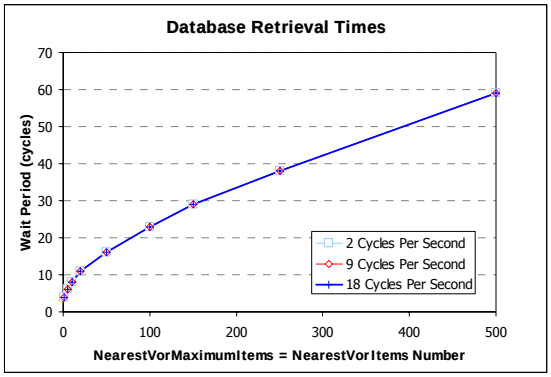
ASYNCHRONOUS OPERATION

Cycle skipping techniques

Information requested in an fs9gps database search is usually not available for use by the gauge, even for simple data display, in the same gauge update cycle that the search is initiated. Often, this causes no problems and it does not need to be addressed. There are some situations, however, where it *may or probably will* adversely affect subsequent code if that code requires the results of the database search before its execution begins. Because of this, it may be necessary to use cycle skipping techniques to delay execution of code that uses the requested data until they become available.

Recapping some discussion from the GPS Database Search section:

- ❑ even small searches can require several cycles for data retrieval
- ❑ waiting on gauge update cycles rather than absolute elapsed time for data retrieval is the key
- ❑ the bigger the search (i.e., the more items), the longer the wait



WHEN IS CYCLE SKIPPING NECESSARY?

I believe the question is not whether a particular fs9gps operation is multi-cycle, that is, whether it requires multiple gauge update cycles to complete. It is whether or not execution of *subsequent* code that uses search data must be carefully timed to <u>not</u> begin until search data are retrieved.

Consider a situation where one may want to find the nearest seaplane base. There are a couple of ways to approach this depending upon what you want to do afterward. For the purposes of this section, one way is to perform a NearestAirport search, then in <Update>, loop through the NearestAirport search results list incrementing the Index pointer (NearestAirportCurrentLine) by 1 each gauge update cycle, searching for an airport with NearestAirportCurrentAirportKind = 3, which is a seaplane base.

The following xml <Update> examples demonstrate what happens if the loop through the NearestAirport list begins before the NearestAirport search has returned data, and two cycle skipping techniques that can be used to "wait" for the Nearest search to conclude before starting the loop through the search result list.

Following that is a discussion of the cycle skipping requirements associated with ICAO Transfers.

EXAMPLE 1 – Nearest Searches

The following shows the aircraft position and the results of a NearestAirport search. Search parameters specify 100 maximum items and a 100 nmile maximum distance. The search is completed, returning the maximum list of 100 airports, on the 6th update cycle. The nearest Airport is Snohomish County (KPAE), while the nearest seaplane base is Kenmore Air Harbor (S60).


```
NEAREST AIRPORT SEARCH
  47.8335 :Current Lat  100 :Max Items 100 :Items Num
-122.3760 :Current Lon  100 :Max Distance

Current  ICAO     111 ---------------------- Current -------------------------
Line     123456789012 Ident Kind Rwy*  Dist  Brg Best Appr  Com     Freq Length
0        A      KPAE   KPAE    1  179   5.8   41   13   ILS  twr  120.20   9005
1        A      S13     S13    1  180   5.8   73    0              0.00   1680
2        A      S60     S60    3  180   6.7  135    0        CTF  122.70  10000
3        A      96WA   96WA    2  180   9.7   66    0              0.00   1100
4        A      W10     W10    1  180  11.4  347    0        CTF  122.90   2400
5        A      S43     S43    1  160  11.8   68    0        CTF  123.00   2660
6        A      W55     W55    3  180  12.4  173    0        CTF  122.90   5000
7        A      83Q     83Q    3  150  12.5  241    0        CTF  122.90  12000
8        A      0W0     0W0    3  201  12.5  172    0        CTF  122.90   9500
9        A      76WA   76WA    2   91  14.2   43    0              0.00   2500
```

(truncated after 10 airports to fit on this page)

What happens when no cycle skipping code is used?

