

```

1 Option Explicit
2
3 Const ReturnNoSolution As Integer = -32767
4 Const ReturnNotMovableDelta As Integer = -1000
5
6 Dim RecurLevel As Integer
7 Dim ListRecurCall As String
8
9
10
11 '-----
12 AU_NPM_ManageMarginProtectedFlights
13 ' modify the FDATime
14 ' lprioModel must be assigned to GPrioModel_TimeMode_OnSchedule or
15 GPrioModel_TimeMode_OnMargin
16 ' in this case we use the schedule or the margin to make the calculation
17
18 ' lAllFl is list off flights to find Margin solution (All except Pflights and ExplicitB
19 flights)
20 ' when finding margin solution, if default priority value is baseline, prio is put to lowest
21 prio
22 ' If Time not after is greater then the Hotspot end, initialize to the end of the hotspot
23 ' BE CAREFULL a B priority on margins flights exclude flight from management
24 Function AU_NPM_ManageMarginPrioFlights_Main(lprioModel As Integer,
25     lAll_AUFlights As CL_AllFlights, lMyFlightsIx As CL_AUFlightsIx, _
26     ByRef lAllFl() As Integer, lAllFl_nb As Integer,
27     ByRef lAddFlight() As Integer, ByRef lMarginflight_Nb As Integer) As Integer
28
29     Dim li As Integer
30     Dim lj As Integer
31     Dim lName As String
32     Dim ltime As Date
33
34     'Dim lAllFl_prio() As Integer ' containt prio off All flight for this AU (Ix in
35 Allflight)
36
37     'Dim lAllFlSortedOnBaseline() As Integer ' containt All flight to manage the margin
38 sorted
39
40     Dim lMarginFl() As Integer ' containt All flight two have a margin
41     Dim lMarginSortedFl() As Integer ' containt All flight to manage the margin sorted
42     Dim lMarginFl_nb As Integer ' nb of flight to manage the margin
43
44     Dim lPrioOnlyFl() As Integer ' containt All flight two have a margin
45     Dim lPrioOnlySortedFl() As Integer ' containt All flight to manage the margin sorted
46     Dim lPrioOnlyFl_nb As Integer ' nb of flight to manage the margin
47
48     Call EX_Mess(EX_MESS_Start, "AU NPMRIO MARGINandPrio: " & lAll_AUFlights.AUName)
49     'ListRecurCall = ""
50
51     ' do nothing if only 1 flight
52     If lAllFl_nb < 2 Then
53         Exit Function
54     End If
55
56     Dim lMargeIx As Integer
57
58     ' slot management
59     Dim lMySlotsValue() As Date
60     Dim lMySlotsValueSorted() As Date
61     Dim lMySlotsUsed() As Integer
62
63     ' get the flight with margin to manage
64     ' get the list of flight with margins to manage
65
66     lMarginFl_nb = 0
67     lPrioOnlyFl_nb = 0
68
69     ' get margin flights and prio only flights
70     ' BE CAREFULL the management of explicit B on margin flight
71     ' explicit B normally not part of the flights

```

```

71 If lAllFl_nb > 0 Then
72     ReDim lMarginFl(lAllFl_nb)
73     ReDim lMySlotsValue(lAllFl_nb)
74     ReDim lPrioOnlyFl(lAllFl_nb)
75     ReDim lMySlotsUsed(lAllFl_nb)
76     For li = 0 To lAllFl_nb - 1
77         lMySlotsValue(li) = lAll_AUFlights.GetFDATime(lAllFl(li))
78         lMySlotsUsed(li) = -1
79         If (lAll_AUFlights.GetMarginNotAfterTimeIsInit(lAllFl(li)) = True) And _
80             (lAll_AUFlights.GetPrio(lAllFl(li)) <> GPrioSuspended) Then
81             lMarginFl(lMarginFl_nb) = lAllFl(li)
82             lMarginFl_nb = lMarginFl_nb + 1
83         Else
84             lPrioOnlyFl(lPrioOnlyFl_nb) = lAllFl(li)
85             lPrioOnlyFl_nb = lPrioOnlyFl_nb + 1
86         End If
87     Next li
88
89     ' get my slots and manage the list of margin or prio flights
90     ReDim lMySlotsValueSorted(lAllFl_nb)
91
92     ' Sort my Time slots (by FDATime)
93     Call AU_NPS_SortATimeTable(lMySlotsValue, lMySlotsValueSorted, lAllFl_nb)
94     Erase lMySlotsValue
95
96     If lMarginFl_nb < 1 Then
97         Erase lMarginFl
98     Else
99
100         ' Sort my Margin flights by prio and Margins and Schedule
101         ReDim lMarginSortedFl(lMarginFl_nb)
102         Call AU_NPS_SortByPrioAndMarginTimeNotAfterAndBaselineTime(lAll_AUFlights,
103 lMarginFl, lMarginSortedFl, lMarginFl_nb)
104         Erase lMarginFl
105
106         ' loop on earch Margin flights
107         ' put Margin flights on available slot
108
109         Call EX_Log_Init
110
111         For lMargeIx = 0 To lMarginFl_nb - 1
112
113             ' for test
114             Dim lCallsign As String
115             Dim lMarginTime As Date
116             Dim lMarginFlightIx As Integer
117
118             lMarginFlightIx = lMarginSortedFl(lMargeIx)
119             lCallsign = lAll_AUFlights.GetCallsignICA0(lMarginFlightIx)
120
121             Dim lSlotAssigned As Integer
122
123             If lMargeIx = 73 Then
124                 lMargeIx = lMargeIx
125             End If
126
127             lMarginTime = lAll_AUFlights.GetMarginNotAfterTime(lMarginFlightIx)
128
129             lSlotAssigned =
130 AU_NPM_ManageMarginPrioFlights_ManageTimeSolution(lAll_AUFlights, _
131 lMySlotsValueSorted(), lMySlotsUsed(), lAllFl_nb, _
132 lMarginTime, lMarginFlightIx)
133
134             ' Test if there is a slot
135             If lSlotAssigned < 0 Then
136                 ' no time solution
137                 Call AU_NPM_MsgboxStop(" ERROR TO ASSIGN Margin flight to a slot : " &
138 lCallsign, _
139 lAll_AUFlights, lMySlotsValueSorted(), lMySlotsUsed(), lAllFl_nb)
140
141             End If
142         Next lMargeIx
143     End If
144

