Kobol 1.4.0
User Manual

Table of Contents

1. Introduction

Kobol is unique in the world of COBOL development systems. Kobol will generate a true executable with no run time requirements other than its support libraries. Kobol has a sophisticated IDE that contains extensive project management capability with integrated CVS support for revision management, syntax highlighting, an optional debugger and integrated syntax reference manual. The application is very straight forward to use with a low learning curve. This manual is arranged in a fashion to make it as quick and easy as possible to get started with Kobol.

When you purchased Kobol you received versions for both Windows and Linux. This chapter will explain how to install Kobol and its various subsystems. If you're reading this document you should already have Kobol, but if you don't, you can download a demo or purchase a copy at http://www.thekompany.com/products/kobol/. When you purchase Kobol you are entitled to free electronic updates for the life of the product for all supported platforms, this does not include add-on modules that are available separately.

2. Requirements and Installation

You will need to be running a 300Mhz or better processor with at least 64MB of RAM, and either a Linux or MS Windows operating system. Koobl for Linux has been statically linked with all the libraries required, so you don't need to worry about installing anything else for Kobol - just run it.

If you are using a Linux system then you need to unpack the Kobol installation package into whatever directory you like (we recommend /usr/local), by using the tar command: tar -zxvf Kobol-1.0.0.tar.gz or tar -jxvf Kobol-1.0.0.tar.bz2 Please note that the version number will change with each release, so the file name may not match the example. The command creates the directory Kobol-1.0.0 and puts all the files there. Now go into the directory with the following command: cd Kobol-1.0.0 Now just run installation script as root: ./install.sh If you have tried to install as a user other than 'root', then you will receive a warning message. The installation script will tell you where the files are installed so if you want to remove Kobol from your Linux system you will only need to remove /usr/local/kobol directory and the binary file /usr/bin/kobol.

Keep in mind that certain Linux distributions do not come with the required development libraries, currently that would be Lindows and Xandros. They are geared for a home desktop user and as such they don’t by default include developer tools like the gcc compiler.

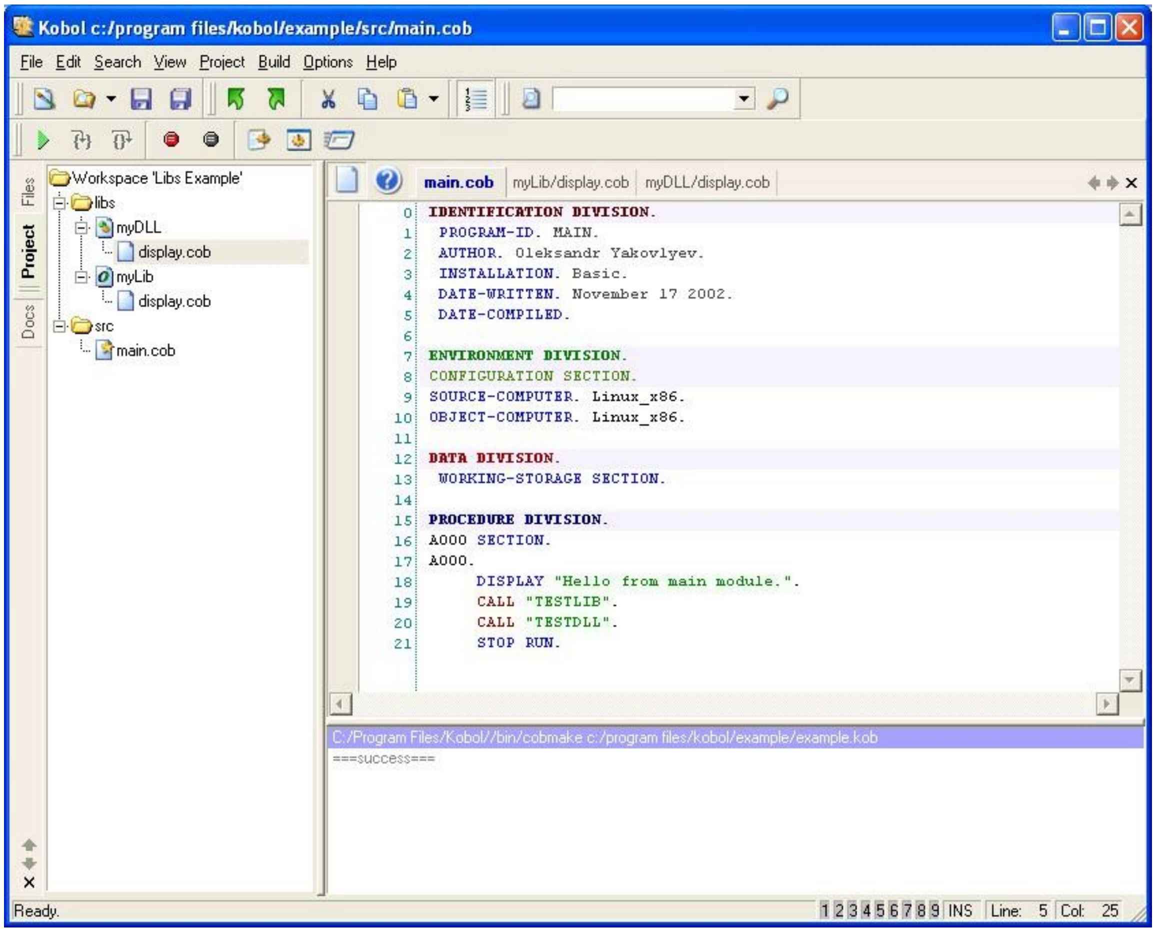
To install Kobol on a Windows system you need to run the exe installation file. Now just follow the prompts as you would with any other Windows installer. If you select a 'typical' installation, then not all features will be installed. If you select a 'full' installation, you can disable features later if you like with 'Control Panel->Removing and Add Programs'. It is also necessary for you to install the ‘CygWin’ application, this contains the compiler and various other pieces needed on Windows.

The installation script on Linux copies the Kobol binary file to /usr/bin. So to run Kobol you just need to type kobol on the command line. If /usr/bin is not in your PATH environment variable you will need to type the full path+filename - /usr/bin/kobol. Please remember that to start Kobol you don't need any other systems (except XServer) running on your Linux system, such as KDE or any other desktop manager. You can use any window manager that you wish. If you are using Windows then you need only open 'Start menu->Programs->theKompany->Kobol' and click on the shortcut for the application.

If you run Kobol from a command line then you can add command line attributes. The X11 version of Kobol knows about -display, -font attributes. For example: kobol -display=Unix:0š-font=9x15bold You can also add the name of file to open in Kobol: kobol myfile.cob Then kobol will start up and load the file into the editor immediately. You can use "open with kobol" in windows by right clicking on Cobol files. Kobol uses various external programs to provide specific features rapidly. Specifically we use cobmake and cob2gcc for compiling. Cobmake creates a Makefile from your project file created in the Kobol IDE. Cob2gcc is the code translator that will generate C++ sources from your COBOL code. The final step will use the make command to build your application. Kobol comes with both cobmake and cob2gcc integrated with the IDE so you don’t really need to worry about them.

