Program: masteroppgaveVers.CR6

```
'CR6 Series Datalogger
'Date: May 2018
'
'Program author:
'Bendik Olai Agdal (M.Sc. Cybernetics and Robotics 2018)
'bendik.agdal@gmail.com

'For syntax see CR6 Measurement and Control System Operators Manual


'////////////////////////////////////////////////////////////////////////
' USER DEFINED CONSTANTS
'////////////////////////////////////////////////////////////////////////

'TIME CONSTANTS
Const MAX_TIMEOUT_LEVEL2 = 10 'System enters fallback after this time
Const MAX_TIMEOUT_RADIO = 5   'System enters fallback after this time

Const INTERVAL_GPS = 1
Const INTERVAL_pwrSYSTEM_STATUS = 1
Const INTERVAL_pwrSYSTEM_DATA = 1
Const INTERVAL_IRIDIUM_RECEIVE = 86400
Const INTERVAL_IRIDIUM_SEND = 86400
Const INTERVAL_GET_LEVEL2_INSTRUCTION = 1
Const INTERVAL_GET_RADIO_INSTRUCTION = 1

Const BOOT_TIME_GPS = 100
Const BOOT_TIME_IRIDIUM = 100

'LEAK DETECTION
Const PUMP_CURRENT_THRESHOLD = 0.9 'A
Const INTERVAL_LEAK_CHECK = 1800 '30 minute interval

'I/O PORTS & PINS
Const LEVEL2_COM_PORT = ComU7
Const GARMIN_COM_PORT = ComC1
Const IRIDIUM_COM_PORT = ComC3
Const BLUESOLAR_COM_PORT = ComU3
Const RADIO_COM_PORT = ComU9

Const THRUSTER_PWM_PIN = u6
Const RUDDER_PWM_PIN = u5

'Relays
Const RUDDER_RELAY_PIN = u8
'Const LEVEL2_RELAY_PIN = u9 'TODO Currently used for 433 MHz radio
'Const LEVEL3_RELAY_PIN = u10 'TODO Currently used for 433 MHz radio
Const LEVEL2_AND_ECHOMAX_AND_AIS_RELAY_PIN = u11
Const PUMPS_RELAY_PIN = u12

'Sleep pins
Const IRIDIUM_SLEEP_PIN = u2
Const GPS_SLEEP_PIN = u1

'////////////////////////////////////////////////////////////////////////

```

```
57   'ERROR DEFINITIONS
58   Const SUCCESS = -100
59   Const ERROR = -101
60   Const NO_ANSWER = -102
61   Const CHECKSUM_ERROR = -103
62   Const MESSAGE_ERROR = -104
63   Const INCORRECT_RETURN_MESSAGE = -105
64   Const DEVICE_ERROR = -106
65   Const ERROR_READING_BUFFER = -107
66   Const BUFFER_EMPTY = -108
67   Const READ_SUCCESS_BUFFER_NOT_CLEARED = -109
68
69   'OPERATING MODES
70   '(Do not change - equally defined in level2 sw and radio control sw)
71   Const NORMAL = 1
72   Const FALLBACK = 2
73   Const MANUAL = 3
74
75   'FALLBACK MODES
76   '(Do not change - equally defined in level2 sw and radio control sw)
77   Const RUDDER_0_DEG = 0
78   Const CIRCLE = 1
79   Const AUTOPILOT = 2
80
81   'PI CONTROLLER
82   Const PI_MAX_U = 20
83   Const PI_MIN_U = -20
84   Const PI_KI = 1
85   Const PI_KP = 0.1
86
87   'OTHER (MISC.)
88   Const IRIDIUM_MO = 0 'mobile originated
89   Const IRIDIUM_MT = 1 'mobile terminated
90   Const ITIDIUM_MT_AND_MO = 2
91   Const NO_CHANGE = -100
92
93   '////////////////////////////////////////////////////////////////////
94   ' VARIABLE DECLARATIONS
95   '////////////////////////////////////////////////////////////////////
96
97   'Fallback related variables
98   Public prevSuccess_level2 As Long
99   Public prevSuccess_radio As Long
100  Public operatingState As Float 'NORMAL / FALLBACK / MANUAL
101  Public fallbackMode As Float  'AUTOPILOT / RUDDER_45DEG / RUDDER_0DEG
102
103  'Public Variables
104  Public debugRuntimeSec As Long
105  Public internalTemperature As Float
106  Public systemVoltage As Float
107
108  'Public debug1 As String * 100
109  'Public debug2 As String * 100
110  'Public debug3 As String * 100
111  'Public debug4 As String * 100
112  'Public debug5 As String * 100
```

```
113  'Public debug6 As String * 100
114
115  Public debugIerror As Float
116  Public debugKerror As Float
117
118  'rudder and thruster instruction variables
119  Public level2_desiredRudderAngle As Float
120  Public level2_desiredThrusterValue As Float
121  Public level2_desiredpwrSettings As String
122  Public level2_desiredFallbackMode As Float
123  Public radio_desiredRudderAngle As Float
124  Public radio_desiredThrusterValue As Float
125  Public radio_desiredFallbackMode As Float
126  Public radio_desiredpwrSettings As String
127
128  'gps variables
129  Dim gps_status As String 'obtained from GPRMC message
130  Dim gps_SOG As String 'GPRMC message
131  Dim gps_COG As String 'GPRMC message
132  Dim gps_date As String 'GPRMC message
133  Dim gps_UTC  As String 'GPGGA message
134  Dim gps_lat  As String 'GPGGA message
135  Dim gps_hemisphereSN  As String 'GPGGA message
136  Dim gps_long  As String 'GPGGA message
137  Dim gps_hemisphereEW  As String 'GPGGA message
138  Dim gps_quality  As String 'GPGGA message
139  Dim gps_numSat  As String 'GPGGA message
140  Dim gps_horizAcc As String 'GPGGA message
141
142  'blueSolar pwr system variables
143  Dim pwrSystem_State As Float
144  Dim pwrSystem_panelpwr As Float
145  Dim pwrSystem_errorCode As Float
146  Dim pwrSystem_loadCurrent As Float
147  Dim pwrSystem_loadpwr As Float
148
149  'Autopilot controller variable
150  Public autopilot_integratedError As Float 'integrated error value
151  Public autopilot_desiredRudderAngle As Float
152  Public autopilot_referenceCourseDeg As Float
153
154  'Error report counter variables
155  Public errors_resetTime As String
156  Public calls_level2 As Float
157  Public calls_garmin As Float
158  Public calls_iridium As Float
159  Public calls_pwrSystem As Float
160  Public error_level2_noResponse As Float
161  Public error_level2_checksum As Float
162  Public error_level2_incorrectReturnMessage As Float
163  Public error_garmin_noResponse As Float
164  Public error_garmin_incorrectMessage As Float
165  Public error_garmin_checksum As Float
166  Public error_garmin_device As Float
167  Public error_iridiumWrite_incorrectMessage As Float
168  Public error_iridiumRead_incorrectMessage As Float
```

```
169  Public error_iridiumRead_noResponse As Float
170  Public error_pwrSystem_noResponse As Float
171  Public error_pwrSystem_checksum As Float
172  Public error_pwrSystem_incorrectMessage As Float
173
174  'Global variables for power settings
175  Public disable_level2 As Float
176  Public disable_level3 As Float
177  Public disable_gps As Float
178  Public disable_iridium As Float
179  Public disable_echomaxAndAIS As Float
180  Public disable_pumps As Float
181  Public leakDetected As Float
182
183
184  DataTable (seaTrial, 1, 14400)
185    Sample (1, runtime, Float)
186    Sample (1, internalTemperature, Float)
187    Sample (1, systemVoltage, Float)
188    Sample (1, pwrSystem_State, Float)
189    Sample (1, pwrSystem_panelpwr,Float)
190    Sample (1, pwrSystem_loadpwr,Float)
191    Sample (1, leakDetected,Float)
192    Sample (1, gps_status,String)
193    Sample (1, gps_SOG,String)
194    Sample (1, gps_COG,String)
195    Sample (1, gps_lat,String)
196    Sample (1, gps_long,String)
197    Sample (1, operatingState,Float)
198    Sample (1, fallbackMode, Float)
199    Sample (1, level2_desiredRudderAngle,Float)
200    Sample (1, radio_desiredRudderAngle,Float)
201    Sample (1, autopilot_desiredRudderAngle, Float)
202    Sample (1, debugKerror, Float)
203    Sample (1, debugIerror, Float)
204  EndTable
205
206  DataTable (seaErrors, 1, 24)
207    Sample (1, errors_resetTime, String)
208    Sample (1, calls_level2, Float)
209    Sample (1, calls_garmin, Float)
210    Sample (1, calls_pwrSystem, Float)
211    Sample (1, error_level2_noResponse,Float)
212    Sample (1, error_level2_checksum,Float)