The following code initiates the NearestAirport and begins inspecting the search results, looking for a seaplane base, in the same gauge update cycle that the Nearest search begins:

```
4       <Macro Name="c">@C</Macro>
5       <Macro Name="C">@C</Macro>
6
7   <Update Frequency="18" Hidden="No">
8   <!-- NearestAirport Search RESET -->
9       (L:Asynchronous Example 1 Gauge Reset, bool) 0 ==
10          if{
11              1 (>L:Asynchronous Example 1 Gauge Reset, bool)
12              -1 (>L:IndexPointer, enum)
13              1 (>L:LoopingThroughNearestAirportList, bool)
14
15
16          }
17
18  <!-- Set Search variables -->
19      100 (>@C:NearestAirportMaximumItems, enum)
20      100 (>@C:NearestAirportMaximumDistance, nmiles)
21
22  <!-- Set Aircraft position reference point -->
23      (A:PLANE LATITUDE, degrees) (>@C:NearestAirportCurrentLatitude, degrees)
24      (A:PLANE LONGITUDE, degrees) (>@C:NearestAirportCurrentLongitude, degrees)
25
26  <!-- Loop through the NearestAirport List searching for Seaplane Bases, AirportKind = 3 -->
27
28
29
30              (L:LoopingThroughNearestAirportList, bool) 1 ==
31                  if{
32                      (L:IndexPointer, enum) ++ (>L:IndexPointer, enum)
33                      (L:IndexPointer, enum) (>@C:NearestAirportCurrentLine)
34                      (@C:NearestAirportCurrentAirportKind) 3 ==
35                          if{
36                              (@C:NearestAirportCurrentDistance, nmiles) (>L:NrstArptCurDist, nmiles)
37                              (@C:NearestAirportCurrentTrueBearing, degrees) (>L:NrstArptCurTruBrg, degrees)
38                              0 (>L:LoopingThroughNearestAirportList, bool)
39                          }
40                  }
41
42  </Update>
```

- **Lines 9–16:** Search RESET. The gauge that this code snippet is from contains a Gauge Reset mouse click button. Clicking it performs the following:

  ```
  0 (>L:Asynchronous Example 1 Gauge Reset, bool)
  (>K:RELOAD_PANELS)
  ```

  Then, in the subsequent gauge update cycle, lines 11 through 15 are executed.

- **Lines 27–41:** Loop through the NearestAirport search results list looking for the first occurrence of NearestAirportCurrentAirportKind = 3. When found, LVars `L:NrstArptCurDist` and `L:NrstArptCurTruBrg` are written and the looping stops, `0 (>L:LoopingThroughNearestAirportList, bool)`.

In this example, looping through the NearestAirport list looking for a seaplane base begins in the same gauge update cycle as the start of the NearestAirport search itself. The problem is that 5 update cycles are consumed before the NearestAirport list is returned and by that time, the loop that checks the search results is already at L:IndexPointer = NearestAirportCurrentLine = 4. Starting at this point in the search results list, the first seaplane base (CurrentAirportKind = 3) found will be in CurrentLine 6, Airport Ident W55, which is not the nearest seaplane base to the aircraft.