3. Getting Started

Kobol has a standard and intuitive style interface with toolbars, menus and tabbed windows for accessing various sub functions quickly. On the left side of the main window you will see a tabbar that has various elements, such as: Project, Files, Docs. The tab bar allows you quick navigation within the work space. The main window is occupied by the editor.



```
IDENTIFICATION DIVISION.
 PROGRAM-ID. MAIN.
 AUTHOR. Oleksandr Yakovlyev.
 INSTALLATION. Basic.
 DATE-WRITTEN. November 17 2002.
 DATE-COMPILED.

ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. Linux_x86.
OBJECT-COMPUTER. Linux_x86.

DATA DIVISION.
 WORKING-STORAGE SECTION.

PROCEDURE DIVISION.
A000 SECTION.
A000.
      DISPLAY "Hello from main module.".
      CALL "TESTLIB".
      CALL "TESTDLL".
      STOP RUN.
```

```
C:/Program Files/Kobol//bin/cobmake c:/program files/kobol/example/example.kob
===success===
```

On the top of the editor you can see a tab with the names of all opened files, this allows you to easily cut/copy/paste parts of code between these windows. We understand how it is to be working hard and not want to take your hands off the keyboard, to this end you can use the Alt+Left and Alt+Right key combinations to navigate between editor windows. Pop up the ToolTips are active pretty much everywhere in the application that they are needed, and these icons are no different.

Finally, under the editor window and mouse buttons is the Message Panel (which we will explain later). You can use Ctrl+M to enable/disable the panel if you want to get some more screen real estate, or if you need it open because you are debugging. Kobol makes heavy use of 'Dock Windows', this allows you to essentially tear off any visible toolbar or panel and position it wherever you want on the screen, either internal or external to the program, docked or free floating. This allows you to customize your work environment so it is comfortable and natural for the way you work.

Kobol has two File views. They are located on the panel named "Files". The first is a list style interface, you can open a directory or select a file by single clicking on it. Here is an example: By right clicking you can switch to "Tree Mode", which is the second interface, to do copy, paste, delete; insert a file or directory in the current project and reload the current dir if there are new elements.

Our next file style is 'Tree Mode', this is where you can expand or collapse directory structures by clicking on the icon next to the name, or double clicking on the name. While this is useful for certain types of operations, it can create a cluttered view as well. You can switch back to 'list mode' by just right clicking and selecting it from the pop up menu.

The Project Panel is for managing your project and the files within it. If no project exists, then the root folder name will say "No project". If you double click on that name, you will get the "Project Open Dialog". This is a nice shortcut to using the menu bar. Right clicking in the Project panel will allow you to add, delete, create and open files as well as perform various CVS operations.

The Documentation Panel provides you ready access to various documents to help you in with your COBOL programming. Kobol has a very friendly integrated help system, however it is currently un-populated as we research and implement an HTML version of the COBOL standard. The help works when you set the cursor on the word you are interested in and press F1 on your keyboard. The documentation window will now appear with help information regarding the selected word. For example entering the word "MOVE" and pressing the F1 button will open a manual page about the attributes of the verb and its description.

The Message Panel is a basic output window where the various programs and functions of Kobol will output their messages. If the line in the message panel contains a line number, then you can highlight the line you are interested in, click on it, and the appropriate window will come up with the appropriate text. Currently Kobol only supports output from cobmake, make, cob2gcc, which makes it very handy for debugging your COBOL code.

When you see an error messages like Error at 20(12) this means that the error was at line 20 column 12.

Kobol provides keyboard shortcuts for almost all the available actions in the application.

F1 - popup help for the word/tag that the cursor is on.
F3 - find next action.
Ctrl+M - Show Message window.

File operations:
Ctrl+N - New file.
Ctrl+O - Open file.
Ctrl+W - Close file.
Ctrl+S - Save file.
Ctrl+Q –
Quit from Kobol.

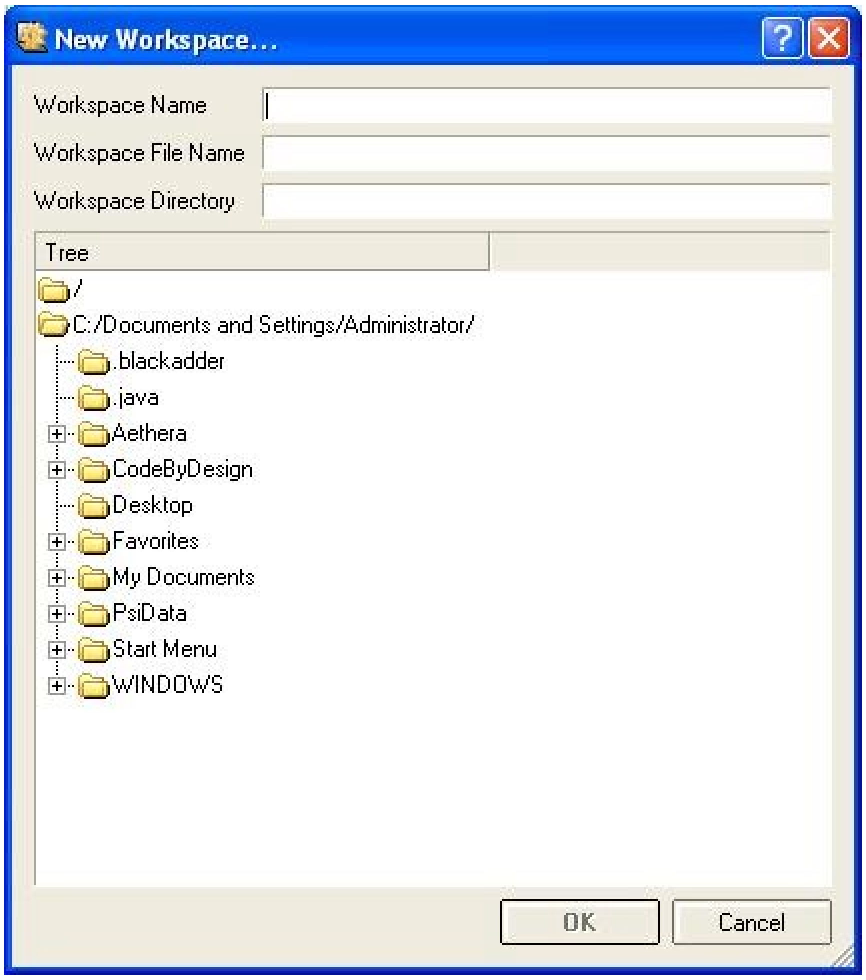
Editor operations:
Ctrl+Z - Undo.
Ctrl+Shift+Z - Redo.
Ctrl+X - Cut.
Ctrl+C - Copy.
Ctrl+V - Paste.
Ctrl+A - Select All.
Ctrl+I - Indent.
Ctrl+U - Unindent.
Ctrl+Space - Complete word.
Alt+Left, Alt+Right - switch to prev,next editor window.

Search operations:
Ctrl+F - Search.
F3 - Search next.
Ctrl+R - Replace.
Ctrl+J - Goto Line.

4. Your First Program

Project support in Kobol allows you to organize your COBOL development in a logical structure you can easily work with. Kobol works with your projects based on the file extension, you need to use .kob for the project name. The files in a project are organized in a tree - like structure that mirror the directories and file structure on your disk. You can perform operations against the entire project such as scanning files and directories, CVS operations and such. Kobol is Project oriented, you must have a project whether it is for one file or 100 or however many. There are some integrated example projects as well as a zip file containing a large selection of example files that you can also look at.

Creating new project is quite simple. By selecting the menu item "Project->New Workspace" you will see a dialog for setting the "Project directory", "Project Filename", and "Project Name" which contains some information for you. So enter in the "Project Name" a short description or name of your project. In "Project Directory" you must select the directory where the files for your project will be. If the directory does not exist you can write the full directory filename in the lineedit then the directory will be created. Also you can use the "Create New Folder" button in "Open dialog" which you can run by clicking on the button "...".



In line edit "Project File Name" you must enter the file name of a project file. The file name must have the extension ".kob". For example - "my_project.kob". Once you've done this, click on "OK" and your project will then be opened in the "Project Panel" where you can start adding files. The terms “Project” and “Workspace” are currently used interchangeably. At some future point a Workspace can be thought of as a meta-Project, in other words a Workspace will contain multiple projects.