```

```

145 ' manage the other type of flights
146 If lPrioOnlyFl_nb < 1 Then
147     'Erase lMySlotsValue
148     Erase lPrioOnlyFl
149     'Erase lMySlotsUsed
150 Else
151     ' AU slot has been assigned to Margin flights
152     ' now manage the other flights (prio only)
153     ReDim lPrioOnlySortedFl(lPrioOnlyFl_nb)
154     Call AU_NPM_SortByPrioAndSchedule(lAll_AUFlights, lPrioOnlyFl,
lPrioOnlySortedFl, lPrioOnlyFl_nb)
155     Erase lPrioOnlyFl
156
157     ' assign prio flights be carrefull baseline flights must be assign first then
flight with number
158     Call AU_NPM_ManageMarginPrioFlights.AssignOtherFlights(lAll_AUFlights, _
lMySlotsValueSorted(), lMySlotsUsed(), lAllFl_nb, _
lPrioOnlySortedFl(), lPrioOnlyFl_nb)
159
160
161     Erase lMarginSortedFl
162
163 End If
164
165 ' return the nb of margin flights
166 lMarginflight_Nb = lMarginFl_nb
167
168 End If
169
170 ' at the en Pack the flights on slots (use the available AU slot
171 ' needed if there is some Suspended flights
172 If lMarginFl_nb > 0 Or lPrioOnlyFl_nb > 0 Then
173     Call AU_NPM_ManageMarginPrioFlights.UseAvailableSlots(lAll_AUFlights, _
lMySlotsValueSorted(), lMySlotsUsed(), lAllFl_nb)
174
175 End If
176
177 ' assign the flights to the slots
178 '----- Update the FDA time -----
179 ' First assigne the FDA time on margins flights
180 For li = 0 To lAllFl_nb - 1
181     ltime = lMySlotsValueSorted(li)
182     ' if the slot is used by a margin flight assign it
183     If lMySlotsUsed(li) <> -1 Then
184         ' assign the Margin flights
185         lName = lAll_AUFlights.GetCallsignICA0(lMySlotsUsed(li))
186         Call lAll_AUFlights.SetFDATime(lMySlotsUsed(li), lMySlotsValueSorted(li))
187     End If
188 Next li
189
190 ' ----- Manage the Suspended flights
191 If lPrioOnlyFl_nb > 0 Then
192     ' Assign it the hotspot end time in fdaValue
193     Call AU_NPM_ManageMarginPrioFlights.AssignSuspendedFlights(lAll_AUFlights, _
lMySlotsValueSorted(), lMySlotsUsed(), lAllFl_nb, _
lPrioOnlySortedFl(), lPrioOnlyFl_nb)
194
195 End If
196
197 Erase lPrioOnlySortedFl
198 Erase lMySlotsValueSorted
199 Erase lMySlotsUsed
200
201 ' return nb of impacted flights to add on next list
202 AU_NPM_ManageMarginPrioFlights_Main = 0
203
204 Call EX_Mess(EX_MESS_End, "AU NPRI0 MARGINandPrio: " & lAll_AUFlights.AUName)
205
206
207 End Function
208
209 '----- AU_NPM_ManageMarginPrioFlights_ManageTimeSolution
210 ' try and manage the solution before the target time
211 ' At this stage the target slot is already used by another flight
212 ' first look if there is an available slot before the target one to shift earlier the others
213 ' if yes :
214 ' try to shift the flights to make a hole at the target slot

```

```

220 ' otherwise give a slot before for this flight
221 ' if no slot return -1
222 Function AU_NPM_ManageMarginPrioFlights_ManageTimeSolution( _
lAll_AUFlights As CL_AllFlights,
223 ByRef lMySlotsValueSorted() As Date, ByRef lMySlotsUsed() As Integer, lAllFl_nb
224 As Integer,
lTargetTime As Date,
225 lFlightIx As Integer) As Integer
226
227 'Dim lSlotEarlierPossible As Integer
228
229 Dim lAvailableSlot_Earlier As Integer
230 Dim lAvailableSlot_Later As Integer
231
232 Dim lEarliestTime As Date
233 Dim lTargetSlot As Integer
234
235 Dim lReturn As Integer
236
237 'lAvailableSlot_Earlier = -1
238
239 RecurLevel = 0
240 'Call EX_Log_Init
241
242 ' earliest time of the margin flight
243 lEarliestTime = lAll_AUFlights.GetRefBlockTime(lFlightIx)
244 - GHspt_FlightEarlyDeparture_forDate
245
246
247 lTargetSlot = AU_NPMF_GetTargetSlots(lMySlotsValueSorted, lAllFl_nb, lTargetTime,
lEarliestTime)
248
249 ' Test if there is a slot
250 If lTargetSlot < 0 Then
251     ' no time solution
252     Call AU_NPM_MsgboxStop(" ERROR TO DO SOMETHING Margin flight with no time solution :
" & lAll_AUFlights.GetCallsignICA0(lFlightIx),
253 lAll_AUFlights, lMySlotsValueSorted(), lMySlotsUsed(), lAllFl_nb)
254
255     Call EX_Log(RecurLevel, " --> No slot solution for " & lFlightIx & " on
targetTime " & lTargetTime)
256     lReturn = -1
257 Else
258     ' try manage solution by shifting flights earlier first
259     ' the slot could ber later then the one asked
260     lAvailableSlot_Earlier =
AU_NPM_ManageMarginPrioFlights_ManageSolutionEarlier(lAll_AUFlights, _
lMySlotsValueSorted(), lMySlotsUsed(), lAllFl_nb, _
lTargetSlot, lFlightIx)
261
262     If lAvailableSlot_Earlier > -1 Then
263         ' Found a place in the slot list earlierassign it
264         lMySlotsUsed(lAvailableSlot_Earlier) = lFlightIx
265         'ListRecurCall = ListRecurCall & " S= " & lAvailableSlot_Earlier & " |"
266         Call EX_Log(RecurLevel, "End FL: " & lFlightIx & " EARLIER Solution is slot: " &
lAvailableSlot_Earlier)
267
268         lReturn = lAvailableSlot_Earlier
269     Else
270         ' No possible slot earlier
271         ' test and get if there is available slot later
272         lAvailableSlot_Later = AU_NPMF_GetLaterAvailableSlots(lMySlotsValueSorted(),
lMySlotsUsed(), lAllFl_nb, _
lTargetSlot + 1, lEarliestTime)
273
274         If lAvailableSlot_Later > -1 Then
275             ' put it at this place
276             lMySlotsUsed(lAvailableSlot_Later) = lFlightIx
277             Call EX_Log(RecurLevel, "End FL: " & lFlightIx & " LATER Solution is slot: "
& lAvailableSlot_Later)
278             lReturn = lAvailableSlot_Later
279         Else
280             ' in this case no possibility to shift flight earlier
281             ' and no slot available after
282             ' otherwise there is other possibilities .....
283
284
285
286
287
288