213    Sample (1, error_level2_incorrectReturnMessage,Float)
214    Sample (1, error_garmin_noResponse,Float)
215    Sample (1, error_garmin_incorrectMessage,Float)
216    Sample (1, error_garmin_device,Float)
217    Sample (1, error_pwrSystem_noResponse,Float)
218    Sample (1, error_pwrSystem_checksum,Float)
219    Sample (1, error_pwrSystem_incorrectMessage,Float)
220  EndTable
221
222  DataTable (seaRelays, 1, 240)
223    Sample (1, disable_level2, Float)
224    Sample (1, disable_level3, Float)
```

```
225     Sample (1, disable_gps, Float)
226     Sample (1, disable_echomaxAndAIS, Float)
227     Sample (1, disable_pumps,Float)
228   EndTable
229
230   DataTable (exeTime, 1, 100) 'For performance check purposes
231     Sample(1,runtime, Float)
232   EndTable
233
234   DataTable (gpsTable, 1, 100)
235     Sample(1, gps_status, String)
236     Sample(1, gps_SOG, String)
237     Sample(1, gps_COG, String)
238     Sample(1, gps_date, String)
239     Sample(1, gps_UTC, String)
240     Sample(1, gps_lat, String)
241     Sample(1, gps_hemisphereSN, String)
242     Sample(1, gps_long, String)
243     Sample(1, gps_hemisphereEW, String)
244     Sample(1, gps_quality, String)
245     Sample(1, gps_numSat, String)
246     Sample(1, gps_horizAcc, String)
247   EndTable
248
249   DataTable (pwrSystemTable, 1, 3600)
250     Sample(1, pwrSystem_panelpwr, Float)
251     Sample(1, pwrSystem_loadpwr, Float)
252     Sample(1, pwrSystem_loadCurrent,Float)
253     Sample(1, systemVoltage, Float)
254     'Sample(1, debugBlueSolarState, String)
255   EndTable
256
257   '////////////////////////////////////////////////////////////////////
258   'PROGRAM FUNCTIONS
259   '////////////////////////////////////////////////////////////////////
260
261   Function NMEA0183_ChecksumOK(input As String * 100) As Long
262       'Reads input string and compares NMEA0183 defined checksum with che
263       Dim checkSumProvided As String * 3
264       Dim messageContents(1) As String * 100
265       SplitStr(checkSumProvided, input,"*",1,4) 'Flag = 4 aka HEADERFILTE
266       SplitStr(messageContents(), input, "*",1,5) 'Flag = 5 aka FOOTERFIL
267       If CheckSum(messageContents,9,0) = HexToDec(checkSumProvided) Then
268         Return TRUE
269       End If
270       Return FALSE
271   EndFunction
272
273   Function setRudder(angle As Float) As Float
274     'input angle as integer value multiplied by 10.
275     'F.ex. angle = 105 --> 10.5 deg rudder angle
276
277     angle = -angle ' Quick fix due to signal definition in provided DC mo
278
279     Dim dutyCycle As Float
280     Dim ANGLE_MAX As Float
```

```
281    Dim ANGLE_MIN As Float
282    ANGLE_MAX = 450
283    ANGLE_MIN = -450
284
285    If angle < ANGLE_MIN OR angle > ANGLE_MAX Then
286      'Set rudder to 0 degrees
287      PWM(0.075, RUDDER_PWM_PIN, 20000, usec)
288      Return ERROR
289    EndIf
290
291    dutyCycle = (((angle/10 + 45) / 90) * 0.05) + 0.05 'Convert -450 to 4
292    PWM(dutyCycle, RUDDER_PWM_PIN, 20000, usec) 'set output signal
293    Return SUCCESS
294
295  EndFunction
296
297  Function setThruster(power As Float) As Float
298    'input desired thrust as integer in range -100 (reverse) to 100. 0 =
299
300    Dim dutyCycle As Float
301    Dim THRUSTER_MAX As Float
302    Dim THRUSTER_MIN As Float
303    THRUSTER_MAX = 100
304    THRUSTER_MIN = -100
305
306    power = -power 'Due to definition in motor control box
307
308    If power < THRUSTER_MIN OR power > THRUSTER_MAX Then
309       'Instruction outside physical constraints.
310      PWM(0.075, THRUSTER_PWM_PIN, 20000, usec) 'Set thruster to 0 pwr
311      Return -1
312    EndIf
313
314    dutyCycle = (((power + 100) / 200) * 0.05) + 0.05 'Convert -100 to 10
315    PWM(dutyCycle, THRUSTER_PWM_PIN, 20000, usec)
316    Return dutyCycle
317
318  EndFunction
319
320  Function setpwrSettings(instructionString As String) As Float
321    'Function sets global variables for relay and sleep pin control.
322    'Output pins are updated at 1 Hz in main loop (scan) in accordance to
323    'Expected input example: "010101". 1 means disable. 0 means enable.
324
325    If instructionString = "NO CHANGE" Then
326      Return 0
327    End If
328
329    If NOT (Len(instructionString) = 6) Then 'Simple error catch on input
330      Return ERROR
331    End If
332
333    disable_level2 = FALSE
334    disable_level3 = FALSE
335    disable_gps = FALSE
336    disable_iridium = FALSE
```

```
337     disable_echomaxAndAIS = FALSE
338     disable_pumps = FALSE
339
340     If Mid(instructionString, 1, 1) = 1 Then
341       disable_level2 = TRUE
342     End If
343     If Mid(instructionString, 2, 1) = 1 Then
344       disable_level3 = TRUE
345     End If
346     If Mid(instructionString, 3, 1) = 1 Then
347       disable_gps = TRUE
348     End If
349     If Mid(instructionString, 4, 1) = 1 Then
350       disable_iridium = TRUE
351     End If
352     If Mid(instructionString, 5, 1) = 1 Then
353       disable_echomaxAndAIS = TRUE
354     End If
355     If Mid(instructionString, 6, 1) = 1 Then
356       disable_pumps = TRUE
357     End If
358
359     Return SUCCESS
360
361   End Function
362
363   Function setFallbackMode(input As Float) As Float
364     If input = NO_CHANGE Then
365       Return 0 'Do not change fallback mode
366     End If
367     fallbackMode = input
368     Return SUCCESS
369
370   End Function
371
372   Function radio_readWrite() As Float
373       'Transmits system data on serial port connected to 433 MHz radio an
374       'Will update the following variables based on received instruction
375       '     - radio_desiredFallbackMode
376       '     - radio_desiredpwrSettings
377       '     - radio_desiredRudderAngle
378       '     - radio_desiredThrusterValue
379       '     - operatingState
380       '
381       'PS: This is the only sub-function that can change operatingState d
382       'OperatingState can only be changed to MANUAL (radio control mode)
383
384       'TRANSMIT MESSAGE
385       Dim pwrSettings As String * 6
386       pwrSettings = ABS(disable_level2) & ABS(disable_level3) & ABS(disab
387       Dim outputString As String * 100
388       outputString = "CR601," & ABS(leakDetected) & "," & pwrSettings & "
389       outputString = "$" & outputString & "*" + CheckSum(outputString,9,0
390       SerialOut(RADIO_COM_PORT, outputString, "", 0, 0)
391
392       'READ RETURNED MESSAGE
```

```
393       Dim serialInput As String * 100
394       Dim bytesReturned As Float
395       SerialInRecord (RADIO_COM_PORT,serialInput,&H24,0,&HA,bytesReturned
396
397       If serialInput = "NAN" OR serialInput = "" Then
398         Return NO_ANSWER
399       End If
400
401       If (NMEA0183_ChecksumOK(serialInput) = TRUE) Then
402         'PARSE MESSAGE
403         Dim message(1) As String * 100
404         Dim messageLine(16) As String * 20
405         SplitStr(message(), serialInput, "*",1,5) 'Flag = 5 aka FOOTERFIL
406         SplitStr(messageLine(), message(1), ",",16,5) 'Flag = 5 aka FOOTE
407
408         Dim msgType As String
409         msgType = messageLine(1)
410
411         If msgType = "RCC01" Then 'Message contains instructions
412
413           'GET DESIRED FALLBACK MODE
