

The GnuCOBOL Interactive Compiler (GCic)

September, 2022 - GCic V2.0 RC2

This document serves as both a user's and programmer's guide for the GnuCOBOL Interactive Compiler (GCic) program. It is intended for those that wish to use, customize, support, and/or enhance that program.

Contents

[Copyright](#)

[Introduction](#)

[Installing GCic](#)

[Using GCic](#)

[The GCic Screen](#)

[Compiling Programs](#)

[Program-Specified Switches](#)

[Screen Features and Options](#)

[The GCic-Generated Source Listing](#)

[The GCic-Generated Cross-Reference Listing](#)

[Customizing GCic](#)

[Screen Automatic Resizing](#)

[The 'gcic-setup.cpy' Proc](#)

[Internals](#)

[The MAIN Program](#)

[The "Button-Tbl" Table](#)

[The "Function-Tbl" Table](#)

[The "Option-Tbl" Table](#)

[The "Temp-Files" Table](#)

[Program-Specified Switches \(PSS\) Internals](#)

[Locking](#)

[The GCINFO Subprogram](#)

[The LISTER Subprogram](#)

[The "All-Verbs-Tbl" Table](#)

[The "Buzzwords-Tbl" Table](#)

[The "Files-And-Statuses-Tbl" Table](#)

[The "Records-And-Files-Tbl" Table](#)

[The "Reserved-Word-Tbl" Table](#)

[The "Stack-Entry" Table](#)

[The "Symbol-Table" Table](#)

[Modifying GCic](#)

[Testing GCic](#)

[Summary of Documentation Changes](#)

[Summary of Software Changes](#)

[Thank You](#)

Copyright

This document and the source code for the GCic program is copyright © 2022 and may not be supplied or any part thereof, embedded or as a separate element, in any program language or form, with any COBOL compiler other than GnuCOBOL ® without the express authority of the copyright holder.

It's usage in conjunction with the COBOL compiler known as GnuCOBOL is herewith granted to all users of said product unconditionally.

Documentation Copyright © 2022 Gary L. Cutler

Permission is granted to copy and distribute this document under the terms of the GNU Free Documentation License, version 1.3 or later published by the Free Software Foundation with no Invariant Sections, Front-Cover Texts, and no Back-Cover Texts other than those originally supplied. A copy of the license is available at <https://www.gnu.org/licenses/fdl-1.3.en.html>.

[TOP](#)

Introduction

The GnuCOBOL Interactive Compiler, or GCic, program provides an interactive Textual User Interface (TUI) to the process of compiling and (optionally) executing a GnuCOBOL program. IT DOES NOT REPLACE THE COBC COMPILER THAT IS PART OF GNUCOBOL, but rather provides an easy-to-use front-end to it.

GCic has been designed to be compiled and run from the following environments:

Windows/MinGW

Windows/Cygwin

These Unix-emulation environments are the typical choice for installing GnuCOBOL on Windows, with MinGW far more common than Cygwin.

Native Windows

It is possible to install GnuCOBOL using native Windows components, and GCic can be compiled for such an environment.

MACOS

GCic has been successfully run on MacOS (formerly OSX) systems all the way back to "Mountain Lion".

UNIX/LINUX (hereafter referred to as "*NIX")

These environments are also suitable to host GCic.

GCic uses some specific features of GnuCOBOL 3.1 and may not compile or function optimally on releases of GnuCOBOL prior to 3.1 - it will not run on any of the older OpenCOBOL versions.

[TOP](#)

Installing GCic

These instructions assume that GnuCOBOL 3.1 (or greater) is installed and is fully operational on your computer. To install GCic on your system:

1. Copy the three files that make up this package...

- `gcic.cbl`
- `gcic-setup.cpy`
- `gcic-readme.html` (you're reading this now)

...to the folder of your choice on your computer.

2. Next, open a `cmd.exe` (or `console` or `terminal`, whichever is appropriate for your computer's OS) window on your computer and `cd` to the folder into which you copied the above three files.
3. Compile GCic using the following command:

```
cobc -x gcic.cbl
```

4. Now execute GCic, with `gcic.cbl` as the one and only argument.

One of three things will happen:

- A. You will receive a screen that looks very much like the one shown [here](#). If that is the case, GCic has recognized your operating system environment and has adjusted to it. It also means that the console session in which GCic is running either had the necessary 35 (or more) lines and 106 (or more) columns to begin with or GCic was able to reconfigure the screen geometry. Click the **CANCEL** (or press the Esc key) to quit GCic, then proceed to step 5.
- B. You received an otherwise blank screen with the following message on it:

**CANNOT DETERMINE OS
Set 'OS' in 'gcic-setup.cpy'
and recompile GCic**

GCic was unable to recognize your OS automatically. Just follow the procedure it asks for on the screen and repeat step #4.

- C. You received an otherwise blank screen with the following message on it:

**SCREEN GEOMETRY MUST BE SET
to 35 lines and 106 columns**

GCic was unable to resize your console window automatically. Increase the window size until it reaches the requested dimensions (at a minimum), and repeat step #4.

- 5. Copy the generated executable and [gcic-readme.html](#) file to ANY folder that exists in your system's PATH. As a recommendation, copy them to the same folder the GnuCOBOL compiler executable resides in.

At your liesure, review the [gcic-setup.cpy](#) proc, along with the [Customizing GCic](#) instructions in this document.

[TOP](#)

Using GCic

GCic is normally executed (from the command-line) as follows:

gcic program-filename

The argument is the name of the file containing the GnuCOBOL program(s) to be compiled, including any PATH information needed to locate the file. That is the only file you may specify on the command-line and there are no switches to specify, unless [debugging features](#) have been enabled. You will have the chance to enter additional files to be compiled or linked on the screen GCic displays.

Once executed, a screen will be presented showing the compilation options that will be used. The following section shows the layout of the 106 column by 35 row GCic screen.

[TOP](#)

The GCic Screen

The sample screen below shows how the screen looks if the [LINEDRAW](#) configuration setting ([gcic-setup.cpy](#)) is set to a value of 1 - this setting means that line-drawing characters will be used to create the lines and frames you see on the sample screen. The [Customizing GCic](#) section will discuss how to change this to other possibilities should the line-drawing characters not be available to you.

GCic 2.0 (2022/06/11 08:52) - GnuCOBOL 3.2.1 23DEC2020 Interactive Compilation

Folder: C:\Users\Gary\Documents\Programs		Automatic COPY Libraries: C:\GnuCOBOL\copy	
Filename: GCic.cbl		C:\Users\Gary\Documents\Programs	
Click a green/yellow option button to change its setting; click OK to compile using the settings			
Debugging	Source & Xref Listing	Truncate COMP to PIC	Save Temporary Files
'D' Lines are Comments	Yes: Wide (Landscape)	No	No
Run-time Err Checking	Listing Produced By	Optimization	
Normal	GCic	None	
Generate Tracing Code	FUNCTION w/ Intrinsic	Dump DATA DIV on Abort	
			OK
			CANCEL
			HELP

No	FUNCTION is Optional	No
Compiler Output A DLL (-m)	Commands and Warnings No Commands/Warnings	Behavior & Standards DEFAULT
Run After Compilation No	Program Source Format Variable	Pgm-Specified Switches Honor
COBCPY COPY Libraries: _____		
C:\Users\Gary\Documents\MyPROCs_____		
Extra 'cobc' Switches: _____		
-fwrite-after_____		
Extra 'cobc' Arguments: _____		

Program Execution Arguments: _____		

GCic Copyright (C) 2009-2022, Gary L. Cutler, GPL

If GCic is run on Windows, through `cmd.exe`, it must run with codepage 437, 85x (x=0125789), or 86x (x=01234569) activated to display the line-drawing characters. One of those will probably be the default on your Windows system's `cmd.exe` environment - use the `chcp` command (no arguments) to see what the default codepage is on your system.

With a Windows/Cygwin build, set the environment variable CYGWIN to a value of `codepage:oem` (this cannot be done from within the program though - you will have to set the environment variable via Control Panel).

MACOS users may use line drawing characters in this and any GnuCOBOL program simply by setting their `terminal` application's font to `Lucida Console`, because that font contains the line-drawing character set at the appropriate code points. This has worked in some *NIX environments also, but it may depend on the settings of the console/terminal application they use.

Users will have the opportunity to do any or all of the following from this screen:

- Specify the switches to be used on cobc by:
 - Clicking the [option buttons](#) under the sixteen different compilation and listing features until they reflect the desired settings.
 - Coding cobc switches that pertain to compilation features not provided by the feature buttons in the **Extra 'cobc' Switches** field.
 - Taking advantage of [Program-Specified Switches](#).
- Specify any additional files to be compiled or linked by entering them in the **Extra 'cobc' Arguments** field, separated with spaces.
- Specify any COPY libraries that are needed in addition to those listed in the upper-right section of the screen by entering them in the **COBCPY COPY Libraries** field, separated by semicolons (Windows, Windows/MinGW) or colons (others). Any pre-existing value for the `COBCPY` environment variable will be displayed on the screen, where it can be deleted, modified, or added to.
- Specify any program execution arguments to be used in the **Program Execution Arguments** area if you wish to select the **Yes - If Compile OK (-j)** option of the **Run After Compilation** feature.

[TOP](#)

Compiling Programs

After the desired options have been selected, and any desired COPY libraries, cobc switches, extra arguments, and/or execution arguments have been entered (in the example, the user has requested that the `-fwrite-after` switch be added to cobc), clicking (or pressing the Enter key) will initiate the compilation. Clicking (or pressing the Esc key) will quit without compiling. Clicking (or pressing either the PgUp or PgDn keys) will display the document you are reading now.

If the compilation is successful, the compiler messages file will be automatically loaded into the viewing application appropriate for the filetype/extension, normally ".lst", assigned to the file. See the [Customizing GCic](#) section for information on how to change ".lst" to what you might prefer. On Windows, it will be opened via the `start` command, on MacOS it will be via the `open -t` command, and on *NIX it will be opened using the `xdg-open` command. Upon a successful compilation, any selected source/xref listing will appended to the messages file before that file gets auto-opened.

If the compilation failed, the messages file will contain the error messages cobc produced. The file will be automatically loaded, but source/xref listings will be suppressed. GCic will remain running, allowing you to fix the errors and recompile by clicking (or pressing the Enter key).

This will continue until you either get a clean compilation or you click (or press the Esc key).

You may have multiple GCic sessions active simultaneously, but only one session at a time for the same program. This is true of both single- and multi-user environments. Anyone attempting to compile a program that is already being compiled will see a black screen with **"xxxxxxx is already being compiled"**. See the [Locking](#) discussion for more information on how this is done. This is not so much intended to manage multiple programmers working simultaneously on the same computer as it is meant to reduce the clutter of multiple GCic sessions *for the same program* - sessions that exist because the programmer initiated a new compilation after making changes to correct errors rather than simply clicking on the compilation screen that was already active.

[TOP](#)

Program-Specified Switches (PSS)

The GCic screen provides two ways to specify the switches that the cobc compiler will use. There is also a THIRD way, by placing comments such as the following anywhere prior to the PROCEDURE DIVISION of the first program in the compilation group that cobc processes:

```
*> cobc switches: switch-1 [switch-n]...
```

Statements of this form are good to have in your programs to document compiler switches the program needs. GCic, however, can honor (or not) the switches on these comments based upon the setting of the **Pgm-Specified Switches** feature.

The following rules apply to the use of this feature:

- 1. The programmer may use multiple comments, if necessary, to specify all the switches the program needs. There is a default limit of ten such comments, and each comment can be as long as the maximum GnuCOBOL line size (255 characters with **FREE** and **VARIABLE** source formats). You may add (or reduce) the number of allowable lines using `gcic-setup.cpy`. See the [Customizing GCic](#) section for additional information.
- 2. The `*>` form of comment must be used, but there is no column requirement other than what is required by cobc.
- 3. The words `cobc` and `switches:` are case-independent, but there must be EXACTLY ONE space between the `*>` and `cobc`, between `cobc` and `switches:`, and after the colon.
- 4. The switches themselves will be passed directly to cobc, so their capitalization, spelling, and syntax must be exactly as cobc expects.