TECHNIQUE 1 - Cycle Counting

The next code employs a cycle counting routine that delays execution of the loop that checks for the first seaplane base until a prescribed number of gauge update cycles have passed since the NearestAirport search was initiated. The goal is to give the NearestAirport search time to return results before looking through them for a seaplane base.

```
    <Macro Name="c">@C:</Macro>
    <Macro Name="C">@C:</Macro>

<Update Frequency="18" Hidden="No">
<!-- NearestAirport Search RESET -->
    (L:Asynchronous Example 1 Gauge Reset, bool) 0 ==
        if{
            1 (>L:Asynchronous Example 1 Gauge Reset, bool)
            -1 (>L:IndexPointer, enum)
            1 (>L:LoopingThroughNearestAirportList, bool)
            4 (>L:NrstArptSearchCyclesToSkip, enum)
            0 (>L:NrstArptSearchCycleSkipCounter, enum)
        }

<!-- Set Search variables -->
    100 (>@C:NearestAirportMaximumItems, enum)
    100 (>@C:NearestAirportMaximumDistance, nmiles)

<!-- Set Aircraft position reference point -->
    (A:PLANE LATITUDE, degrees) (>@C:NearestAirportCurrentLatitude, degrees)
    (A:PLANE LONGITUDE, degrees) (>@C:NearestAirportCurrentLongitude, degrees)

<!-- Loop through the NearestAirport List searching for Seaplane Bases, AirportKind = 3 -->
    (L:NrstArptSearchCycleSkipCounter, enum) 1 + (>L:NrstArptSearchCycleSkipCounter, enum)
    (L:NrstArptSearchCycleSkipCounter, enum) (L:NrstArptSearchCyclesToSkip, enum) &gt;
        if{
            (L:LoopingThroughNearestAirportList, bool) 1 ==
                if{
                    (L:IndexPointer, enum) ++ (>L:IndexPointer, enum)
                    (L:IndexPointer, enum) (>@C:NearestAirportCurrentLine)
                    (@C:NearestAirportCurrentAirportKind) 3 ==
                        if{
                            (@C:NearestAirportCurrentDistance, nmiles) (>L:NrstArptCurDist, nmiles)
                            (@C:NearestAirportCurrentTrueBearing, degrees) (>L:NrstArptCurTruBrg, degrees)
                            0 (>L:LoopingThroughNearestAirportList, bool)
                        }
                }
        }
</Update>
```

- **Lines 9–16:** Same comments apply, as above.
- **Lines 27–41:** Same comments apply, as above.
- **Line 14:** The pre-selected number of update cycles to skip.
- **Line 15:** The cycle skip counter. It is reset to 0 when the Search RESET mouse button is clicked.
- **Line 27:** The cycle skip counter is incremented by 1 each update cycle.
- **Lines 28-29:** When the cycle skip counter is greater than the number of cycles to skip, looping through the NearestAirport search list is allowed to begin (lines 30–40).

When `L:NrstArptSearchCyclesToSkip` is set too low, lines 30 - 40 begin execution before the NearestAirport search has returned data, and the same thing happens as in the first example, and the nearest seaplane base may be missed because the L:IndexPointer = NearestAirportCurrentLine may be already past it before NearsetAirport results are available.

In this example, when `L:NrstArptSearchCyclesToSkip` = 4 or less, the first seaplane base found is ident W55, which is incorrect. When `CyclesToSkip` is set to 5 or more, the first seaplane base found is ident S60, which is correct.

TECHNIQUE 2 - Let fs9gps tell you when it's ready

In this example, progress of the NearestAirport search is checked each gauge update cycle. Line 28 checks if NearestAirportItemsNumber, which is the total number of airports found in the search, is greater than zero. When starting the database search, this value is zero, but when the NearestAirport search returns data, ItemsNumber reflects the number of airports found. Assuming the NearestAirport search found at least one airport within the specified search distance, ItemsNumber becomes greater than zero, and only then are lines 30 – 40 executed. This time, the NearestAirport list is available before the loop begins and the nearest seaplane base, S60, is not bypassed.

```
4       <Macro Name="c">@C</Macro>
5       <Macro Name="C">@C</Macro>
6
7   <Update Frequency="18" Hidden="No">
8   <!-- NearestAirport Search RESET -->
9       (L:Asynchronous Example 1 Gauge Reset, bool) 0 ==
10          if{
11              1 (>L:Asynchronous Example 1 Gauge Reset, bool)
12              -1 (>L:IndexPointer, enum)
13              1 (>L:LoopingThroughNearestAirportList, bool)
14
15
16          }
17
18  <!-- Set Search variables -->
19      100 (>@C:NearestAirportMaximumItems, enum)
20      100 (>@C:NearestAirportMaximumDistance, nmiles)
21
22  <!-- Set Aircraft position reference point -->
23      (A:PLANE LATITUDE, degrees) (>@C:NearestAirportCurrentLatitude, degrees)
24      (A:PLANE LONGITUDE, degrees) (>@C:NearestAirportCurrentLongitude, degrees)
25
26  <!-- Loop through the NearestAirport List searching for Seaplane Bases, AirportKind = 3 -->
27
28      (@C:NearestAirportItemsNumber) 0 &gt;
29          if{
30              (L:LoopingThroughNearestAirportList, bool) 1 ==
31                  if{
32                      (L:IndexPointer, enum) ++ (>L:IndexPointer, enum)
33                      (L:IndexPointer, enum) (>@C:NearestAirportCurrentLine)
34                      (@C:NearestAirportCurrentAirportKind) 3 ==
35                          if{
36                              (@C:NearestAirportCurrentDistance, nmiles) (>L:NrstArptCurDist, nmiles)
37                              (@C:NearestAirportCurrentTrueBearing, degrees) (>L:NrstArptCurTruBrg, degrees)
38                              0 (>L:LoopingThroughNearestAirportList, bool)
39                          }
40                  }
41          }
42  </Update>
```