414           If messageLine(5) = "" Then 'Fallback mode not defined in input
415             radio_desiredFallbackMode = NO_CHANGE
416           Else
417             radio_desiredFallbackMode = messageLine(5)
418           End If
419
420           'GET DESIRED pwr SETTINGS
421           radio_desiredpwrSettings = messageLine(4)
422
423           'GET DESIRED RUDDER AND THRUSTER VALUES
424           radio_desiredRudderAngle = messageLine(2)
425           radio_desiredThrusterValue = messageLine(3)
426
427           If messageLine(6) = 1 Then
428             operatingState = MANUAL
429           Else
430             operatingState = FALLBACK
431           End If
432
433           Return SUCCESS
434
435         End If
436       End If
437
438       Return CHECKSUM_ERROR
439
440   End Function
441
442
443   Function level2_readWrite() As Float
444
445       'TRANSMIT MESSAGE
446       Dim pwrSettings As String * 6
447       pwrSettings = ABS(disable_level2) & ABS(disable_level3) & ABS(disab
448       Dim outputString As String * 100
```

```
449       outputString = "CR601," & ABS(leakDetected) & "," & pwrSettings & "
450       outputString = "$" & outputString & "*" + Hex(CheckSum(outputString
451
452       SerialOut(LEVEL2_COM_PORT, outputString, "", 0, 0)
453
454       'READ RETURNED MESSAGE
455       Dim serialInput As String * 100
456       Dim bytesReturned As Float
457       SerialInRecord (LEVEL2_COM_PORT,serialInput,&H24,0,&HA,bytesReturne
458
459       If serialInput = "NAN" OR serialInput = "" Then
460         Return NO_ANSWER
461       End If
462
463       If NMEA0183_ChecksumOK(serialInput) = TRUE Then
464         'PARSE MESSAGE
465         Dim message(1) As String * 100
466         Dim messageContents(16) As String * 20
467         SplitStr(message(), serialInput, "*",1,5) 'Flag = 5 aka FOOTERFIL
468         SplitStr(messageContents(), message(1), ",",16,5) 'Flag = 5 aka F
469
470         Dim msgType As String
471         msgType = messageContents(1)
472
473         If msgType = "BBB01" Then 'Message contains instructions
474
475           'GET DESIRED RUDDER AND THRUSTER VALUES
476           level2_desiredRudderAngle = messageContents(2)
477           level2_desiredThrusterValue = messageContents(3)
478
479           'GET DESIRED pwr SETTINGS
480           If messageContents(4) = "" Then 'pwr settings not defined in in
481             level2_desiredpwrSettings = "NO CHANGE"
482           Else
483             level2_desiredpwrSettings = messageContents(4)
484           End If
485
486           'GET DESIRED FALLBACK MODE
487           If messageContents(5) = "" Then 'Fallback mode not defined in i
488             level2_desiredFallbackMode = NO_CHANGE
489           Else
490             level2_desiredFallbackMode = messageContents(5)
491           End If
492
493           Return SUCCESS
494
495         Else
496           Return INCORRECT_RETURN_MESSAGE
497         End If
498
499       End If
500
501       Return CHECKSUM_ERROR
502
503   End Function
504
```

```
505  Function garmin_read() As Float
506    Dim SubStrings(16) As String * 32
507    Dim rawdata As String * 500
508    Dim CalculatedChecksum As Long, ReportedChecksum As Long
509
510    SerialIn(rawdata, GARMIN_COM_PORT, 1, 0, 500)
511
512    If Len(rawdata) = 0 Then
513      Return NO_ANSWER
514    EndIf
515
516    Dim inputLine(4) As String * 100
517    SplitStr(inputLine(), rawdata,"$", 4, 4)
518
519    Dim GPRMCmsg As String *100
520    Dim GPGGAmsg As String *100
521
522    'Get data from buffer. Attempt to get newest data (line4) first.
523    'At 1 Hz sampling, line 3 and 4 will be empty
524
525    If InStr (1,inputLine(4),"GPRMC",2) Then 'GPRMC MESSAGE
526      GPRMCmsg = inputLine(4)
527    ElseIf InStr (1,inputLine(4),"GPGGA",2) Then
528      GPGGAmsg = inputLine(4)
529    EndIf
530
531    If InStr (1,inputLine(3),"GPRMC",2) Then 'GPRMC MESSAGE
532      GPRMCmsg = inputLine(3)
533    ElseIf InStr (1,inputLine(3),"GPGGA",2) Then
534      GPGGAmsg = inputLine(3)
535    Else
536      'Line 3 and 4 were empty. Get data from line 1 and 2
537      If InStr (1,inputLine(2),"GPRMC",2) Then 'GPRMC MESSAGE
538      GPRMCmsg = inputLine(2)
539      ElseIf InStr (1,inputLine(2),"GPGGA",2) Then
540      GPGGAmsg = inputLine(2)
541      Else
542        Return INCORRECT_RETURN_MESSAGE
543      EndIf
544      If InStr (1,inputLine(1),"GPRMC",2) Then 'GPRMC MESSAGE
545      GPRMCmsg = inputLine(1)
546      ElseIf InStr (1,inputLine(1),"GPGGA",2) Then
547      GPGGAmsg = inputLine(1)
548      Else
549        Return INCORRECT_RETURN_MESSAGE
550      EndIf
551    EndIf
552
553    'GPRMC message
554    SplitStr (SubStrings(),GPRMCmsg,",",16,5)
555    gps_status = SubStrings(3) 'validity - A-ok, V-invalid
556    gps_SOG=SubStrings(8)
557    gps_COG=SubStrings(9)
558    gps_date=SubStrings(10)
559    CalculatedChecksum = CheckSum (GPRMCmsg,9,Len(GPRMCmsg) - 5)
560    CalculatedChecksum = CalculatedChecksum AND 255
```

```
561     ReportedChecksum = HexToDec(Mid(GPRMCmsg,Len(GPRMCmsg) - 3,2))
562
563     If NOT (CalculatedChecksum = ReportedChecksum) Then
564       Return CHECKSUM_ERROR
565     EndIf
566
567     'Parse GPGGA message
568     SplitStr (SubStrings(),GPGGAmsg,",",16,5)
569     gps_UTC=SubStrings(2)
570     gps_lat=SubStrings(3)
571     gps_hemisphereSN=SubStrings(4)
572     gps_long=SubStrings(5)
573     gps_hemisphereEW=SubStrings(6)
574     gps_quality=SubStrings(7) '0=invalid; 1=GPS fix; 2=Diff. GPS fix
575     gps_numSat=SubStrings(8)
576     gps_horizAcc=SubStrings(9)
577     CalculatedChecksum = CheckSum (GPGGAmsg,9,Len(GPGGAmsg) - 5)
578     CalculatedChecksum = CalculatedChecksum AND 255
579     ReportedChecksum = HexToDec(Mid(GPGGAmsg,Len(GPGGAmsg) - 3,2))
580
581     If NOT (CalculatedChecksum = ReportedChecksum) Then
582       Return CHECKSUM_ERROR
583     EndIf
584
585     If (gps_status = "V") OR (gps_quality = "0") Then
586       Return DEVICE_ERROR
587     End If
588
589     Return SUCCESS
590
591   EndFunction
592
593   Function iridium_messageOK(pointerToMessage As Long) As Long
594       'The last two chars in any message should read "OK"
595       If Mid(!pointerToMessage, Len(!pointerToMessage) - 3, 2) = "OK" The
596         Return 1
597       End If
598       Return 0
599   EndFunction
600
601   Function iridium_writeToOutbox(message As String * 100) As Float
602
603     Dim instruction As String * 100
604     Dim serialInput As String * 100
605     instruction = "AT+SBDWT=" & message & CHR (13) '13 = Carriage return
606     SerialOut(IRIDIUM_COM_PORT, instruction, "", 0, 0)
607     SerialIn(serialInput, IRIDIUM_COM_PORT, 5, 0, 100)
608
609     If Mid(serialInput, Len(!serialInput) - 3, 2) = "OK" Then
610       Return SUCCESS
611     End If
612
613     Return ERROR
614
615   End Function
616
```

```
617  Function iridium_readFromInbox(pointerToMessage As Long) As Long
618
619      Dim C3Input As String * 160
620      Dim instruction As String
621
622      'GET STATUS FOR MT BUFFER (Iridium inxbox)
623      instruction = "AT+SBDS" & CHR (13) 'CHR(13) = Carriage return
624      SerialOut(IRIDIUM_COM_PORT, instruction, "", 0, 0)
625      SerialIn(C3Input, IRIDIUM_COM_PORT, 5, 0, 160)
626
627      Dim messageWaiting As Float