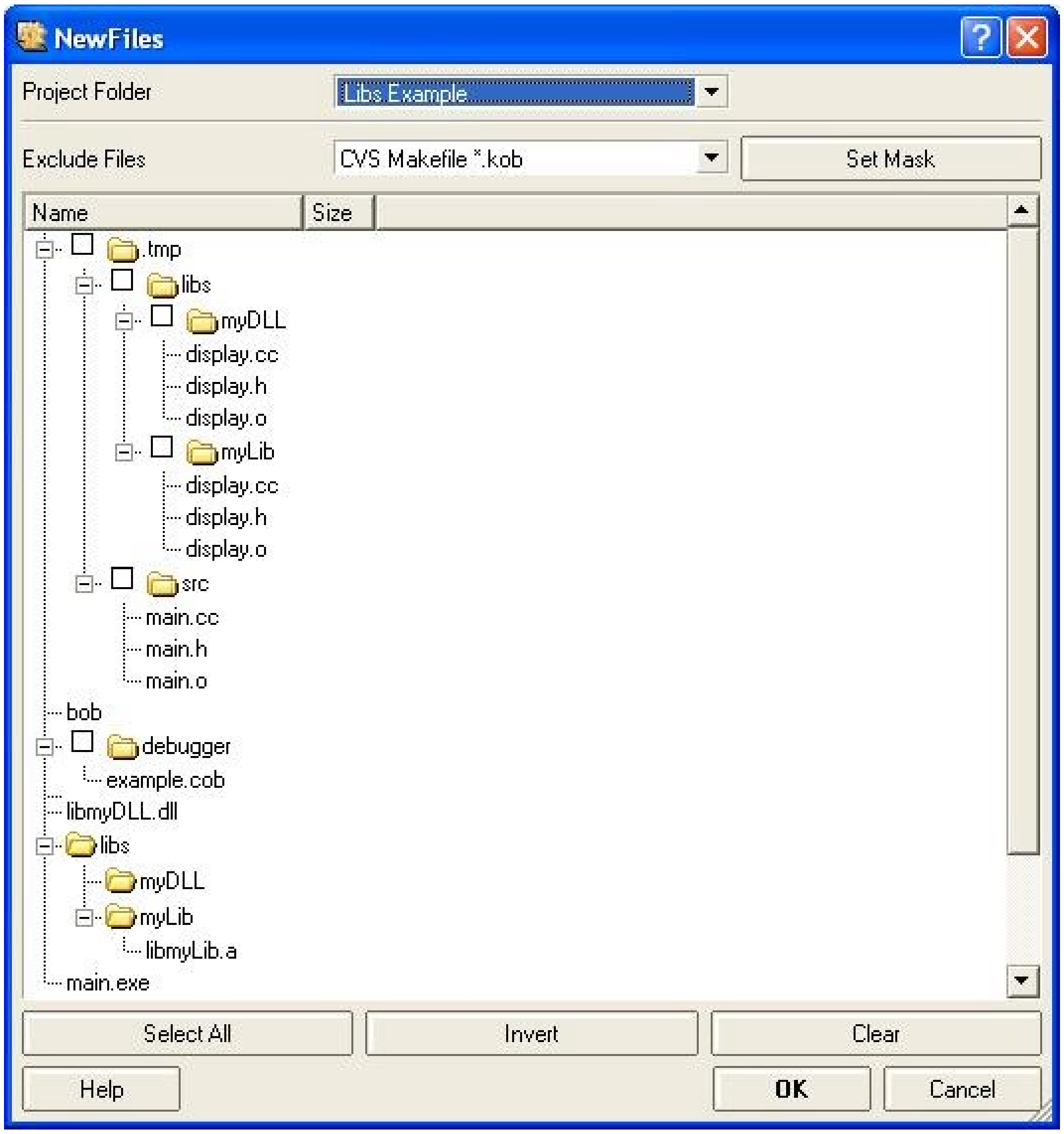
To add files to a project, you can right click on the projects folder in the "Project Panel" and select "Add files to folders". This will open up a dialog that allows you to select what files you want to add. If you select a different directory for the files to include, then they will be copied into your work directory. You can create a new folder inside a project by selecting the 'New Folder' option. Simply enter the folder name and click on 'Ok'. This will create the directory on your disk at the same time.

The next feature is available from the menu "Project->Add new files" (note that the option "Project->Add existing file..." is the same as the right click in the project tree). When you add a new file, Kobol will prompt you with a 'save file' dialog so it knows where to put the new file and what to call it.

A powerful companion feature is the menu item 'Project->Scan for new files'. When you select this you will get a checkable tree like file/directory structure which is not included in the current project. From here you can easily select missing files. In the dialog on top you can select the project's folder which you want to check for existing new files. Below you can enter a mask for files that you want to exclude. In the tree you can select files for addition by clicking once on files that you are interesting in. When you click on the checkbox you will have all files and subdirectories with files selected for addition. When you clear the checkbox on the folder you will clear all selections inside. So this feature allows you to very quickly navigate a project directory for new files and select them for addition.

In all cases a project needs a 'main' source file, this is the outer block. When you are creating a new source file in a new project for the first time, Kobol will try to set this one as main. A main file is indicated by a 'star' on it's icon in the project list view, sometimes this doesn't show up until you close and re-open Kobol if the setting has been done automatically, this is currently a bug. To specifically set a file as main, right click on its name in the file list and select the menu option for it. Here is a sequence of events that is popular with our customers:

1. Build the project and workspace,
2. Build the directory structure,
3. Copying the source files into the folder via Explorer,
4. Add the files to the folder via "Add files to folder" option
5. Save and close the workspace
6. Delete the bookmark on the project page.
7. Open the ".kob" file directly.



```
.tmp
  libs
    myDLL
      display.cc
      display.h
      display.o
    myLib
      display.cc
      display.h
      display.o
  src
    main.cc
    main.h
    main.o
bob
debugger
  example.cob
libmyDLL.dll
libs
  myDLL
  myLib
    libmyLib.a
main.exe
```

5. Additional Features

Kobol also has very basic CVS support which will continued to be enhanced in the future. Now you can update or commit changes of the entire project directory by selecting "Project->Update/Commit Project". This feature will save changes in your files and run the "cvs" command. The command will output all messages in the "Message window" and there you can see which files were modified or which files are updated from other developers. You can also update/commit for one file by selecting "Project->Update/Commit" or use "Project Panel" by right mouse clicking. Note that using cvs commands for a folder will updates all files and directories recursively inside the folder.

As previously mentioned, basic CVS support has been added to Kobol. Soon we will be adding features such as log browsing, merging and resolving conflicts among other features. You must have CVS installed on your Linux system to use it, on Windows you need to select "CVS support" in the installation wizard. To work with CVS in Kobol on a project you need to have the project uploaded by using the "cvs checkout" command from a console. After that you will have a project with directories CVS inside folders. Once you have done this, then CVS operations will be available in Kobol. CVS documentation will be added to the Kobol help system to keep it consistent with the other integrated "sub" systems.

The Editor that is used in Kobol is another product developed at theKompany by Max Judin and is used in a variety of our products. The editor has standard features for editing, powerful highlighting system which allow you easily navigate in your code, on the fly checking for, and repairing errors.

Word completion has also been implemented for speeding up your work even more. By selecting the menu item "Options->Fonts" you can modify the font used in the editor window. Monospace font is really more useful for editing, but you can select any font available in the 'font dialog'. By selecting the menu item "Options->Editor" you can choose your own settings to control the behavior of the editor. Other options available are:

Auto indent: Allows you to save the indentation used in the previous line.

Backspace Indent: Remove indent by pressing backspace.