- **Lines 9–16:** Same comments apply, as above.
- **Lines 27–41:** Same comments apply, as above.

This technique is employed in the FS9 gps_500 gauge where screen display loop of Nearest search results does not begin until ItemsNumber value is non-zero (for example, see line 2016). As a side comment, it isn't actually necessary that the gps_500 does that in this particular situation – to delay a *display* loop until the search concludes. If the user is only *displaying* the Nearest list, the display script will simply display blank values until the search concludes, at which point, the full list will be displayed as normal.

Edit line 2016 to read:

```
%((@c:NearestAirportItemsNumber) s2 )
```

and delete lines 2017 and 2034 to check this.

EXAMPLE 2 – Cycle Counting Technique for ICAO Transfers

ICAO Transfers are another multi-cycle operation. Data from the Waypoint Group are not available during the same update cycle that the ICAO Transfer is executed. In some circumstances, data may not be accessible even in the subsequent cycle and a cycle counting technique to skip more than one cycle is needed.

Building on the NearestAirport-seaplane base search example, the following statements have been added to the example code: a NearestAirport to WaypointAirport ICAO Transfer, cycle counting code following the ICAO Transfer, and a WaypointAirport Group write-to-LVar statement.

- **Line 41:** The ICAO Transfer. NearestAirport to WaypointAirport Group.
- **Line 42:** Write WaypointAirportLatitude to an LVar, same cycle as ICAO Xfer.
- **Line 43:** Write WaypointAirportLongitude to an LVar, same cycle as ICAO Xfer.
- **Lines 50 –59:**
  - If Line 44 `LoopingThroughNearestAirportList` is set to 2, then Lines 52 – 57 will be executed, otherwise they won't.
  - Starting in Line 52, the `ICAOXferCycleSkipCounter` is incremented by one each update cycle.
  - Only when it is equal to or greater than the prescribed `ICAOTransferCyclesToSkip` (Line 53, 54),
  - are the WaypointAirport Group variables Latitude and Longitude written to LVars (Lines 55, 56).
  - Finally, Line 57 sets `LoopingThroughNearestAirportList` to zero, terminating the entire Nearest Search and ICAO Transfer process.
- **Lines 14 –19:** Resets cycle counters and LVar values.

