A Hitchhikers Guide to forIGES

Version 0.1 (Beta Release 1)

Written by: Richard Weed

August 19, 2024

Introduction

forIGES is a set of Modern Fortran utilities for reading and writing a subset of the IGES CAD file entities described in the IGES 5.3 specification. The focus of forIGES is on geometric entities such as NURBS curves and surfaces. Therefore, the complete IGES specification is not supported. Users should obtain a copy of the IGES 5.3 specification prior to attempting to use this software. A PDF of the 5.3 specification can be obtained from here

The succeeding sections of this document will describe the file structure of a typical IGES file, the IGES entity types supported by forIGES, a description of the software that makes up the forIGES utilities and the test programs, a discussion of typical use cases for the library, and instructions on how to build the software.

Overview of the IGES file structure

Users should refer to the IGES 5.3 specification reference above to get a detailed description of IGES files. This section will give a brief overview of the file structure. A typical IGES file is an ASCII text file containing 80 character records. The file is broken into five distinct sections. Each section is identified by a unique character in column 73 followed by a sequence number for each line in the section. The sequence numbers start at one for each section. These sections in the order they must appear in the file are:

1. The Start Section is identified by a letter S in column 73 and serves as a human readable prolog to the file.
2. The Global Section is identified by a letter G in column 73. A typical Global section contains a variety of information such as the IGES file name, the units (inches, centimeters etc.) used for all geometric objects in the file, the allowable precisions for floating point numbers, the author of the file and several other entries that define, the organization of the author etc. forIGES defines a set of default values for each entry in the Global section so most forIGES users will only be be interested in things like the geometric units along with author and organization name.
3. The Directory Entry Section is identified by a letter D in column 73. This section contains two eighty character records for each IGES entity in the file. This section provides an index to the file and contains attribute information about each entity. The data in the fields consist of integer data or "pointer" data in eight fixed eight character fields. Note in IGES a "pointer" is a sequence number in the Directory Entry Section. The

three most important entries in a Directory Section entry are the entity type number(Field 1), the Parameter Data field which identifies the sequence number of the starting location in the Parameter Section of the Parameter Data entity (Field 2) that corresponds to the Data Section entry and the Parameter Line Count that defines the number of lines that make up the corresponding Parameter Data entity definition (Field 14). Two other important Directory Entry fields are the Form number (Field 15) and the Transformation Matrix Field (Field 7). Several of the entities in the IGES specification have different forms that control what data must be specified. In additions each IGES entity (even drawing and viewing entities that are not supported by forIGES) can have an associated Transformation Matrix entity. All of the other fields in the individual IGES Directory Entries in forIGES have default values. With the exception of the Color Number, users should not have to change the values of the other fields. As will be discussed in the code description below, forIGES stores Directory entry data in an array of derived types. There the sequence number "pointer" values in a Directory Entry such as the Transformation Matrix pointer are converted into an array index. Users will reference an index value in the Directory Entry array instead of an IGES file sequence number. Complete details of the Directory Entry section are described in the IGES specification

4. The Parameter Data Section entities are identified by the letter P in column 73. This section contains the parameter data associated with each entity. Data is specified in a comma delimited free field format in columns one through 64. Column 65 is a space. Columns 66 through 72 holds the sequence number of the first line of the corresponding Directory Entry for each entity. Data in a Parameter Data entity can be a mixture of integers, floating point values, character strings denoted as old Fortran type Hollerith strings, and "logical" values defined as unsigned integer ones or zeros. When creating an IGES file with forIGES, the number of decimal places for floating point numbers can be controlled by the user. The current default is nine decimal places but a smaller number can be set to reduce the final IGES file size

5. The final data section is the Termination Section which consists of a single line. The values in the Termination Section define the number of records in each section.

An example of an IGES file containing the defintion of a single NURBS surface can be found here

NOTE: There is another section called the Flag section that can appear before the Start Section to indicate if the file is in a "Compressed ASCII" format. This is seldom used anymore. However, forIGES will look for it and try to convert the compressed ASCII format to standard IGES ASCII using refactored versions of the Fortran 77 code listed in the IGES specification for this purpose.