[TOP](#)

Screen Features and Options

There are currently sixteen categories of compilation features you can control. These are labeled on the screen as **Debugging**, **Source & Xref Listing**, and so on. Directly underneath each feature's caption will be a button with a caption describing the option that is currently selected for that feature. For the two aforementioned features, those option buttons currently have the captions **'D' Lines Are Comments** and **Yes: Wide (Landscape)**, respectively, on the sample screen. All option buttons colored green on that sample screen are showing captions for their default values. Those buttons that are yellow are showing non-default option values.

The following table shows each of the sixteen features and the options available for them. Those options that are **highlighted** are the defaults for their respective features, as GCic is distributed. The number of each option will come into play if you [reconfigure GCic](#) to change the assumed defaults.

Feature	No	Available Options	Cobc Options and Other Actions
Debugging (F1)	1	'D' Lines Are Comments	None
	2	Compile 'D' Lines	<code>-fdebugging-line</code>
Run-time Err Checking (F2)	1	Normal	None
	2	Enhanced	<code>-debug</code>
Generate Tracing Code (F3)	1	No	None
	2	Yes: Procedures Only	<code>-ftrace</code>
	3	Yes: Procedures+Stmnts	<code>-ftraceall</code>
Compiler Output (F4) See note 1, below	1	An EXE (-x)	<code>-x</code>
	2	A DLL (-m)	<code>-m</code>
	3	Save C Source (-C)	<code>-C</code>

	4	Save Asm Source (-S)	-S
	5	Save Object Code (-c)	-c
Run After Compilation (F5)	1	No	None
	2	Yes: If Comp. OK	-j
Source & Xref Listing (F6)	1	None	None
	2	Yes: Wide (Landscape)	-T gcic\$output.txt -t gcic\$output.txt or -T gcic\$output.txt
	3	Yes: Narrow (Portrait)	If the Listing Prepared By feature is set to "cobc", the first switch will be used. If "GCic" is selected, the second switch is used and GCic will reformat the wide listing to the narrow format.
	4	Yes: Preproc. COBOL	-E
Listing Produced By (F7)	1	GCic	-ftsymbols -save-temps The LISTER subroutine needs to work with the expanded source file produced by cobc in order to produce the cross-reference listing - hence the inclusion of -save-temps. GCic will handle temporary file cleanup.
	2	cobc	-X The -T or -t option will be added by the previous feature
Commands and Warnings (F8)	1	No Commands/Warnings	-q -w
	2	Cmnds, Minimal Warnings	None
	3	Cmnds, Most Warnings	-W
	4	Cmnds, All Warnings	-Wall -Wextra -Wadditional
FUNCTION w/ Intrinsics (F9)	1	FUNCTION Is Optional	-fintrinsics=all
	2	Managed By REPOSITORY	None
Program Source Format (F10) See note 4, below	1	Free	-free
	2	Fixed	-fixed
	3	Variable	-fixed -ftext-column=255
Truncate COMP to PIC (F11)	1	No	-fnotrunc
	2	Yes	None
Optimization (F12)	1	None	-O0
	2	Size Only	-Os
	3	Size+Speed (Level 1)	-O
	4	Size+Speed (Level 2)	-O2
Dump DATA DIV on Abort (F13)	1	No	None
	2	Yes	-fdump=all
Behavior & Standards (F14)	1	AcuCOBOL	-std=acu
	2	AcuCOBOL (Strict)	-std=acu-strict
	3	BS2000	-std=bs2000
	4	BS2000 (Strict)	-std=bs2000-strict
	5	COBOL2002	-std=cobol2002
	6	COBOL2014	-std=cobol2014
	7	COBOL85	-std=cobol85
	8	DEFAULT	-std=default
	9	Bull GCOS	-std=gcos
	10	Bull GCOS (Strict)	-std=gcos-strict

	11	IBM Enterprise	-std=ibm
	12	IBM Enterprise (Strict)	-std=ibm-strict
	13	MicroFocus	-std=mf
	14	MicroFOCUS (Strict)	-std=mf-strict
	15	REALIA	-std=realia
	16	REALIA (Strict)	-std=realia-strict
	17	RmCOBOL	-std=rm
	18	RmCOBOL (Strict)	-std=rm-strict
	19	IBM VSCOBOL II	-std=mvs
	20	IBM VSCOBOL II (Strict)	-std=mvs-strict
	21	XOPEN	-std=xopen
Pgm-Specified Switches (F15)	1	Ignore	None
	2	Honor	None
Save Temporary Files (F16)	1	No	None
	2	Yes	-save-temps

The [Customizing GCic](#) section will show you how you may change any of these defaults. To change an option for a single compilation, just left-click an option button on the screen to advance the selection forward (i.e. downward on the above list) and use a right-click to move the selection backward (i.e. upward on the above list).

Some of these features/options deserve special consideration, as follows.

1. The **Compiler Output** feature has the default shown, but if the 1st program within the source code file specified as the argument to GCic has a **LINKAGE SECTION**, the option will default to **A DLL (-m)**. If the **LINKAGE SECTION** cannot be found in that program, the **An EXE (-x)** option will be assumed. If you configure GCic to make the **A DLL (-m)** option the default, this checking will not be performed. The terms "EXE" and "DLL" will change, depending on the OS and environment GCic is configured for, as follows:
 - When GCic is compiled for MacOS, they will be **Executable File (-x)** and **A Dylib (-m)**, respectively
 - When GCic is compiled for *NIX, they will be **Executable File (-x)** and **An SO (-m)**, respectively
2. The **Listing Produced By** feature allows you to specify whether source and cross-reference listings should be produced by GCic or by the native functionality built-in to the cobc compiler.
3. You'll probably want to experiment with the options for the **Commands and Warnings** feature to find the setting you like best.
4. The three options of the **Program Source Format** feature are equivalent to using **>>SOURCE FORMAT IS FREE**, **>>SOURCE FORMAT IS FIXED**, and **>>SOURCE FORMAT IS VARIABLE**, respectively, in your programs.
5. Use the **No** option of the **Truncate COMP to PIC** feature to get the full range of values out of **BINARY** and **COMP** fields. For example, **PIC S99 COMP** fields will normally be limited to a range of values of -99 to +99. By turning this truncation off, the range of possible values would become -127 to +127 (0-255 if unsigned). As a bonus, turning truncation off will yield a substantial performance boost to arithmetic operations performed on **BINARY** and **COMP** fields.
6. GCic generates a messages file when it invokes cobc to compile a program. The contents of that file are controlled by the **Commands and Warnings** feature. This dictates whether or not the text of the generated cobc command, along with the compilation and linking commands cobc generates, will be included. It also controls how in-depth any cobc-generated warnings will be.

Error messages generated by cobc are always included

If source and cross-reference listings are selected by the **Source & Xref Listing** feature, they will be appended to the messages file. If cobc issued error messages, source/xref listings will NOT be generated.

7. Should the compilation fail, GCic will NOT terminate, but will display the following bottom line on the screen:

Compilation Failed - Correct error and try again

As the message suggests, you may correct the errors and click (or press the Enter key) to recompile. This will continue until you've either corrected all outstanding errors and have gotten a clean compilation, or until you click (or press the Esc key). Remember that the messages file, which will be refreshed with each [re]compilation, will contain all error and warning messages generated by cobc.

The GCic-Generated Source Listing

GCic 2.0 (2022/08/27 09:28) for GnuCOBOL 3.1.2 (23DEC2020)		Source Listing		2022/08/28
C:/Users/Gary/Documents/Programs/GCic.cbl		MAIN		
LineNo	Source Statement; LineNo: Statement locn in expanded source; Source: Statement locn in its file			
379	+55 \$SET CONSTANT FASTSIZE 50			
380	+56 \$SET CONSTANT GCICVER '2.0'			
381	+57 \$SET CONSTANT LINEDRAW 1			
382	+58 \$SET CONSTANT LISTEXT '.lst'			
383	+59 \$SET CONSTANT LPP 51			
384	+60 \$SET CONSTANT LPPP 78			
385	+61 \$SET CONSTANT OS 'MINGW'			
386	+62 \$SET CONSTANT PROMPTCHAR ' '			
387	+63 \$SET CONSTANT RAFTSIZE 150			
388	+64 \$SET CONSTANT REF ' '			
389	+65 \$SET CONSTANT RMTSIZE 2047			
390	+66 \$SET CONSTANT STACKSIZE 50			
391	+67 \$SET CONSTANT STSIZE 7500			
392	+68 \$SET CONSTANT UPD ' '			
393	+69			
394	+70 \$END-IF			
395	+71			
396	+72			
397	+73 *-> DO NOT CHANGE ANYTHING AFTER THIS POINT UNLESS YOU ADD A NEW DEBUG **			
398	+74 *-> SWITCH **			
399	+75 *->*****			
400	+76			
NOT COMPILED	DEBUG D01 Debug-Switches.			
NOT COMPILED	DEBUG D 05 INFO	PIC 9(1).		
NOT COMPILED	DEBUG D 05 SOURCE-Sw	PIC 9(1).		
NOT COMPILED	DEBUG D 05 SPLIT	PIC 9(1).		
NOT COMPILED	DEBUG D 05 SUB	PIC 9(1).		
NOT COMPILED	DEBUG D 05 TRACE-Sw	PIC 9(1).		
NOT COMPILED	DEBUG D 05 USER-Sw	PIC 9(1).		
NOT COMPILED	DEBUG D 05 X-OPTIONS.	PIC 9(1).		
NOT COMPILED	DEBUG D 10 XFAST	PIC 9(1).		
NOT COMPILED	DEBUG D 10 XFSM	PIC 9(1).		
NOT COMPILED	DEBUG D 10 XPARSE	PIC 9(1).		
NOT COMPILED	DEBUG D 10 XRAFT	PIC 9(1).		
NOT COMPILED	DEBUG D 10 XREAD	PIC 9(1).		
NOT COMPILED	DEBUG D 10 XREF	PIC 9(1).		
NOT COMPILED	DEBUG D 10 XTOKEN	PIC 9(1).		
NOT COMPILED	DEBUG D 10 XWORDS	PIC 9(1).		
417	129			
418	130 >>IF OS = "CYGWIN"			
NOT COMPILED	CYWIN 01 05-Dir-Chr	VALUE "/"	PIC X(1).	
NOT COMPILED	CYWIN 78 05-Exe-Txt	VALUE " "	An EXE (-x)	" "
NOT COMPILED	CYWIN 78 05-Lib-Txt	VALUE " "	A DLL (-m)	" "
NOT COMPILED	CYWIN 78 05-Path-Char	VALUE ":",		
NOT COMPILED	CYWIN 01 05-Type-Code	VALUE 2	PIC 9(1).	
424	136 >>ELIF OS = "MINGW"			
425	137 MINGW 01 05-Dir-Chr	VALUE "\"	PIC X(1).	
426	138 MINGW 78 05-Exe-Txt	VALUE " "	An EXE (-x)	" "
427	139 MINGW 78 05-Lib-Txt	VALUE " "	A DLL (-m)	" "
428	140 MINGW 78 05-Path-Char	VALUE ":",		
429	141 MINGW 01 05-Type-Code	VALUE 5	PIC 9(1).	
430	142 >>ELIF OS = "MACOS"			
NOT COMPILED	MACOS 01 05-Dir-Chr	VALUE "/"	PIC X(1).	
NOT COMPILED	MACOS 78 05-Exe-Txt	VALUE "A Standalone Executable (-x)		" "
NOT COMPILED	MACOS 78 05-Lib-Txt	VALUE " "	A DYLIB (-n)	" "
NOT COMPILED	MACOS 78 05-Path-Char	VALUE ":",		
NOT COMPILED	MACOS 01 05-Type-Code	VALUE 4	PIC 9(1).	
436	148 >>ELIF OS = "NIX"			
NOT COMPILED	*NIX 01 05-Dir-Chr	VALUE "/"	PIC X(1).	
NOT COMPILED	*NIX 78 05-Exe-Txt	VALUE "A Standalone Executable (-x)		" "
NOT COMPILED	*NIX 78 05-Lib-Txt	VALUE " "	An SO (-m)	" "
NOT COMPILED	*NIX 78 05-Path-Char	VALUE ":",		
NOT COMPILED	*NIX 01 05-Type-Code	VALUE 3	PIC 9(1).	
442	154 >>ELIF OS = "WINDOWS"			
NOT COMPILED	WINDOZ 01 05-Dir-Chr	VALUE "\"	PIC X(1).	
NOT COMPILED	WINDOZ 78 05-Exe-Txt	VALUE " "	An EXE (-x)	" "
NOT COMPILED	WINDOZ 78 05-Lib-Txt	VALUE " "	A DLL (-m)	" "
NOT COMPILED	WINDOZ 78 05-Path-Char	VALUE ":",		
=====				
GCic for Windows/MinGW Copyright (C) 2009-2022, Gary L. Cutler, GPL		Page: 6		

This is a sample of a GCic source listing. These listings show every line of COBOL code in the entire compilation group, including comments and code introduced by COPY statements.