```

```

289     lReturn = -1
290     Call EX_Log(RecurLevel, " --> ERR: No slot for " & lFlightIx)
291
292     Call AU_NPM_MsgboxStop("ManageTimeSolution PB of NB of available slot not OK
for : " & _
293         lAll_AUFlights.GetCallsignICAO(lFlightIx) & " id= " & lFlightIx, _
294         lAll_AUFlights, lMySlotsValueSorted, lMySlotsUsed, lAllFL_nb)
295
296     End If
297 End If
298
299     AU_NPM_ManageMarginPrioFlights_ManageTimeSolution = lReturn
300
301 End Function
302
303
304
305
306 '-----
307 AU_NPM_ManageMarginPrioFlights_ManageSolutionEarlier
308 ' Manage the solution at or before the target time or later if not found earlier
309 ' At this stage the target slot is already used by another flight
310 ' try to manage the solution by shifting the flight who occupy the slot earlier
311 ' because this flight has a higher priority because it's assigned before
312
313 'Input :
314 ' - the list of slot
315 ' - the slot used in this list by a previous assigned flight,
316 ' - the flight to be managed
317 ' - the started target slot
318
319 'Output:
320 ' - the available slot for this flight
321
322 ' first check if there is an available slot before the target one (needed to find a solution
earlier)
323 ' if no available slot before : stop this function and return -1
324 ' if there is an empty slot before (minimum constraint to have a earlier solution)
325 ' loop from the current needed slot to the latest slot of the list
326 ' (look also on later slot to return a slot later if no solution
before)
327 ' Call "MoveFlightEarlier" the recursive function to try to shift the flights on
earlier slot
328 ' to make a hole to assign the flight on a slot
329 ' if a slot is found: return the slot found (end of loop)
330 ' End of the loop (at this stage the slot found could be later)
331
332 ' if no possible slot return -1
333 ' otherwise return the slot found
334
335
336 Function AU_NPM_ManageMarginPrioFlights_ManageSolutionEarlier( _
337     lAll_AUFlights As CL_AllFlights, _
338     ByRef lMySlotsValueSorted() As Date, ByRef lMySlotsUsed() As Integer, lAllFL_nb
As Integer,
339     lTargetSlot As Integer, _
340     lMarginFlightIx As Integer) As Integer
341
342     Dim lSlotEarlierPossible As Integer
343     Dim lAvailableSlot_Earlier As Integer
344     Dim lTargetMove As Integer
345
346     lAvailableSlot_Earlier = -1
347
348 ' to test
349 If lAll_AUFlights.GetCallsignICAO(lMarginFlightIx) = "AFR165Z" Then
350     lMarginFlightIx = lMarginFlightIx
351 End If
352
353
354     lSlotEarlierPossible = AU_NPMF_GetIdxOfEarlierFlightCanMove(lAll_AUFlights, _
355     lMySlotsValueSorted(), lMySlotsUsed(), lAllFL_nb, lTargetSlot,
lMarginFlightIx)
356
357 ' some time say no but there is SO overwrite it
358 'lSlotEarlierPossible = 0