Smart Home: With this feature when you press the "Home" button the cursor jumps to the first non' space symbol in the line and then with the second pressing the cursor jumps to the beginning of the line.

Wrap cursor: If you have a long line, the text will automatically go to the next line.

Replace tabs: Sometimes it is better to uses an actual space instead of a tab to indent code, this option will go through and replace the tabs with a specified number of spaces.

Remove Trailing Spacers: This will eliminate embedded spaces at the end of the document.

Margins: Indicator Margin: Not currently used, but a future release will be used to indicate code folding, bookmarks and more.

Selection Margin: Allows you to select a range or area more flexibly than would otherwise be possible. Word completion can be a big time saver when working on extensive document. It is very simple to use, while typing just press "Ctrl+Space" and you receive a popup menu that contains all word that you already have entered in the text that match the symbols currently entered. Just select what you want and press 'Enter'.

General Tips

Try to use "Select Assign Using" to avoid hard coding file names in your program. This allows you to specify input and output file names on the command line at execution time.

The "Accept From command-line" contains system data in addition to what is passed, make sure you parse it correctly.

Be sure to check your file-status after opening a file to see if it exists. If it does not your program runs on an empty dataset and never hits eof.

The concept of ‘Nested’ source is not supported in Kobol. This isn’t used often, and is typically misunderstood. Essentially you have multiple programs in one source file and they can share variables, you see multiple DIVISIONS for each sub program. If you try to use this concept, it will appear to compile but not ‘make’, but it actually isn’t compiling.

6. Copylib and $INCLUDE Files

COPY members are included from separate files. This means that a copylib that contains multiple members must be split into one file per member. These files can be placed in a directory that has the same name as the library. Members from the default copylib must be placed directly in one of the include directories.

Example:

Assume that have a copy library named MYLIB with members A, B and C. You can create a directory named MYLIB and within that directory you can create files A, B and C.

The COBOL statement

```
COPY A IN MYLIB
```

will then read the file MYLIB/A.

cob2c allows to indicate include directories (via the -I option, just like gcc). The copy libraries can be in any of these directories or in the current working directory, which is always searched first. $INCLUDE files are included from the same directories. Since COBOL names are not case sensitive,cob2c will look for file names both in uppercase and in lowercase.

7. Accessing Command Line Arguments

Here is a small example on how to access command-line arguments and environment variables from Kobol. This method can be extended to create CGI programs to access variables from a web page.

```
identification division.
program-id. test123.
environment division.
configuration section.
data division.
working-storage section.
 01 cmd pic x(80).
 01 arg-0 pic x(80).
 01 arg-1 pic x(80).
 01 arg-3 pic x(80).
 01 path pic x(256).
 01 my-var pic x(80).
 procedure division.
 A000 section.
 A0.
* get the full command line
 accept cmd from command-line.
 display cmd.

* get argument 0 from the command line (i.e. the program name)
 display 0 upon argument-number.
 accept arg-0 from argument-value.
 display arg-0.

* get the next argument from the command line
 accept arg-1 from argument-value.
 display arg-1.

* get argument 3 from the command line
 display 3 upon argument-number.
 accept arg-3 from argument-value.
 display arg-3.

* read the environment variable with name PATH
 display "PATH" upon environment-name.
 accept path from environment-value.
  display path.

* write and read an environment variable
 display "MYVAR" upon environment-name.
 display "my value" upon environment-value.
 accept my-var from environment-value.
 display my-var
```

8. Binary/Integer Data Types

A number of compiler vendors have implemented different versions of the COMPUTATIONAL type, notably there is COMP-1 through COMP-6 and COMP-X. In the new ANSI standard for COBOL these have all mostly been lumped together into BINARY and the compiler needs to know weather to use big or little endian (Intel chips are little endian for example). We currently have support for COMP, BINARY, COMP-3, PACKED-DECIMAL, and COMP-5.

At this moment, COMP, BINARY and COMP-5 are all the same native format. This means that they are little endian on Intel systems. Many other compilers define COMP and BINARY as big endian

COMP-3 and PACKED-DECIMAL are the same, which is to say they are 4 bits per digit. Our PACKED-DECIMAL format always has a sign nibble, which is in the samebyte as the low order digit. All PACKED-DECIMAL values are byte aligned.

We also support numeric DISPLAY format aka zoned decimal, this is for both
signed and unsigned, as well as with a leading or trailing separate sign.

9. Distributing Binaries

One of the many great things about Kobol is that you can distribute royalty free binaries of your application. This is very straight forward on Linux, just use the IDE to generate your executable and then hand it around. On Windows it can be slightly more confusing because of the use of Cygwin, and some support files are required. It will be up to you to properly package your application, but this will tell you what files are important.

These 2 dll's, along with your executable, are all you need to distribute a Kobol application:

DOS	Windows
CYGWIN1	cygwin1.dll
CYGNCU~5	cygncurses6.dll

We've listed how the file name will look under DOS and under Windows, you have to make sure that you don't alter these file names in any way or you will not be successful.

If you are having any trouble because of paths not finding critical files, then the following list is the critical files, they should all be in the cygwin directory that was installed, you can either move them to a directory and set your PATH or set your PATH to the directory where they are located.

** Important Files

DOS	Windows
LIBCOM~1	libcompsupp.a
COB2C EXE	cob2c.exe
G__~1 EXE	g++.exe
COBOL2~1 H	cobol2gcc.h
QT-MT311 DLL	QT-MT311.DLL
GCC EXE	gcc.exe
CYGICO~1 DLL	cygiconv-2.dll
CYGFORM5 DLL	cygform5.dll
CYGMENU5 DLL	cygmenu5.dll
CYGNCU~2 DLL	cygncurses++5.dll
CYGNCU~3 DLL	cygncurses5.dll
CYGPAN~1 DLL	cygpanel5.dll
CYGFORM6 DLL	cygform6.dll
CYGMENU6 DLL	cygmenu6.dll
CYGNCU~4 DLL	cygncurses++6.dll
CYGNCU~5 DLL	cygncurses6.dll
CYGPAN~2 DLL	cygpanel6.dll
CYGWIN1 DLL	cygwin1.dll
LIBCOM~2 DLL	libcompsupp.dll

Alternatively you can simply:

C:\Program Files\Kobol\cygwin\bin\*.dll to c:\Windows

Assuming that you’ve installed the cygwin files to that directory.

10. Working Without the IDE

Sometimes you have an editor or IDE that you are already happy with and don't want to use the one included with Kobol. The software is partioned so that you can access the compiler directly if you so choose. Remember, what Kobol does is use a highly sophisticated translator to create C++ code out of your COBOL code and then use GCC (GNU Compiler Collection) to compile it. This is what provides us such a great degree of flexibility in porting to other platforms and supporting Object Oriented features. You can use Kobol to generate intermediate C++ code on a Windows machine and then port it over to a Sun box and compile it there for example.

So there are some basic steps to follow, first generate the C++ code using our cob2c:

```
cob2c -M -i myprog.cob -D .
```
Typing cob2c with no parameters will provide a list of options for the application. You should probably use a script or batch file and pass 'myprog.cob' as a parameter for maximum flexbility. Here are some example steps:

1) Compile prog1
```
cob2c prog1.cob -D .
g++ -c -o prog1.o prog1.cc
```

2) Compile prog2
```
cob2c prog2.cob -D .
g++ -c -o prog2.o prog2.cc
```

3) Create a dll containing both programs
```
g++ -shared -o mydll.dll prog1.o prog2.o -L. -lcompsupp
```

4) Compile and link calling program
```
cob2c -M -i caller.cob -D .
g++ -c -o caller.o caller.cc
g++ -o caller caller.o -L. -lcompsupp -lncurses
```

A good reference book for the GNU system is:

GCC, The Complete Reference
by Arthur Griffith

11. Writing CGI Applications

CGI is the common abbreviation for applications that run in web servers to feed data back to a web page, they can also accept data from the web page using the GET or POST method, we are going to describe the GET method here. What we have to do is retrieve the contents of the system variable "QUERY_STRING" that will contain the name/data pairs that get submitted from the web page form. The URL will look something like:

http://gonzo.ukrweb.net/cgi-bin/test.cgi?name1=value1&name2=value2

Where the relevant part that gets posted to QUERY_STRING is

name1=value1&name2=value2

'name' is the name of the variable and 'value' is the actual contents of the variable.