Note that in the following <Update> section, LVAR write statements (Lines 42, 43) are included immediately following the ICAO Transfer (Line 41) to illustrate that the WaypointAirport Group variables including Latitude and Longitude are not accessible in the same cycle as the ICAO Transfer.

```
<Update Frequency="2" Hidden="No">
<!-- ICAOSearch RESET -->
    (L:Asynch Example 3 Gauge Reset, enum) 1 ==
        if{
            0 (>L:Asynch Example 3 Gauge Reset, enum)
            -1 (>L:IndexPointer, enum)
            1 (>L:LoopingThroughNearestAirportList, enum)
            -1 (>L:ICAOXferCycleSkipCounter, enum)
            0 (>L:ICAOXferCyclesToSkip, enum)
            0 (>L:NrstArptCurDist, nmiles)
            0 (>L:NrstArptCurTruBrg, degrees)
            0 (>L:WptArptLat, degrees)
            0 (>L:WptArptLon, degrees)
        }

<!-- Set Search variables -->
    10 (>@C:NearestAirportMaximumItems, enum)
    100 (>@C:NearestAirportMaximumDistance, nmiles)

<!-- Set Aircraft position reference point -->
    (A:PLANE LATITUDE, degrees) (>@C:NearestAirportCurrentLatitude, degrees)
    (A:PLANE LONGITUDE, degrees) (>@C:NearestAirportCurrentLongitude, degrees)

<!-- Loop through the NearestAirport list searching for Seaplane Base, AirportKind = 3 -->
    (L:LoopingThroughNearestAirportList, enum) 1 ==
        if{
            (@C:NearestAirportItemsNumber) 0 !=
                if{
                    (L:IndexPointer, enum) ++ (>L:IndexPointer, enum)
                    (L:IndexPointer, enum) (>@C:NearestAirportCurrentLine)
                    (@C:NearestAirportCurrentAirportKind) 3 ==
                        if{
                            (@C:NearestAirportCurrentDistance, nmiles) (>L:NrstArptCurDist, nmiles)
                            (@C:NearestAirportCurrentTrueBearing, degrees) (>L:NrstArptCurTruBrg, degrees)
                            (@C:NearestAirportCurrentICAO) (>@C:WaypointAirportICAO)
                            (@C:WaypointAirportLatitude, degrees) (>L:WptArptLat, degrees)
                            (@C:WaypointAirportLongitude, degrees) (>L:WptArptLon, degrees)
                            0 (>L:LoopingThroughNearestAirportList, enum)
                        }
                }
        }

<!-- After ICAO Transfer, delay accessing Waypoint Group variables using a Cycle Counting Technique -->
    (L:LoopingThroughNearestAirportList, enum) 2 ==
        if{
            (L:ICAOXferCycleSkipCounter, enum) ++ (>L:ICAOXferCycleSkipCounter, enum)
            (L:ICAOXferCycleSkipCounter, enum) (L:ICAOXferCyclesToSkip, enum) >=
                if{
                    (@C:WaypointAirportLatitude, degrees) (>L:WptArptLat, degrees)
                    (@C:WaypointAirportLongitude, degrees) (>L:WptArptLon, degrees)
                    0 (>L:LoopingThroughNearestAirportList, enum)
                }
        }
</Update>
```

When Line 44 is set to zero, the loop terminates after the LVar write statements (Lines 42, 43) that follow the ICAO Transfer (Line 41). The WaypointAirport variables Latitude and Longitude are not yet accessible, so the LVars remain 0.0000, as shown below.

Similarly, when Line 44 is set to 2, but with `ICAOXferCyclesToSkip` = 0 (Line 15), then the LVar write statements again occur in the *same* update cycle as the ICAO Transfer, the WaypointAirport variables Latitude and Longitude are not yet accessible, and the LVars remain 0.0000, as shown below (see lower right corner of the figure).

```
NEAREST AIRPORT SEARCH
  47.8335 :Current Lat   10 :Max Items  10 :Items Num
-122.3760 :Current Lon  100 :Max Distance

Current  ICAO     111 ---------------------- Current -------------------------
Line     123456789012 Ident Kind Rwy*  Dist  Brg Best Appr  Com     Freq Length
0        A      KPAE   KPAE    1  179   5.8   41   13   ILS  twr  120.20   9005
1        A      S13     S13    1  180   5.8   73    0              0.00   1680
2        A      S60     S60    3  180   6.7  135    0        CTF  122.70  10000
3        A      96WA   96WA    2  180   9.7   66    0              0.00   1100
4        A      W10     W10    1  180  11.4  347    0        CTF  122.90   2400
5        A      S43     S43    1  160  11.8   68    0        CTF  123.00   2660
6        A      W55     W55    3  180  12.4  173    0        CTF  122.90   5000
7        A      83Q     83Q    3  150  12.5  241    0        CTF  122.90  12000
8        A      OW0     OW0    3  201  12.5  172    0        CTF  122.90   9500
9        A      76WA   76WA    2   91  14.2   43    0              0.00   2500

NEAREST AIRPORT > LVars                  WAYPOINT AIRPORT > LVars

(L:NrstArptCurDist, nmiles):     6.7     (L:WptArptLat, degrees):    0.0000
(L:NrstArptCurTruBrg, degrees):  135     (L:WptArptLon, degrees):    0.0000
```