```

```

359
360 If lSlotEarlierPossible > -1 Then
361     ' move flight earlier is possible
362     'ListRecurCall = ListRecurCall & " | T:" & lMarginFlightIx & " " & lTargetSlot & " -> "
363
364     ' loop until solution found or end of slots
365     ' apply the recursive move flight earlier function from the target slot to the end
of the slots
366
367     ' until a solution is found
368     For lTargetMove = lTargetSlot To lAllFL_nb - 1
369         'ListRecurCall = "FL: " & lMarginFlightIx & " Slot: " & lTargetMove & " -> "
370         Call EX_Log(RecurLevel, "Start FL: " & lMarginFlightIx & " Slot: " & lTargetMove
& " -> ")
371
372         ' test and manage if there is a slot earlier
373         lAvailableSlot_Earlier =
AU_NPM_ManageMarginPrioFlights_MoveFlightEarlier(lAll_AUFlights, _
374             lMySlotsValueSorted(), lMySlotsUsed(), lAllFL_nb, _
375             lTargetMove, lMarginFlightIx)
376
377         'ListRecurCall = ListRecurCall & " Target= " & lTargetMove & " Out=" &
lAvailableSlot_Earlier")
378
379         If lAvailableSlot_Earlier > -1 Then
380             ' Found a place in the slot list earlier assign it
381             'lMySlotsUsed(lAvailableSlot_Earlier) = lMarginFlightIx
382             lTargetMove = lAllFL_nb
383             'ListRecurCall = ListRecurCall & " S= " & lAvailableSlot_Earlier & " |"
384             Call EX_Log(RecurLevel, "En FL: " & lMarginFlightIx & " Slot: " &
lAvailableSlot_Earlier)
385
386         Else
387             'ListRecurCall = ListRecurCall & " ..Next.."
388             Call EX_Log(RecurLevel, " FL: " & lMarginFlightIx & " Slot Not OK ..
Next")
389
390         End If
391
392     Next lTargetMove
393
394
395     If lAvailableSlot_Earlier < 0 Then
396
397         ' LG2018-07 No now it's OK because use of all earlier slots
398         ' If a flight can't be on the earlier slot it's mean that ther is a slot later
!!!
399         ' SO only return -1
400
401         Call EX_Log(RecurLevel, "En FL: " & lMarginFlightIx & " No Solution found
earlier even with hole")
402         GoTo lNext1:
403         ' a solution can exist
404         ' here it's because an affected flight lock the list because it's after another
one
405
406         If AU_NPM_ManageMarginPrioFlights_IsFlightScheduleCompatible(lAll_AUFlights, _
407             lMySlotsValueSorted(lSlotEarlierPossible), lMarginFlightIx) = True Then
408             ' this flight is compatible
409             lAvailableSlot_Earlier = lSlotEarlierPossible
410             Call EX_Log(RecurLevel, "En FL: " & lMarginFlightIx & " Direct Slot: " &
lAvailableSlot_Earlier)
411         Else
412             ' no solution because current and earlier slot not compatible with schedule
413             ' something wrong here
414             Call EX_Log(RecurLevel, "En FL: " & lMarginFlightIx & " ERROR No slot ")
415
416         Call AU_NPM_MsgboxStop("Margins PB possible slot before but not found
solution !!! : " & _
417             lAll_AUFlights.GetCallsignICAO(lMarginFlightIx) & " id= " &
lMarginFlightIx & vbCr & ListRecurCall,
418             lAll_AUFlights, lMySlotsValueSorted, lMySlotsUsed, lAllFL_nb)
419
420         End If
421
422     lNext1:
423
424 End If

```

```

424 Else
425     lAvailableSlot_Earlier = -1
426 End If
427
428
429 AU_NPM_ManageMarginPrioFlights_ManageSolutionEarlier = lAvailableSlot_Earlier
430
431 End Function
432
433
434
435
436
437
438 ' Recursive function to find and return a slot for a flight
439 ' If the slot is not free it make it available by shifting already assign flights earlier
440 ' (the already assign flights have higher priority)
441 ' this function make the shift of the flights only if all flights can be shifted (recursive
442 ' test before shifting)
443 'Input :
444 ' - the list of slot
445 ' - the slot used in this list by a previous assigned flight,
446 ' - the flight to be managed
447 ' - the started target slot
448
449 'Output:
450 ' - return >-1 : the available slot position set to free for this flight
451 ' - return = -1 blocking point, no possible shift earlier
452 ' - return = a negative value starting at -1000
453 ' + the negative value of the slot corresponding to a Unmovable
454 flight
455 ex: -1051 : the slot 51 is occupied by a Unmovable flight
456
457 Function AU_NPM_ManageMarginPrioFlights_MoveFlightEarlier(lAll_AUFlights As CL_AllFlights, _
458 ByRef lMySlotsValueSorted() As Date, ByRef lMySlotsUsed() As Integer, lSlots_nb _
459 As Integer, lTargetIx As Integer, lFlightIx_ToMove As Integer) As Integer
460
461 Dim li As Integer
462 Dim lReturn As Integer
463 Dim lTo As Integer
464 Dim lFrom As Integer
465 Dim lTryPreviousFlight As Boolean
466 Dim lEarliestTime As Date
467 Dim lPos As Integer
468 Dim lCallsign As String
469
470 ' to test
471 RecurLevel = RecurLevel + 1
472 lCallsign = lAll_AUFlights.GetCallsignICAO(lFlightIx_ToMove)
473
474 Call EX_Log(RecurLevel, "Mv FL: " & lFlightIx_ToMove & " to Slot: " & lTargetIx & "(used
475 by " & lMySlotsUsed(lTargetIx) & ")")
476
477 ' ca boucle ici !!!!!!!
478 If lFlightIx_ToMove = 284 And lTargetIx = 57 And lMySlotsUsed(lTargetIx) = 272 Then
479     lTargetIx = lTargetIx
480 End If
481
482 ' try to make the target slot available by shifting it earlier
483 lTryPreviousFlight = True
484 lFrom = lTargetIx
485 lTo = lTargetIx - 1
486
487 ' initial condition
488 FromSlot = Target slot
489 ToSlot = Target slot - 1 slot to be use for the first shift in case the target one
490 is used
491
492 ' loop on the flights until a solution is found or no possible solution
493 While lTryPreviousFlight = True
494     ' test if the flight to put on the target slot (FromSlot) is time compatible
495     ' (with its reference time - Airport early schedule duration)
496     If AU_NPM_ManageMarginPrioFlights_IsFlightScheduleCompatible(lAll_AUFlights, _
497         lMySlotsValueSorted(lFrom), lFlightIx_ToMove) = False Then

