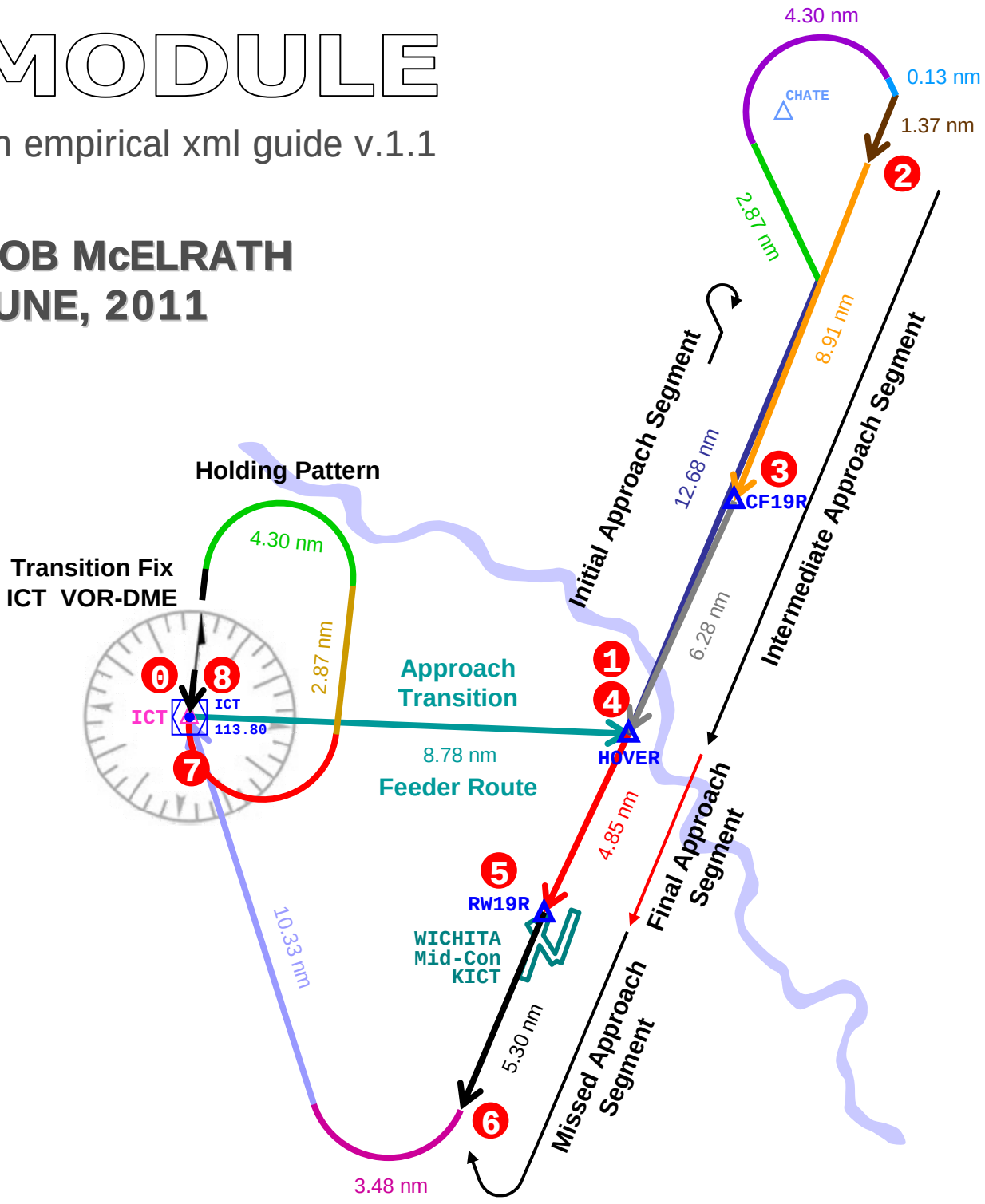
# FS9GPS MODULE

an empirical xml guide v.1.1

**BOB McELRATH**

**JUNE, 2011**



# TABLE OF CONTENTS

# INTRODUCTION

This is an empirical guide for working with the FS9 gps module. It's a collection of 'notes to myself' on utilization of the fs9gps module, definition and examples of its 320 variables, and discussion of some xml coding techniques that are used when working with the fs9gps. Started out as the GPSViewer spreadsheet then grew into a book.

What it's not - This isn't a discussion of the stock gps_500 xml gauge. None of the 80+ custom @g functions found in that gauge are discussed (however, *none* of those are necessary to fully use all of the capabilities of the gps module). Nor is this a pilot's handbook on the operation of a gps unit. It doesn't directly address the FSX gps module, although in my opinion, if you know FS9 gps, you will have little difficulty with FSX which seems easier to work with (less need for ICAO transfers, for example). Finally, it does not explain, per se, how to write sophisticated gauges such as the gps_500 or the Garmin 1000 or autopilots that use the gps module.

I have learned much from forum posts by several gps veterans such as Jan Van Harten, Paul (PVE), and especially, Tom Aguilo, Roman Stoviak, and Ed Wilson. Luckily, our community has some professional and experienced programmers that have always been willing to offer advice. Bill Leaming heads that list. Still, a significant number of things having to do with the fs9gps are not addressed by MSFT or in the forums, and that which is out there is scattered around … and vanishing sometimes, so I have attempted to fill in the 'white space' and pull it together into one document.