However, if Line 44 is set to 2 and Line 15 is edited to read

```
1 (>L:ICAOXferCyclesToSkip, enum)
```

then the write statements in Lines 55 & 56 are delayed one update cycle from the ICAO Transfer and the WaypointAirport Lat & Lon for seaplane base S60 are now accessed:

```
NEAREST AIRPORT SEARCH
  47.8335 :Current Lat   10 :Max Items  10 :Items Num
-122.3760 :Current Lon  100 :Max Distance

Current  ICAO     111 ---------------------- Current -------------------------
Line     123456789012 Ident Kind Rwy*  Dist  Brg Best Appr  Com     Freq Length
0        A      KPAE   KPAE    1  179   5.8   41   13   ILS  twr  120.20   9005
1        A      S13     S13    1  180   5.8   73    0              0.00   1680
2        A      S60     S60    3  180   6.7  135    0        CTF  122.70  10000
3        A      96WA   96WA    2  180   9.7   66    0              0.00   1100
4        A      W10     W10    1  180  11.4  347    0        CTF  122.90   2400
5        A      S43     S43    1  160  11.8   68    0        CTF  123.00   2660
6        A      W55     W55    3  180  12.4  173    0        CTF  122.90   5000
7        A      83Q     83Q    3  150  12.5  241    0        CTF  122.90  12000
8        A      OW0     OW0    3  201  12.5  172    0        CTF  122.90   9500
9        A      76WA   76WA    2   91  14.2   43    0              0.00   2500

NEAREST AIRPORT > LVars                  WAYPOINT AIRPORT > LVars

(L:NrstArptCurDist, nmiles):     6.7     (L:WptArptLat, degrees):   47.7548
(L:NrstArptCurTruBrg, degrees):  135     (L:WptArptLon, degrees): -122.2593
```

An important note is that in some circumstances, **skipping more than one gauge update cycle following an ICAO Transfer may be required** before the Waypoint Group variables are accessible. With ICAO Transfers, unfortunately there is no gps variable that can be used with the "let fs9gps tell you when it is ready" technique, so cycle counting is necessary. Reading through the forums, you may run across examples advocating the use of cycle skipping toggles – very simple code that results in a one cycle skip. But if you unknowingly run into the situation where you need to skip more than one cycle following an ICAO Transfer but have used a single-cycle toggle, then

figuring out what is wrong with your code can be pretty frustrating. Been there, done that. Experiment to determine what you need.

To get a better sense of the sequence of events, slow the update frequency rate (Line 7) down to 2, and watch the display after Gauge Reset (lines 11 – 19). You should be able to discern the slight hesitation during the Nearest search, before the NearestAirport list is displayed. Next, the NearestAirport LVar values appear, followed a moment later by the WaypointAirport LVar values.

Finally, just to demonstrate a point, the LVar assignments could have been pulled out of the loop and placed by themselves elsewhere in the Update section. This results in continuous* writing of WaypointAirportLatitude and Longitude to LVars. That would eliminate the need for specific cycle skipping code because, eventually, the ICAO Transfer is completed, the Waypoint Group variables are finally accessible, and the WaypointAirport Latitude and Longitude will be written to the LVars.

```
7  <Update Frequency="18" Hidden="No">
8  <!-- ICAOSearch RESET -->
9      (L:Asynch Example 3 Gauge Reset, enum) 1 ==
10         if{
11             0 (>L:Asynch Example 3 Gauge Reset, enum)
12             -1 (>L:IndexPointer, enum)
13             1 (>L:LoopingThroughNearestAirportList, enum)
14             0 (>L:NrstArptCurDist, nmiles)
15             0 (>L:NrstArptCurTruBrg, degrees)
16             0 (>L:WptArptLat, degrees)
17             0 (>L:WptArptLon, degrees)
18         }
19
20 <!-- Set Search variables -->
21     10 (>@C:NearestAirportMaximumItems, enum)
22     100 (>@C:NearestAirportMaximumDistance, nmiles)
23
24 <!-- Set Aircraft position reference point -->
25     (A:PLANE LATITUDE, degrees) (>@C:NearestAirportCurrentLatitude, degrees)
26     (A:PLANE LONGITUDE, degrees) (>@C:NearestAirportCurrentLongitude, degrees)
27
28 <!-- Loop through the NearestAirport list searching for Seaplane Base, AirportKind = 3 -->
29     (L:LoopingThroughNearestAirportList, enum) 1 ==
30         if{
31             (@C:NearestAirportItemsNumber) 0 !=
32                 if{
33                     (L:IndexPointer, enum) ++ (>L:IndexPointer, enum)
34                     (L:IndexPointer, enum) (>@C:NearestAirportCurrentLine)
35                     (@C:NearestAirportCurrentAirportKind) 3 ==
36                         if{
37                             (@C:NearestAirportCurrentDistance, nmiles) (>L:NrstArptCurDist, nmiles)
38                             (@C:NearestAirportCurrentTrueBearing, degrees) (>L:NrstArptCurTruBrg, degrees)
39                             (@C:NearestAirportCurrentICAO) (>@C:WaypointAirportICAO)
40                             0 (>L:LoopingThroughNearestAirportList, enum)
41                         }
42                 }
43         }
44
45 <!-- Write WaypointAirport Latitude and Longitude to LVars each update cycle -->
46     (@C:WaypointAirportLatitude, degrees) (>L:WptArptLat, degrees)
47     (@C:WaypointAirportLongitude, degrees) (>L:WptArptLon, degrees)
48
49  </Update>
```