```

```

495 'if the flight is not time compatible with the FromSlot
496 'stop the loop and return the index of the tested slot as negative value
497 ' add -1000 to be sur the value 0 is well managed
498
499 lReturn = ReturnNotMovableDelta - lFrom
500 lTryPreviousFlight = False
501 Call EX_Log(RecurLevel, " FL: " & lFlightIx_ToMove & " Slot: " & lFrom & "
502 To Early .. Return : " & lReturn)
503
504 Else
505 'Else : the flight is compatible with the FromSlot
506 If lMySlotsUsed(lFrom) = -1 Then
507     'if the FromSlot is empty: no flight assigned on it
508     'return this slot and stop the loop
509     lReturn = lFrom
510     lTryPreviousFlight = False
511     Call EX_Log(RecurLevel, " FL: " & lFlightIx_ToMove & " Slot empty OK ..
512 Return : " & lReturn)
513
514 Else
515 'Else : the slot is used
516 'try to move the flight assigned in the used slot at an earlier position (to
517 the ToSlot)
518 'if the ToSlot is < 0 (it is not possible to move before because it was the
519 first slot)
520 ' in this case there is no solution: and end the loop
521 If lTo < 0 Then
522     ' we are at the beginning of the slot list without finding a solution
523     ' no possible solutions return -1
524     lReturn = ReturnNoSolution
525     lTryPreviousFlight = False
526     Call EX_Log(RecurLevel, " FL: " & lFlightIx_ToMove & " Slot -- No
527 SOLUTION -- At the end of Slots list Return : " & lReturn)
528
529 Else
530 'Else: the slot is used
531 'try to move the flight using the ToSlot to a earlier position by using
532 this same recursive function
533 'call lPos = MoveflightEarlier with in parameter: with the ToSlot
534 target and with the flight in the FromSlot position
535
536 lPos = AU_NPM_ManageMarginPrioFlights_MoveFlightEarlier(lAll_AUFlights,
537 _
538 lMySlotsValueSorted(), lMySlotsUsed(), lSlots_nb, _
539 lTo, lMySlotsUsed(lFrom))
540
541 'if the function return a positive value : (lPos > -1)
542
543 If lPos > -1 Then
544     'a compatible slot has been found
545     'make the use of this slot effective (assign it)
546     'put the flight use to call the recursive function on the Slot
547 position returned by it
548 'empty the slot used by it previously
549 'end the loop and return the empty slot
550 lMySlotsUsed(lPos) = lMySlotsUsed(lFrom)
551 lMySlotsUsed(lFrom) = -1
552
553 lReturn = lFrom 'return the slot put to free by the shift
554 lTryPreviousFlight = False
555
556 Call EX_Log(RecurLevel, " FL: " & lFlightIx_ToMove & " Recur OK
557 return " & lReturn & _
558 " Flight move from: " & lFrom & " To: " & lPos)
559
560 ElseIf lPos = ReturnNoSolution Then
561 'Else if the function return a -1 value (no possible solution) : lPos =
562 -1
563 ' return no possible solution and close the loop
564 ' no solution because flights impossible to move
565 ' cannot create a hole
566 lReturn = ReturnNoSolution
567 lTryPreviousFlight = False
568 Call EX_Log(RecurLevel, " FL: " & lFlightIx_ToMove & " Recur NO
569 SOLUTION ")
570
571 Else

```

3/8/2020

alg.vb

```

559 'Else : the function return a negative value < -1 a flight is blocked on
its position lPos = -lxxx
560 'continue to loop with an earlier position
561
562 'FromSlot = Slot position used to find solution on a flight blocked
:
563 ' Slot position -1 of the flight blocked
564 ' (be careful : corresponding to the ToSlot position when the
function is called)
565 'ToSlot = FromSlot - 1
566
567 ' test if not the earliest possible Slot to test
568 ' If FromSlot position is > -1 , continue the
loop
569 ' otherwise stop the loop and end by a -1
solution (no solution)
570 lFrom = -lPos + ReturnNotMovableDelta
571 lTo = lFrom - 1
572
573 If lFrom < 0 Then
574 ' From from next loop must be >= 0 otherwise no solution, stop
the loop
575 lTryPreviousFlight = False
576 lReturn = ReturnNoSolution
577
578 Call EX_Log(RecurLevel, " FL: " & lFlightIx_ToMove & " Recur NO
solution no more Slot to check " &
579 lFrom)
580 Else
581 lTryPreviousFlight = True
582
583 Call EX_Log(RecurLevel, " FL: " & lFlightIx_ToMove & " Recur
Next target Dde: " & _
584 lTargetIx & " Check: " & lFrom)
585
586 End If
587
588 End If
589 End If
590 End If
591 End If
592 Wend
593
594 AU_NPM_ManageMarginPrioFlights_MoveFlightEarlier = lReturn
595 RecurLevel = RecurLevel - 1
596 End Function
597
598
599
600
601
602
603
604
605
606 ' Recursive function to find and return a slot for a flight
607 ' If the slot is not free it make it available by shifting already assign flights earlier
(the already assign flights have higher priority)
608 ' this function make the shift of the flights only if all flights can be shifted (recursive
test before shifting)
609 'Input :
610 ' - the list of slot
611 ' - the slot used in this list by a previous assigned flight,
612 ' - the flight to be managed
613 ' - the started target slot
614
615 'Output:
616 ' - return >-1 : the available slot position set to free for this flight
617 ' - return = -1 blocking point, no possible shift earlier
618 ' - return = a negative value starting at -1000
619 ' + the negative value of the slot corresponding to a Unmovable
flight
620 ' ex: -1051 : the slot 51 is occupied by a Unmovable flight
621
622
623 Function AU_NPM_ManageMarginPrioFlights_MoveFlightEarlierOLD(lAll_AUFlights As
CL_AllFlights, _