Some areas of this sample have been highlighted:

1. The YELLOW area on the listing will be the name of the file whose contents were compiled. It will appear exactly as it was entered on the command line or in the "Extra cobc Arguments" field on the screen.
2. The BLUE area on the listing will be the **PROGRAM-ID** or **FUNCTION-ID** of the program whose code is listed on the page.
3. The LineNo column on the listing shows the relative position of the current line of source code with respect to the total of all code that was compiled. These line numbers will be the ones that appear on the cross-reference listing.
4. The Source column on the listing shows the relative position of the current line of source code within the file in which that source line is found. Lines that are COPYed will be prefixed with a +, and the number is the relative line number in the proc.
5. The NOT COMPILED markers on the listing show source lines that were skipped by the compiler due to \$IF or >>IF statements, or debugging lines that were treated as comments by cobc.

GCic determined that the debugging lines were not compiled because NONE of the following were detected by GCic:

1. A **SOURCE-COMPUTER** paragraph **WITH DEBUGGING MODE**.
2. Debugging mode being requested by the user via the **Compile 'D' Lines** option of the **Debugging** feature.
3. Debugging mode being implied by the user having selected the **Enhanced** option of the **Run-time Err Checking** feature.
4. Debugging mode being requested by the user via the **-fdebugging-line** switch being included in the **Extra 'cobc' Switches** area on the GCic screen, or on a **Program-Specified Switch** comment.
5. Debugging mode being implied by the user having specified the **-debug** switch being included in the **Extra 'cobc' Switches** area on the GCic screen, or on a **Program-Specified Switch** comment.

GCic will start the source listing of each file it compiles at the top of a new page. The same is true of any non-nested programs in that file. You may use the **/**, **eject**, **space1**, **space2**, and **space3** directives to control the source listing; you may wish to include a **" / "** (or an **"eject"**) between programs in your source files so that each program begins on a new page.

The GCic-Generated Cross-Reference Listing

GCic 2.0 (2022/08/27 09:28) for GnuCOBOL 3.1.2 (23DEC2020)		Cross-Reference Listing		2022/08/28
C:/Users/Gary/Documents/Programs/GCic.cbl		LISTER		

This is a sample of a GCic cross-reference


```

User-Defined Name; '^'=Defn, '*'=Upd, ' '=Ref
=====
CT-Div-Code ..... WORKING-STORAGE SECTION, ALPHANUMERIC, Level: 05, Bytes: 1, Desc: 'X(1)'
3261^ 5129 5208 5222 5306 5322 5350 5384 5449

CT-Is-Buzzword ..... WORKING-STORAGE SECTION, CONDITIONAL, Level: 88
3250^ 5330

CT-Is-Device ..... WORKING-STORAGE SECTION, CONDITIONAL, Level: 88
3251^                                     *** NOT REFERENCED ***

CT-Is-Feature ..... WORKING-STORAGE SECTION, CONDITIONAL, Level: 88
3254^                                     *** NOT REFERENCED ***

CT-Is-LvNo ..... WORKING-STORAGE SECTION, CONDITIONAL, Level: 88
3255^ 5446

CT-Is-Name ..... WORKING-STORAGE SECTION, CONDITIONAL, Level: 88
3256^ 5303 5381 5593

CT-Is-NDF ..... WORKING-STORAGE SECTION, CONDITIONAL, Level: 88
3252^ 4945 5219

CT-Is-NDS ..... WORKING-STORAGE SECTION, CONDITIONAL, Level: 88
3253^ 4950 5126 5666

CT-Is-RsvdWid ..... WORKING-STORAGE SECTION, CONDITIONAL, Level: 88
3257^ 5205

CT-Is-Switch ..... WORKING-STORAGE SECTION, CONDITIONAL, Level: 88
3258^                                     *** NOT REFERENCED ***

CT-Is-UserDefn ..... WORKING-STORAGE SECTION, CONDITIONAL, Level: 88
3259^ 5072 5319 5619 5681 5721

CT-Is-Verb ..... WORKING-STORAGE SECTION, CONDITIONAL, Level: 88
3260^ 5035

CT-Line ..... WORKING-STORAGE SECTION, NUMERIC, Level: 05, Bytes: 6, Desc: '9(6)'
3248^ 5127 5206* 5220 5304 5320 5348 5382 5447 5937 5959 5982
6002 6026 6063 6118* 6154* 6499 6524

CT-Name ..... WORKING-STORAGE SECTION, ALPHANUMERIC, Level: 05, Bytes: 63, Desc: 'X(63)'
3263^ 3264 4492 4496 4500 4533 5124 5209 5217 5297 5300 5385
5386 5435 5579 5596 5598 5647 5685 5688 5741 5747 5776 5780
5784 5795 5798 5802 5806 5816 5831 5855 5859 5863 5866 5900
5936 5953 5981 6088 6117 6153 6161 6291 6317

CT-Paren-Level ..... WORKING-STORAGE SECTION, NUMERIC, Level: 05, Bytes: 3, Desc: '9(3)'
3262^ 5128 5207 5221 5305 5321 5349 5383 5448 6091 6098 6150

CT-Sub ..... WORKING-STORAGE SECTION, NUMERIC, Level: 05, Bytes: 3, Desc: '9(3)'
3247^ 5425 5433 5577 5578 5579

CT-Type ..... WORKING-STORAGE SECTION, ALPHANUMERIC, Level: 05, Bytes: 1, Desc: 'X(1)'
3249^ 5328

CT-UC-Name ..... WORKING-STORAGE SECTION, ALPHANUMERIC, Level: 05, Bytes: 63, Desc: 'X(63)'
3268^ 4961 4993 5037 5124 5209 5217 5297 5299 5300 5324 5336
5385 5386 5450 5899 5935 5980 5998 6021 6086 6116 6152 6160

CT-Valid-Level-Num ..... WORKING-STORAGE SECTION, CONDITIONAL, Level: 88
3266^ 5437

Curr-Division ..... WORKING-STORAGE SECTION, ALPHANUMERIC, Level: 01, Bytes: 1, Desc: 'X(1)'
3234^ 5129 5208 5222 5306 5322 5350 5384 5449

Curr-FD-Filename ..... WORKING-STORAGE SECTION, ALPHANUMERIC, Level: 01, Bytes: 63, Desc: 'X(63)'
3240^ 5685 5688 6290

Curr-Filename ..... WORKING-STORAGE SECTION, ALPHANUMERIC, Level: 01, Bytes: 255, Desc: 'X(255)'
3242^ 5493 5496 5502 5503

=====
GCic for Windows/MinGW Copyright (C) 2009-2022, Gary L. Cutler, GPL
Page: 128

```

("xref", for short) listing. These listings are generated by analyzing every non-comment line of COBOL code in the complete compilation group.

Some points of note about the listing have been highlighted:

1. The YELLOW area on the listing will be the name of the file whose contents were compiled. It will appear exactly as it was entered on the command line or in the "Extra cobc Arguments" field on the screen.
2. The BLUE area on the listing will be the PROGRAM-ID or FUNCTION-ID of the program whose user-defined items are listed on the page.
3. The PURPLE area of the report shows places in the total compilation group where the data item is defined (denoted by a trailing "^"), referenced (denoted by a trailing space), or updated (denoted by a trailing "*"). These line numbers track back to the "LineNo" column in the source listing.
4. The GREEN area highlights the data item attributes that will be presented for items on the report.
5. The *** NOT REFERENCED *** markers on the listing flag data items that are never referenced or updated.

Here are some points of interest regarding the cross-reference process:

1. The three characters used to flag definition, references, and updates of user-defined data items and procedures are configurable by the user. See the discussions of **DEF**, **REF**, and **UPD** in the [Customizing GCic](#) section.
2. Updates to data-items that are part of a group item will propagate update cross-references upward to the top of the group hierarchy (the 01-level data item that "roots" the tree). Similarly, update cross-references will also propagate downward through the subtree rooted by the originally-updated item. This will not happen with references. How this is done is explained in the [The "Symbol-Table" Table](#) section.
3. I/O references to files/records will propagate to equivalent references to their records/files, but will not currently propagate to references to the key fields of the records of **RELATIVE** or **INDEXED** files. How this is done is explained in the [The "Records-And-Files-Tbl" Table](#) section.
4. The **OPENing** and **CLOSEing** of files logs only references to the file(s).
5. Cross-referencing programs that contain reserved or user-defined words that have been split across lines (using a - in column 7 of the continuation lines) will consider each of the word segments on the two lines to be separate words. I do have to ask though - "Why are you coding like that to begin with?" The splitting of character strings in that manner will not be a problem.
6. The cross-reference listing preserves full (63-character) user-defined names, but the process of propagating updates upward and downward group item structures matches names only on the first 30 characters. This is because that propagation relies on the "Symbol Table" produced by cobc (via the **-ftsymbols** switch) and that report currently truncates names at 30 characters.

[TOP](#)

Customizing GCic

Screen Automatic Resizing

GCic expects to run in a terminal window that has at least 35 rows and 106 columns. In some OS environments, it is possible for GCic to automatically resize the terminal window to that geometry, if any aspect of the current screen size is smaller than that.

Native Windows

GCic *will assume* you are using cmd.exe and will use the command `mode con: cols=106 lines=35` to resize the window to 35 lines x 106 columns.

Windows/Cygwin

GCic will assume you are using the `rxvt` terminal emulator and will use the command `printf "\x1B[8;35;106t"` to issue the VT100 command sequence to resize the window.

MacOS

GCic will use the command sequence `resize -s 35 106&&stty rows 35&&stty cols 106` to resize the terminal window to 35 lines x 106 columns.

*NIX

GCic *will assume* you are using an X11-based terminal emulator and will use the command string `resize -s 35 106&&stty rows 35&&stty cols 106` to resize the window.

Windowsb/MinGW

GCic *will assume* you are using cmd.exe and will use the command `mode con: cols=106 lines=35` to resize the window.

If you don't find the right combination of environments and assumptions for your setup, automatic resizing *may* not be possible for you and you may just have to resize your terminal window manually.

Before giving up on automatic resizing, do some research into the terminal or console emulator you are using. Perhaps it *does* have a way to resize its window from the command line. If so, you can [configure GCic](#) to issue the necessary command(s) using the `RESIZECMD` configuration constant, provided the total size of the command(s) needed (including ";", "|", or "&&" command separator characters) does not exceed 256 characters. If you do need more than a linear sequence of commands 256 characters long, consider writing a shell script to do the job and then configure GCic to execute that script.

If your system runs *NIX and you do NOT use an X11 emulator (i.e. xterm) and you CANNOT identify a command sequence to resize the screen window, [set the XTERM configuration constant](#) to 0 (zero) to at least stop trying to run `resize`.

[TOP](#)

The 'gcic-setup.cpy' Proc

The `gcic-setup.cpy` proc, distributed as part of GCic, provides these functions:

1. It defines the color palette used for the GCic screen, and allows it to be customized.
2. It allows for the customization of default options for the features, the sizing of various tables used within the program, specification of the host environment GCic will be running in, and more.
3. It defines the data structure used to manage the various debugging switches that can be activated in GCic - see [Testing GCic](#) for more information on debugging.
4. It defines the data structure that the `GCINFO` subroutine populates with information about the GnuCOBOL version and operating system type.

All color-palette configuration is performed by modifying the values of level-78 constants defined in the proc. Each 78-level data item's relationship to the screen is documented in the proc.

Functional configuration settings and table sizings are accomplished by changing `$SET` statements for the following constants. All changes made to `gcic-setup.cpy` other than changes to comments, of course, require a recompilation of GCic.cbl to take effect.