Here is a short example that will get the contents of QUERY_STRING and then parse the results into a table:

```
 WORKING-STORAGE SECTION.
     01 M1                   PIC s9(4) COMP VALUE 0.
     01 M2                   PIC S9(4) COMP VALUE 0.
     01 INPUT-BUFFER         PIC X(80) VALUE SPACES.
     01 INPUT-TABLE.
        03 IPT               PIC X(30) OCCURS 10.
     01 NAME-TABLE.
        03 NT-ENTRY          PIC X(10) OCCURS 20.
     01 VALUE-TABLE.
        03 VT-ENTRY          PIC X(10) OCCURS 20.

 PROCEDURE DIVISION.
* indicate what environment variable you want to access
     display "QUERY_STRING" upon environment-name.
* get the value for the indicated environment variable
     accept INPUT-BUFFER from environment-value.

     MOVE SPACES                          TO NAME-TABLE
                                             VALUE-TABLE
                                             INPUT-TABLE.
* assume INPUT-BUFFER looks like
* name1=value1&name2=value2&name3=value3&name4=value4

     INSPECT INPUT-BUFFER TALLYING M1 FOR ALL '&'.
     ADD 1 TO M1.
     UNSTRING INPUT-BUFFER DELIMITED BY "&" INTO
```

```
         IPT(1) IPT(2) IPT(3) IPT(4).
PERFORM VARYING M2 FROM 1 BY 1 UNTIL M2 > M1
        UNSTRING IPT(M2) DELIMITED BY '=' INTO
                 NT-ENTRY(M2) VT-ENTRY(M2)
        DISPLAY 'Name Token : ' NT-ENTRY(M2)
        DISPLAY 'Value Token: ' VT-ENTRY(M2)
END-PERFORM.
```

12. Calling non-COBOL sub-programs

Calling non-cobol sub-programs.

The CALL statement is used to call a sub-program, whether it is written in COBOL or not.

By default, the Kobol compiler will assume that a called sub-program is written in COBOL, and generate code according to this assumption. When calling a non-cobol sub-program, the calling sequence may not be compatible with what Kobol assumes. The programmer must inform the compiler about the correct calling sequence.

The calling sequence for a sub-program is expressed as a prototype.

The prototype has the following format:

```
name (parameter_type, parameter_type, ...)
```

The name is the external function name, with appropriate capitalization.

Each parameter_type is a C-style parameter type declaration. This type declaration shall not contain any parentheses or commas itself. The type declaration shall not contain a parameter name (this implies that arrays should be expressed as pointers, and that pointers to functions are not supported).

The Kobol compiler will not verify the parameter types for correctness.

Some examples of prototypes for well know functions:

```
int system (const char *)
void *memcpy (void *, const void *, size_t)
```

Specifying a prototype that is not compatible with the actual parameters of the called sub-program will result in undefined behaviour. In the best case, an error will be issued during the compilation of the generated intermediate code:

```
declaration of C function `foo' conflicts with previous declaration
```

The prototype for a called subprogram can be specified in the AS clause of a CALL statement. This requires that every call statement must specify the prototype. This is a good solution when one or two CALL statements must be modified.

A better way to specify prototypes is in a prototype file. This file contains prototypes for external functions. Each line of the file contains exactly one prototype. The name of the prototype file must be provided using the '-c' option when invoking the compiler (see Compiler options.)

The format for a prototype declaration in the prototype file is:

```
internal_name = prototype
```

The internal name is the name as it is specified in a CALL statement. Every CALL statement that references the given sub-program will behave as if the prototype had been mentioned in an AS clause for that statement.
Important notes.

Unlike in C or C++, COBOL does not put a terminating zero in any variable. Therefore, the programmer must provide this terminating zero when the called sub-program expects it. The STRING statement in the following examples does just that.

One of the examples demonstrates a call to 'memcpy'. The same effect will result from a simple MOVE statement. When there is such a COBOL alternative for calling a sub-program, then usually the COBOL statement should be preferred.
Examples.

An example using the AS clause:

```
IDENTIFICATION DIVISION.
PROGRAM-ID. SYS.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
 01 COMMAND PIC X(80).
PROCEDURE DIVISION.
 BEGIN.
     STRING "/bin/ls" x"0" INTO COMMAND.
     CALL "system" AS "int system (const char *)"
               USING COMMAND.

     STOP RUN.
```

An example using a prototype file. The prototype file is:

```
system =int system (const char *)
memcpy=void *memcpy(void *, const void *, size_t)
```

The program is:

```
IDENTIFICATION DIVISION.
PROGRAM-ID. SYS.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
 01 COMMAND PIC X(80).
 01 COMMAND1 PIC X(80).
PROCEDURE DIVISION.
BEGIN.
     STRING "/bin/ls" x"0" INTO COMMAND.
     CALL "system" USING COMMAND.

     CALL "memcpy" USING COMMAND1, COMMAND,
                     BY VALUE LENGTH OF COMMAND.
* the CALL "memcpy" is equivalent to
*      MOVE COMMAND TO COMMAND1
* the MOVE statement is usualy preferable.

     CALL "system" USING COMMAND1.

     STOP RUN.
```

13. SELECT..ASSIGN

Indexed files are actually kept as a pair of files, one with the '.data' extension and one with the '.index' extension. The name given in the ASSIGN TO clause must be the name without these extensions.

ORGANIZATION SEQUENTIAL files are byte streams that have no embedded structure. Such a file can contain binary or ASCII data. No record delimiters are added at all. The file simply consists of a set of records, all with the same length, and this length is defined by the program that accesses the file.

ORGANIZATION LINE SEQUENTIAL is also supported, and uses new-line characters as record separators.

14. FILE-STATUS Codes

A program compiled using the Kobol compiler can get the following FILE-STATUS codes.

00 Succesfull completion.

05 Optional file not present.
An OPEN statement is successfully executed but the file is described as optional and the physical file is not present at the time the OPEN statement is executed. If the open mode is I-O or extend, the physical file has been created.

10 At end.
A sequential read is attempted and no record is available.
NEXT was specified or implied and the end of the file has been reached or a sequential READ statement is attempted for the first time on a file described as OPTIONAL and the file is not present.

22 Duplicate key.
A WRITE or REWRITE is attempted that would create a duplicate key, where no such duplicate is allowed.

23 Invalid key.
A START or READ statement attempts to access a record, identified by a key, and the record is not present.

24 Boundary violation.
An attempt is made to write beyond the externally-defined boundaries of a physical relative or indexed file.

34 Boundary violation.
An attempt is made to write beyond the externally-defined boundaries of a physical sequential file.

35 File not found.
An OPEN statement with the INPUT, I-O, or EXTEND phrase is attempted on a file that is not described as optional and the physical file is not present.

37 Invalid open mode.
An OPEN statement is attempted on a file and that file will not support the open mode specified in the OPEN statement.

41 File already open.
An OPEN statement is attempted for a file that is already open.

42 File not open.
A CLOSE statement is attempted for a file that is not opened.

43 No valid record.
A DELETE or REWRITE statement on a file in sequential access mode was not preceded by a successful READ statement.