```

localhost:4649/?mode=vb

9/15

3/8/2020

alg.vb

```

624 - ByRef lMySlotsValueSorted() As Date, ByRef lMySlotsUsed() As Integer, lSlots_nb
As Integer,
625 lTargetIx As Integer, lFlightIx_ToMove As Integer) As Integer
626
627 Dim li As Integer
628 Dim lReturn As Integer
629 Dim lTo As Integer
630 Dim lFrom As Integer
631 Dim lTryPreviousFlight As Boolean
632 Dim lEarliestTime As Date
633 Dim lPos As Integer
634 Dim lCallsign As String
635
636 ' to test
637 RecurLevel = RecurLevel + 1
638 lCallsign = lAll_AUFlights.GetCallsignICAO(lFlightIx_ToMove)
639
640 Call EX_Log(RecurLevel, "Rec Start Mv FL: " & lFlightIx_ToMove & " to Slot: " &
lTargetIx & "(used by " & lMySlotsUsed(lTargetIx) & ")")
641
642
643 ' make it available by shifting earlier previous flights
644 lTryPreviousFlight = True
645 lFrom = lTargetIx
646 lTo = lTargetIx - 1
647
648
649 ' loop on the flights because some of them could not be moved earlier due to schedule
650 While lTryPreviousFlight = True
651 ' test if the target slot is compatible (with reference time - Airport early
schedule duration)
652 If AU_NPM_ManageMarginPrioFlights_IsFlightScheduleCompatible(lAll_AUFlights, _
653 lMySlotsValueSorted(lFrom), lFlightIx_ToMove) = False Then
654 ' the flight is not compatible with the reference time of the target
655 ' stop the loop on this flight and return the last tested flight (as
negative value)
656 ' add -1000 to be sur the value 0 is managed
657 ' in recursive calling function, its the To target (in the current one it's
the from or an earlier one)
658 lReturn = ReturnNotMovableDelta - lFrom
659 lTryPreviousFlight = False
660 Call EX_Log(RecurLevel, "Rec - FL: " & lFlightIx_ToMove & " Slot: " & lFrom
& " To Early .. Return : " & lReturn)
661 Else
662 If lMySlotsUsed(lFrom) = -1 Then
663 'GOOD the slot is empty
664 lReturn = lFrom
665 lTryPreviousFlight = False
666 Call EX_Log(RecurLevel, "Rec - FL: " & lFlightIx_ToMove & " Slot empty OK ..
Return : " & lReturn)
667
668 Else
669 ' the slot is used : try to move the used slot at an earlier position
670 If lTo < 0 Then
671 ' no possible solutions
672 ' we are at the beginning of the slot list without finding a solution
673 lReturn = ReturnNoSolution
674 lTryPreviousFlight = False
675 Call EX_Log(RecurLevel, "Rec - FL: " & lFlightIx_ToMove & " Slot -- NO
SOLUTION -- At the end of Slots list Return : " & lReturn)
676
677 Else
678 ' slot used, try to move the flight using the slot to the lTo position
679 ' use recursive call to move the used slot earlier
680
681 lPos = AU_NPM_ManageMarginPrioFlights_MoveFlightEarlier(lAll_AUFlights,
682 lMySlotsValueSorted(), lMySlotsUsed(), lSlots_nb, _
683 lTo, lMySlotsUsed(lFrom))
684
685 If lPos > -1 Then
686 ' found a empty good slot
687 ' make the swap slot
688 lMySlotsUsed(lPos) = lMySlotsUsed(lFrom)
689 lMySlotsUsed(lFrom) = -1
690
691 lReturn = lFrom 'return the slot put to free by the shift

```

localhost:4649/?mode=vb

10/15

3/8/2020

alg.vb

```

692         lTryPreviousFlight = False
693
694         Call EX_Log(RecurLevel, "Rec - FL: " & lFlightIx_ToMove & " OK
        return " & lReturn & _
            " Flight move from: " & lFrom & " To: " & lPos)
695
696     ElseIf lPos = ReturnNoSolution Then
697         ' no solution because flights impossible to move
698         ' cannot create a hole
699         ' Normally a later solution is possible
700         lReturn = ReturnNoSolution
701         lTryPreviousFlight = False
702         Call EX_Log(RecurLevel, "Rec - FL: " & lFlightIx_ToMove & " NO
703     SOLUTION ")
704
705     Else
706         ' a negative value is returned if the position is not movable
707         ' continu to loop with an earlier position
708
709         'from is initiated and used for next test of ealier slot in this
        loop
710             lFrom = -lPos + ReturnNotMovableDelta
711             lTo = lFrom - 1
712
713             If lFrom < 0 Then
714                 ' From from next loop must be >= 0 therwhiqse no solution, stop
715
716                 lTryPreviousFlight = False
717                 lReturn = ReturnNoSolution
718
719                 Call EX_Log(RecurLevel, "Rec - FL: " & lFlightIx_ToMove & " NO
720                 solution no more Slot to check " & _
721                 lFrom)
722                 Else
723                     lTryPreviousFlight = True
724                     Call EX_Log(RecurLevel, "Rec - FL: " & lFlightIx_ToMove & "
725                     Initial: " & lTargetIx & _
726                     " reloop Next target: " & lFrom)
727                 End If
728             End If
729         End If
730     End If
731 End If
732 Wend
733
734 Call EX_Log(RecurLevel, "Rec EndLevel : " & lFlightIx_ToMove & " return : " & lReturn)
735
736 AU_NPM_ManageMarginPrioFlights_MoveFlightEarlier = lReturn
737
738 RecurLevel = RecurLevel - 1
739 End Function
740
741
742 '-----
743 AU_NPM_ManageMarginPrioFlights_AssignPrioOnlyFlight
744 ' Assign priority only flights in the remaining slot
745 ' list containt prio only + baseline + suspended
746 Sub AU_NPM_ManageMarginPrioFlights_AssignOtherFlights(lAll_AUFlights As CL_AllFlights, _
747     ByRef lSlotTime() As Date, ByRef lSlotList() As Integer, lSlot_nb As Integer, _
748     ByRef lPrioFlightSortedFl() As Integer, lPrioFlight_nb As Integer)
749
750     'don't manage the suspended flight here
751
752     Dim lFl As Integer
753     Dim lFlIx As Integer
754
755     Dim lBaselineFlights() As Integer
756     Dim lBaselineFlightsNb As Integer
757
758     Dim lPrioOnlyFlights() As Integer
759     Dim lPrioOnlyFlightsNb As Integer
760
761     'Dim lSuspendedFlights() As Integer