628      If iridium_messageOK(@C3Input) Then
629        Dim statusArray(4) As String
630        C3Input = (Mid(C3Input, 18, 100)) 'Discard first 17 chars in mess
631        'Parse message
632        SplitStr(statusArray(), C3Input, ",", 4, 5)
633        'debugX = statusArray(1) 'MO flag
634        'debugX = statusArray(2) 'MOMSN
635        messageWaiting = statusArray(3) 'MT flag
636        'debug4 = statusArray(4) 'MTMSN
637      Else
638        Return INCORRECT_RETURN_MESSAGE 'UNABLE TO OBTAIN MT BUFFER STATU
639      EndIf
640
641      If NOT messageWaiting = 1 Then
642        Return BUFFER_EMPTY
643      EndIf
644
645
646      'READ MT BUFFER (Iridium inbox)
647      instruction = "AT+SBDRT" & CHR (13) 'CHR(13) = Carriage return
648      SerialOut(ComC3, instruction, "", 0, 0)
649      SerialIn(C3Input, ComC3, 5, 0, 160)
650
651      If iridium_messageOK(@C3Input) Then
652        'Extract message contents
653        !pointerToMessage = Mid(C3Input, 21, Len(C3Input) - 26) 'Discard
654      Else
655        Return ERROR_READING_BUFFER 'MESSAGE WAITING, BUT ERROR READING M
656      EndIf
657
658
659      'CLEAR MT BUFFER (Iridium Inbox)
660      instruction = "AT+SBDD1" & CHR (13) 'CHR(13) = Carriage return
661      SerialOut(IRIDIUM_COM_PORT, instruction, "", 0, 0)
662      SerialIn(C3Input, ComC3, 5, 0, 160)
663
664      If iridium_messageOK(@C3Input) Then
665        If (Mid(C3Input, 12, 1)) = 0 Then 'parse message and check value
666          'MT Buffer successfully cleared
667          Return SUCCESS 'Read buffer, MT buffer successfully cleared
668        Else
669          'An error occurred while clearing the buffer
670           Return READ_SUCCESS_BUFFER_NOT_CLEARED 'Read buffer, An error
671        EndIf
672      EndIf
```

```
673
674       Return READ_SUCCESS_BUFFER_NOT_CLEARED 'Read buffer, but present MT
675
676   EndFunction
677
678   Function iridium_CopyOutboxToInbox() As Float
679     'Used for debug/testing
680     Dim instruction As String
681     Dim serialInput As String * 100
682     instruction = "AT+SBDTC" & CHR (13) '13 = Carriage return
683     SerialOut(IRIDIUM_COM_PORT, instruction, "", 0, 0)
684     SerialIn(serialInput, IRIDIUM_COM_PORT, 5, 0, 100)
685   End Function
686
687   Function iridium_clearBuffer(selectedBuffer As Float) As Float
688     'Used for debug/testing - Clear MT and MO buffer
689     Dim instruction As String
690     Dim serialInput As String * 100
691     instruction = "AT+SBDD" & selectedBuffer & CHR (13) 'CHR(13) = Carria
692     SerialOut(IRIDIUM_COM_PORT, instruction, "", 0, 0)
693     SerialIn(serialInput, IRIDIUM_COM_PORT, 5, 0, 100)
694     Dim inputLine(5) As String
695     SplitStr(inputLine(), serialInput, CHR(13), 5, 5)
696     If NOT inputLine(1) = "AT+SBDD" & selectedBuffer Then
697       Return INCORRECT_RETURN_MESSAGE
698     Else If NOT inputLine(5) = "OK" Then
699       Return  MESSAGE_ERROR
700     End If
701     If InStr (1,inputLine(3),"0",2) Then 'Device should return "0" on lin
702       Return SUCCESS
703     EndIf
704     'Message is OK, but buffer was not cleared.
705     Return DEVICE_ERROR
706   End Function
707
708   Function iridium_receive() As Float
709     Dim instruction As String
710     Dim serialInput As String * 100
711     instruction = "AT+SBDI" & CHR (13) 'CHR(13) = Carriage return
712     SerialOut(IRIDIUM_COM_PORT, instruction, "", 0, 0)
713     SerialIn(serialInput, IRIDIUM_COM_PORT, 5, 0, 100)
714
715     If Len(serialInput) = 0 Then
716       Return NO_ANSWER
717     End If
718
719     If iridium_messageOK(@serialInput) = FALSE Then
720       Return INCORRECT_RETURN_MESSAGE 'Comm. error
721     End If
722
723     Dim statusArray(6) As String
724     serialInput = (Mid(serialInput, 8, 100)) 'Discard first 8 chars in me
725
726     'Parse message
727     '+SBDI:<MO status>,<MOMSN>,<MT status>,<MTMSN>,<MT length>,<MTqueued>
728     Dim MO_status, MOMSN, MT_status, MTMSN, MT_length, MT_queued As Float
```

```
729     SplitStr(statusArray(), serialInput, ",", 4, 5)
730     MO_status = statusArray(1)
731     MOMSN = statusArray(2)
732     MT_status = statusArray(3)
733     MTMSN = statusArray(4)
734     MT_length = statusArray(5)
735     MT_queued = statusArray(6)
736
737     If MT_status = 1 Then 'Ref ISU AT Command Reference v5
738       Return SUCCESS
739     End If
740
741     Return ERROR
742
743   End Function
744
745   Function iridium_send() As Float
746     Dim instruction As String
747     Dim serialInput As String * 100
748     instruction = "AT+SBDI" & CHR (13) 'CHR(13) = Carriage return
749     SerialOut(IRIDIUM_COM_PORT, instruction, "", 0, 0)
750     SerialIn(serialInput, IRIDIUM_COM_PORT, 5, 0, 100)
751
752     If Len(serialInput) = 0 Then
753       Return NO_ANSWER
754     End If
755
756     If Mid(serialInput, Len(!serialInput) - 3, 2) = "OK" Then
757       Return SUCCESS
758     End If
759
760     Dim statusArray(6) As String
761     serialInput = (Mid(serialInput, 8, 100)) 'Discard first 8 chars in me
762
763     'Parse message
764     '+SBDI:<MO status>,<MOMSN>,<MT status>,<MTMSN>,<MT length>,<MTqueued>
765     Dim MO_status, MOMSN, MT_status, MTMSN, MT_length, MT_queued As Float
766     SplitStr(statusArray(), serialInput, ",", 4, 5)
767     MO_status = statusArray(1)
768     MOMSN = statusArray(2)
769     MT_status = statusArray(3)
770     MTMSN = statusArray(4)
771     MT_length = statusArray(5)
772     MT_queued = statusArray(6)
773
774     If MO_status = 1 Then 'Ref ISU AT Command Reference v5
775       Return SUCCESS
776     EndIf
777
778     Return ERROR
779
780   End Function
781
782   Function iridium_signalQuality() As Float
783     Dim instruction As String
784     Dim serialInput As String * 100
```

```
785     instruction = "AT+CSQ" & CHR (13) 'CHR(13) = Carriage return
786     SerialOut(IRIDIUM_COM_PORT, instruction, "", 0, 0)
787     SerialIn(serialInput, IRIDIUM_COM_PORT, 5, 0, 100)
788     If iridium_messageOK(@serialInput) Then
789       Dim signalStrength As Float
790       signalStrength = (Mid(serialInput, 15, 1)) 'Extract char
791       Return signalStrength
792     End If
793     Return INCORRECT_RETURN_MESSAGE 'Comm. error
794   End Function
795
796   Function pwrSystem_checkSumOK(message As String) As Float
797     'For info. see VE.Direct HEX protocoll
798     'message format: ":[instruction][dataValue1]...[dataValueN][checkSum]
799     Dim instruction As String * 1
800     Dim dataValueStr As String * 2
801     Dim providedSum As String * 2
802     providedSum = Mid(message,Len(message) - 1,2) 'get checksum
803     instruction = Mid(message, 2, 1)
804     Dim sum As Float
805     sum = 0
806     Dim n As Float
807     For n = 1 To (Len(message)-3)/2 'sum all data values
808       dataValueStr = Mid(message, 3 + (n-1)*2, 2) 'extract data value fro
809       sum = sum + HexToDec(dataValueStr)
810     Next
811     sum = sum + HexToDec(instruction) + HexToDec(providedSum)
812     sum = sum MOD 256
813     If sum = 85 Then '0d85 = 0x55
814       Return 1
815     EndIf
816     Return -1
817   End Function
818
819   Function pwrSystem_getLoadCurrent() As Float
820     Dim instruction As String
821     Dim serialInput As String * 100
822     Dim message As String