46 No valid record.
A sequential READ statement is attempted on a file open in INPUT or I-O mode and no valid next record has been established because the preceding START or READ statement referencing that file was unsuccessful.

47 File not open for input.
The execution of a READ or START statement is attempted on a file that is not open in INPUT or I-O mode.

48 File not open for output.
The execution of a WRITE statement is attempted on a file that is not open in I-O, EXTEND or OUTPUT mode.

49 File not open for I-O.
The execution of a DELETE or REWRITE statement is attempted on a file that is not open in I-O mode.

15. Plug Ins

Over time we will release additional functionality to Kobol via optional plug ins. Currently we have two of these, they are the “debugger” that provides for a source debugger interface, the second is to provide for “MPE/HP3000” support. What the latter one does is provide support for the HP extensions to the COBOL language such as macros, pseudo-intrinsics and system intrinsics. Please note for the system intrinsics we do not provide an intrinsic emulation library, just the stubs so it can work with a 3rd party library.

When you install a plug in, it will either add the functionality to the IDE if appropriate, or open up the feature to the compiler so that it can be used. The installation goes very quickly and will seem as though almost nothing happened. In some cases with the HP3000 plug in you might have to force the plug in directory, which can be done using the -p switch to cob2c and giving the specific path where the plug in got installed.

16. Support

In general we will do our best to support you free of charge, should your needs go beyond our freely available services then send to support@thekompany.com

Bugs take priority over features and we will address them as quickly as possible. We have a mail list for Kobol customers set up. If you purchased from our web site you would have received information on it already, if you purchased from another source then you will need to register your product so that you can join the mail list. The mail list is called kobol@thekompany.com

Our web site has product information available as well if this document doesn't answer your questions. Please see www.thekompany.com/products/kobol. You can also send an email to info@thekompany.com. Kobol is available from many online and brick and mortar retailers, but it can also be purchased directly from our website at www.thekompany.com/products/kobol.

17. Credits

Kobol IDE: Alexander Yakovlev
cob2gcc: Jacques V Damme
cobmake: Alexander Yakovlev & Dmitry Poplavsky
tkEditor used in Kobol by: Max Judin
Icons, graphics, splash screen and packaging by: Bogdan Munteanu & John Grantham

Appendix A (Reserved Words)