Constant Name	Setting as Distributed	Description
CMDMAXSIZE	8191	This will define the size (in characters) of the buffer GCic uses to submit commands (like "cobc") to the operating system. It is recommended that this be set to the maximum size of a command in the operating system GCic will be running under. Windows (including MinGW and Cygwin) allows 8191 characters. Most *NIX versions allow 4096 and MacOS allows 262144 (as do many BSD UNIX systems) - it might be a just a LITTLE absurd to go THAT HIGH though.
DEF	'^'	The character that will be appended to line numbers in the GCic-generated cross-reference listing to indicate that the item in question was DEFINED at that line number in the merged and expanded source listing (the number in the

		"LineNo") column in the source listing.
F1	1	Debugging feature default option number (see Using GCic).
F2	1	Run-time Err Checking feature default option number (see Using GCic).
F3	1	Generate Tracing Code feature default option number (see Using GCic).
F4	1	Compiler Output feature default option number (see Using GCic).
F5	1	Run After Compilation feature default option number (see Using GCic).
F6	1	Source & Xref Listing feature default option number (see Using GCic).
F7	1	Listing Generated By feature default option number (see Using GCic).
F8	2	Commands and Warnings feature default option number (see Using GCic).
F9	1	FUNCTION w/ Intrinsics feature default option number (see Using GCic).
F10	3	Program Source Format feature default option number (see Using GCic).
F11	1	Truncate COMP to PIC feature default option number (see Using GCic).
F12	1	Optimization feature default option number (see Using GCic).
F13	1	Dump DATA DIV on Abort feature default option number (see Using GCic).
F14	8	Behavior & Standards feature default option number (see Using GCic).
F15	1	Pgm-Specified Switches feature default option number (see Using GCic).
F16	1	Save Temporary Files feature default option number (see Using GCic).
FASTSIZE	50	Set this to the desired size of the Files And Statuses table. This should have a value of the expected number of data items registered as FILE STATUS or SORT STATUS items across all programs being compiled and cross-referenced. This has no bearing on cross-referencing performed by cobc/cobxref.
GCICVER	'2.0'	The version.release.update.subupdate number for GCic. Set to: 0 To use spaces (no lines) 1 To use the line-drawing character set (PC codepage 437) 2 To use conventional ASCII characters (+, -,)
LINEDRAW	1	MACOS USERS - To use the linedrawing character set, set your 'terminal' font to 'Lucida Console'
LISTEXT	'lst'	The desired extension for the file containing any "cobc" output as well as the source and cross-reference listings. In prior versions you were stuck with ".gclst" but as of V2.0 you can choose your own extension (including ".gclst" if you want to stick with that). DON'T FORGET THE LEADING PERIOD!
LPP	51	Set to the maximum number of printable lines per page when a GCic-generated listing should be generated for landscape orientation (can be overridden at runtime time using the GCXREF_LINES environment variable). This value plus 6 (the number of lines in page headers) plus 3 (the number of lines in page footers) should add up to the total number of theoretically-printable lines on the printer you use. Lines-per-page control for cobc-generated listings uses the cobc --tlines=Lines switch, which must be entered in the Extra 'cobc' Switches field on the GCic screen.
LPPP	78	Similar to LPP , but applies when a GCic-generated listing should be generated for portrait orientation (can be over-ridden at runtime time using the GCXREF_LINES_PORT environment variable).
MAXSWITCH	200	Set to the maximum number of switches you believe will EVER need to be specified on a cobc command. This value is probably much more than sufficient.
OS	0	Defines the Operating System and environment. Set to: 0 Attempt to determine the OS and environment automatically 1 Native Windows 2 Windows/Cygwin 3 *NIX 4 MacOS 5 Windows/MinGW
PROMPTCHAR	' '	Set to the character that will serve as the fill (i.e. "PROMPT") character for the four input areas on the screen. A common character used instead of a space is

an underscore ("_"); you're welcome to use that! of course! but it may make the screen look a little too "busy".

PSSQTY	10	Set this to the maximum number of <code>*> cobc switches:</code> lines that can be processed
RAFTSIZE	150	Set this to the desired size of the Records And Files table. This should be a value greater than the expected total number of 01 level data items defined in the FILE SECTIONS of all programs being compiled and cross-referenced by GCic. This has no bearing on cross-referencing performed by cobc/cobxref.
REF	<code>' '</code>	The character that will be appended to line numbers in the GCic-generated cross-reference listing to indicate that the item in question was REFERENCED at that line number in the merged and expanded source listing (the number in the "LineNo") column in the source listing.
RESIZECMD	<code>' '</code>	If GCic as-distributed cannot programmatically resize the terminal/console window, and you know of a command-line sequence that <i>will</i> do the job, code that sequence here. The command sequence is limited to a single line of code in the proc; command separator sequences (" <code>;</code> ", " <code> </code> ", " <code>&&</code> ") <u>can be used</u> .
RWTSIZE	2047	Set to the size of the Reserved Word Table . This should have a value of a power of 2, minus one. As of GnuCOBOL 3.1 there are a little more than 1000 reserved words, so this value here should be good for quite a while, if not forever.
STACKSIZE	50	Set to the size of the stack used in the scanning of <code>ADD</code> , <code>SUBTRACT</code> , <code>MULTIPLY</code> , and <code>DIVIDE</code> statements. This should have a value of the expected number of numeric identifiers used in any ONE of these statements, in any single program being cross-referenced. It's hard to imagine one of these statements involving more than THIS number of identifiers.
STSIZE	7500	Set to the size of the Symbol Table which contains an entry for each non-FILLER, non-77, non-66, and non-78 data item defined in all programs currently being compiled and cross-referenced.
UPD	<code>'*'</code>	The character that will be appended to line numbers in the GCic-generated cross-reference listing to indicate that the item in question was UPDATED at that line number in the merged and expanded source listing (the number in the "LineNo") column in the source listing.
X11	1	A YES (1) / NO (0) flag indicating whether or not X11 is installed. If 1, GCic will attempt to use the X11 <code>resize</code> command to resize the terminal window.

[TOP](#)

Internals

This section will provide a glimpse into the structure, data structures, and algorithms that make up the GCic program. It consists of a main program (named "MAIN") and six subprograms called by MAIN, all found in a single file named `gcic.cbl`. The programs in this file are separated from one another using `END PROGRAM` statements.

The following table summarizes each of the seven programs, in the order in which they occur in `gcic.cbl`.

PROGRAM-ID		Description
MAIN	Purpose:	This is the main program for GCic. It is responsible for all user interaction and for generating and submitting the cobc command per the user's wishes.
DBGCOL	Purpose:	This subroutine generates a dump of the contents of up to six <code>USAGE DISPLAY</code> data items, separated from one another by <code> </code> characters.
	Syntax:	<pre>CALL "DBGCOL" USING debug-switch "switch-name" spacer identifier-n... END-CALL</pre>
	Arguments:	<code>debug-switch</code> This is the name of one of the debug switches, as defined in the <code>gcic-setup.cpy</code>

proc - for example, **TRACE-Sw**, **XFSM-Sw**, **USER-Sw**, and so on. The value of this data item will be either 1 or 0, depending on whether the switch was coded on the **gcic** command (1) or not (0). if the switch value is 0, the subroutine immediately returns without generating any output.

"switch-name"

This is the name of the switch (without the trailing "-Sw"), and will appear starting in column 1 of all output generated by the subroutine.

spacer

This argument is either a literal string or **PIC X(n)** identifier whose contents will serve as a padding sequence, separating the dump contents from the dump header (see example).

identifier-n

The **USAGE DISPLAY** identifier(s) that are to be dumped. There can be anywhere from one to six of these.

Example:

```

DEBUG D    CALL "DBGCOL" USING XPARSE
DEBUG D                                "XPARSE"
DEBUG D                                "Released (435):  "
DEBUG D                                SWR-Prog-ID
DEBUG D                                SWR-Line
DEBUG D                                SWR-Name-UC
DEBUG D    END-CALL

```

Assuming the three identifiers being dumped contain "PrntB", "2433", and "Qty" and also assuming that the **XPARSE** switch has a value of 1, the generated output would be:

```
XPARSE.....Released (435): |PrntB|2433|Qty
```

Purpose: This subroutine generates a dump of the contents of up to three **USAGE DISPLAY** data items, each presented in a **keyword="value"** format. There can be as many as three sets of the last two arguments.

Syntax:

```

CALL "DBGKWV" USING debug-switch,
                  "switch-name"
                  { "text-n" identifier-n }...
END-CALL

```

debug-switch

This is the name of one of the debug switches, as defined in the **gcic-setup.cpy** proc - for example, **TRACE-Sw**, **XFSM-Sw**, **USER-Sw**, and so on. The value of this data item will be either 1 or 0, depending on whether the switch was coded on the **GCic** command (1) or not (0). if the switch value is 0, the subroutine immediately returns without generating any output.

DBGKWV Arguments: **"switch-name"**
This is the name of the switch (without the trailing "-Sw"), and will appear starting in column 1 of all output generated by the subroutine.

"text-n"

A text string that will appear to the left of the "=". Usually this will be the next argument's name.

identifier-n

The **USAGE DISPLAY** data item whose contents will appear to the right of the "=".

Example:

```

DEBUG D    CALL "DBGKWV" USING XTOKEN
DEBUG D                                "XTOKEN"
DEBUG D                                "SWR-Prog-ID" SWR-Prog-ID
DEBUG D                                "Line No"      SWR-Line
DEBUG D                                "SWR-Name-UC"  SWR-Name-UC
DEBUG D    END-CALL

```

Assuming the three identifiers being dumped contain "PrntB", "2433", and "Qty" and also assuming that the **XTOKEN** switch has a value of 1, the generated output would be:

Purpose: This subprogram will be CALLED by MAIN, GCINFO, LISTER and LOADER to generate debugging output - in this case, a text message passed by the CALLER. This routine is invoked only when debugging features have been enabled (see the [Testing GCic](#) topic).

Syntax:

```
CALL "DBGTXT" USING debug-switch,  
                    "switch-name",  
                    "text"  
  
END-CALL
```

DBGTXT

debug-switch
This is the name of one of the debug switches, as defined in the [gcic-setup.cpy](#) proc - for example, [TRACE-Sw](#), [XFSM-Sw](#), [USER-Sw](#), and so on. The value of this data item will be either 1 or 0, depending on whether the switch was coded on the GCic command (1) or not (0). if the switch value is 0, the subroutine immediately returns without generating any output.

Arguments:

"switch-name"
This is the name of the switch (without the trailing "-Sw", if any), and will appear starting in column 1 of all output generated by the subroutine.

"text"
This is the text to be displayed It could also be specified as a [USAGE DISPLAY](#) identifier.

Example:

```
DEBUG D    CALL "DBGTXT" USING XTRACE-Sw  
DEBUG D                                "XTRACE"  
DEBUG D                                "100-Initialization"  
DEBUG D    END-CALL
```

If the XTRACE-Sw switch has the value 1, the following output will be produced:

```
XTRACE.....100-Initialization
```

Purpose: MAIN CALLs this subroutine to collect information about the GnuCOBOL installation on your system. The information will be collected by executing cobc with just the "-i" option, and parsing the output to collect the information.

Syntax:

```
DEBUG D    CALL "GCINFO" USING BY REFERENCE identifier-1  
                                BY CONTENT   identifier-3  
                                END-CALL
```

GCINFO

identifier-1
The group item that will receive the information:

Arguments:

```
01 GCInfo-Arg.  
  05 GA-Version                PIC X(20).  
  05 GA-Version-No-X.  
    10 GA-Version-No          PIC 9(8).  
  05 GA-Release-No-X.  
    10 GA-Release-No          PIC 9(8).  
  05 GA-Update-Major-No-X.  
    10 GA-Update-Major-No     PIC 9(8).  
  05 GA-Update-Minor-No-X.  
    10 GA-Update-Minor-No     PIC 9(8).  
  05 GA-Release-Date           PIC X(9).  
  05 GA-Build-Date             PIC X(9).
```

05 GA-Build-Env	PIC X(30).
05 GA-OS-Type	PIC 9(1).
88 GA-OS-Unknown	VALUE 0.
88 GA-OS-Windows	VALUE 1.
88 GA-OS-Cygwin	VALUE 2.
88 GA-OS-SplatNIX	VALUE 3.
88 GA-OS-MacOS	VALUE 4.
88 GA-OS-MinGW	VALUE 5.

identifier-2

This argument is passed to GCINFO only if debugging mode is active. This is passed to provide GCINFO access to the "INFO" switch. See the [Testing GCic](#) section for more information.