* It is not a best practice to continuously execute code in the Update section when it is not necessary, so this code snippet is just for the purpose of illustrating a point.

ICAO SEARCH – No Cycle-Skipping Required

ICAOSearch is a single cycle operation, therefore, no cycle skipping code is required after ICAOSearch and before an ICAO Transfer.

CONDITIONAL TEXT DISPLAY

The section, "When Is Cycle Skipping Necessary?", began by saying that depending upon what the user wants to do next, there are alternative approaches to finding the nearest seaplane base. If simple display of the list of nearest seaplane bases is all that is required and no ICAO Transfer is needed, then conditional text display statements can be used to display only NearestAirportCurrentAirportKind = 3 airports, thus eliminating the need for the loop through the NearestAirport list in the <Update> section.
The display <Element> I use to display all of the nearest airports is:

```
62  <Element Name="NEAREST AIRPORT LOOP DISPLAY">
63      <Position X="10" Y="5"/>
64      <FormattedText X="800" Y="700" Font="Courier New" FontSize="12"
65      LineSpacing="12" Color="#100000" Bright="Yes">
66          <Color Value="blue"/>
67          <Color Value="darkgreen"/>
68          <String>
69          \{clr2}NEAREST AIRPORT SEARCH\n
70          \{clr3}%((C:fs9gps:NearestAirportCurrentLatitude, degrees))%!9.4f! :Current Lat
71          %((C:fs9gps:NearestAirportMaximumItems, enum))%!5d! :Max Items
72          %((@c:NearestAirportItemsNumber))%!4d! :Items Num\n
73          %((C:fs9gps:NearestAirportCurrentLongitude, degrees))%!9.4f! :Current Lon
74          %((C:fs9gps:NearestAirportMaximumDistance, nmiles))%!5d! :Max Distance\n
75          \n
76          \{clr2}Current  ICAO     111 ---------------------- Current -------------------------\n
77          \{clr2}Line     123456789012 Ident Kind Rwy*  Dist  Brg Best Appr  Com    Freq Length\n\{clr)
78              %((@c:NearestAirportItemsNumber) s2 0 !=)
79                  %{if}
80                      %(0 sp1)
81                      %{loop}
82                          %(l1 (>@c:NearestAirportCurrentLine))
83
84
85                                      %((@c:NearestAirportCurrentLine))%!-5d!
86                                      %((@c:NearestAirportCurrentICAO))%!16s!
87                                      %((@c:NearestAirportCurrentIdent))%!6s!
88                                      %((@c:NearestAirportCurrentAirportKind))%!5d!
89                                      %((@c:NearestAirportCurrentLongestAirportDirection, degrees))%!5d!
90                                      %((@c:NearestAirportCurrentDistance, nmiles))%!6.1f!
91                                      %((@c:NearestAirportCurrentTrueBearing, degrees))%!5d!
92                                      %((@c:NearestAirportCurrentBestApproachEnum))%!4d!
93                                      %((@c:NearestAirportCurrentBestApproach))%!6s!
94                                      %((@c:NearestAirportCurrentComFrequencyName))%!5s!
95                                      %((@c:NearestAirportCurrentComFrequencyValue, mhz))%!8.2f!
96                                      %((@c:NearestAirportCurrentLongestRunwayLength, feet))%!7d!\n
97
98                                  %(l1 ++ s1 l2 &lt;)
99                      %{next}
100                 %{end}
101         </String>
102     </FormattedText>
103 </Element>
```