Note that the second column contains a '1' if the corresponding reserved word can start a new sentence. The third column says in what dialect the word is reserved.

```
  { "ABSENT",                   0, diaCobol2000 },
  { "ACCEPT",                   1, diaAll },
  { "ACCESS",                   0, diaAll },
  { "ACTIVE-CLASS",             0, diaCobol2000 },
  { "ACTUAL",                   0, diaCobol3000 },
  { "ADD",                      1, diaAll },
  { "ADDRESS",                  0, diaCobol2000 },
  { "ADVANCING",                0, diaAll },
  { "AFTER",                    0, diaAll },
  { "ALL",                      0, diaAll },
  { "ALLOCATE",                 0, diaCobol2000 },
  { "ALLOW",                    1, diaCobol2000 },
  { "ALPHABET",                 0, diaAll },
  { "ALPHABETIC",               0, diaAll },
  { "ALPHABETIC-LOWER",         0, diaAll },
  { "ALPHABETIC-UPPER",         0, diaAll },
  { "ALPHANUMERIC",             0, diaAll },
  { "ALPHANUMERIC-EDITED",      0, diaAll },
  { "ALSO",                     0, diaAll },
  { "ALTER",                    1, diaAll },
  { "ALTERNATE",                0, diaAll },
  { "AND",                      0, diaAll },
  { "ANY",                      0, diaAll },
  { "ARE",                      0, diaAll },
  { "AREA",                     0, diaAll },
  { "AREAS",                    0, diaAll },
  { "AS",                       0, diaCobol2000 },
  { "ASCENDING",                0, diaAll },
  { "ASSIGN",                   0, diaAll },
  { "AT",                       0, diaAll },
  { "B-AND",                    0, diaCobol2000 },
  { "B-NOT",                    0, diaCobol2000 },
  { "B-OR",                     0, diaCobol2000 },
  { "B-XOR",                    0, diaCobol2000 },
  { "BEFORE",                   0, diaAll },
  { "BEGINNING",                0, diaCobol3000 },
  { "BINARY",                   0, diaAll },
  { "BINARY-CHAR",              0, diaCobol2000 },
  { "BINARY-DOUBLE",            0, diaCobol2000 },
  { "BINARY-LONG",              0, diaCobol2000 },
  { "BINARY-SHORT",             0, diaCobol2000 },
  { "BIT",                      0, diaCobol2000 },
  { "BLANK",                    0, diaAll },
  { "BLOCK",                    0, diaAll },
  { "BOOLEAN",                  0, diaCobol2000 },
  { "BOTTOM",                   0, diaAll },
  { "BY",                       0, diaAll },
  { "CALL",                     1, diaAll },
  { "CALL-CONVENTION",          0, diaCobol2000 },
  { "CANCEL",                   1, diaAll },
  { "CD",                       0, diaAll },
  { "CF",                       0, diaAll },
  { "CH",                       0, diaAll },
  { "CHARACTER",                0, diaAll },
  { "CHARACTERS",               0, diaAll },
  { "CLASS",                    0, diaAll },
  { "CLASS-ID",                 0, diaCobol2000 },
  { "CLOSE",                    1, diaAll },
  { "CODE",                     0, diaAll },
  { "CODE-SET",                 0, diaAll },
  { "COL",                      0, diaCobol2000 },
  { "COLLATING",                0, diaAll },
  { "COLS",                     0, diaCobol2000 },
  { "COLUMN",                   0, diaAll },
  { "COLUMNS",                  0, diaCobol2000 },
  { "COMMA",                    0, diaAll },
  { "COMMON",                   0, diaAll },
  { "COMMUNICATION",            0, diaAll },
  { "COMP",                     0, diaAll },
  { "COMPUTATIONAL",            0, diaAll },
  { "COMPUTE",                  1, diaAll },
  { "CONFIGURATION",            0, diaAll },
  { "CONSTANT",                 0, diaCobol2000 },
  { "CONTAINS",                 0, diaAll },
  { "CONTENT",                  0, diaAll },
  { "CONTINUE",                 1, diaAll },
  { "CONTROL",                  1, diaAll },
  { "CONTROLS",                 0, diaAll },
  { "CONVERTING",               0, diaAll },
  { "COPY",                     0, diaAll },
  { "CORR",                     0, diaAll },
  { "CORRESPONDING",            0, diaAll },
  { "COUNT",                    0, diaAll },
  { "CRT",                      0, diaCobol2000 },
  { "CURRENCY",                 0, diaAll },
  { "CURRENT-DATE",             0, diaCobol3000 },
  { "CURSOR",                   0, diaCobol2000 },
  { "DATA",                     0, diaAll },
  { "DATE",                     0, diaAll },
  { "DAY",                      0, diaAll },
  { "DAY-OF-WEEK",              0, diaAll },
  { "DE",                       0, diaAll },
  { "DEBUGGING",                0, diaAll },
  { "DECIMAL-POINT",            0, diaAll },
  { "DECLARATIVES",             0, diaAll },
  { "DEFAULT",                  0, diaCobol2000 },
  { "DELETE",                   1, diaAll },
  { "DELIMITED",                0, diaAll },
  { "DELIMITER",                0, diaAll },
  { "DEPENDING",                0, diaAll },
  { "DESCENDING",               0, diaAll },
  { "DESTINATION",              0, diaAll },
  { "DETAIL",                   0, diaAll },
  { "DISABLE",                  1, diaAll },
  { "DISPLAY",                  1, diaAll },
  { "DIVIDE",                   1, diaAll },
  { "DIVISION",                 0, diaAll },
  { "DOWN",                     0, diaAll },
  { "DUPLICATES",               0, diaAll },
  { "DYNAMIC",                  0, diaAll },
  { "EGI",                      0, diaAll },
  { "ELSE",                     0, diaAll },
  { "EMI",                      0, diaAll },
  { "ENABLE",                   1, diaAll },
  { "END",                      0, diaAll },
  { "END-ACCEPT",               0, diaAll },
  { "END-ADD",                  0, diaAll },
  { "END-CALL",                 0, diaAll },
  { "END-COMPUTE",              0, diaAll },
  { "END-DELETE",              0, diaAll },
  { "END-DISPLAY",             0, diaAll },
  { "END-DIVIDE",              0, diaAll },
  { "END-EVALUATE",            0, diaAll },
  { "END-IF",                  0, diaAll },
  { "END-MULTIPLY",            0, diaAll },
  { "END-OF-PAGE",             0, diaAll },
  { "END-PERFORM",             0, diaAll },
  { "END-READ",                0, diaAll },
  { "END-RECEIVE",             0, diaAll },
  { "END-RETURN",              0, diaAll },
  { "END-REWRITE",             0, diaAll },
  { "END-SEARCH",              0, diaAll },
  { "END-START",               0, diaAll },
  { "END-STRING",              0, diaAll },
  { "END-SUBTRACT",            0, diaAll },
  { "END-UNSTRING",            0, diaAll },
  { "END-WRITE",               0, diaAll },
  { "ENDING",                  0, diaCobol3000 },
  { "ENTRY",                   1, diaCobol3000 },
  { "ENVIRONMENT",             0, diaAll },
  { "EOP",                     0, diaAll },
  { "EQUAL",                   0, diaAll },
  { "ERROR",                   0, diaAll },
  { "ESI",                     0, diaAll },
  { "EVALUATE",                1, diaAll },
  { "EXCEPTION",               0, diaAll },
  { "EXCEPTION-OBJECT",        0, diaCobol2000 },
  { "EXCLUSIVE",               0, diaCobol3000 },
  { "EXIT",                    1, diaAll },
  { "EXTEND",                  0, diaAll },
  { "EXTERNAL",                0, diaAll },
  { "FACTORY",                 0, diaAll },
  { "FALSE",                   0, diaAll },
  { "FD",                      0, diaAll },
  { "FILE",                    0, diaAll },
  { "FILE-CONTROL",            0, diaAll },
  { "FILE-LIMIT",              0, diaCobol3000 },
  { "FILE-LIMITS",             0, diaCobol3000 },
  { "FILLER",                  0, diaAll },
  { "FINAL",                   0, diaAll },
  { "FIRST",                   0, diaAll },
  { "FLOAT-EXTENDED",          0, diaCobol2000 },
  { "FLOAT-LONG",              0, diaCobol2000 },
  { "FLOAT-SHORT",             0, diaCobol2000 },
  { "FOOTING",                 0, diaAll },
  { "FOR",                     0, diaAll },
  { "FORMAT",                  0, diaAll },
  { "FREE",                    1, diaCobol3000 | diaCobol2000 },
  { "FROM",                    0, diaAll },
  { "FUNCTION",                0, diaAll },
  { "FUNCTION-ID",             0, diaCobol2000 },
  { "GENERATE",                1, diaAll },
  { "GET",                     0, diaCobol2000 },
  { "GIVING",                  0, diaAll },
  { "GLOBAL",                  0, diaAll },
  { "GO",                      1, diaAll },
  { "GOBACK",                  1, diaAll },
  { "GREATER",                 0, diaAll },
  { "GROUP",                   0, diaAll },
  { "HEADING",                 0, diaAll },
  { "HIGH-VALUE",              0, diaAll },
  { "HIGH-VALUES",             0, diaAll },
  { "I-O",                      0, diaAll },
  { "I-O-CONTROL",              0, diaAll },
  { "IDENTIFICATION",           0, diaAll },
  { "IF",                       1, diaAll },
  { "IN",                       0, diaAll },
  { "INDEX",                    0, diaAll },
  { "INDEXED",                  0, diaAll },
  { "INDICATE",                 0, diaAll },
  { "INHERITS",                 0, diaCobol2000 },
  { "INITIAL",                  0, diaAll },
  { "INITIALIZE",               1, diaAll },
  { "INITIATE",                 1, diaAll },
  { "INPUT",                    0, diaAll },
  { "INPUT-OUTPUT",             0, diaAll },
  { "INSPECT",                  1, diaAll },
  { "INTEGER",                  0, diaCobol2000 },
  { "INTERFACE",                0, diaCobol2000 },
  { "INTERFACE-ID",             0, diaCobol2000 },
  { "INTO",                     0, diaAll },
  { "INTRINSIC",                0, diaCobol3000 },
  { "INVALID",                  0, diaAll },
  { "INVOKE",                   0, diaCobol2000 },
  { "IS",                       0, diaAll },
  { "JUST",                     0, diaAll },
  { "JUSTIFIED",                0, diaAll },
  { "KEY",                      0, diaAll },