823     instruction = ":7ADEDB4" & CHR (10) 'CHR(10) = newline
824     SerialOpen (BLUESOLAR_COM_PORT,19200,19,0,24)
825     SerialOut(BLUESOLAR_COM_PORT, instruction, "", 0, 0)
826     SerialIn(serialInput, BLUESOLAR_COM_PORT, 5, 0, 200)
827     SerialClose(BLUESOLAR_COM_PORT)
828
829     'Remove excessive characters
830     SplitStr(message, serialInput, CHR(10), 1,5) 'Discard bytes that succ
831     SplitStr(message, message, CHR(58), 1, 4)    'Discard bytes that prec
832
833     'Error checks
834     If message = "" Then
835       Return NO_ANSWER
836     ElseIf  pwrSystem_checkSumOK(message) = FALSE Then
837       Return CHECKSUM_ERROR
838     ElseIf NOT (Mid(instruction, 3,4) = Mid(message,2,4)) Then
839       Return INCORRECT_RETURN_MESSAGE
840     EndIf
```

```
841     'Parse data
842     Dim dataIn As String * 2
843     dataIn = Mid(message, 8, 2)
844     Return (HexToDec(dataIn) * 0.1) 'Unit: A
845   End Function
846
847   Function pwrSystem_getDeviceState() As Float
848     Dim instruction As String
849     Dim serialInput As String * 100
850     Dim message As String
851     instruction = ":701024B" & CHR (10) 'CHR(10) = newline
852     SerialOpen (BLUESOLAR_COM_PORT,19200,19,0,24)
853     SerialOut(BLUESOLAR_COM_PORT, instruction, "", 0, 0)
854     SerialIn(serialInput, BLUESOLAR_COM_PORT, 1, 0, 200)
855     SerialClose(BLUESOLAR_COM_PORT)
856
857     'Remove excessive characters
858     SplitStr(message, serialInput, CHR(10), 1,5) 'Discard bytes that succ
859     SplitStr(message, message, CHR(58), 1, 4)    'Discard bytes that prec
860
861     'Error checks
862     If message = "" Then
863       Return NO_ANSWER
864     ElseIf pwrSystem_checkSumOK(message) = FALSE Then 'TODO Case not test
865       Return CHECKSUM_ERROR
866     ElseIf NOT Mid(instruction, 3,4) = Mid(message,2,4) Then
867       Return INCORRECT_RETURN_MESSAGE
868     EndIf
869     'Parse messsage
870     Return Mid(message, 9, 1) 'Note: TODO: Test that the correct characte
871   End Function
872
873   Function pwrSystem_getPanelpwr() As Float
874     Dim instruction As String
875     Dim serialInput As String * 100
876     Dim message As String
877     instruction = ":7BCEDA5" & CHR (10) 'CHR(10) = newline
878     SerialOpen (BLUESOLAR_COM_PORT,19200,19,0,24)
879     SerialOut(BLUESOLAR_COM_PORT, instruction, "", 0, 0)
880     SerialIn(serialInput, BLUESOLAR_COM_PORT, 5, 0, 200) 'FIX TIMEOUT TIM
881     SerialClose(BLUESOLAR_COM_PORT)
882     'Remove excessive characters
883
884     SplitStr(message, serialInput, CHR(10), 1,5) 'Discard bytes that succ
885     SplitStr(message, message, CHR(58), 1, 4)    'Discard bytes that prec
886
887     'debugGetPanelpwr = serialInput
888
889     'Error checks
890     If message = "" Then
891       Return NO_ANSWER
892     ElseIf pwrSystem_checkSumOK(message) = FALSE Then
893       Return CHECKSUM_ERROR
894     ElseIf NOT Mid(instruction, 3,4) = Mid(message,2,4) Then
895       Return INCORRECT_RETURN_MESSAGE
896     EndIf
```

```
897
898     'Parse messsage
899     Dim dataVal As String * 10
900     dataVal = Mid(message,10,2) & Mid(message,8,2) 'Little endian. Only t
901     Return HexToDec(dataVal) * 0.01 'returned value [Watts]
902   End Function
903
904
905   Function pwrSystem_getChargerErrorCode() As Float
906     Dim instruction As String
907     Dim serialInput As String * 100
908     Dim message As String
909     instruction = ":7DAED87" & CHR (10) 'CHR(10) = newline
910     'instruction = ":7EFED38"
911     SerialOpen (BLUESOLAR_COM_PORT,19200,19,0,24)
912     SerialOut(BLUESOLAR_COM_PORT, instruction, "", 0, 0)
913     SerialIn(serialInput, BLUESOLAR_COM_PORT, 5, 0, 200)
914     SerialClose(BLUESOLAR_COM_PORT)
915     SplitStr(message, serialInput, CHR(10), 1,5) 'Discard bytes that succ
916     SplitStr(message, message, CHR(58), 1, 4)    'Discard bytes that prec
917
918     'Error checks
919     If message = "" Then
920       Return NO_ANSWER
921     ElseIf pwrSystem_checkSumOK(message) = FALSE Then
922       Return CHECKSUM_ERROR
923     ElseIf NOT Mid(instruction, 3,4) = Mid(message,2,4) Then
924       Return INCORRECT_RETURN_MESSAGE
925     EndIf
926     'Parse messsage
927     Dim dataVal As String * 10
928     dataVal = Mid(message,8,2) & Mid(message, 6,2)
929     Return HexToDec(dataVal) 'Note: Little endian used
930   End Function
931
932   Function pwrSystem_loadRelayEnable(enableRelay As Float) As Float
933     Dim instruction As String
934     Dim serialInput As String * 100
935     Dim message As String
936     If enableRelay Then
937       instruction = ":8ABED0004B1" & CHR (10) 'CHR(10) = newline
938     Else
939       instruction = ":8ABED0000B5" & CHR (10) 'Disable
940     EndIf
941     SerialOpen (BLUESOLAR_COM_PORT,19200,19,0,24)
942     SerialOut(BLUESOLAR_COM_PORT, instruction, "", 0, 0)
943     SerialIn(serialInput, BLUESOLAR_COM_PORT, 10, 0, 200)
944     SerialClose(BLUESOLAR_COM_PORT)
945     SplitStr(message, serialInput, CHR(10), 1,5) 'Discard bytes that succ
946     SplitStr(message, message, CHR(58), 1, 4)    'Discard bytes that prec
947
948     If message = "" Then
949       Return NO_ANSWER
950     ElseIf pwrSystem_checkSumOK(message) = FALSE Then 'TODO case not test
951       Return CHECKSUM_ERROR
952     ElseIf NOT Mid(instruction, 3,4) = Mid(message,2,4) Then
```

```
953       Return INCORRECT_RETURN_MESSAGE
954     EndIf
955     Return enableRelay
956   End Function
957
958   Function toHexString(x As Float) As String * 2
959     'Takes a number and returns a two character string. Maximum represen
960     If x > 255 Then
961       x = 255
962     End If
963     Dim output As String * 2
964     output = Hex(x)
965     If Len(output) = 1 Then
966       output = "0" & output
967     End If
968     Return output
969   End Function
970
971   Function report_systemErrorDetected() As Float
972     If error_level2_noResponse > 0 Then
973       Return 1
974     ElseIf error_level2_checksum > 0 Then
975       Return 1
976     ElseIf error_garmin_noResponse > 0 Then
977       Return 1
978     ElseIf error_garmin_incorrectMessage > 0 Then
979       Return 1
980     ElseIf error_garmin_checksum > 0 Then
981       Return 1
982     ElseIf error_garmin_device > 0 Then
983       Return 1
984     ElseIf error_iridiumWrite_incorrectMessage > 0 Then
985       Return 1
986     Else If error_iridiumRead_incorrectMessage > 0 Then
987       Return 1
988     Else If error_iridiumRead_noResponse > 0 Then
989       Return 1
990     Else If error_pwrSystem_noResponse > 0 Then
991       Return 1
992     Else If error_pwrSystem_checksum > 0 Then
993       Return 1
994     Else If error_pwrSystem_incorrectMessage > 0 Then
995       Return 1
996     End If
997     Return 0
998   End Function
999
1000  Function getDTG() As String
1001    'returns [DAY OF MONTH][HOUR][MINUTE]
1002    Dim rTime(9) As Long
1003    RealTime(rTime())
1004    Return rTime(3) & rTime(4) & rTime(5)
1005  End Function
1006
1007  Function resetErrorCount() As Float
1008    'Error report counter variables
```

```
1009    errors_resetTime = getDTG()
1010    calls_level2 = 0
1011    calls_garmin = 0
1012    calls_iridium = 0
1013    calls_pwrSystem = 0
1014    error_level2_noResponse = 0
1015    error_level2_checksum = 0