This program is responsible for generating GCic's source and cross-reference listings. This subroutine will ONLY be called if:

- Purpose:**
1. You select the **GCic** option of the **Listing Produced By** feature, **and...**
 2. You choose either the **Yes - Wide (Landscape)** or **Yes - Narrow (Portrait)** options of the **Source & Xref Listing** feature.

Syntax:

```
CALL "LISTER" USING arg-1 arg-2 arg-3 arg-4 arg-5 arg-6
                    arg-7 arg-8 arg-9 arg-10 arg-11 [ arg-12 ]

END-CALL
```

arg-1

The name of the messages file to which the source and xref listings should be appended.

arg-2

The filename argument from the command line.

arg-3

9(2) listing code (02=landscape, 03=portrait)

arg-4

The banner line (line 2) from the screen (line 1 is completely blank)

arg-5

The copyright line (line 34) from the screen

arg-6

The GCINFO return record

Arguments:

arg-7

The name of the cobc temp files folder

arg-8

The group item that contains variable information for error messages

arg-9

The "-std" switch used to compile the program(s) being listed.

arg-10

A **PIC 9(4) DISPLAY** item LISTER uses to pass an error number back to MAIN.

arg-11

A one-character Y/N flag item that tells LISTER whether the user that compiled the program being listed specified either the **-debug** or **-fdebugging-line** switch.

arg-12

The debugging switches record. This is present only if debugging features are enabled.

LISTER

This subroutine is CALLED by MAIN to:

LOADER

Purpose:

1. Load the messages file into the user's text editor, and...
2. Load this file into a web browser when the user clicks , presses PgUp, or

presses PgDn

Syntax: CALL "LOADER" USING *file-name* END-CALL

Arguments: *file-name*
A character string or PIC X(n) identifier containing the fully-qualified name of the file to be loaded.

[TOP](#)

The GCic MAIN Program

As was stated earlier, this is the program responsible for all interaction with the user. All of its actions are managed through the use of four key tables.

[TOP](#)

The "Button-Tbl" Table

```
01 Button-FILLER.
*>*****
*> Button coordinates for the four two-line entry fields      **
*>*****
    05 PIC X(21) VALUE "025003104NOP".      *> COBCPY COPY Libraries
    05 PIC X(21) VALUE "027003104NOP".      *> Additional 'cobc' Switches
    05 PIC X(21) VALUE "029003104NOP".      *> Additional 'cobc' Arguments
    05 PIC X(21) VALUE "031003104NOP".      *> Program Execution Args (line 1)
    05 PIC X(21) VALUE "032003104NOP".      *> Program Execution Args (line 2)
*>*****
*> Button coordinates for the sixteen function option buttons **
*>*****
    05 PIC X(21) VALUE "010004025F01".      *> Debugging
    05 PIC X(21) VALUE "013004025F02".      *> Run-time Err Checking
    05 PIC X(21) VALUE "016004025F03".      *> Generate Tracing Code
    05 PIC X(21) VALUE "019004025F04".      *> Compiler Output
    05 PIC X(21) VALUE "022004025F05".      *> Run After Compilation
    05 PIC X(21) VALUE "010027048F06".      *> Source & Xref Listing
    05 PIC X(21) VALUE "013027048F07".      *> Listing Produced By
    05 PIC X(21) VALUE "016027048F08".      *> FUNCTION w/ Intrinsics
    05 PIC X(21) VALUE "019027048F09".      *> Commands and Warnings
    05 PIC X(21) VALUE "022027048F10".      *> Program Source Format
    05 PIC X(21) VALUE "010050071F11".      *> Truncate COMP to PIC
    05 PIC X(21) VALUE "013050071F12".      *> Optimization
    05 PIC X(21) VALUE "016050071F13".      *> Dump DATA DIV on Abort
    05 PIC X(21) VALUE "019050071F14".      *> Behavior & Standards
    05 PIC X(21) VALUE "022050071F15".      *> Pgm-Specified Switches
    05 PIC X(21) VALUE "010073094F16".      *> Save Temporary Files
*> 05 PIC X(21) VALUE "013073094F17".      *> Unused
*> 05 PIC X(21) VALUE "016073094F18".      *> Unused
*> 05 PIC X(21) VALUE "019073094F19".      *> Unused
*> 05 PIC X(21) VALUE "022073094F20".      *> Unused
*>*****
*> Button coordinates for the main screen static buttons    **
*>*****
    05 PIC X(21) VALUE "010095103OK   OK   8".*> All captions must have at least
    05 PIC X(21) VALUE "012095103CAN CANCEL 8".*> leading and ONE trailing space
    05 PIC X(21) VALUE "014095103HLP  HELP 8".
    05 PIC X(21) VALUE LOW-VALUES.          *> Must be last entry
01 All-Buttons REDEFINES Button-FILLER.
    05 Button-Tbl OCCURS 25 TIMES
                        INDEXED BY Button-Idx.
        10 BT-Row PIC 9(3).      *> Screen row no where button resides
        10 BT-Col-Start PIC 9(3).  *> Screen column where button begins
        10 BT-Col-End PIC 9(3).   *> Screen column where button ends
        10 BT-ID.      *> Three-char button function, or...
```

15 BT-ID-Type	PIC X(1).	*> ...Letter "F", followed by...
15 BT-Function-No	PIC 9(2).	*> ...2-digit function button # (01-16)
10 BT-Caption	PIC X(8).	*> Fixed button caption text
10 BT-Caption-Size	PIC 9(1).	*> The size (chars) of the caption

Whenever the Switches-Screen is **ACCEPT**ed, and the **COB-STATUS-CODE** register has a value of **COB-SCR-LEFT-PRESSED** or **COB-SCR-RIGHT-PRESSED**, the **Buttons-Tbl** table will be searched to find out what button or field (if any) the user clicked.

The **Cursor-Coordinates** group will receive the six-digit cursor coordinates:

```

01 Cursor-Coordinates          PIC 9(6).  *> Locn of cursor on screen ACCEPTs
01 REDEFINES Cursor-Coordinates.
   05 CC-Row                    PIC 9(3).
   05 CC-Col                    PIC 9(3).
```

The **Cursor-Coordinates** data item was defined in **SPECIAL-NAMES** as the data item to receive the cursor coordinates when a mouse-click occurs. The table will be searched (sequentially, via **SEARCH**) for the first entry where **BT-Row = CC-Row** AND **BT-Col-Start <= CC-Col** AND **CC-Col <= BT-Col-End**. If no such entry was found, **MAIN** will ring the bell, signaling a user input error.

If such an entry is found, **MAIN** will then know what button was clicked, and the **BT-ID(Button-Idx)** field value in the table ("NOP", "F01"... "F15", "OK ", "CAN", "HLP") is ready-made for an **EVALUATE** statement to be able to take the desired action. The NOP fields represent the five data-entry fields on the screen.

There is a procedure in **GCic** - **015-Identify-Click** - whose job it is to perform the aforementioned searching. It also, in the instance where a function option button was clicked, populates a data-item named **Function-No** with the number of the function button (1-16).

When adding (or removing) entries in the **FILLER** portion of this table, don't forget to adjust the **OCCURS** value accordingly. This table will be "sanity-checked" during initialization and an incorrect **OCCURS** count will result in the following fatal error (the actual numbers may differ):

Fatal Error: 'Button-Tbl' OCCURS count is 0026, should be 0027

This is easily correctable by making the source code change the message recommends and recompiling **gcic.cbl**.

[TOP](#)

The "Function-Tbl" Table

This table exists to track which option is currently selected for each of the sixteen features.

```

01 Function-FILLER.              *> Function button characteristics
   05 Function-No PIC 9(2) VALUE 0.  *> Subscript of most-recently clicked function button
*> -----
*> Function 1 - Debugging
*> -----
   05 PIC 9(2) VALUE F1.          *> FT-Curr-Opt-No    (1)
   05 PIC 9(2) VALUE F1.          *> FT-Default-Opt-No(1)
   05 PIC 9(2) VALUE 0.          *> FT-Opt-Qty      (1)  Computed by 100-Initialization
*> -----
*> Function 2 - Run-time Err Checking
*> -----
   05 PIC 9(2) VALUE F2.          *> FT-Curr-Opt-No    (2)
   05 PIC 9(2) VALUE F2.          *> FT-Default-Opt-No(2)
   05 PIC 9(2) VALUE 0.          *> FT-Opt-Qty      (2)  Computed by 100-Initialization
*> -----
*> Function 3 - Generate Tracing Code
*> -----
   05 PIC 9(2) VALUE F3.          *> FT-Curr-Opt-No    (3)
   05 PIC 9(2) VALUE F3.          *> FT-Default-Opt-No(3)
   05 PIC 9(2) VALUE 0.          *> FT-Opt-Qty      (3)  Computed by 100-Initialization
*> -----
*> Function 4 - Compiler Output
*> -----
   05 PIC 9(2) VALUE F4.          *> FT-Curr-Opt-No    (4)
   05 PIC 9(2) VALUE F4.          *> FT-Default-Opt-No(4)
   05 PIC 9(2) VALUE 0.          *> FT-Opt-Qty      (4)  Computed by 100-Initialization
```

```

*> -----
*> Function 5 - Run After Compilation
*> -----
05 PIC 9(2) VALUE F5.          *> FT-Curr-Opt-No    (5)
05 PIC 9(2) VALUE F5.          *> FT-Default-Opt-No(5)
05 PIC 9(2) VALUE 0.           *> FT-Opt-Qty      (5)  Computed by 100-Initialization
*> -----
*> Function 6 - Source & Xref Listing
*> -----
05 PIC 9(2) VALUE F6.          *> FT-Curr-Opt-No    (6)
05 PIC 9(2) VALUE F6.          *> FT-Default-Opt-No(6)
05 PIC 9(2) VALUE 0.           *> FT-Opt-Qty      (6)  Computed by 100-Initialization
*> -----
*> Function 7 - Listing Produced By
*> -----
05 PIC 9(2) VALUE F7.          *> FT-Curr-Opt-No    (7)
05 PIC 9(2) VALUE F7.          *> FT-Default-Opt-No(7)
05 PIC 9(2) VALUE 0.           *> FT-Opt-Qty      (7)  Computed by 100-Initialization
*> -----
*> Function 8 - FUNCTION w/ Intrinsics
*> -----
05 PIC 9(2) VALUE F8.          *> FT-Curr-Opt-No    (8)
05 PIC 9(2) VALUE F8.          *> FT-Default-Opt-No(8)
05 PIC 9(2) VALUE 0.           *> FT-Opt-Qty      (8)  Computed by 100-Initialization
*> -----
*> Function 9 - Commands and Warnings
*> -----
05 PIC 9(2) VALUE F9.          *> FT-Curr-Opt-No    (9)
05 PIC 9(2) VALUE F9.          *> FT-Default-Opt-No(9)
05 PIC 9(2) VALUE 0.           *> FT-Opt-Qty      (9)  Computed by 100-Initialization
*> -----
*> Function 10 - Program Source Format
*> -----
05 PIC 9(2) VALUE F10.         *> FT-Curr-Opt-No    (10)
05 PIC 9(2) VALUE F10.         *> FT-Default-Opt-No(10)
05 PIC 9(2) VALUE 0.           *> FT-Opt-Qty      (10) Computed by 100-Initialization
*> -----
*> Function 11 - Truncate COMP to PIC
*> -----
05 PIC 9(2) VALUE F11.         *> FT-Curr-Opt-No    (11)
05 PIC 9(2) VALUE F11.         *> FT-Default-Opt-No(11)
05 PIC 9(2) VALUE 0.           *> FT-Opt-Qty      (11) Computed by 100-Initialization
*> -----
*> Function 12 - Optimization
*> -----
05 PIC 9(2) VALUE F12.         *> FT-Curr-Opt-No    (12)
05 PIC 9(2) VALUE F12.         *> FT-Default-Opt-No(12)
05 PIC 9(2) VALUE 0.           *> FT-Opt-Qty      (12) Computed by 100-Initialization
*> -----
*> Function 13 - Dump DATA DIV on Abort
*> -----
05 PIC 9(2) VALUE F13.         *> FT-Curr-Opt-No    (13)
05 PIC 9(2) VALUE F13.         *> FT-Default-Opt-No(13)
05 PIC 9(2) VALUE 0.           *> FT-Opt-Qty      (13) Computed by 100-Initialization
*> -----
*> Function 14 - Behavior & Standards
*> -----
05 PIC 9(2) VALUE F14.         *> FT-Curr-Opt-No    (14)
05 PIC 9(2) VALUE F14.         *> FT-Default-Opt-No(14)
05 PIC 9(2) VALUE 0.           *> FT-Opt-Qty      (14) Computed by 100-Initialization
*> -----
*> Function 15 - Pgm-Specific Switches
*> -----
05 PIC 9(2) VALUE F15.         *> FT-Curr-Opt-No    (15)
05 PIC 9(2) VALUE F15.         *> FT-Default-Opt-No(15)
05 PIC 9(2) VALUE 0.           *> FT-Opt-Qty      (15) Computed by 100-Initialization
*> -----
*> Function 16 - Save Temporary Files
*> -----
05 PIC 9(2) VALUE F16.         *> FT-Curr-Opt-No    (16)