To display only AirportKind = 3, Lines 83, 84, and 97 are added as follows:

```
<Element Name="NEAREST AIRPORT LOOP DISPLAY">
    <Position X="10" Y="5"/>
    <FormattedText X="800" Y="700" Font="Courier New" FontSize="12"
    LineSpacing="12" Color="#100000" Bright="Yes">
        <Color Value="blue"/>
        <Color Value="darkgreen"/>
        <String>
        \{clr2}NEAREST AIRPORT SEARCH\n
        \{clr3}%((C:fs9gps:NearestAirportCurrentLatitude, degrees))%!9.4f! :Current Lat
        %((C:fs9gps:NearestAirportMaximumItems, enum))%!5d! :Max Items
        %((@c:NearestAirportItemsNumber))%!4d! :Items Num\n
        %((C:fs9gps:NearestAirportCurrentLongitude, degrees))%!9.4f! :Current Lon
        %((C:fs9gps:NearestAirportMaximumDistance, nmiles))%!5d! :Max Distance\n
        \n
        \{clr2}Current  ICAO     111 ---------------------- Current -------------------------\n
        \{clr2}Line     123456789012 Ident Kind Rwy*  Dist  Brg Best Appr  Com    Freq Length\n\{clr}
            %((@c:NearestAirportItemsNumber) s2 0 !=)
                %{if}
                %(0 sp1)
                    %{loop}
                        %(l1 (>@c:NearestAirportCurrentLine))
                        %((@c:NearestAirportCurrentAirportKind) 3 ==)
                            %{if}\{nr}
                                %((@c:NearestAirportCurrentLine))%!-5d!
                                %((@c:NearestAirportCurrentICAO))%!16s!
                                %((@c:NearestAirportCurrentIdent))%!6s!
                                %((@c:NearestAirportCurrentAirportKind))%!5d!
                                %((@c:NearestAirportCurrentLongestAirportDirection, degrees))%!5d!
                                %((@c:NearestAirportCurrentDistance, nmiles))%!6.1f!
                                %((@c:NearestAirportCurrentTrueBearing, degrees))%!5d!
                                %((@c:NearestAirportCurrentBestApproachEnum))%!4d!
                                %((@c:NearestAirportCurrentBestApproach))%!6s!
                                %((@c:NearestAirportCurrentComFrequencyName))%!5s!
                                %((@c:NearestAirportCurrentComFrequencyValue, mhz))%!8.2f!
                                %((@c:NearestAirportCurrentLongestRunwayLength, feet))%!7d!\n
                            %{end}
                        %(l1 ++ s1 l2 &lt;)
                    %{next}
                %{end}
        </String>
    </FormattedText>
</Element>
```

which results in the following screen display:

```
NEAREST AIRPORT SEARCH
  47.8335 :Current Lat  100 :Max Items 100 :Items Num
-122.3760 :Current Lon  100 :Max Distance

Current  ICAO     111 ---------------------- Current -------------------------
Line     123456789012 Ident Kind Rwy*  Dist  Brg Best Appr  Com    Freq Length
2        A       S60      S60     3   180   6.7  135    0        CTF  122.70  10000
6        A       W55      W55     3   180  12.4  173    0        CTF  122.90   5000
7        A       83Q      83Q     3   150  12.5  241    0        CTF  122.90  12000
8        A       0W0      0W0     3   201  12.5  172    0        CTF  122.90   9500
19       A       W36      W36     3   140  20.4  163    0        CTF  124.70   5000
41       A       2WA2    2WA2     3   200  27.4  186    0                0.00  15000
45       A       WN19    WN19     3   110  30.9  196    0                0.00   6000
69       A       21H      21H     3   156  40.9  343    0        CTF  128.25   5000
74       A       W37      W37     3    40  42.2  190    0        CTF  122.90   5500
91       A       WA81    WA81     3    70  46.4  332    0                0.00   4000
```