A warning is that I am a layman, neither a long time FS gauge hobbyist or a professional programmer. Frankly, I'm not an especially strong programmer for that matter and I realize there may be alternative approaches to those I suggest. Also, the pros will find some mistakes in here relating to nomenclature, syntax, or basic understanding of the way Flight Simulator and the gps.dll module really work. When you do, I'd appreciate receiving an email or a forum response so I can correct the errors.

I have relied a lot on some basic tools such as BlackBox2, GPSViewer, and many fit for purpose xml scripts to study the fs9gps … test, test, test, and re-test. I mention the tools to acknowledge the contribution of my son, Robbie McElrath – the author of BlackBox (and the professional programmer of the family).

Bob McElrath
Jakarta, Indonesia
March, 2011

# PRELIMINARIES

Some odds and ends:

- FS9 Version 9.01, Build 40901.01 was used in development of this guidebook.
- Stock FS9 gps database. Not updated with third party navaid, terrain mesh, or magnetic declination files.
- I indicate gps variables using blue font, NearestNdbCurrentIdent, and xml code typically in green Courier New font **`(A:PLANE LONGITUDE, degrees)`**.
- Very often, I abbreviate gps variable names by omitting the Group Name. For example, in the Flight Plan Data Group, I refer to FlightPlanWaypointAproachTarget simply as WaypointApproachTarget.
- I use FS9 xml syntax throughout.
- I do not get into the Airport Design Editor world when discussing Approaches. I discuss only the stock fs9gps variables and approach segment definitions.
- The aircraft used for sim testing was predominantly Flight 1's Cessna C421 twin. The aircraft configuration remains unchanged from the default settings except for correction of the max_indicated_speed and cruise_speed reversal. Flight simulation testing was performed with wind speed set to zero and gyro drift disabled.
- An Autopilot was used on all flight testing. FS9's stock Bendix-King Radio AP.
- Aviation nomenclature used is predominantly USA standard from Federal Aviation Administration resources (Aeronautical Information Manual, Instrument Flying Handbook, Instrument Procedures Handbook, in particular) http://www.faa.gov/library/manuals/aviation/, or Microsoft's FS9 Help section.
- English units were used in preparation of this guidebook.
- Flight Sim tools and fs9gps resources used: BlackBox2, GPSViewer1.2, numerous specific use xml gauge scripts, and the gauge and panel sections of several flight sim forums (AVSIM, FS Developer, Freeflight Design, FlightSim, Simviation).

# GET, SET, and UNITS

## GET AND SET

Fs9gps variables are classified as follows:

- **Get.** Read-only. Strings and numbers associated with these variable types can be displayed in <Elements>.

- **Set.** Write-only. These are a little more difficult to work with because you cannot directly display the values entered into these variables in display Elements. The use of a 'shadowing' L:Var is sometimes necessary in order to view what value was Set. As an example, I use the following to view confirmation of what was actually *entered* into the Set-only variable, FlightPlanLoadApproach:

  ```
  (L:LoadApprEnum, enum) d (>c:fs9gps:FlightPlanLoadApproach)
  (>L:LoadApprEnumEntered, enum)
  ```

  and then display `(L:LoadApprEnumEntered, enum)` in an <Element>.

  I have LoadApprEnum to begin with, but if I want to confirm what was really *entered*, I need LoadApprEnumEntered because FlightPlanLoadApproach cannot be displayed.

- **Get and Set.** Read and Write capable.

## UNITS

Like most other FS9 variables (e.g., A:, L:, P:, E:), gps variables have units. Throughout this guidebook, however, I omit specifying the units for string, enum and bool variables, but include units for distance, direction, time, speed, and frequency-type gps variables.

Examples:

- (c:fs9gps:NearestVorCurrentFilter) - enum variable
- (c:fs9gps:NearestVorCurrentDistance, NMiles) – distance variable
- (c:fs9gps:NearestVorCurrentFrequency, MHz) – frequency variable
- (c:fs9gps:NearestVorCurrentLongitude, degrees) – direction variable
- (c:fs9gps:NearestVorCurrentICAO) – string variable

## fs9gps VARIABLE GROUPS

There are 320 gps variables in the fs9gps gps module that are organized into the following functional data groups:

- ❑ Waypoint Data Groups:
  - WaypointAirport Group
  - WaypointIntersection Group
  - WaypointNdb Group
  - WaypointVor Group
- ❑ Search Data Groups:
  - Nearest Search Groups:
    - o NearestAirport Group
    - o NearestIntersection Group
    - o NearestVor Group
    - o NearestNdb Group
    - o NearestAirspace Group
  - ICAOSearch Group
  - NameSearch Group
- ❑ Message Data Group
- ❑ FlightPlan Data Group
- ❑ Facility Data Group
- ❑ GeoCalc Data Group

The FSX gps module uses the same functional groups but contains additional variables. Many of the new variables simply augment the Nearest Groups with variables from the Waypoint Groups making Nearest searches more robust and reducing the need for ICAO Transfer to access some Waypoint Group information.

## FS REGIONS

The FS9 world is divided into approximately 285 Regions following **I**nternational **C**ivil **A**viation **O**rganization format. The first letter of the two letter Region code is generally determined by continent. The second letter generally represents a country within that continent. The exception is some larger countries that have single-letter country codes such as the USA which uses country code "K".

The Region code is part of the fs9gps ICAO Identifier, in character positions 2 and 3.