```

localhost:4649/?mode=vb

11/15

3/8/2020

alg.vb

```

762     'Dim lSuspendedFlightsNb As Integer
763
764
765     ReDim lBaselineFlights(lPrioFlight_nb)
766     lBaselineFlightsNb = 0
767
768     ReDim lPrioOnlyFlights(lPrioFlight_nb)
769     lPrioOnlyFlightsNb = 0
770
771     'ReDim lSuspendedFlights(lPrioFlight_nb)
772     'lSuspendedFlightsNb = 0
773
774     For lFl = 0 To lPrioFlight_nb - 1
775         lFlIx = lPrioFlightSortedFl(lFl)
776         If lAll_AUFlights.GetPrio(lFlIx) = GPrioBaseline Then
777             lBaselineFlights(lBaselineFlightsNb) = lFlIx
778             lBaselineFlightsNb = lBaselineFlightsNb + 1
779         ElseIf lAll_AUFlights.GetPrio(lFlIx) = GPrioSuspended Then
780             'lSuspendedFlights(lSuspendedFlightsNb) = lFlIx
781             'lSuspendedFlightsNb = lSuspendedFlightsNb + 1
782         Else
783             lPrioOnlyFlights(lPrioOnlyFlightsNb) = lFlIx
784             lPrioOnlyFlightsNb = lPrioOnlyFlightsNb + 1
785         End If
786     Next lFl
787
788     '----- manage the baseline flights on schedule
789     If lBaselineFlightsNb > 0 Then
790         Call AU_NPM_ManageMarginPrioFlights_AssignBaselineFlights(lAll_AUFlights, _
791             lSlotTime(), lSlotList(), lSlot_nb, _
792             lBaselineFlights(), lBaselineFlightsNb)
793     End If
794     Erase lBaselineFlights
795
796     '----- Manage the prio flights
797     If lPrioOnlyFlightsNb > 0 Then
798         Call AU_NPM_ManageMarginPrioFlights_AssignPrioFlights(lAll_AUFlights, _
799             lSlotTime(), lSlotList(), lSlot_nb, _
800             lPrioOnlyFlights(), lPrioOnlyFlightsNb)
801     End If
802     Erase lPrioOnlyFlights
803
804     '----- Manage the Suspended flights
805     'If lSuspendedFlightsNb > 0 Then
806         ' Call AU_NPM_ManageMarginPrioFlights_AssignSuspendedFlights(lAll_AUFlights, _
807             ' lSlotTime(), lSlotList(), lSlot_nb, _
808             ' lSuspendedFlights(), lSuspendedFlightsNb)
809     'End If
810     'Erase lSuspendedFlights
811
812
813
814
815 End Sub
816
817
818
819 '-----
820 AU_NPM_ManageMarginPrioFlights_AssignBaselineFlights
821 ' Assign default Baseline flights in the remaining slot
822 ' PB what we do if no possible slot for baseline ??????
823 Sub AU_NPM_ManageMarginPrioFlights_AssignBaselineFlights(lAll_AUFlights As CL_AllFlights, _
824     ByRef lSlotTime() As Date, ByRef lSlotList() As Integer, lSlot_nb As Integer, _
825     ByRef lFlightSorted() As Integer, lFlight_nb As Integer)
826
827     Dim lFl As Integer
828     Dim lSlotAssigned As Integer
829     Dim lEarliestTime As Date
830     Dim lBaselineTime As Date
831     Dim lFlAssigned As Integer
832     Dim lFlIx As Integer
833
834     '----- manage the baseline flights on schedule
835     lFlAssigned = 0
836     ' loop on baseline flights
837     For lFl = 0 To lFlight_nb - 1

```

localhost:4649/?mode=vb

12/15


```

838 lFlIx = lFlightSorted(lFl)
839 ' idem part then for Margins
840 ' find a slot corresponding to the Margin value to put the flight
841 lBaselineTime = lAll_AUFlights.GetBaselineTime(lFlIx)
842
843 lSlotAssigned = AU_NPM_ManageMarginPrioFlights_ManageTimeSolution(lAll_AUFlights, _
844     lSlotTime(), lSlotList(), lSlot_nb, _
845     lBaselineTime, lFlIx)
846
847 If lSlotAssigned < 0 Then
848     Call AU_NPM_MsgboxStop("Baseline flight PB of NB of available slot not OK for :
849 " & _
850     lAll_AUFlights.GetCallsignICA0(lFlIx) & " id= " & lFlIx, _
851     lAll_AUFlights, lSlotTime, lSlotList, lSlot_nb)
852 End If
853 Next lFl
854 End Sub
855
856 '-----
857 AU_NPM_ManageMarginPrioFlights_AssignPrioOnlyFlight
858 ' Assign priority only flights in the remaining slot
859
860 Sub AU_NPM_ManageMarginPrioFlights_AssignPrioFlights(lAll_AUFlights As CL_AllFlights, _
861     ByRef lSlotTime() As Date, ByRef lSlotList() As Integer, lSlot_nb As Integer, _
862     ByRef lFlightSorted() As Integer, lFlight_nb As Integer)
863
864 Dim lFl As Integer
865 Dim lFlIx As Integer
866 Dim lSlotIx As Integer
867
868 Dim lTargetTime As Date
869 Dim lSlotAssigned As Integer
870
871 Dim lFlightHaveSolution As Boolean
872
873 ' ----- Manage the prio flights
874
875 For lFl = 0 To lFlight_nb - 1
876     lFlIx = lFlightSorted(lFl)
877
878     ' Manage Prio Flights try to find a free slot compatible with the schedule
879     lFlightHaveSolution = False
880     For lSlotIx = 0 To lSlot_nb - 1
881         If lSlotList(lSlotIx) = -1 Then
882             If AU_NPM_ManageMarginPrioFlights_IsFlightScheduleCompatible(lAll_AUFlights,
883                 lSlotTime(lSlotIx), lFlIx) Then
884                 ' the schedule is compatible with the slot time
885                 ' lTime = lMySlotsValueSorted(lSlotIx)
886                 lSlotList(lSlotIx) = lFlIx
887                 'Call lAll_AUFlights.SetFDATime(lFlIx, lSlotTime(lSlotIx))
888                 lSlotIx = lSlot_nb ' stop the loop
889                 lFlightHaveSolution = True
890             End If
891         End If
892     Next lSlotIx
893
894 ' test if no solution found because slot too early
895 If lFlightHaveSolution = False Then
896     ' no slot available due to schedule time until the end of the slot list
897
898     lTargetTime = lAll_AUFlights.GetHotspotEndTime
899
900     lSlotAssigned =
901     AU_NPM_ManageMarginPrioFlights_ManageTimeSolution(lAll_AUFlights, _
902         lSlotTime(), lSlotList(), lSlot_nb, _
903         lTargetTime, lFlIx)
904
905     If lSlotAssigned < 0 Then