Overview of the forIGES software

The forIGES library software consists of nine source files. Each file contains the source for a Fortran module. The files in the order they need to be compiled are:

1. IGES_params.F90 - This module contains global parameters and other data used by the other modules.
2. commandLineUtils.F90 - These are a set of utilities used by the test programs to read users specified input and output filenames from the command line.
3. IGES_utils.F90 - These are utility routines for determining if an entity is supported, converting an array of variable length P section field strings to an 80 character, record, and functions for converting back and forth between the forIGES Directory entry array index and an IGES file sequence number.
4. IGES_Gsection.F90 - This module defines a Gsection_t derived type and procedures for reading, writing, and printing all of the data defined in an IGES file Global section. The data components of Gsection_t use the same names for most values as defined for the Global section in the IGES 5.3 specification.
5. IGES_Dsection.F90 - This module defines an Dentry_t derived type whose components contain the data specified in the data fields of a Directory Entry for an individual entity. Again, component names reflect the names used in the IGES specification. A separate Dsection_t derived type is provided that holds an array of Dentry_t types. Dsection_t is used to hold the data for the complete Directory Entry section of an IGES file. As with the Gsection_t type, methods are provided to read, write, and print, and de-allocate Dentry_t data.
6. IGES_Psection.F90 - This module holds the code used to process IGES Parameter sections. The heart of the code is a set of 39 polymorphic derived types extended from an abstract type (entity_t) that contain the data for the 38 supported by forIGES. Each entity type has methods for reading, writing, printing, initializing, and de-allocating the IGES entity type data specific to each entity. See the IGES 5.3 specification for a complete description of each entity type. Also included are a PE_t derived type holding an array of entity_t classes that are used to provide to hold the different polymorphic specific entities. This array can be processed with a DO loop to extract the data for each entity. An additional unsupported_t derived type can be used as a placeholder in a PE_t array to preserve the order of the entities as they appear in an IGES file. Like the G and P section modules, a Psection_t type is used to hold a PE_t array containing the entire P section of an IGES file.
7. PElist.F90 - This module holds routines to create a circular doubly-linked list of the polymorphic IGES P section entity types. There are several methods available but most users will only use the init, append, move, and delete methods. The PElist and associated DElist routines are designed to be used in building user defined output files

and extracting only desired entity types from a large IGES file.

8. DElist.F90 - This module holds routines to create a circular doubly-linked list of Dentry_t types. There are several methods available but most users will only use the init, append, move, and delete methods. DElist routines are designed to be used in conjunction with PElist routines for creating user defined IGES output files.

9. forIGES.F90 - This module is the heart of the forIGES library. It defines an IGES_t derived type and associated methods for reading a complete IGES file, printing its contents, and constructing user define IGES output files. IGES_t holds a variety of data the most important of which are type instances for the S, G, D, and P section derived types. IGES_t serves as a bridge between the data read from and IGES file and the user. Additional routines are supplied for copying a Dsection_t or Psection_t array to a list and vice versa. Other routines are supplied to allow users to extract to a list only the values specified by the user in an array containing desired type numbers.

A more detailed description of the API defined by these files can be obtained by running the FORD documentation program to generate a complete set of HTML documentation for the entire code base.

IGES Entity Types Supported by forIGES

The following IGES entities are supported by forIGES. The entities chosen are a mixture of entities supported by previous work documented in the References.

Type number Description

- 0 Null
- 100 Circular arc
- 102 Composite curve
- 104 Conic arc
- 106 Copious data (forms 1-3, 11-13, 63)
- 108 Plane
- 110 Line
- 112 Parametric Spline Curve
- 114 Parametric Spline Surface
- 116 Point
- 118 Ruled Surface
- 120 Surface of Revolution
- 122 Tabulated Cylinder
- 123 Direction
- 124 Transformation Matrix
- 126 Rational B-spline curve
- 128 Rational B-spline surface
- 130 Offset Curve
- 140 Offset Surface
- 141 Boundary
- 142 Curve on a Parametric Surface

- 143 Bounded Surface
- 144 Trimmed (parametric) surface
- 190 Plane Surface
- 192 Right Circular Cylinder Surface
- 194 Right Circular Conical Surface
- 196 Spherical Surface
- 198 Toroidal Surface
- 212 General Note
- 308 Subfigure Definition
- 314 Color Definition
- 402 Associativity Instance
- 406 Property (forms 1 and 15)
- 408 Singular Subfigure instance
- 502 Vertex list
- 504 Edge
- 508 Loop
- 510 Face

Test and Example Programs for forIGES

The test subdirectory holds six test programs that also serve as examples for the basic functionality of the forIGES routines. The test/test_files directory holds several IGES files that form the basis for the majority of the testing performed to date on the forIGES library. These files were borrowed from the pyIGES repository and were selected because they referenced several of the IGES entities supported by forIGES.