```

```

05 PIC 9(2) VALUE F16.          *> FT-Default-Opt-No(16)
05 PIC 9(2) VALUE 0.            *> FT-Opt-Qty      (16) Computed by 100-Initialization
*> -----
*> Function 17 - ???
*> -----
*> 05 PIC 9(2) VALUE F17.          *> FT-Curr-Opt-No   (17)
*> 05 PIC 9(2) VALUE F17.          *> FT-Default-Opt-No(17)
*> 05 PIC 9(2) VALUE 0.            *> FT-Opt-Qty      (17) Computed by 100-Initialization
*> -----
*> Function 18 - ???
*> -----
*> 05 PIC 9(2) VALUE F18.          *> FT-Curr-Opt-No   (18)
*> 05 PIC 9(2) VALUE F18.          *> FT-Default-Opt-No(18)
*> 05 PIC 9(2) VALUE 0.            *> FT-Opt-Qty      (18) Computed by 100-Initialization
*> -----
*> Function 19 - ???
*> -----
*> 05 PIC 9(2) VALUE F19.          *> FT-Curr-Opt-No   (19)
*> 05 PIC 9(2) VALUE F19.          *> FT-Default-Opt-No(19)
*> 05 PIC 9(2) VALUE 0.            *> FT-Opt-Qty      (19) Computed by 100-Initialization
*> -----
*> Function 20 - ???
*> -----
*> 05 PIC 9(2) VALUE F20.          *> FT-Curr-Opt-No   (20)
*> 05 PIC 9(2) VALUE F20.          *> FT-Default-Opt-No(20)
*> 05 PIC 9(2) VALUE 0.            *> FT-Opt-Qty      (20) Computed by 100-Initialization
*> -----
05 PIC X(6) VALUE LOW-VALUES.    *> Must be the last entry
01 Function-Redef REDEFINES Function-FILLER.
05 PIC 9(2).
05 Function-Tbl OCCURS 17 TIMES.  *> # Supported funcs + 1 for end sentinel
10 FT-Curr-Opt-No PIC 9(2).        *> Current displayed option (varies)
10 FT-Default-Opt-No PIC 9(2).     *> Default displayed option (unchanging)
10 FT-Opt-Qty PIC 9(2).            *> Number of option choices (unchanging)
01 Function-Names REDEFINES Function-FILLER.
05 PIC 9(2).
05 F1-Debugging-Opt-No PIC 9(2).
05 PIC X(4).
05 F2-ErrCheck-Opt-No PIC 9(2).
05 PIC X(4).
05 F3-Tracing-Opt-No PIC 9(2).
05 PIC X(4).
05 F4-Output-Opt-No PIC 9(2).
05 PIC X(4).
05 F5-Run-Opt-No PIC 9(2).
05 PIC X(4).
05 F6-Listing-Opt-No PIC 9(2).
88 F6-No-Listing-Wanted VALUE 1.
88 F6-Landscape-Listing-Wanted VALUE 2.
88 F6-Portrait-Listing-Wanted VALUE 3.
88 F6-Exp-Srce-Listing-Wanted VALUE 4.
05 PIC X(4).
05 F7-Lister-Opt-No PIC 9(2).
88 Listing-Generated-By-GCic VALUE 1.
88 Listing-Generated-By-Cobc VALUE 2.
05 PIC X(4).
05 F8-Function-Opt-No PIC 9(2).
05 PIC X(4).
05 F9-CmdWarn-Opt-No PIC 9(2).
05 PIC X(4).
05 F10-SrceFmt-Opt-No PIC 9(2).
05 PIC X(4).
05 F11-Truncation-Opt-No PIC 9(2).
05 PIC X(4).
05 F12-Optimization-Opt-No PIC 9(2).
05 PIC X(4).
05 F13-Dump-Opt-No PIC 9(2).
05 PIC X(4).
05 F14-Config-Opt-No PIC 9(2).
05 PIC X(4).

```

```

05 F15-PrgSpcSwTchs-Opt-No PIC 9(2).
05 PIC X(4).
05 F16-SaveTemps-Opt-No PIC 9(2).
05 PIC X(4).
*> 05 F17-XXXX-Opt-No PIC 9(2).
*> 05 PIC X(4).
*> 05 F18-XXXX-Opt-No PIC 9(2).
*> 05 PIC X(4).
*> 05 F19-XXXX-Opt-No PIC 9(2).
*> 05 PIC X(4).
*> 05 F20-XXXX-Opt-No PIC 9(2).
*> 05 PIC X(4).
05 PIC X(6).

```

*> Lines up w/ end-of-table sentinel

Once a mouse button has been clicked, the **Button-Tbl** will be consulted to identify which button, if any, was clicked. Assuming it was one of the function buttons, the **Function-Tbl** comes into play as follows.

Depending on whether the option button was left-clicked or right-clicked, the value of the **FT-Curr-Opt-No** item for the "**Function-No**"th button will be incremented by 1 or decremented by 1, respectively. The **FT-Opt-Qty** items are there so the code knows how many options there are for each function (those values are calculated automatically by the **100-Initialization** routine). With that knowledge, the value will be set back to 1 if you left-click past the last option for the function, or will be set to the last option if you attempt to right-click to 0.

The **Function-Names** group provides a non-subscripted and more meaningful name for each function's current option number, for use when appropriate. As you can see, there are also four additional unused entries already coded for future expansion.

When adding (or removing) entries in the **FILLER** portion of this table, don't forget to adjust the **OCCURS** value accordingly. This table will be "sanity-checked" during initialization and an incorrect **OCCURS** count will result in the following fatal error (the actual numbers may differ):

Fatal Error: 'Function-Tbl' OCCURS count is 0015, should be 0016

This is easily correctable by making the source code change the message recommends and recompiling **gcic.cbl**.

[TOP](#)

The "Option-Tbl" Table

This table defines every option for every function, the caption that option will have on its feature button when it is selected, and the switches (if any) that option will pass to the cobc command.

As the user clicks option buttons, their captions will change. When **Function-No** gets set as a result of the click and the **FT-Curr-Opt-No(Function-No)** value gets incremented or decremented, this table will be consulted when it comes time to repaint the screen so that the new caption can be displayed. This is done via a **SEARCH ALL** using **Function-No** and **FT-Curr-Opt-No(Function-No)** as the search key against the **Option-Func** table.

```

01 Option-FILLER.
*> -----
*> Function 1 - Debugging
*> -----
05 PIC X(5) VALUE "0101X".
05 PIC X(29) VALUE " Treat 'D' Lines as Comments ".
05 PIC X(94) VALUE SPACES.
*> -----
05 PIC X(5) VALUE "0102 ".
05 PIC X(29) VALUE " 'D' Lines Will Be Compiled ".
05 PIC X(94) VALUE "-fdebugging-line".
*> -----
*> -----
*> Function 2 - Runtime Error Checking
*> -----
05 PIC X(5) VALUE "0201X".
05 PIC X(29) VALUE " Normal ".
05 PIC X(94) VALUE SPACES.
*> -----
05 PIC X(5) VALUE "0202 ".

```

```

05 PIC X(29)          VALUE "          Enhanced          ".
05 PIC X(94)          VALUE "-debug".
*> -----
*> -----
*> Function 3 - Generate Tracing Code
*> -----
05 PIC X(5)           VALUE "0301X".
05 PIC X(29)          VALUE "          No          ".
05 PIC X(94)          VALUE SPACES.
*> -----
05 PIC X(5)           VALUE "0302 ".
05 PIC X(29)          VALUE "          Yes - Procedures Only          ".
05 PIC X(94)          VALUE "-ftrace".
*> -----
05 PIC X(5)           VALUE "0303 ".
05 PIC X(29)          VALUE "Yes - Procedures & Statements".
05 PIC X(94)          VALUE "-ftraceall".
*> -----
.
.
.
*> -----
*> Must be the last entry
*> -----
05 PIC X(5)           VALUE LOW-VALUES.
05 PIC X(29)          VALUE LOW-VALUES.
05 PIC X(94)          VALUE LOW-VALUES.

01 Option-Redef       REDEFINES Option-FILLER.
05 Option-Tbl         OCCURS 59 TIMES
                      ASCENDING KEY Option-Func
                      INDEXED BY Option-Idx.

    10 Option-Func.
      15 OF-Func-No    PIC 9(2).  *> Function # (static) - 01-16
      15 OF-Opt-No     PIC 9(2).  *> Option # (static)   - 01-FT-Opt-Qty(n)
    10 Option-Text     PIC X(22).  *> Function button text for option (static)
    10 Option-Switch   PIC X(30).  *> "cobc" switches for this option (static)

```

Once the appropriate entry has been found, MAIN knows both the caption to be used for the currently-selected option and the option switches (if any) that need to be added to the cobc command that will be built and submitted.

The option numbers for any given function must be coded in strictly ascending and consecutive sequence (01, 02, 03, 04, ...). Failure to adhere to this will result in the following fatal error:

Fatal Error: Missing Option in Option-tbl (ffoo)

Where "ff" is the function number and "oo" is the option number, the combination of which is not present in the table.

When adding (or removing) entries in the **FILLER** portion of this table, don't forget to adjust the **OCCURS** value accordingly. This table will be "sanity-checked" during initialization and an incorrect **OCCURS** count will result in the following fatal error (the actual numbers may differ):

Fatal Error: 'Option-Tbl' OCCURS count is 0059, should be 0061

This is easily correctable by making the source code change the message recommends and recompiling **gcic.cbl**.

[TOP](#)

The "Temp-Files" Table

The process of producing the source and cross-reference listings relies heavily on files produced by the cobc compiler - some of which the compiler will delete before LISTER can "get its hands on them". Because of that, GCic will code the "-save-temps" switch on the cobc command it generates. Since the user will undoubtedly get annoyed as these files start to build up, GCic will clean up after itself and delete those files itself UNLESS the user *actually specified the* "-save-temps" switch in the "Extra cobc Switches" field on the screen. Also included are the various "gcic\$xxxx.txt" files that GCic creates in the process of doing its job.

The cleanup process is managed by this table. Comments in the code (shown here) explain the conventions used in

the table entries.

```
01 Temp-Files-FILLER.                                *> Files that we have to clean up on normal finish
*> *****
*> ** Entries containing "xxxxxx" will have the command-line program **
*> ** filename (minus extension) substituted for the "xxxxxx". Those **
*> ** entries with a TF-Wildcard value of "*" will be deleted using **
*> ** a "SYSTEM" command while all others will be deleted via the **
*> ** CBL_DELETE_FILE subroutine. **
*> *****
05 PIC X(21) VALUE "xxxxxx.c.l*.h". *> ..."C" header for local storage from "cobc"
05 PIC X(21) VALUE " gcic-info.txt". *> ..."cobc -i" output
05 PIC X(21) VALUE " gcic-list.txt". *> ..."cobc -list-reserved", etc. output
05 PIC X(21) VALUE " gcic-output.txt". *> ..."cobc -T|t [-ftsymbols]" output
05 PIC X(21) VALUE " gcic-source.txt". *> ..."cobc -T" source listing
05 PIC X(21) VALUE " gcic-symbols.txt". *> ..."cobc -ftsymbols" listing
05 PIC X(21) VALUE " xxxxxx.c". *> ..."C" source code from "cobc"
05 PIC X(21) VALUE " xxxxxx.c.h". *> ..."C" header file from "cobc"
05 PIC X(21) VALUE " xxxxxx.i". *> ...Expanded COBOL code from "cobc"
05 PIC X(21) VALUE " xxxxxx.o". *> ...Object code from "cobc"
05 PIC X(21) VALUE " xxxxxx.s". *> ...Assembler source from "cobc -S"
05 PIC X(21) VALUE LOW-VALUES. *> ...End-of-table sentinel
01 Temp-Files-Tbl REDEFINES Temp-Files-FILLER.
05 Temp-File OCCURS 12.
10 TF-Wildcard PIC X(1).
10 TF-Filename PIC X(20).
```

When adding (or removing) entries in the **FILLER** portion of this table, don't forget to adjust the **OCCURS** value accordingly. This table will be "sanity-checked" during initialization and an incorrect **OCCURS** count will result in the following fatal error (the actual numbers may differ):

Fatal Error: 'Temp-File' OCCURS count is 0012, should be 0011

This is easily correctable by making the source code change the message recommends and recompiling **gcic.cbl**.