```

```

910 Call AU_NPM_MsgboxStop("Prio flights PB of NB of available slot not OK for :
911 " & _
912     lAll_AUFlights.GetCallsignICA0(lFlIx) & " id= " & lFlIx, _
913     lAll_AUFlights, lSlotTime, lSlotList, lSlot_nb)
914 End If
915 End If
916 Next lFl
917 End Sub
918
919 Sub AU_NPM_ManageMarginPrioFlights_AssignSuspendedFlights(lAll_AUFlights As CL_AllFlights, _
920     ByRef lSlotTime() As Date, ByRef lSlotList() As Integer, lSlot_nb As Integer, _
921     ByRef lPrioFlightSortedFl() As Integer, lPrioFlight_nb As Integer)
922
923 Dim lFl As Integer
924 Dim lFlIx As Integer
925 Dim ltime As Date
926
927 ' get the suspended flights
928 For lFl = 0 To lPrioFlight_nb - 1
929     lFlIx = lPrioFlightSortedFl(lFl)
930     If lFlIx > -1 Then ' normally never
931         If lAll_AUFlights.GetPrio(lFlIx) = GPrioSuspended Then
932             ltime = lAll_AUFlights.GetHotspotEndTime - G_OneSec_AsDate
933             Call lAll_AUFlights.SetFDATime(lFlIx, ltime)
934         End If
935     End If
936 Next lFl
937 End Sub
938
939 '-----
940 AU_NPM_ManageMarginPrioFlights_AssignPrioOnlyFlight
941 ' Assign priority only flights in the remaining slot
942
943 Sub AU_NPMOLD_ManageMarginPrioFlights_AssignSuspendedFlights(lAll_AUFlights As
944     CL_AllFlights, _
945     ByRef lSlotTime() As Date, ByRef lSlotList() As Integer, lSlot_nb As Integer, _
946     ByRef lFlightSorted() As Integer, lFlight_nb As Integer)
947
948 Dim lFl As Integer
949 Dim lFlIx As Integer
950 Dim ltime As Date
951
952 ' ----- manage the baseline flights on schedule
953 ' ----- manage the suspended flights
954 For lFl = 0 To lFlight_nb - 1
955     lFlIx = lFlightSorted(lFl)
956     If lFlIx < -1 Then
957         ' suspended flights at the end of the hotspot
958         ' Dont use a slot in the middle
959         ' the slot will be use when compacting at the end
960
961         ltime = lAll_AUFlights.GetHotspotEndTime - G_OneSec_AsDate
962
963         Call lAll_AUFlights.SetFDATime(lFlIx, ltime)
964     End If
965 Next lFl
966 End Sub
967
968 '-----
969 AU_NPM_ManageMarginPrioFlights_UseAvailableSlots
970 ' Assign priority only flights in the remaining slot
971
972 Sub AU_NPM_ManageMarginPrioFlights_UseAvailableSlots(lAll_AUFlights As CL_AllFlights, _
973     ByRef lSlotTime() As Date, ByRef lSlotList() As Integer, lSlot_nb As Integer)
974
975 Dim lFl As Integer
976 Dim lFlIx As Integer
977
978 Dim lFlChg As Integer
979 Dim lFlChgIx As Integer
980
981 Dim lEarliestTime As Date

```

```
983
984 ' compact the list
985 For lFl = 0 To lSlot_nb - 1
986     lFlIx = lSlotList(lFl)
987     If lFlIx = -1 Then
988         ' there is a hole
989         ' find a flight to put here
990         For lFlChg = lFl + 1 To lSlot_nb - 1
991             lFlChgIx = lSlotList(lFlChg)
992             If lFlChgIx > -1 Then
993                 ' there is a flight here
994                 If
AU_NPM_ManageMarginPrioFlights_IsFlightScheduleCompatible(lAll_AUFlights, _
                    lSlotTime(lFl), lFlChgIx) Then
995
996                     'lEarliestTime = lAll_AUFlights.GetBaselineTime(lFlChgIx)
997                     'If lSlotTime(lFl) >= lEarliestTime Then
998                         ' use this flight to fill the hole
999                         'Call lAll_AUFlights.SetFDATime(lFlChgIx, lSlotTime(lFl))
1000                         lSlotList(lFl) = lFlChgIx
1001                         lSlotList(lFlChg) = -1 ' free the slot
1002                         lFlChg = lSlot_nb ' stop the loop
1003                     End If
1004                 End If
1005             Next lFlChg
1006         End If
1007     Next lFl
1008 End Sub
1009
1010
1011
1012
```