Test Program Descriptions

testReadList.F90

Reads a list of IGES files found in the test_files directory and then processes each file to create a IGES_t object. Unsupported entities are flagged and replaced by an unsupported_t placeholder. The data stored in each IGES_t object is then printed out to a FILE.out file where FILE is the name of the IGES file in the list.

testReadFile.F90

Reads a single IGES files then processes each file to create a IGES_t object. Unsupported entities are flagged and replaced by an unsupported_t placeholder. The data stored in the resulting IGES_t object is then printed out to a user specified file.

testWriteFile.F90

Reads an IGES file, creates and IGES_t object and then writes the IGES_t object to an new IGES file without the unsupported entities

testArrayFilter.F90

Reads an IGES file and generates lists of Directory section entries and Parameter section entities that are contain only the entity types defined in an array by the user. The list contents are echoed to stdout. The IGES file

that is read is hardwired to ./test_files/surf128.igs because it contains both transformation array entities and NURBS surface entities which are extracted from the input file

testFilterWrite.F90

testFilterWrite extends testArrayFilter to create a new IGES file based on the data in the filtered lists.

testGenNURBSsurf.F90

testGenNURBSsurf generates a cubic B-spline surface using data from Rogers *Mathematical Elements of Computer Graphics*, p459. It illustrates the case where a user creates there own NURBS surface and wants to output it to a CAD system. A file named RogersBspline.igs will be created.

Additional test programs will probably be added in future releases based on user feedback.

Procedures Used to Test forIGES

The testing procedure for forIGES to date has consisted of reading all the files in the /test/test_files directory and comparing the output generated by testReadList.F90 and testReadFile.F90 for each supported entity with the data in the original IGES file. The IGES file creation capability of forIGES was tested using testWriteFile.F90, testFilterWrite.F90 and testGenNURBSsurf.F90. The resulting IGES files were visually compared with either the original IGES file or the NURBS surface data in testGenNURBSsurf.F90. The final test of the file creation capability was to use either FreeCad and Paraview to read and display the IGES files generated by forIGES. The ability to correctly read IGES files for all the supported entities for which test data existed was demonstrated. FreeCad appears to correctly read and display all of the forIGES generated files I've fed it so far. I've only made three attempt to read a file with ParaView. The NURBS surface file I generated was read and displayed correctly. However, ParaView refused to read and display either one of the original files from pyIGES or a file I generated from an original file so either the original files were malformed or there is a bug in ParaView's IGES reader. I think its probably the latter. FreeCad uses the openCASCADE IGES reader which I would expect to be bulletproof by now. The results of these tests give me confidence that the basic functionality for reading and writing IGES files with forIGES has been verified. That doesn't mean the code is bug free, it probably isn't. However, it should perform correctly with most IGES files that users will throw at it.

Using forIGES

The following gives a brief overview of how to read, create, and print IGES files using forIGES. The test programs described above should be consulted for more detailed examples.

Reading an IGES file

The steps required to read an IGES file, create and IGES_t object and echo

its contents to stdout are:

```
  Use forIGES ! brings in IGES_t type and methods
  Type(IGES_t) :: iges ! Create an IGES_t instance
  Character(:), Allocatable :: filename !! define the input filename
  Character(:), Allocatable :: outfile !! define the output filename
  filename = "anIGESFile.igs"
  outfile  = "anIGESFile.out"
  Call iges%readFile(filename) ! Reads the IGES file and creates the iges S, G, D, and P
  Call iges%output(outfile) ! outputs the contents of the iges component arrays for eac
  Call iges%dealloc() ! cleans up all arrays in iges
```

Once the D and P section arrays have been obtained the user can access the data in the P and corresponding D section with a DO loop.

Creating an IGES file from user data.