[TOP](#)

Program-Specified Switches (PSS)

As seen back in the [Using GCic](#) section, this feature of GCic allows for specially-formatted comments to be included in GnuCOBOL programs. In addition to serving as documentation of compilation requirements these can also be collected by GCic and added to the cobc statement before it is submitted.

The **025-Scan-PSS** routine scans the [Options-Tbl](#) table, looking for entries with a SPACE in their **PSS-Exclude** field. Those will be the options having a single option switch defined for them. The complete set of PSS options accumulated during the **020-Is-Prog-A-Subprogram** scan of the source file supplied as the argument to GCic will be searched (via **INSPECT TALLYING**, not **SEARCH**) for an occurrence of the entry in the [Options-Tbl](#) table. If not found, **025-Scan-PSS** will move on.

The **025-Scan-PSS** routine is executed at two points during GCic's execution:

1. During initialization, immediately after **020-Is-Prog-A-Subprogram**. Switch matches found here will have the corresponding options selected in time for the first presentation of the screen to show their captions.
2. Any time the Pgm-Specified Switches function has its option changed from **Ignore** to **Honor**.

[TOP](#)

Locking

When you start GCic:

1. It generates a filename of **gcic\$lock\$xxxxxx**, where "xxxxxx" is the filename portion of the file that was specified on the command-line.
2. It then checks for the existence of that file (using **CBL_CHECK_FILE_EXIST**) in the same folder in which the file specified on the command-line resides.

3. If the file *does not exist*, it is created using an OPEN/CLOSE sequence.
4. If the file *does exist*, the **"xxxxxxx is already being compiled"** message is displayed for 3 seconds and GCic is terminated.

Immediately after the **ACCEPT** statement completes execution and has read the key/mouse input, the `gcic$lock$xxxxxx` file is deleted, thus releasing the "lock".

If a Ctrl-C or Ctrl-Break is issued while the **ACCEPT** is in progress, the lock file will NOT be deleted, necessitating that the lock file be manually deleted before that program can be compiled again.

[TOP](#)

The GCINFO Subprogram

This subroutine:

1. Executes a `cobc -i`, piping the output of that command to a file named `gcic-info.txt` (in the same folder as the file being compiled)
2. Reads the contents of that file, converting each record read to lowercase (to make it easier to look for specific words without worrying about case)
3. Parses each record read, extracting the first five (5) words of each record into `Token-1`, `Token-2`, ... , `Token-5`
4. Checks the extracted tokens, looking for certain key words, and then extracts the information it is looking for

The following table shows how a variety of `cobc -i` results would be mined for their configuration information. Words shown in **yellow** are the words GCINFO looks for while those in **green** are the ones providing the actual information.

cobc -i Output	Returned Results
<code>cobc (gnucobol) 3.2-dev.20220908</code>	GA-Version "3.2-dev.20220908"
<code>copyright (c) 2022 free software foundation, ...</code>	GA-Version-No 00000003
<code>license gplv3+: gnu gpl version 3 or later ...</code>	GA-Release-No 00000002
<code>this is free software; see the source for ...</code>	GA-Update-Major-No "dev "
<code>warranty; not even for merchantability or ...</code>	GA-Update-Minor-No 20220908
<code>written by keisuke nishida, roger while, ...</code>	GA-Release-Date "08SEP2022"
<code>built sep 08 2022 20:15:25</code>	GA-Build-Date "08SEP2022"
<code>packaged sep 08 2022 20:15:02 utc</code>	GA-Build-Env "?????????????????" *
<code>c version (microsoft) 1500</code>	GA-OS-Type 1 (i.e. Native Windows)
 build information	GA-Version "2.0.0"
<code>build environment : ??????????????????</code>	GA-Version-No 00000002
 <code>cobc (gnu cobol) 2.0.0</code>	GA-Release-No 00000000
<code>copyright (c) 2001,2002,2003,2004,2005,2006,2007 ...</code>	GA-Update-Major-No 00000000
<code>copyright (c) 2006-2012 roger while</code>	GA-Update-Minor-No 00000000
<code>copyright (c) 2009,2010,2012,2014 simon sobisch</code>	GA-Release-Date "11JUL2014"
<code>this is free software; see the source for copying ...</code>	GA-Build-Date "20JAN2014"
<code>warranty; not even for merchantability or fitness ...</code>	GA-Build-Env "i686-pc-cygwin"
<code>built jul 11 2014 07:13:23</code>	GA-OS-Type 2 (i.e. Windows/Cygwin)
<code>packaged jan 20 2014 07:40:53 utc</code>	GA-Version "3.2-dev.0"
<code>c version "4.8.3"</code>	GA-Version-No 00000003
 build information	GA-Release-No 00000002
<code>build environment : i686-pc-cygwin</code>	GA-Update-Major-No "dev "
 <code>cobc (gnucobol) 3.2-dev.0</code>	
<code>copyright (c) 2022 free software foundation, ...</code>	
<code>license gplv3+: gnu gpl version 3 or later ...</code>	
<code>this is free software; see the source for ...</code>	
<code>warranty; not even for merchantability or ...</code>	

written by keisuke nishida, roger while, ...

built **aug 15 2021** 19:53:08

packaged **aug 15 2021** 17:52:29 utc

c version "10.3.0"

build information

build environment : **x86_64-pc-linux-gnu**

cobc (glibcobol) **3.1.2.0**

copyright (c) 2020 free software foundation, ...

license gplv3+: gnu gpl version 3 or later ...

this is free software; see the source for ...

warranty; not even for merchantability or ...

written by keisuke nishida, roger while, ...

built **sep 24 2021** 12:31:22

packaged **dec 23 2020** 12:04:58 utc

c version "4.2.1 compatible apple llvm 11.0.3 ...

build information

build environment : **x86_64-apple-darwin19.6.0**

cobc (glibcobol) **3.1.2.0**

copyright (c) 2020 free software foundation, ...

license gplv3+: gnu gpl version 3 or later ...

this is free software; see the source for ...

warranty; not even for merchantability or ...

written by keisuke nishida, roger while, ...

built **jun 28 2022** 10:38:00

packaged **dec 23 2020** 12:04:58 utc

c version (mingw) "6.3.0"

build information

build environment : **i686-pc-mingw32**

GA-Update-Minor-No 00000000

GA-Release-Date "15AUG2021"

GA-Build-Date "15AUG2021"

GA-Build-Env "x86_64-pc-linux-gnu"

GA-OS-Type 3 (i.e. *NIX)

GA-Version "3.1.2.0"

GA-Version-No 00000003

GA-Release-No 00000001

GA-Update-Major-No 00000002

GA-Update-Minor-No 00000000

GA-Release-Date "24SEP2021"

GA-Build-Date "23DEC2020"

GA-Build-Env "x86_64-apple-darwin19.6.0"

GA-OS-Type 4 (i.e. MacOS)

GA-Version "3.1.2.0"

GA-Version-No 00000003

GA-Release-No 00000001

GA-Update-Major-No 00000002

GA-Update-Minor-No 00000000

GA-Release-Date "28JUN2022"

GA-Build-Date "23DEC2020"

GA-Build-Env "i686-pc-mingw32"

GA-OS-Type 5 (i.e. Windows/MinGW)

* Being only able to obtain a sample of **cobc -v** output for a native Windows build, the actual "build environment" string for such remains a mystery.

[TOP](#)

The LISTER Subprogram

This program, CALLED by GCic only when the user has elected to create a source and cross-reference listing AND the user chose GCic (instead of cobc) as the creator of those listings. There is NO interaction with the user in this program; even the error messages generated by LISTER are actually *delivered* to the user by the GCic main program via an **Exit-Code** value passed back to GCic.

[TOP](#)

The "All-Verbs-Tbl" Table

LISTER obtains its list of reserved words directly from cobc via the **-list-reserved**, **list-intrinsic**, and **-list-mnemonic** switches, and uses the reports produced by cobc when those switches are used to populate the [Reserved Word Table](#). Unfortunately, cobc doesn't distinguish verbs (actual statement names) from any of the other reserved words. Since LISTER needs to know any time a new **PROCEDURE DIVISION** statement has been encountered while it's generating the cross-reference listing, this table is used to flag those entries in the [Reserved Word Table](#) with a **RWT-Type-Code** value of "V".

```
01 ALL-Verbs.                                *> All GnuCOBOL verbs
   05 PIC X(32)                               VALUE "ACCEPT".
   05 PIC X(32)                               VALUE "ADD".
   05 PIC X(32)                               VALUE "ALLOCATE".
```

```

05 PIC X(32)          VALUE "ALTER".
05 PIC X(32)          VALUE "CALL".
05 PIC X(32)          VALUE "CANCEL".
05 PIC X(32)          VALUE "CLOSE".
05 PIC X(32)          VALUE "COMMIT".
05 PIC X(32)          VALUE "COMPUTE".
05 PIC X(32)          VALUE "CONTINUE".
05 PIC X(32)          VALUE "DELETE".
05 PIC X(32)          VALUE "DISABLE".
05 PIC X(32)          VALUE "DISPLAY".
05 PIC X(32)          VALUE "DIVIDE".
05 PIC X(32)          VALUE "ELSE".
05 PIC X(32)          VALUE "END".      *> Pseudo-verb to pick up "END PROGRAM"
05 PIC X(32)          VALUE "ENABLE".
.
.
.
05 PIC X(32)          VALUE "START".
05 PIC X(32)          VALUE "STOP".
05 PIC X(32)          VALUE "STRING".
05 PIC X(32)          VALUE "SUBTRACT".
05 PIC X(32)          VALUE "SUPPRESS".
05 PIC X(32)          VALUE "TERMINATE".
05 PIC X(32)          VALUE "TRANSFORM".
05 PIC X(32)          VALUE "UNLOCK".
05 PIC X(32)          VALUE "UNSTRING".
05 PIC X(32)          VALUE "USE".
05 PIC X(32)          VALUE "WHEN".
05 PIC X(32)          VALUE "WRITE".
05 PIC X(32)          VALUE "XML".      *> Only GENERATE
05 PIC X(32)          VALUE LOW-VALUES. *> Must be last entry
01 All-Verbs-Tbl
05 Verb-Name          REDEFINES All-Verbs.
                       OCCURS 61 TIMES
                       PIC X(32).

```

When adding (or removing) entries in the **FILLER** portion of this table, don't forget to adjust the **OCCURS** value accordingly. This table will be "sanity-checked" during initialization and an incorrect **OCCURS** count will result in the following fatal error (the actual numbers may differ):

Fatal Error: 'All-Verbs-Tbl' OCCURS count is 0061, should be 0059

This is easily correctable by making the source code change the message recommends and recompiling **gcic.cbl**.

[TOP](#)

The "Buzzwords-Tbl" Table

While LISTER is parsing a program during the process of creating a cross-reference listing, some reserved words unimportant to the generation of the xref listing need to be ignored by the parser, if they are found. This table contains those words, and is used to flag [Reserved Word Table](#) entries as "buzzwords" to be ignored.

```

01 Buzzwords. *> Reserved words ignored to simplify parsing
05 PIC X(32)          VALUE "ADDRESS".
05 PIC X(32)          VALUE "ARE".
05 PIC X(32)          VALUE "AS".
05 PIC X(32)          VALUE "AWAY-FROM-ZERO".
05 PIC X(32)          VALUE "CHARACTERS".
05 PIC X(32)          VALUE "IN".
05 PIC X(32)          VALUE "IS".
05 PIC X(32)          VALUE "KEY".
05 PIC X(32)          VALUE "MODE".
05 PIC X(32)          VALUE "NEAREST-AWAY-FROM-ZERO".
05 PIC X(32)          VALUE "NEAREST-EVEN".
05 PIC X(32)          VALUE "NEAREST-TOWARD-ZERO".
05 PIC X(32)          VALUE "NOT".
05 PIC X(32)          VALUE "OF".
05 PIC X(32)          VALUE "OPTIONAL".
05 PIC X(32)          VALUE "PROCEED".