**I**nternational **C**ivil **A**viation **O**rganization Region Code


http://en.wikipedia.org/wiki/International_Civil_Aviation_Organization_airport_code

***A - Western South Pacific***

| | |
|---|---|
| AG | Solomon Islands |
| AN | Nauru |
| AY | Papua New Guinea |

***B - Iceland/Greenland and Kosovo***

| | |
|---|---|
| BG | Greenland |
| BI | Iceland |
| BK | Kosovo |

***C - Canada***

| | |
|---|---|
| C | Canada (Multiple) |

***D - West Africa***

| | |
|---|---|
| DA | Algeria |

***N - South Pacific***

| | |
|---|---|
| NC | Cook Islands |
| NF | Fiji, Tonga |
| NG | Kiribati (Gilbert Islands), Tuvalu |
| NI | Niue |
| NL | Wallis and Futuna |
| NS | Samoa, American Samoa |
| NT | French Polynesia |
| NV | Vanuatu |
| NW | New Caledonia |
| NZ | New Zealand, Antarctica |

***O - Southwest Asia (excluding Israel and Turkey), Afghanistan and Pakistan***

DB Benin
DF Burkina Faso
DG Ghana
DI Côte d'Ivoire
DN Nigeria
DR Niger
DT Tunisia
DX Togolese Republic

***E - Northern Europe***

EB Belgium
ED Germany (civil)
EE Estonia
EF Finland
EG United Kingdom
EH Netherlands
EI Ireland
EK Denmark
EL Luxembourg
EN Norway
EP Poland
ES Sweden
ET Germany (military)
EV Latvia
EY Lithuania

***F - Southern Africa***

FA South Africa
FB Botswana
FC Republic of the Congo
FD Swaziland
FE Central African Republic
FG Equatorial Guinea
FH Ascension Island
FI Mauritius
FJ British Indian Ocean Territory
FK Cameroon
FL Zambia
FM Comoros, Madagascar, Mayotte, Réunion
FN Angola
FO Gabon
FP São Tomé and Príncipe
FQ Mozambique
FS Seychelles
FT Chad
FV Zimbabwe
FW Malawi
FX Lesotho
FY Namibia
FZ Democratic Republic of the Congo

***G - Northwestern Africa***

OA Afghanistan
OB Bahrain
OE Saudi Arabia
OI Iran
OJ Jordan and the West Bank
OK Kuwait
OL Lebanon
OM United Arab Emirates
OO Oman
OP Pakistan
OR Iraq
OS Syria
OT Qatar
OY Yemen

***P - Eastern North Pacific***

PA Alaska only
PB Baker Island
PC Kiribati (Canton Airfield, Phoenix Islands)
PF Fort Yukon, Alaska
PG Guam, Northern Marianas
PH Hawaii only
PJ Johnston Atoll
PK Marshall Islands
PL Kiribati (Line Islands)
PM Midway Island
PO Oliktok Point, Alaska
PP Point Lay, Alaska
PT Federated States of Micronesia, Palau
PW Wake Island

***R - Western North Pacific***

RC Republic of China (Taiwan)
RJ Japan (most of country)
RK South Korea
RO Japan (Okinawa Prefecture and Yoron)
RP Philippines

***S - South America***

SA Argentina
SB Brazil (also SD, SI, SJ, SN, SS and SW)
SC Chile
SD Brazil (also SB, SI, SJ, SN, SS and SW)
SE Ecuador
SF Falkland Islands
SG Paraguay
SI Brazil (also SB, SD, SJ, SN, SS and SW)
SJ Brazil (also SB, SD, SI, SN, SS and SW)
SK Colombia
SL Bolivia
SM Suriname
SN Brazil (also SB, SD, SI, SJ, SS and SW)

GA Mali
GB The Gambia
GC Canary Islands (Spain)
GE Ceuta and Melilla (Spain)
GF Sierra Leone
GG Guinea-Bissau
GL Liberia
GM Morocco
GO Senegal
GQ Mauritania
GS Western Sahara
GU Guinea
GV Cape Verde

***H - Northeastern Africa***

HA Ethiopia
HB Burundi
HC Somalia (including Somaliland because of disputes)
HD Djibouti (also HF)
HE Egypt
HF Djibouti (also HD)
HH Eritrea
HK Kenya
HL Libya
HR Rwanda
HS Sudan
HT Tanzania
HU Uganda

***K - United States (excluding Alaska and Hawaii)***

K Contiguous United States (K1, K2, K3 …K7 )

***L - Southern Europe, Israel and Turkey***

LA Albania
LB Bulgaria
LC Cyprus
LD Croatia
LE Spain
LF France, including Saint-Pierre and Miquelon
LG Greece
LH Hungary
LI Italy
LJ Slovenia
LK Czech Republic
SO French Guiana
SP Peru
SS Brazil (also SB, SD, SI, SJ, SN and SW)
SU Uruguay
SV Venezuela
SW Brazil (also SB, SD, SI, SJ, SN and SS)
SY Guyana

***T - Caribbean***

TA Antigua and Barbuda
TB Barbados
TD Dominica
TF Guadeloupe
TG Grenada
TI U.S. Virgin Islands
TJ Puerto Rico
TK Saint Kitts and Nevis
TL Saint Lucia
TN Netherlands Antilles, Aruba
TQ Anguilla
TR Montserrat
TT Trinidad and Tobago
TU British Virgin Islands
TV Saint Vincent and the Grenadines
TX Bermuda

***U - Russia and former Soviet States***

U Russia (except UA, UB, UD, UG, UK, UM and UT)
UA Kazakhstan, Kyrgyzstan
UB Azerbaijan
UD Armenia
UG Georgia
UK Ukraine
UM Belarus and Kaliningrad, Russia
UT Tajikistan, Turkmenistan, Uzbekistan