This example uses the case of a user creating a file containing a user defined NURBS Surface. The typical workflow for this case looks like.

```
  USE forIGES

! Define some required temporary and local data arrays and types

  Type(Gsection_t)  :: G
  Type(Dentry_t)    :: D
  Type(entity128_t) :: surf128
  Type(DElist_t)    :: Dlist
  Type(PElist_t)    :: Plist

  Character(:),  Allocatable :: filename
  Character(80), Allocatable :: Ssec(:)

  Character(80), Allocatable :: Ssection(:)
  Character(80), Allocatable :: Gsection(:)
  Character(80), Allocatable :: Dsection(:)
  Character(80), Allocatable :: Psection(:)

  filename = "yourIGESfile.igs"

! Define the data and arrays required to construct an NURBS surface entity (Type 128)
! and then call the surf128 entity init method. See the API documentation for a descrip
! of the subroutine arguments.

! Call init method to build a entity128_t entity

  Call surf128%init(1, 0, K1, K2, M1, M2, PROP1, PROP2, PROP3, PROP4, PROP5,  &
                    S, T, W, X, Y, Z, U, V)

! Call init method to build a Dentry_t entry
```

```
  Call D%init(entity_type=128, param_data=1, transformation_matrix=0,      &
              form_number=0)

! Create Plist and Dlist to hold the entity and directory entry
! Probably could have just made these single dimension arrays and skipped
! the lists but I think most folks will use a list to hold there data
! since you can add things without worrying about the array dimension

  Call Plist%init()
  Call Dlist%init()
  Call Plist%append(surf128)
  Call Dlist%append(D)

! Set filename and sender_id to filename and author to me

  Call G%init(filename=filename, author="me")

! Define S section strings and make an Ssection

  ALLOCATE(Ssec(1))
  Ssec(1) = REPEAT(" ",80)
  Ssec(1) = " Test of creating an IGES file for a B-spline surface"

 Call makeSsection(Ssec, Ssection)

! Make the G section

  Call makeGsection(G, Gsection)

! Make the D and P sections

  Call makeDPsections(Dlist, Plist, Dsection, Psection)

! Set number of decimal points output for all real values in output file.
! Default is 9 decimal places. We set it to 7 here to reduce the size of
! the output file.

  Call setRealFormat(7)

! Write the new IGES file

  Call writeIGESfile(filename, Ssection, Gsection, Dsection, Psection)

! Clean up lists etc

  Call Plist%delete()
  Call Dlist%delete()
```

Accessing list nodes in PElist and DElist

```
! Assume you have created an existing PElist or DElist named Plist and Dlist
! and have given them a TARGET attribute. Next create a node pointer for
! each list.

  Type(PEnode_t), pointer :: pnode
  Type(DEnode_t), pointer :: dnode

  np = Plist%size() ! gives the size of the PElist
  nd = Dlist%size() ! gives the size of the DE list (should be the same as np)

  pnode => Plist%first() ! sets pnode to first node in list
  Do i=1,np
    Call pnode%outputEntity() ! prints pnode entity information to default stdout
    If (i < np) Call Plist%move(pnode,1) ! move node pointer to next node
  End Do

! Similarly for Dlist

  dnode => Dlist%first()
  Do i=1,nd
    Call dnode%output() ! prints dnode information to default stdout
    If (i < np) Call Dlist%move(dnode,1) ! move node pointer to next node
  End Do
```

This example just cycles through the lists printing out their contents. The same method can be used to extract the component information for the users purposes.

Some Potential Issues That Users Need To Be Aware Of.

1. The circular "pointer" system that IGES uses to reference back and forth between the P and D sections requires care in creating IGES files that use a transformation matrix or use entities like the composite curve entity that rely on pointers to other entities through the associated Directory Entry. Users will currently have to reconcile these dependencies manually until I can think of a workable way to automate this process.

2. While the test files in /test/test_files allowed me to test the entities that most users will care about, there are several entities that have not been tested due to lack of test data.

3. Using lists and the associated reliance on pointers always opens up the possibility of a memory leak. I have not tested for leaks yet due to my experience with leak checkers such as Valgrind giving false positives for Fortran codes but running leak checks is on my to-do list.

Building forIGES

Currently forIGES has only been built and tested on Linux systems. Support for Windows and Macs is planned for the future but they are not directly

supported in this release. However, given the relatively small number of files that make up forIGES it should be simple for Windows users to create their own Visual Studio project from the files in the src and test directories. forIGES has been built and tests run with the following compilers:

Intel ifx version 2024.1
Intel ifort version 2021.12
GCC gfortran version 13 and 14
NVIDIA nvfortran version 24.7
AMD AOCC flang version 16.0.3

Both Intel compilers, GCC, and NVIDIA appear to compile without error and generate correct results for all the forIGES test cases. flang appears to compile without error and runs the read tests correctly but produces erroneous results for the IGES file creation tests so flang cannot be recommended at this point.

