT1 [ 0 ] THEN
292     i2c_single_1 . DATAOUT [ 0 ] := BMP_DATAOUT1 [ 0 ] ;
293     BMP_DATAOUT_old1 [ 0 ] := BMP_DATAOUT1 [ 0 ] ;
294 END_IF
295
296
```