***V - South Asia (except Afghanistan and Pakistan), mainland Southeast Asia, Hong Kong and Macau***

VA India (West Zone, Mumbai Center)
VC Sri Lanka
VD Cambodia
VE India (East Zone, Kolkata Center)
VG Bangladesh
VH Hong Kong
VI India (North Zone, Delhi Center)

LL Israel
LM Malta
LN Monaco
LO Austria
LP Portugal, including the Azores
LQ Bosnia and Herzegovina
LR Romania
LS Switzerland
LT Turkey
LU Moldova
LV Areas Under the Control of the Palestinian Authority
LW Macedonia
LX Gibraltar
LY Serbia and Montenegro
LZ Slovakia

***M - Central America and Mexico***

MB Turks and Caicos Islands
MD Dominican Republic
MG Guatemala
MH Honduras
MK Jamaica
MM Mexico
MN Nicaragua
MP Panama
MR Costa Rica
MS El Salvador
MT Haiti
MU Cuba
MW Cayman Islands
MY Bahamas
MZ Belize
VL Laos
VM Macau
VN Nepal
VO India (South Zone, Chennai Center)
VQ Bhutan
VR Maldives
VT Thailand
VV Vietnam
VY Myanmar

***W - Maritime Southeast Asia (except the Philippines)***

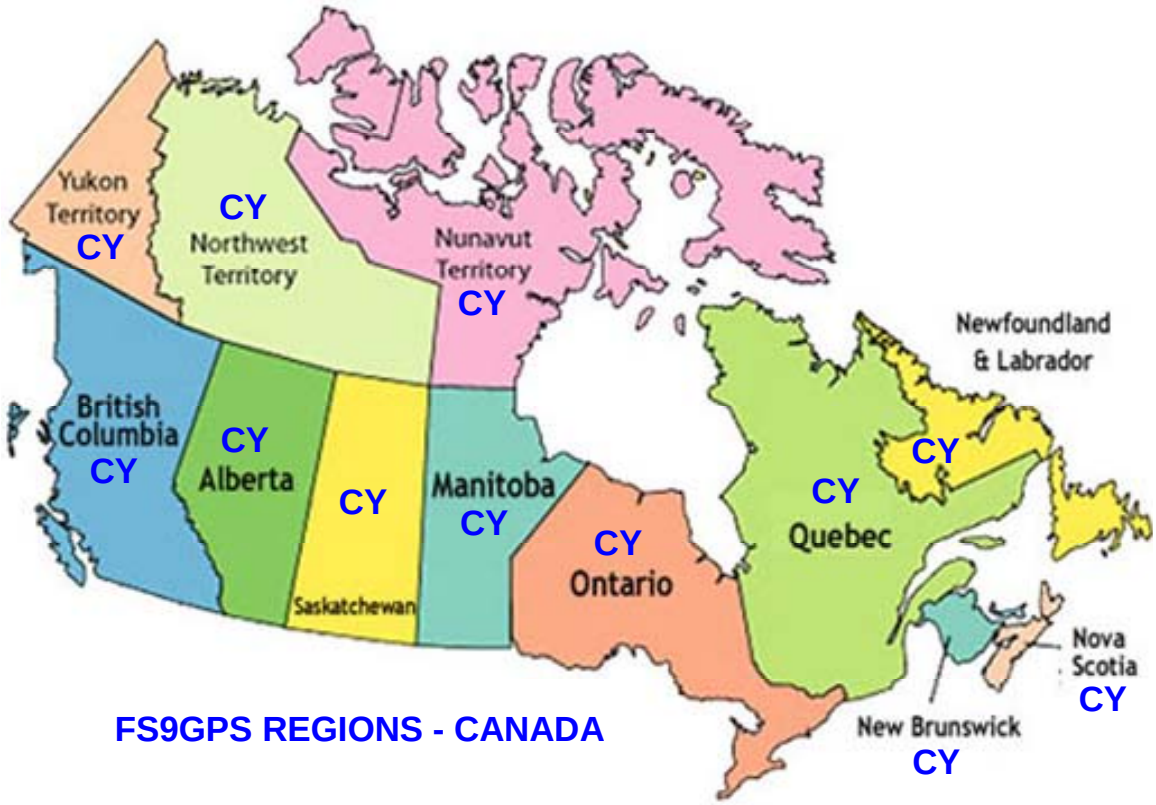
WA Indonesia (also WI, WQ and WR)
WB Malaysia (East Malaysia), Brunei
WI Indonesia (also WA, WQ and WR)
WM Malaysia (Peninsular Malaysia)
WP Timor-Leste
WQ Indonesia (also WA, WI and WR)
WR Indonesia (also WA, WI and WQ)
WS Singapore