1016    error_level2_incorrectReturnMessage = 0
1017    error_garmin_noResponse = 0
1018    error_garmin_incorrectMessage = 0
1019    error_garmin_checksum = 0
1020    error_garmin_device = 0
1021    error_iridiumWrite_incorrectMessage = 0
1022    error_iridiumRead_incorrectMessage = 0
1023    error_iridiumRead_noResponse = 0
1024    error_pwrSystem_noResponse = 0
1025    error_pwrSystem_checksum = 0
1026    error_pwrSystem_incorrectMessage = 0
1027    Return 0
1028  End Function
1029
1030  Function report_generate() As String * 200
1031    'Outputs a system report string that can be transmitted on Iridium o
1032    'TODO: Generated report string desired is a suggestion. Can easily b
1033
1034    Dim report As String * 200
1035    '[DATE][TIME][MODE][AVG_PV][AVG_LOAD][POSITION][COG][SOG][RELAYS][LA
1036
1037    Dim pwrSettings As String * 6
1038    pwrSettings = ABS(disable_level2) & ABS(disable_level3) & ABS(disabl
1039
1040    'Generate device calls string. Representation: Two hexadecimal chara
1041    Dim deviceCalls As String
1042    deviceCalls =  errors_resetTime & "," & toHexString(calls_level2) &
1043
1044    'Multiple redefintions of report for increased readability. Can be w
1045    report = deviceCalls
1046    report = report & "," & toHexString(error_level2_noResponse) & toHex
1047    report = report & toHexString(error_garmin_checksum ) & toHexString(
1048    report = report & toHexString(error_iridiumRead_noResponse ) & toHex
1049
1050    report = gps_date & gps_UTC & gps_hemisphereSN & gps_lat & gps_hemis
1051
1052    If report_systemErrorDetected() Then
1053     'Append log of errors
1054     report = report & "," & toHexString(error_level2_noResponse) & toHe
1055     report = report & toHexString(error_garmin_checksum ) & toHexString
1056     report = report & toHexString(error_iridiumRead_noResponse ) & toHe
1057    End If
1058
1059    Return report
1060
1061  End Function
1062
1063  Function report_leak(currentIncrease As Float) As Float
1064    Dim result As Float
```

```
1065    Dim outputMessage As String
1066    outputMessage = "LEAK DETECTED " & getDTG & ",+" & currentIncrease &
1067    result = iridium_writeToOutbox(outputMessage)
1068    If result = SUCCESS Then
1069      result = iridium_send()
1070      If result = SUCCESS Then
1071        'Message sent. Clear the ouput buffer
1072        result = iridium_clearBuffer(IRIDIUM_MO)
1073        If result = SUCCESS Then
1074          'Procedure complete
1075          Return SUCCESS
1076        End If
1077      End If
1078    EndIf
1079  End Function
1080
1081  Function autopilot_piController(desiredCourse As Float) As Float
1082    'Will be called every second (1 Hz)
1083    'Uses gps_COG
1084
1085     'CHECK IF GPS SIGNAL IS VALID
1086     If NOT (gps_status = "A") Then
1087       'GPS signal bad
1088       autopilot_integratedError = 0 'reset integrated error
1089       Return 0 'Set rudder angle to zero
1090     End If
1091
1092     'PI CONTROLLER
1093
1094     Dim error_value As Float
1095     Dim new_integrated_error_value As Float
1096     Dim output As Float
1097     Dim saturation As Float
1098     saturation = FALSE
1099
1100     error_value = desiredCourse - gps_COG
1101     If (error_value > 180) Then
1102       error_value = error_value - 360
1103     ElseIf (error_value < -180) Then
1104       error_value = error_value + 360
1105     End If
1106
1107     new_integrated_error_value = autopilot_integratedError + error_valu
1108     output = error_value * PI_KP + new_integrated_error_value * PI_KI
1109
1110     debugKerror = error_value * PI_KP
1111     debugIerror = new_integrated_error_value * PI_KI
1112
1113    'CHECK OUTPUT SATURATION LIMITS
1114
1115     If (output < PI_MIN_U) Then
1116       output = PI_MIN_U
1117
1118       If (error_value < 0) Then
1119         '=> integrated error value was further DECREASED
1120         'don't save the further integrated signal value
```

```
1121          Return output * 10
1122        Else
1123          'ouput saturation, but integrated error value was increased.
1124          autopilot_integratedError = new_integrated_error_value
1125          Return output * 10
1126        EndIf
1127
1128      ElseIf (output > PI_MAX_U) Then
1129        output = PI_MAX_U
1130        If (error_value > 0) Then
1131          '=> integrated error value was further INCREASED
1132          'don't save the further integrated signal value
1133          Return output * 10
1134        Else
1135          'ouput saturation, but integrated error value was decreased.
1136          autopilot_integratedError = new_integrated_error_value
1137          Return output * 10
1138        EndIf
1139
1140      End If
1141
1142      'NO SATURATION:
1143      autopilot_integratedError = new_integrated_error_value 'update inte
1144      Return output * 10
1145
1146  End Function
1147
1148
1149  SequentialMode
1150
1151  BeginProg
1152
1153  '////////////////////////////////////////////////////////////////////
1154  'INITIAL USV LEVEL 1 SETTINGS [STARTS HERE]
1155  '////////////////////////////////////////////////////////////////////
1156
1157  operatingState = NORMAL
1158  fallbackMode = RUDDER_0_DEG
1159
1160  prevSuccess_level2 = 3 'Will not enter fallback during first 3 secs of
1161  prevSuccess_radio = 3 'Will not enter fallback during first 3 secs of
1162
1163  autopilot_referenceCourseDeg = 180 'South
1164
1165  'INITIAL POWER SETTINGS
1166  disable_gps = FALSE
1167  disable_iridium = FALSE
1168  disable_pumps = FALSE
1169  disable_echomaxAndAIS = FALSE
1170  disable_level2 = FALSE
1171  disable_level3 = FALSE
1172
1173  '////////////////////////////////////////////////////////////////////
1174  'INITIAL USV LEVEL 1 SETTINGS [ENDS HERE]
1175  '////////////////////////////////////////////////////////////////////
1176
```

```
1177  'INITIAL CHECK TIMES
1178  Dim next_gpsUpdate As Long
1179  Dim next_level2 As Long
1180  Dim next_getpwrSystemStatus As Long
1181  Dim next_getpwrSystemData As Long
1182  Dim next_iridiumInboxCheck As Long
1183  Dim next_iridiumSendReport As Long
1184  Dim next_LeakCheck As Long
1185  Dim next_radio As Long
1186
1187  next_level2 = INTERVAL_GET_LEVEL2_INSTRUCTION
1188  next_gpsUpdate = INTERVAL_GPS
1189  next_getpwrSystemStatus = INTERVAL_pwrSYSTEM_STATUS
1190  next_getpwrSystemData = INTERVAL_pwrSYSTEM_DATA
1191  next_iridiumInboxCheck = INTERVAL_IRIDIUM_RECEIVE
1192  next_iridiumSendReport = INTERVAL_IRIDIUM_SEND
1193  next_LeakCheck = INTERVAL_LEAK_CHECK
1194  next_radio = INTERVAL_GET_RADIO_INSTRUCTION
1195
1196  level2_desiredRudderAngle = 0
1197  level2_desiredThrusterValue = 0
1198  radio_desiredRudderAngle = 0
1199  radio_desiredThrusterValue = 0
1200  level2_desiredFallbackMode = NO_CHANGE
1201  radio_desiredFallbackMode = NO_CHANGE
1202  level2_desiredpwrSettings = "NO CHANGE"
1203  radio_desiredpwrSettings = "NO CHANGE"
1204
1205  Dim currentMeasurement(6) As Float 'Used for leak calculation. Must be
1206
1207  Dim runtime As Long 'Time since boot in seconds
1208  runtime = 0
1209
1210  'Initial integrated error value for autopilot PI-controller error sign
1211  autopilot_integratedError = 0
1212
1213  'TODO - Try to keep the ports closed and open only when needed.