Building forIGES Using Make

On Linux systems the easiest way to build forIGES is using the supplied makefiles and configure bash script.

The steps are as follows:

1. Run configure to set the desired compiler and optimization level. The defaults are gfortran and no optimization
2. Type make - this will build the library and associated mod files and place them in the lib and module directories. By default both an archive named libforIGES.a and a shared object named libforIGES.so are build
3. Type make tests to build the executables for the test programs.

Examples:

default gfortran:

```
source ./configure
```

Intel compilers:

```
source ./configure --compiler=intel-ifort
or
source ./configure --compiler=intel-ifx
```

NVIDIA compiler:

```
source ./configure --compiler=nvidia
```

AMD compiler:

```
source ./configure --compiler=amd
```

```
make
make tests
```

make cleanall - deletes all files not needed for a new build

Additonal configure script options

The configure script used with the make build also provides the following four additional options

--warn= to set the warning level for the compiler usually warn=all
--define= "some macro value" - sets a -D flag to some user macro
--opt-level= 1, 2, or 3 - some higher level of optimization than none
--build-debug=yes - sets a -DDBG macro to turn on debug sections of code

Building forIGES Using FPM

The forIGES libraries can also be built using the Fortran Package Manager FPM

To build using fpm do the following:

fpm build
fpm install --prefix=your_install_path

These will install a libforIGES.a archive in lib and the mod files in include. The default compiler for FPM is gfortran. Other compilers can be used by setting the FPM_FC and FPM_FFLAGS environment variables.

Building Code HTML Documentation with FORD

A complete set of HTML documentation describing the forIGES API can be generated using FORD. The resulting HTML pages will describe each subroutine and function interface along with definitions of various variables and derived types.

To build the FORD documentation on your local system type the following:

ford ford.md

The resulting documentation will be placed in a local doc directory

Note FORD and FPM are Python applications and can be installed using pipx or pip. FORD also requires a recent copy of graphViz.

Getting Help, Reporting Bugs etc.

Please report any bugs, new feature requests etc. on the forIGES Github repository under issues.

Future Plans

Based on user feedback, support for more entities can be added. Support for Windows and Macs is also planned. The ultimate goal of this project is to provide a way to extract non-NURBS geometric entities from an IGES file and then convert them to a NURBS description (where possible) as was done in References 2 through 5. I would also like to stand up a Github Pages site with the API documentation generated by FORD when I can figure out how to do it. Obviously, more testing will be required to move to a

production level release of the code.

References

1. US PRO, "Initial Graphics Exchange Specification IGES 5.3", 1996 https://web.archive.org/web/20120821190122/http://www.uspro.org/documents/IGES5-3_forDownload.pdf

2. Yu, Tzu-Yi, "CAGD Techniques in Grid Generation," Ph.D. Dissertation, Mississippi State University, Miss. State MS, December 1995 https://ntrs.nasa.gov/api/citations/19970014211/downloads/19970014211.pdf

3. Yu, T. and Soni, B., "NURBS in Structured Grid Generation", *Handbook of Grid Generation*, Chapter 30, Ed. J.F Thompson, et al, Taylor Francis Publishing, December 28, 1998 https://doi.org/10.1201/9781420050349 also http://ebrary.free.fr/Mesh%20Generation/Handbook_of_Grid_%20Generation,1999/chap30.pdf

4. Blake, M., Chou, J., Kerr, P., Thorp, S., "NASA Geometry Data Exchange Specification for Computational Fluid Dynamics (NASA IGES)", NASA RP 1338, 1994 https://ntrs.nasa.gov/api/citations/19950005054/downloads/19950005054.pdf

5. Evans, A. and Miller D., "NASA IGES and NASA-IGES NURBS-Only Standard," *Handbook of Grid Generation*, Chapter 31, Ed. J.F Thompson, et al, Taylor Francis Publishing, December 28, 1998 https://doi.org/10.1201/9781420050349

6. SINTEF GoTools https://www.sintef.no/en/software/software-applied-mathematics/gotools/