***Y - Australia***

Y Australia

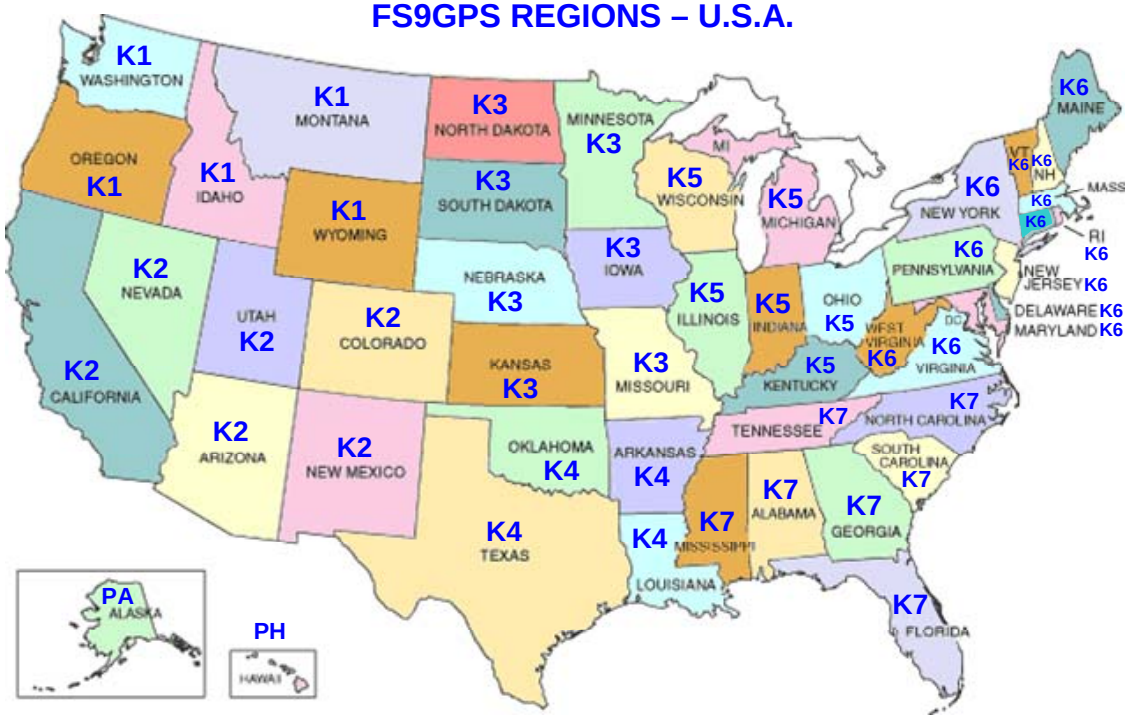
***Z - East Asia (excluding Hong Kong, Japan, Macau, South Korea and Taiwan)***

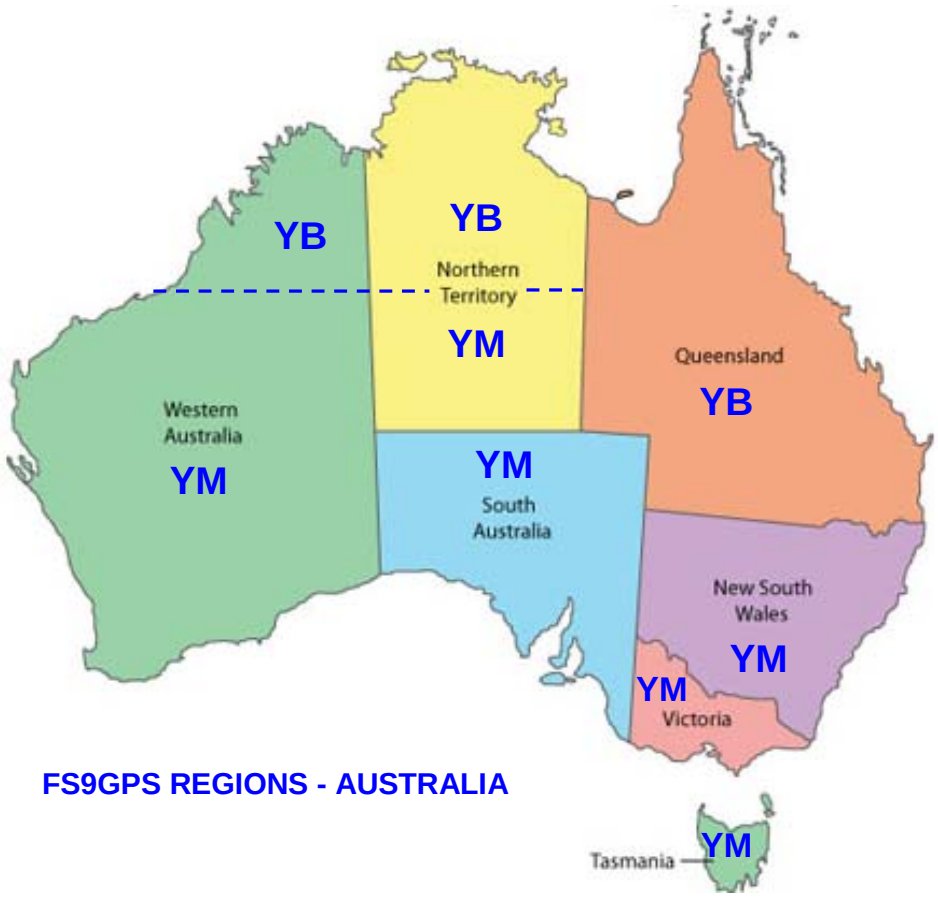
Z People's Republic of China (except ZK and ZM)
ZK North Korea
ZM Mongolia




**FS9GPS REGIONS - CANADA**

**FS9GPS REGIONS – U.S.A.**






**FS9GPS REGIONS - AUSTRALIA**

# ICAO

The ICAO is a 12 character* long unique identification string for all facilities in the fs9gps database. It is required for access to almost all of the variables within the Waypoint Groups, where most of the fs9gps database information is located. The exception is that latitude, longitude and Ident of the facility the ICAO represents are accessible without the need to retrieve Lat and Lon from the specific Waypoint Group.

All single point facilities (Airport, VOR, NDB, Intersection) in the database have a unique ICAO. Airspaces are not single point facilities and consequently do not have an ICAO.