1214  'Cannot be done with GPS at 1 HZ since it's a data stream - not polled
1215  SerialOpen (LEVEL2_COM_PORT,9600,16,0,100,1)
1216  SerialOpen (IRIDIUM_COM_PORT,19200,3,0,200,0)
1217  SerialOpen (RADIO_COM_PORT,57600,16,0,200)
1218  SerialOpen (GARMIN_COM_PORT,38400,3,0,157)
1219
1220
1221  'MAIN PROGRAM LOOP
1222
1223    Scan (1,Sec,0,0)
1224
1225      runtime += 1
1226      debugRuntimeSec = runtime
1227
1228      PanelTemp (internalTemperature,15000)
1229      Battery (systemVoltage)
1230      CallTable seaTrial 'Save system data
1231
1232      If (runtime MOD 10 = 0) Then
```

```
1233        CallTable seaRelays
1234      EndIf
1235
1236      If (runtime MOD 30 = 0) Then 'Save error count every 30 secs
1237        CallTable seaErrors
1238        resetErrorCount()
1239      EndIf
1240
1241
1242
1243      '////////////////////////////////////////////////////////////////
1244      'SET RELAYS
1245      '////////////////////////////////////////////////////////////////
1246
1247      'GPS
1248      If (runtime >= (next_gpsUpdate - BOOT_TIME_GPS)) AND (NOT (disable
1249        PortSet (GPS_SLEEP_PIN, 0) 'GPS-->NOT Sleep
1250      Else
1251        PortSet (GPS_SLEEP_PIN, 1) 'GPS-->Sleep
1252      EndIf
1253
1254      'IRIDIUM
1255      If (runtime >= (next_iridiumInboxCheck - BOOT_TIME_IRIDIUM) OR (ru
1256        PortSet (IRIDIUM_SLEEP_PIN, 1)
1257      Else
1258        PortSet (IRIDIUM_SLEEP_PIN, 0)
1259      EndIf
1260
1261      'LEVEL 2
1262      If (disable_level2 = TRUE) Then
1263        PortSet(LEVEL2_AND_ECHOMAX_AND_AIS_RELAY_PIN, 0)
1264      Else
1265        PortSet(LEVEL2_AND_ECHOMAX_AND_AIS_RELAY_PIN, 1)
1266      End If
1267
1268      'LEVEL 3
1269      'If (disable_level3 = TRUE) Then
1270      '  PortSet(LEVEL3_RELAY_PIN, 0)
1271      'Else
1272      '  PortSet(LEVEL3_RELAY_PIN, 1)
1273      'End If
1274
1275      'ECHOMAX & AIS
1276      'If (disable_echomaxAndAIS = TRUE) Then
1277      '  PortSet(ECHOMAX_AND_AIS_RELAY_PIN, 0)
1278      'Else
1279      '  PortSet(ECHOMAX_AND_AIS_RELAY_PIN, 1)
1280      'End If
1281
1282      'PUMPS
1283      If (leakDetected = TRUE) AND (NOT disable_pumps = TRUE) Then
1284        PortSet(PUMPS_RELAY_PIN, 1)
1285      End If
1286
1287      '////////////////////////////////////////////////////////////////
1288      'UPDATA GPS DATA
```

```
1289      '////////////////////////////////////////////////////////////////
1290
1291      Dim result As Float
1292
1293      'UPDATE GPS DATA
1294      If runtime >= next_gpsUpdate Then
1295        calls_garmin += 1
1296        result = garmin_read() 'Updates public variables
1297
1298        CallTable gpsTable
1299        If result = NO_ANSWER Then
1300          error_garmin_noResponse += 1
1301          gps_status = "V" 'gps invalid data
1302        ElseIf result = DEVICE_ERROR Then
1303          error_garmin_device += 1
1304          gps_status = "V"  'gps invalid data
1305        ElseIf result = CHECKSUM_ERROR Then
1306          error_garmin_checksum  += 1
1307          gps_status = "V"  'gps invalid data
1308        EndIf
1309        next_gpsUpdate = runtime + INTERVAL_GPS
1310      EndIf
1311
1312      '////////////////////////////////////////////////////////////////
1313      'AUTOPILOT DESIRED ANGLE
1314      '////////////////////////////////////////////////////////////////
1315
1316      'Should run early in loop to ensure 1 Hz frequency for integrator
1317      autopilot_desiredRudderAngle = autopilot_piController(autopilot_re
1318
1319      '////////////////////////////////////////////////////////////////
1320      'PING LEVEL 1 AND RADIO CONTROL
1321      '////////////////////////////////////////////////////////////////
1322
1323      'PING 433 MHz RADIO (MANUAL CONTROL)
1324      If runtime >= next_radio Then
1325        calls_level2 += 1
1326        result = radio_readWrite()
1327        If result = SUCCESS Then
1328          prevSuccess_radio = runtime
1329          'radio_desiredRudderAngle was set
1330          'radio_desiredThrusterValue was set
1331          'radio_desiredpwrSettings was set
1332          'radio_desiredFallbackMode was set
1333
1334        ElseIf result = CHECKSUM_ERROR Then
1335          'radioMode = FALSE
1336        ElseIf result = NO_ANSWER Then
1337          'radioMode = FALSE
1338          'error_level2_noResponse += 1
1339        EndIf
1340        next_radio = runtime + INTERVAL_GET_RADIO_INSTRUCTION
1341      EndIf
1342
1343      'PING LEVEL 2 CONTROL SYSTEM
1344      If (runtime >= next_level2) Then
```

```
1345        calls_level2 += 1
1346        result = level2_readWrite()
1347        If result = SUCCESS Then
1348          prevSuccess_level2 = runtime
1349          'level2_desiredRudderAngle was set
1350          'level2_desiredThrusterValue was set
1351          'level2_desiredpwrSettings was set
1352          'level2_desiredFallbackMode was set
1353
1354        ElseIf result = CHECKSUM_ERROR Then
1355          error_level2_checksum += 1
1356        ElseIf result = INCORRECT_RETURN_MESSAGE Then
1357          error_level2_incorrectReturnMessage += 1
1358        ElseIf result = NO_ANSWER Then
1359          error_level2_noResponse += 1
1360        EndIf
1361        next_level2 = runtime + INTERVAL_GET_LEVEL2_INSTRUCTION
1362      EndIf
1363
1364
1365      '////////////////////////////////////////////////////////////////
1366      'ENTER OR EXIT FALLBACK MODE?