  { "LABEL",                    0, diaCobol85 | diaCobol3000 },
  { "LAST",                     0, diaAll },
  { "LEADING",                  0, diaAll },
  { "LEFT",                     0, diaAll },
  { "LENGTH",                   0, diaAll },
  { "LESS",                     0, diaAll },
  { "LIMIT",                    0, diaAll },
  { "LIMITS",                   0, diaAll },
  { "LINAGE",                   0, diaAll },
  { "LINAGE-COUNTER",           0, diaAll },
  { "LINE",                     0, diaAll },
  { "LINE-COUNTER",             0, diaAll },
  { "LINES",                    0, diaAll },
  { "LINKAGE",                  0, diaAll },
  { "LOCAL-STORAGE",            0, diaCobol2000 },
  { "LOCK",                     1, diaAll },
  { "LOW-VALUE",                0, diaAll },
  { "LOW-VALUES",               0, diaAll },
  { "MERGE",                    1, diaAll },
  { "MESSAGE",                  0, diaAll },
  { "METHOD",                   0, diaCobol2000 },
  { "METHOD-ID",                0, diaCobol2000 },
  { "MODE",                     0, diaAll },
  { "MORE-LABELS",              0, diaCobol3000 },
  { "MOVE",                     1, diaAll },
  { "MULTIPLY",                 1, diaAll },
  { "NATIONAL",                 0, diaCobol2000 },
  { "NATIONAL-EDITED",          0, diaCobol2000 },
  { "NATIVE",                   0, diaAll },
  { "NEGATIVE",                 0, diaAll },
  { "NESTED",                   0, diaCobol2000 },
  { "NEXT",                     1, diaAll },
  { "NO",                       0, diaAll },
  { "NOLIST",                   0, diaCobol3000 },
  { "NOT",                      0, diaAll },
  { "NULL",                     0, diaCobol2000 },
  { "NUMBER",                   0, diaAll },
  { "NUMERIC",                  0, diaAll },
  { "NUMERIC-EDITED",           0, diaAll },
  { "OBJECT",                   0, diaCobol2000 },
  { "OBJECT-COMPUTER",          0, diaAll },
  { "OCCURS",                   0, diaAll },
  { "OF",                       0, diaAll },
  { "OFF",                      0, diaAll },
  { "OMITTED",                  0, diaAll },
  { "ON",                       1, diaAll },
  { "OPEN",                     1, diaAll },
  { "OPTIONAL",                 0, diaAll },
  { "OPTIONS",                  0, diaCobol2000 },
  { "OR",                       0, diaAll },
  { "ORDER",                    0, diaAll },
  { "ORGANIZATION",             0, diaAll },
  { "OTHER",                    0, diaAll },
  { "OUTPUT",                   0, diaAll },
  { "OVERFLOW",                 0, diaAll },
  { "OVERRIDE",                 0, diaCobol2000 },
  { "PACKED-DECIMAL",           0, diaAll },
  { "PADDING",                  0, diaAll },
  { "PAGE",                     0, diaAll },
  { "PAGE-COUNTER",             0, diaAll },
  { "PERFORM",                  1, diaAll },
  { "PF",                       0, diaAll },
  { "PH",                       0, diaAll },
  { "PIC",                      0, diaAll },
  { "PICTURE",                  0, diaAll },
  { "PLUS",                     0, diaAll },
  { "POINTER",                  0, diaAll },
  { "POSITION",                 0, diaAll },
  { "POSITIVE",                 0, diaAll },
  { "PRESENT",                  0, diaCobol2000 },
  { "PRINTING",                 0, diaAll },
  { "PROCEDURE",                0, diaAll },
  { "PROCEDURES",               0, diaAll },
  { "PROCESSING",               0, diaCobol3000 },
  { "PROGRAM",                  0, diaAll },
  { "PROGRAM-ID",               0, diaAll },
  { "PROGRAM-POINTER",          0, diaCobol2000 },
  { "PROPERTY",                 0, diaCobol2000 },
  { "PROTOTYPE",                0, diaCobol2000 },
  { "PURGE",                    1, diaAll },
  { "QUEUE",                    0, diaAll },
  { "QUOTE",                    0, diaAll },
  { "QUOTES",                   0, diaAll },
  { "RAISE",                    0, diaCobol2000 },
  { "RAISING",                  0, diaCobol2000 },
  { "RANDOM",                   0, diaAll },
  { "RD",                       0, diaAll },
  { "READ",                     1, diaAll },
  { "RECEIVE",                  1, diaAll },
  { "RECORD",                   0, diaAll },
  { "RECORDING",                0, diaCobol3000 },
  { "RECORDS",                  0, diaAll },
  { "REDEFINES",                0, diaAll },
  { "REEL",                     0, diaAll },
  { "REFERENCE",                0, diaAll },
  { "REFERENCES",               0, diaAll },
  { "RELATIVE",                 0, diaAll },
  { "RELEASE",                  1, diaAll },
  { "REMAINDER",                0, diaAll },
  { "REMOVAL",                  0, diaAll },
  { "RENAMES",                  0, diaAll },
  { "REPLACE",                  0, diaAll },
  { "REPLACING",                0, diaAll },
  { "REPORT",                   0, diaAll },
  { "REPORTING",                0, diaAll },
  { "REPORTS",                  0, diaAll },
  { "REPOSITORY",               0, diaCobol2000 },
  { "RESERVE",                  0, diaAll },
  { "RESET",                    1, diaCobol2000 },
  { "RESUME",                   0, diaCobol2000 },
  { "RETRY",                    0, diaCobol2000 },
  { "RETURN",                   1, diaAll },
  { "RETURNING",                0, diaCobol2000 },
  { "REWIND",                   0, diaAll },
  { "REWRITE",                  1, diaAll },
  { "RF",                       0, diaAll },
  { "RH",                       0, diaAll },
  { "RIGHT",                    0, diaAll },
  { "ROUNDED",                  0, diaAll },
  { "RUN",                      1, diaAll },
  { "SAME",                     0, diaAll },
  { "SCREEN",                   0, diaCobol2000 },
  { "SD",                       0, diaAll },
  { "SEARCH",                   1, diaAll },
  { "SECTION",                  0, diaAll },
  { "SEEK",                     0, diaCobol3000 },
  { "SEGMENT",                  0, diaAll },
  { "SELECT",                   0, diaAll },
  { "SELF",                     0, diaCobol2000 },
  { "SEND",                     1, diaAll },
  { "SENTENCE",                 0, diaAll },
  { "SEPARATE",                 0, diaAll },
  { "SEQUENCE",                 0, diaAll },
  { "SEQUENTIAL",               0, diaAll },
  { "SET",                      1, diaAll },
  { "SHARING",                  0, diaCobol2000 },
  { "SIGN",                     0, diaAll },
  { "SIZE",                     0, diaAll },
  { "SORT",                     1, diaAll },
  { "SORT-MERGE",               0, diaAll },
  { "SOURCE",                   0, diaAll },
  { "SOURCE-COMPUTER",          0, diaAll },
  { "SOURCES",                  0, diaCobol2000 },
  { "SPACE",                    0, diaAll },
  { "SPACES",                   0, diaAll },
  { "SPECIAL-NAMES",            0, diaAll },
  { "STANDARD",                 0, diaAll },
  { "STANDARD-1",               0, diaAll },
  { "STANDARD-2",               0, diaAll },
  { "STANDARD-3",               0, diaCobol2000 },
  { "START",                    1, diaAll },
  { "STATUS",                   0, diaAll },
  { "STOP",                     1, diaAll },
  { "STRING",                   1, diaAll },
  { "SUB-QUEUE-1",              0, diaAll },
  { "SUB-QUEUE-2",              0, diaAll },
  { "SUB-QUEUE-3",              0, diaAll },
  { "SUBTRACT",                 1, diaAll },
  { "SUM",                      0, diaAll },
  { "SUPER",                    0, diaCobol2000 },
  { "SUPPRESS",                 1, diaAll },
  { "SYMBOLIC",                 0, diaAll },
  { "SYNC",                     0, diaAll },
  { "SYNCHRONIZED",              0, diaAll },
  { "SYSTEM-DEFAULT",            0, diaCobol2000 },
  { "TABLE",                     0, diaCobol2000 },
  { "TALLY",                     0, diaCobol3000 },
  { "TALLYING",                  0, diaAll },
  { "TERMINAL",                  0, diaAll },
  { "TERMINATE",                 1, diaAll },
  { "TEST",                      0, diaAll },
  { "TEXT",                      0, diaAll },
  { "THAN",                      0, diaAll },
  { "THEN",                      0, diaAll },
  { "THROUGH",                   0, diaAll },
  { "THRU",                      0, diaAll },
  { "TIME",                      0, diaAll },
  { "TIME-OF-DAY",               0, diaCobol3000 },
  { "TIMES",                     0, diaAll },
  { "TO",                        0, diaAll },
  { "TOP",                       0, diaAll },
  { "TRAILING",                  0, diaAll },
  { "TRUE",                      0, diaAll },
  { "TYPE",                      0, diaAll },
  { "TYPEDEF",                   0, diaCobol2000 },
  { "UN-EXCLUSIVE",              0, diaCobol3000 },
  { "UNIT",                      0, diaAll },
  { "UNIVERSAL",                 0, diaCobol2000 },
  { "UNLOCK",                    1, diaCobol2000 },
  { "UNSTRING",                  1, diaAll },
  { "UNTIL",                     0, diaAll },
  { "UP",                        0, diaAll },
  { "UPON",                      0, diaAll },
  { "USAGE",                     0, diaAll },
  { "USE",                       1, diaAll },
  { "USER-DEFAULT",              0, diaCobol2000 },
  { "USING",                     0, diaAll },
  { "VALID",                     0, diaCobol2000 },
  { "VALIDATE",                  0, diaCobol2000 },
  { "VALUE",                     0, diaAll },
  { "VALUES",                    0, diaAll },
  { "VARYING",                   0, diaAll },
  { "WHEN",                      0, diaAll },
  { "WHEN-COMPILED",             0, diaCobol3000 },
  { "WITH",                      0, diaAll },
  { "WORKING-STORAGE",           0, diaAll },
  { "WRITE",                     1, diaAll },
  { "ZERO",                      0, diaAll },
  { "ZEROES",                    0, diaAll },
  { "ZEROS",                     0, diaAll },
```