## ICAO Format

### fs9gps ICAO - Examples

| | | Character position | | | | | | | | | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FACILITY | SLEN | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 |
| Airport | 12 | A | | | | | | | E | G | N | X | |
| Airport | 12 | A | | | | | | | W | 3 | 6 | | |
| Airport ILS | 12 | V | | | K | A | S | T | I | A | S | T | |
| VOR | 12 | V | E | G | | | | | M | I | D | | |
| Intersection | 12 | W | K | 1 | | | | | E | G | R | E | T |
| Waypoint | 12 | W | V | T | V | T | B | D | C | F | 0 | 3 | R |
| NDB | 12 | N | P | A | | | | | C | M | Q | | |
| NDB | 12 | N | E | B | E | B | B | R | O | Z | | | |
| | | Type | Region | | "Owning" Airport Ident | | | | Facility Ident | | | | |

The ICAO is assembled from four parts that are concatenated to form the 12 character ICAO identifier. The four parts are:

- **Type**. Character position 1. A single letter representing the type of facility.
  - "A" = Airport
  - "V" = VOR, ILS, LOC
  - "N" = NDB
  - "W" = Waypoint, Intersection
  - "R" = Runway
  - "X" = NDB
- **Region**. Character positions 2 and 3. The two letter FS Region code. Note that the Airport Group, which includes ILS and LOC, does not include Region in the ICAO (which is why WaypointAirportRegion always returns a blank string).

- **Owning Airport Ident**. Character positions 4 through 7. For navigation facilities (ILS, NDB) and points (Waypoints, Intersections) that are part of an approach procedure in the fs9gps database, the Ident of the airport to which the procedure belongs is indicated. Without Owning Airport Ident, the ICAO would not be unique for Computer Navigation Fix and unnamed waypoints. As an example, 'CF19R' is the Ident of a computer fix waypoint which may be part of an approach at multiple airports having a runway 19R. Without including the associated, or owning, airport, the ICAO "WK3____CF19R" would probably not be unique. However, adding the owning airport Ident makes it unique, "WK3KICTCF19R".

- **Ident**. Character positions 8 through 12. The 1 to 5 character long Ident of the facility. Note that while ICAOs are unique, Idents are not necessarily unique other than airport Idents. For example, there are many occurrences in the database of VORs having the same 3 letter Ident.

***** The String Length (SLEN) of an ICAO is always 12, even in cases where the Ident is not 5 characters long. However, when directly entering an ICAO, such as:

```
'A      W36' (>C:fs9gps:WaypointAirportICAO)
```

it is acceptable to omit the trailing spaces.

# GPS DATABASE SEARCH: Search> Index> Display

Most information needed from the fs9gps database must be obtained through a sequence of 1) Search, 2) Index, and then 3) Display. It is important to understand that the Search step can be computationally extensive. Database information cannot be manipulated or displayed until it is retrieved from the database. More than one gauge update cycle is often required to *retrieve* data from the database. Consequently, the gps module has been described by some as having asynchronous behavior – data cannot be retrieved from the database and then displayed or otherwise utilized in the same gauge update cycle.

## SEARCH (and EXTRACT)

Search is the action of requesting and receiving a small (tiny) portion of the global data base for subsequent manipulation by the gauge – manipulation that can be as simple as display of the information.

In general, the first step is to define 1) the type of information desired, 2) the geographic location of interest, and 3) the amount of information (the limit) the user wants to extract from the gps data base. For example, if a list of VORs nearest the aircraft is needed, a NearestVor search can be initiated using the following statements:

```
(A:PLANE LATITUDE, degrees)
(>c:fs9gps:NearestVorCurrentLatitude, degrees)

(A:PLANE LONGITUDE, degrees)
(>c:fs9gps:NearestVorCurrentLongitude, degrees)

20 (>c:fs9gps:NearestVorMaximumItems, enum)
100 (>c:fs9gps:NearestVorMaximumDistance, nmiles)
```

From these instructions, the gps engine knows the type of information requested (nearest VORs), the geographic location of interest (lat and lon of the aircraft) and the maximum amount of information desired (max items and max search distance, or radius).