1367      '////////////////////////////////////////////////////////////////
1368
1369      If operatingState = NORMAL Then
1370        If (runtime - prevSuccess_level2) >= MAX_TIMEOUT_LEVEL2 Then
1371          operatingState = FALLBACK
1372        EndIf
1373      EndIf
1374
1375      If operatingState = FALLBACK Then
1376        If (runtime - prevSuccess_level2) < MAX_TIMEOUT_LEVEL2 Then
1377          operatingState = NORMAL
1378        ElseIf (runtime - prevSuccess_level2) < MAX_TIMEOUT_RADIO Then
1379          operatingState = MANUAL
1380        EndIf
1381      EndIf
1382
1383      If operatingState = MANUAL Then
1384        If (runtime - prevSuccess_radio) >= MAX_TIMEOUT_RADIO Then
1385          operatingState = FALLBACK
1386        EndIf
1387      EndIf
1388
1389  '
1390  '
1391  '
1392  '
1393  '
1394  '
1395  '
1396  '
1397  '
1398  '
1399  '
1400  '
```

```
1401  '
1402
1403
1404
1405
1406      '////////////////////////////////////////////////////////////////
1407      'SET RUDDER AND THRUSTER
1408      '////////////////////////////////////////////////////////////////
1409
1410      Select Case operatingState
1411
1412      Case NORMAL
1413        setRudder(level2_desiredRudderAngle)
1414        setThruster(level2_desiredThrusterValue)
1415        setpwrSettings(level2_desiredpwrSettings)
1416        setFallbackMode(level2_desiredFallbackMode)
1417
1418      Case MANUAL
1419        setRudder(radio_desiredRudderAngle)
1420        setThruster(radio_desiredThrusterValue)
1421        setpwrSettings(radio_desiredpwrSettings)
1422        setFallbackMode(radio_desiredFallbackMode)
1423
1424      Case Else 'FALLBACK
1425        If fallbackMode = AUTOPILOT Then
1426          setRudder(autopilot_desiredRudderAngle)
1427          setThruster(0)
1428        Else If fallbackMode = CIRCLE Then
1429          setRudder(450)
1430          setThruster(0)
1431        Else 'IDLE
1432          setRudder(0)
1433          setThruster(0)
1434        EndIf
1435
1436      EndSelect
1437
1438      '////////////////////////////////////////////////////////////////
1439      'SYSTEM MONITORING
1440      '////////////////////////////////////////////////////////////////
1441
1442      'UPDATE POWER SYSTEM STATUS
1443      If runtime >= next_getpwrSystemStatus Then
1444        calls_pwrSystem += 1
1445        result = pwrSystem_getDeviceState()
1446        If result = NO_ANSWER Then
1447          error_pwrSystem_noResponse += 1
1448        ElseIf result = INCORRECT_RETURN_MESSAGE Then
1449          error_pwrSystem_incorrectMessage += 1
1450        ElseIf result = CHECKSUM_ERROR Then
1451          error_pwrSystem_checksum += 1
1452        Else
1453          pwrSystem_State = result
1454        EndIf
1455
1456        If pwrSystem_State = 2 Then 'Indicates fault
```

```
1457          calls_pwrSystem += 1
1458          result = pwrSystem_getChargerErrorCode()
1459          If result = NO_ANSWER Then
1460          ElseIf result = INCORRECT_RETURN_MESSAGE Then
1461            'DO SOMETHING
1462          ElseIf result = CHECKSUM_ERROR Then
1463            'DO SOMETHING
1464          Else
1465            pwrSystem_errorCode = result
1466          EndIf
1467        EndIf
1468
1469        next_getpwrSystemStatus = runtime + INTERVAL_pwrSYSTEM_STATUS
1470      EndIf
1471
1472      'UPDATE POWER SYSTEM DATA
1473      If runtime >= next_getpwrSystemData Then
1474
1475        calls_pwrSystem += 1
1476        result = pwrSystem_getLoadCurrent()
1477        If result = NO_ANSWER Then
1478          error_pwrSystem_noResponse += 1
1479
1480        ElseIf result = INCORRECT_RETURN_MESSAGE Then
1481          error_pwrSystem_incorrectMessage += 1
1482
1483        ElseIf result = CHECKSUM_ERROR Then
1484          error_pwrSystem_checksum += 1
1485
1486        Else
1487          pwrSystem_loadCurrent = result
1488          pwrSystem_loadpwr = pwrSystem_loadCurrent * systemVoltage
1489        EndIf
1490
1491        calls_pwrSystem += 1
1492        result = pwrSystem_getPanelpwr()
1493        If result = NO_ANSWER Then
1494          error_pwrSystem_noResponse += 1
1495          pwrSystem_panelpwr = 0
1496        ElseIf result = INCORRECT_RETURN_MESSAGE Then
1497          error_pwrSystem_incorrectMessage += 1
1498          pwrSystem_panelpwr = 0
1499        ElseIf result = CHECKSUM_ERROR Then
1500          error_pwrSystem_checksum += 1
1501          pwrSystem_panelpwr = 0
1502        Else
1503          pwrSystem_panelpwr = result
1504        EndIf
1505
1506        'update table 'FIX!
1507        CallTable pwrSystemTable
1508        next_getpwrSystemData = runtime + INTERVAL_pwrSYSTEM_DATA
1509      EndIf
1510
1511
1512
```

```
1513      '////////////////////////////////////////////////////////////////
1514      'LEAK DETECTION
1515      '////////////////////////////////////////////////////////////////
1516      'TODO: Rewrite for easy adjustment of measuring time with pumps ac
1517      'During testing it was found that 3 seconds could be insufficient
1518
1519      If runtime >= next_LeakCheck AND NOT(disable_pumps = True) Then
1520
1521        'Measure load current
1522        result = pwrSystem_getLoadCurrent()
1523        If (result = NO_ANSWER) OR (result = INCORRECT_RETURN_MESSAGE) O
1524          next_LeakCheck += 30 'Retry in 30 seconds
1525        Else
1526          If (runtime = next_LeakCheck + 0) Then 'FIX MOTSATT REKKEFLGE
1527            PortSet(PUMPS_RELAY_PIN, 0) 'Disable pumps
1528            leakDetected = FALSE 'Reset leak status
1529            currentMeasurement(1) = result
1530          ElseIf (runtime = next_LeakCheck + 1) Then
1531            currentMeasurement(2) = result
1532          ElseIf (runtime = next_LeakCheck + 2) Then
1533            currentMeasurement(3) = result
1534            'Enable pumps
1535            PortSet(PUMPS_RELAY_PIN, 1)
1536          ElseIf (runtime = next_LeakCheck + 3) Then
1537            currentMeasurement(4) = result
1538          ElseIf (runtime = next_LeakCheck + 4) Then
1539            currentMeasurement(5) = result
1540          ElseIf (runtime = next_LeakCheck + 5) Then
1541            currentMeasurement(6) = result
1542            'Disable pumps 'FIX
1543            PortSet(PUMPS_RELAY_PIN, 0)
1544            'Evaluate result
1545            Dim averageCurrent_inactive As Float
1546            Dim averageCurrent_active As Float
1547            Dim difference As Float
1548            averageCurrent_inactive = (currentMeasurement(1) + currentMe
1549            averageCurrent_active = (currentMeasurement(4) + currentMeas
1550            difference = averageCurrent_active - averageCurrent_inactive
1551
1552            If difference >= PUMP_CURRENT_THRESHOLD Then
1553              'Possibly water in bilge
1554              leakDetected = TRUE
1555              report_Leak(difference)
1556            Else
1557              leakDetected = FALSE
1558            EndIf
1559            next_LeakCheck += INTERVAL_LEAK_CHECK
1560          EndIf
1561        EndIf
1562      EndIf
1563
1564      '////////////////////////////////////////////////////////////////
1565      'IRIDIUM
1566      '////////////////////////////////////////////////////////////////
1567      'TODO: Only tested using the function iridium_CopyOutboxToInbox()
1568      'The subscription was not active in May, so the real functionality
```

```
1569
1570      'CHECK IRIDIUM INBOX
1571      If runtime >= next_iridiumInboxCheck Then
1572
1573        result = iridium_Receive()
1574        If result = NO_ANSWER Then
1575          error_iridiumRead_noResponse += 1
1576        Else If result = INCORRECT_RETURN_MESSAGE Then
1577          error_iridiumRead_incorrectMessage += 1
1578        Else
1579          'SUCCESS
1580
1581          If result = SUCCESS Then 'Message has been received from ESS.
1582
1583            'Read from iridium device buffer
1584            Dim inputMessage As String
1585            result = iridium_readFromInbox(@inputMessage)
1586            If result = NO_ANSWER Then
1587              error_iridiumRead_noResponse += 1
1588            Else If result = INCORRECT_RETURN_MESSAGE Then
1589              error_iridiumRead_incorrectMessage += 1
1590            Else
1591              'SUCCESS
1592              'FIX checksum
1593              setpwrSettings(Mid(inputMessage, 1, 5))
1594              next_iridiumInboxCheck = runtime + INTERVAL_IRIDIUM_RECEIV
1595            End If
1596
1597          Else If result = 0 Then
1598            'No messages waiting
1599            next_iridiumInboxCheck = runtime + INTERVAL_IRIDIUM_RECEIVE
1600          End If
1601        End If
1602      EndIf
1603
1604      'SEND IRIDIUM
1605      If runtime >= next_iridiumSendReport Then
1606        Dim outputMessage As String * 100
1607        outputMessage = report_generate()
1608        result = iridium_writeToOutbox(outputMessage)
1609        If result = INCORRECT_RETURN_MESSAGE Then
1610          error_iridiumRead_incorrectMessage += 1
1611        Else If result = SUCCESS Then
1612          result = iridium_send()
1613          If result = SUCCESS Then
1614            'Message sent. Clear the ouput buffer
1615            result = iridium_clearBuffer(IRIDIUM_MO)
1616            If result = SUCCESS Then
1617              'Procedure complete
1618              resetErrorCount()
1619              next_iridiumSendReport = runtime + INTERVAL_IRIDIUM_SEND
1620            End If
1621          EndIf
1622        EndIf
1623      EndIf
1624
```

```
1625    TriggerSequence(1,0) 'For testing (check execution time of loop)
1626
1627    NextScan
1628
1629    SlowSequence 'For testing (check execution time of loop)
1630      Do
1631        WaitTriggerSequence
1632        CallTable(exeTime)
1633      Loop
1634
1635  EndProg
1636
1637
1638
1639
```