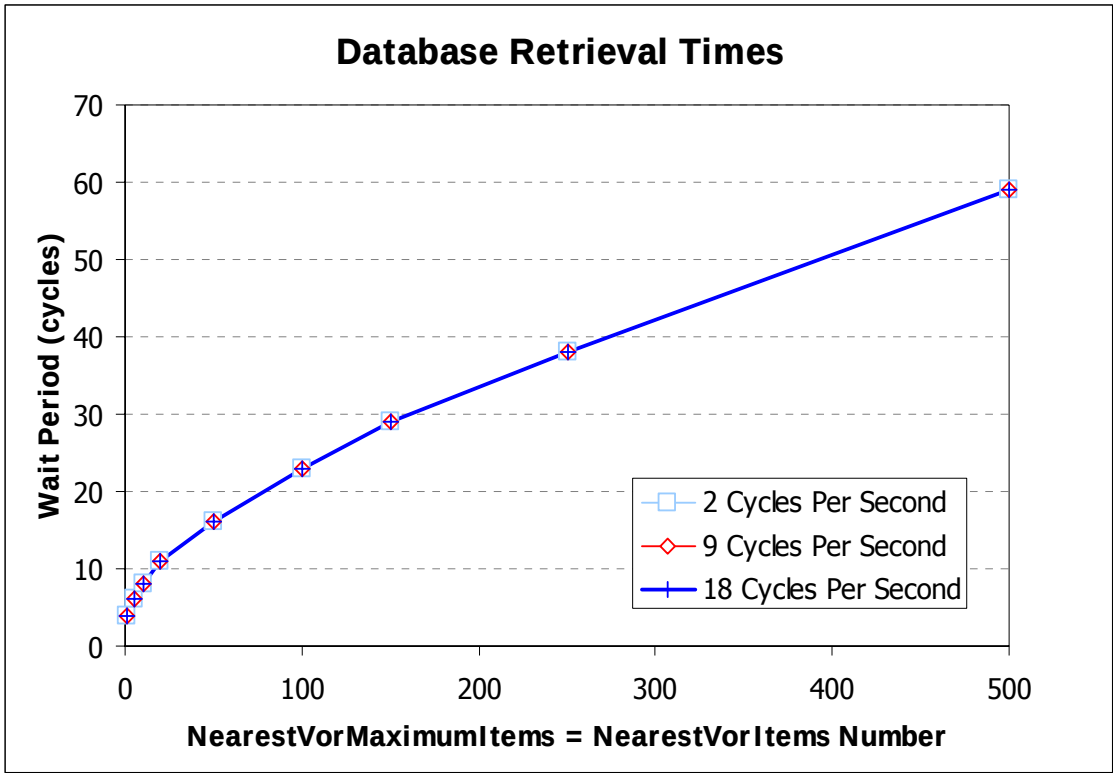
As soon as these statements are executed, a gps database search automatically begins. Some amount of time, however small, is required to extract the data, so you must wait for database search results to be delivered. How many gauge update cycles you wait is predominantly a function of the amount information requested: max distance and max items. In the stock gps_500 xml gauge, max items and max distance values are set realistically low (e.g., lines 416-419, max items = 9) and it seems that data are sometimes displayed *almost* instantly. However, if one wants a list of nearest airports within 1000 nmiles of Chicago O'Hare airport with a max item limit of 10,000, be prepared to wait. That order will take a minute or so to prepare. Regardless of the gauge update cycle rate, several, and in unrealistically large searches, thousands of update cycles may pass before search and extract is complete.

The figure below shows the gauge update cycle wait plotted against MaximumItems for an arbitrary NearestVor search. Three things are noteworthy:

- ❑ First, retrieval of just <u>one</u> item in this particular NearestVor search required 4 gauge update cycles before the retrieved data could be accessed and stored as an L:Var, or even just displayed on the screen.

- ❑ Second, the same number of gauge update cycles are required for the search regardless of gauge update cycle frequency. So, the key is to wait on update cycles, not absolute elapsed time, for data retrieval.

- ❑ Third, the bigger the search, the longer the wait.




The **multiple update cycle** database operations include:

1) **Nearest searches**, and

2) **ICAO transfer**

Does this matter? Often, no. Waiting on data extraction may not harm the function of the gauge you are building. This is especially true if searches are simple and retrieved data are only displayed on the screen, which describes most searches used in the gps_500 gauge. For example, a display loop within an <Element> could, in effect, just

sit there displaying blank lines, waiting for search data to become available, and it does not matter how few or how many update cycles pass before that happens. Often, the delay is short, almost imperceptible, and of no consequence.

However, there are situations where the user must wait until data have been retrieved from the database before executing subsequent code. These situations, as well as knowing how many gauge update cycles to wait, are the topics of the **Asynchronous Operation** section.

## INDEX

Much of the information retrieved from the gps database is structured, that is, organized in the form of lists: the list of nearest airports or VORs, the list of radio frequencies or runways at an airport, the list of waypoints in a Flight Plan. In a list of nearest airports, for example, all retrieved data from a specific, individual airport can be thought of as occupying one line of the list. The lines are numbered, or indexed, and to display or access data from any particular line, the line number must first be specified. This is accomplished by assigning a number to an index pointer such as the CurrentLine or Index variable.

```
2 (>c:fs9gps:NearestVorCurrentLine, enum)
```

selects the third VOR of a NearestVor list (indices and line numbers start at 0 for the first line).

The CurrentLine / Index pointers and Total Number variables in fs9gps include:

| **Index Pointer** (all are enum) | **Total Number** (all are enum) |
|---|---|
| WaypointAirportCurrentFrequency | WaypointAirportFrequenciesNumber |
| WaypointAirportCurrentRunway | WaypointAirportRunwaysNumber |
| WaypointAirportCurrentApproach | WaypointAirportApproachesNumber |
| WaypointAirportApproachCurrentTransition | WaypointAirportApproachTransitionsNumber |
| NearestAirportCurrentLine | NearestAirportItemsNumber |
| NearestIntersectionCurrentLine | NearestIntersectionItemsNumber |
| NearestVorCurrentLine | NearestVorItemsNumber |
| NearestNdbCurrentLine | NearestNdbItemsNumber |
| NearestAirspaceCurrentLine | NearestAirspaceItemsNumber |
| FlightPlanWaypointIndex | FlightPlanWaypointsNumber |
| IcaoSearchMatchedIcao | IcaoSearchMatchedIcaosNumber |
| MessageCurrentLine | MessageItemsNumber |
| FlightPlanWaypointApproachIndex | FlightPlanApproachWaypointsNumber |

The table below shows the results of a NearestVor search. It demonstrates the line-by-line list nature of the retrieved data and the different types of VOR information available from a FS9 NearestVor search - the ICAO, VOR Ident, Type, Frequency, Distance and True Bearing to VOR. The *nearest* VOR to the reference latitude and longitude is CurrentLine=0 (Index 0), the FFA VOR, which is 12.4 nmiles distant.

NEAREST VOR SEARCH

| | | | | | |
|---|---|---|---|---|---|
| 51.4943 | :Current Lat | 10 | :Max Items | 10 | :Items Num |
| -1.6551 | :Current Lon | 100 | :Max Dist | 62 | :Filter |

NearestVorCurrent

| | ICAO | 111 | | | | | |
|---|---|---|---|---|---|---|---|
| Line | 123456789012 | | Ident | Type | Freq | Dist | Brg |
| 0 | VEG | FFA | FFA | 3 | 113.40 | 12.4 | 334 |
| 1 | VEG | LYE | LYE | 3 | 109.80 | 12.6 | 274 |
| 2 | VEG | BZN | BZN | 3 | 111.90 | 15.4 | 7 |
| 3 | VEG | CPT | CPT | 2 | 114.35 | 16.3 | 90 |
| 4 | VEG | BDN | BDN | 3 | 108.20 | 21.0 | 190 |
| 5 | VEG | OX | OX | 3 | 117.70 | 23.8 | 31 |
| 6 | VEG | GOS | GOS | 3 | 115.55 | 30.6 | 322 |
| 7 | VEG | ODH | ODH | 3 | 109.60 | 30.8 | 120 |
| 8 | VEG | BLC | BLC | 3 | 116.20 | 32.0 | 108 |
| 9 | VEG | SAM | SAM | 2 | 113.35 | 34.4 | 160 |

## DISPLAY (use of the extracted data)

Extracted data are either displayed to the pilot on the screen of a gauge like the gps_500, or used in calculations in the gauge's code. In either case, the Index pointer / CurrentLine must first be specified to access any information that is indexed, that is in the form of a list. It's important to note that defining the index value and then display or other calculation using the extracted data occur in the same gauge update cycle. Only the database search or an ICAO Transfer consume multiple gauge update cycles.

Simple displays of extracted lists are common. The easiest way to do this is through the use of {loop} … {next} in a <String> section within an <Element>. The xml used to display the NearestVor list is shown below:

```
<Element Name="NEAREST VOR LOOP DISPLAY">
        <Position X="10" Y="25"/>
        <FormattedText X="800" Y="800" Font="Courier New" FontSize="12" LineSpacing="12"
        Color="#101010" Bright="Yes" >
        <Color Value="blue"/>
        <Color Value="darkgreen"/>
            <String>
            \{clr2}NEAREST VOR SEARCH\n\n\{clr3}
            %((C:fs9gps:NearestVorCurrentLatitude, degrees))%!9.4f! :Current Lat
            %((C:fs9gps:NearestVorMaximumItems, enum))%!5d! :Max Items
            %((@c:NearestVorItemsNumber))%!4d! :Items Num\n
            %((C:fs9gps:NearestVorCurrentLongitude, degrees))%!9.4f! :Current Lon
            %((C:fs9gps:NearestVorMaximumDistance, nmiles))%!5d! :Max Dist
            %((@c:NearestVorCurrentFilter))%!5d! :Filter
            \n\n\{clr2}
            ---------------- NearestVorCurrent ----------------\n
           %      ICAO     111\n
            Line  123456789012  Ident Type     Freq    Dist   Brg\n
                %((@c:NearestVorItemsNumber) s2 0 !=)
                    %{if}
                        %(0 sp1)
                        %{loop}
                            %(11 (>@c:NearestVorCurrentLine))
                                \{clr}%((@c:NearestVorCurrentLine))%!-5d!
                                %((@c:NearestVorCurrentICAO))%!13s!
                                %((@c:NearestVorCurrentIdent))%!7s!
                                %((@c:NearestVorCurrentType))%!5d!
                                %((@c:NearestVorCurrentFrequency, mhz))%!9.2f!
                                %((@c:NearestVorCurrentDistance, nmiles))%!8.1f!
                                %((@c:NearestVorCurrentTrueBearing, degrees))%!6d!\n
                                %(11 ++ s1 12 &lt;)
                        %{next}
                    %{end}
            </String>
        </FormattedText>
</Element>
```

The display is constructed one line at a time as the required index pointer, NearestVorCurrentLine, is incremented each pass through the loop. Line 82 is the indexing action, lines 83 through 89 are the display instructions for the list of VORs, and line 90 is the 'incrementer'. Display Loops are discussed further in the <ELEMENT> DISPLAY LOOPS chapter.

Similarly, data used in calculations rather than screen display must first be extracted from the database, then an index pointer specified.

```
2 (>c:fs9gps:NearestVorCurrentLine, enum)
```

places the index pointer to the third line of the NearestVor data extraction. Then, in the case of the NearestVor search above,

```
(C:fs9gps:NearestVorCurrentFrequency, MHz)
```

returns 111.90.

In another gauge in the panel, `(c:fs9gps:NearestVorCurrentFrequency, MHz)` will return 0.00 (or be empty for a string value) because indexed gps variables are local to the host gauge. As Tom Aguilo pointed out in one of his forum posts (edited to address NearestVor):

"the visibility/status of an indexed gps var is local to the gauge itself - its "instance" - In the other gauge case, NearestVorCurrentFrequency is a different "instance" of the same NearestVorCurrentFrequency found in the gauge containing the NearestVor search, therefore it needs to be initialized too.

There are other gpsvars that are "public", or visible through the entire panelset, for example FlightPlanIsActiveFlightPlan, FlightPlanTitle, etc.

It is important to add also that gpsvar names are case sensitive."