```

```

05 PIC X(32)          VALUE "PROHIBITED".
05 PIC X(32)          VALUE "ROUNDED".
05 PIC X(32)          VALUE "SEQUENCE".
05 PIC X(32)          VALUE "STATUS".
05 PIC X(32)          VALUE "TOWARD-GREATER".
05 PIC X(32)          VALUE "TOWARD-LESSER".
05 PIC X(32)          VALUE "TRUNCATION".
05 PIC X(32)          VALUE "WITH".
05 PIC X(32)          VALUE LOW-VALUES. *> Must be last entry
01 Buzzwords-Tbl     REDEFINES Buzzwords.
05 Buzzword          OCCURS 25 TIMES
                     PIC X(32).

```

When adding (or removing) entries in the **FILLER** portion of this table, don't forget to adjust the **OCCURS** value accordingly. This table will be "sanity-checked" during initialization and an incorrect **OCCURS** count will result in the following fatal error (the actual numbers may differ):

Fatal Error: 'Buzzwords-Tbl' OCCURS count is 0025, should be 0030

This is easily correctable by making the source code change the message recommends and recompiling **gcic.cbl**.

[TOP](#)

The "Files-And-Statuses-Tbl" Table

Whenever a program references a file, any **FILE STATUS** or **SORT STATUS** item associated with that file will be updated with the two-digit status code. This table exists so the cross-reference listing can reflect those implied updates.

```

01 Files-And-Statuses. *> Relates files in parsed program
                      *> w/ FILE STATUS items defined for them

05 FAST-Sub          USAGE BINARY-LONG *> Subscript into table
                      UNSIGNED
                      VALUE 0.
05 FAST-Hold-Name     PIC X(63). *> Working save area
05 Files-And-Statuses-Tbl OCCURS FASTSIZE TIMES.
10 FAST-Filename      PIC X(63). *> File-name
10 FAST-Status        PIC X(63). *> FILE-STATUS item (if > 1,
                      *> multiple table entries)

```

This table is set to a fixed size by a configuration constant in the **gcic-setup.cpy** proc. If the table fills up at runtime the following fatal error will result (the actual numbers may differ):

Fatal Error: 'Files-And-Statuses-Tbl' is full - Increase FASTSIZE

This is easily correctable by making the source code change the message recommends (to FASTSIZE in **gcic-setup.cpy**) and recompiling **gcic.cbl**.

[TOP](#)

The "Records-And-Files-Tbl" Table

Whenever a program reads a file, the various records defined for that file are updated. Any time a program writes a record to a file, not only is the record data item referenced, by the file contents are updated. This table exists so the cross-reference listing can reflect those implied references and updates.

```

01 Records-And-Files. *> Relates files in parsed program with the records defined for them
05 RAFT-Sub          USAGE BINARY-LONG *> Subscript into table
                      UNSIGNED
                      VALUE 0.
05 Records-And-Files-Tbl OCCURS RAFTSIZE TIMES.
10 RAFT-Filename      PIC X(63). *> Filename
10 RAFT-Recordname    PIC X(63). *> Record name (Multiple recs, multiple entries)

```

This table is set to a fixed size by a configuration constant in the `gcic-setup.cpy.cpy` proc. If the table fills up at runtime the following fatal error will result (the actual numbers may differ):

Fatal Error: 'Records-And-Files-Tbl' is full - Increase RAFTSIZE

This is easily correctable by making the source code change the message recommends (to RAFTSIZE in `gcic-setup.cpy`) and recompiling `gcic.cbl`.

[TOP](#)

The "Reserved-Word-Tbl" Table

This table contains the various reserved words of the language dialect the program compilation used (-std). After being loaded, the table will be sorted according to it's description to allow it to be searched via `SEARCH ALL`. While GCic makes the distinction between different classes of reserved words documented in the comments, it is not necessarily using all of the `RWT-Type-Code` values - just call it preparation for future possibilities.

```
01 Reserved-Words.  *> Populated using the output of "cobc" run
                    *> with the various "-list-xxxx" switches
05 Reserved-Word-Tbl OCCURS RWTSIZE TIMES
                    ASCENDING KEY RWT-Word
                    INDEXED BY RWT-Idx.
10 RWT-Type-Code   PIC X(1).    *> "B": A buzzword (to be ignored)
                                *> "D": A device
                                *> "F": A feature
                                *> "I": An intrinsic (function)
                                *> "M": A device mnemonic
                                *> "R": Just a plain 'ol reserved word
                                *> "V": A verb
                                *> "W": A switch name

10 RWT-Word        PIC X(32).
```

This table is set to a fixed size by a configuration constant in the `gcic-setup.cpy` proc. If the table fills up at runtime the following fatal error will result (the actual numbers may differ):

Fatal Error: 'Reserved-Word-Tbl' is full - Increase RWTSIZE

This is easily correctable by making the source code change the message recommends (to RWTSIZE in `gcic-setup.cpy`) and recompiling `gcic.cbl`.

[TOP](#)

The "Stack-Entry" Table

The `ADD`, `SUBTRACT`, `MULTIPLY`, `DIVIDE`, `RELEASE`, `REWRITE`, and `WRITE` statements present a challenge to producing a cross-reference listing that differentiates between simple references and updates because all of these statements allow the use of a clause that reverses initial assumptions made the reference/update status of user-defined items named on the statements.

For the arithmetic statements, it's the presence or absence of a `GIVING` clause. Observe these two `ADD` statements:

```
ADD A TO B
ADD A TO B GIVING C
```

In the first case, `A` is referenced while `B` is updated. In the second, both `A` and `B` are referenced and it's `C` that's actually updated. With the I/O statements, it is the `FROM` clauses that cause problems:

```
WRITE A
WRITE A FROM B
```

The first statement references `A`, while the second references `B` and *updates* `A`

The solution to this problem, is the following:

```
01 Stack.
05 Stack-Sub      PIC 9(4).
05 Stack-Entry    OCCURS STACKSIZE TIMES
```

PIC X(244). *> Must be the same size as Sort-Work-Rec

The data collection portion of the cross-referencing process actually runs as the **INPUT PROCEDURE** of a **SORT** statement, with the **OUTPUT PROCEDURE** being the actual report production. During the **INPUT PROCEDURE**, references and updates are usually easily differentiated from each other and are immediately released to the sort as they are found. In the cases of these problem-statements, the sort records are still built, for whatever mode would be the case if no **GIVING** or **FROM** were present, but are then "pushed" onto the stack rather than getting released to the sort. If a **GIVING** or **FROM** is found before the end of the statement, the entries are popped off the stack, one at a time, *reversed* so that references become updates and updates become references, and are then released to the sort. If the statement ended before a **GIVING** or **FROM** was found, the entries on the stack each get popped and released to the sort as-is.

This table is set to a fixed size by a configuration constant in the **gcic-setup.cpy** proc. If the table fills up at runtime the following fatal error will result (the actual numbers may differ):

Fatal Error: 'Stack-Entry' is full - Increase STACKSIZE

This is easily correctable by making the source code change the message recommends (to **STACKSIZE** in **gcic-setup.cpy**) and recompiling **gcic.cbl**.

[TOP](#)

The "Symbol-Table" Table

```
01 Symbol-Table.
  05 ST-Sub          PIC 9(4).          *> Subscript into "ST-Entry"
  05 ST-Entry        OCCURS STSIZE TIMES *> Entry for a non-FILLER, 66, 77, 78 item
                        INDEXED BY ST-Idx. *> Second way to point to "ST-Entry"
      10 ST-Level     PIC X(2).          *> Data item's level number
      10 ST-Name      PIC X(30).         *> 1st 30 chars of data item's name
      10 ST-Name-UC   PIC X(30).         *> UPPER-CASE copy of "ST-Name"
```

This structure is used during LISTER's creation of the cross-reference listing. The structure is loaded from the symbol table report built by cobc (-ftsymbols). Here's a sample group item:

```
01 Employee-Record.
  05 ER-Name.
    10 ER-First      PIC X(15).
    10 ER-Last       PIC X(20).
  05 ER-Address.
    10 ER-Street     PIC X(20).
    10 ER-City       PIC X(15).
    10 ER-State      PIC X(2).
    10 ER-Zip.
      15 ER-Zip-5    PIC 9(5).
      15 ER-Zip-4    PIC 9(4).
```

Here's what the portion of the table that pertains to the above structure would look like:

```
..... Previous Structure...
01Employee-Record      EMPLOYEE-RECORD
05ER-Name              ER-NAME
10ER-First             ER-FIRST
10ER-Last              ER-LAST
05ER-Address           ER-ADDRESS
10ER-Street            ER-STREET
10ER-City              ER-CITY
10ER-State             ER-STATE
10ER-Zip               ER-ZIP
15ER-Zip-5             ER-ZIP-5
15ER-Zip-4             ER-ZIP-4
01... Next Structure ...
```

Consider the statement **MOVE SPACES TO ER-Zip**, which occurs at line 677 of a compilation group for which source and cross-reference listings, prepared by GCic, have been requested. This results in the generation of a "677*" xref for "ER-Zip".

One of the design goals for V2.0 of GCic was to propagate update xrefs both upward and downward in the tree structures represented by group items. So, the **MOVE** will also generate additional "677*" xrefs for other items in the

structure by applying the following algorithm.

1. First, determine if the data item that just had an UPDATE cross-reference entry created for it (ER-Zip) is part of a group item. That can be determined by searching the ST-Entry table for a match between the ST-Name-UC value and the uppercase version of the data item in question. If it cannot be found, there won't be any extra update xrefs produced. Why do an uppercase-to-uppercase comparison? That way there will not be a problem if the programmer is inconsistent with his/her use of case.
2. Assuming a match WAS found, the item found in the symbol table, hereafter known as the BASE ITEM, will have its location in the table saved for later.
3. Now, work *backwards* in the symbol table, ignoring entries with level numbers EQUAL TO or GREATER THAN that of the BASE ITEM. If an item with a level number STRICTLY LESS THAN that of the base item is found (meaning this item is *higher* than the BASE ITEM in the group hierarchy) an update reference for that item is generated, the item is made the *new* BASE ITEM, and - if the level number is **NOT** 01 - THIS STEP IS REPEATED (otherwise fall into the next step).
4. Finally, proceed *forward* through the subtree rooted by the original BASE ITEM (this why we saved its location earlier). Each item found with a level number STRICTLY GREATER THAN that of the original BASE ITEM has an update xref generated for it because it belongs to the original BASE ITEM. Continue moving forward until either a blank entry is found in the symbol table (end of table) or an entry with a level number LESS THAN OR EQUAL TO that of the original BASE ITEM is encountered.

When applied to the "MOVE" statement mentioned earlier, this algorithm will generate additional "677*" xrefs for "ER-Address", "Employee-Address", "ER~Zip-5", and "ER-Zip-4".

This table is set to a fixed size by a configuration constant in the `gcic-setup.cpy` proc. If the table fills up at runtime the following fatal error will result (the actual numbers may differ):

Fatal Error: 'ST-Entry' is full - Increase STSIZE

This is easily correctable by making the source code change the message recommends (to STSIZE in `gcic-setup.cpy`) and recompiling `gcic.cbl`.

[TOP](#)

Modifying GCic

Before attempting to modify any of the code in GCic, you should be aware of the following design principles if you plan on sharing those changes with the community.

1. Wherever possible, data items in each of the **DATA DIVISION** sections have been defined in alphabetical order of their top-level data names.
2. **GO TO** statements have been avoided wherever possible. The only place they are used is within the Finite State Machine in LISTER.
3. When procedural **PERFORM** statements are used, they always reference **SECTIONS**. What few **GO TO** statements exist only reference paragraphs within the same **SECTION** as they are.
4. Think of the procedural structure of all programs as a tree similar to an organization chart, with **000-Main** always being the root of the tree (that is, the one at the very top). The sections at the next layer - numbered 100, 200, 300, and so on, represent the top-levels of up to nine common threads of activity. In the LISTER program, for example:
 - All initialization functions belong to the subtree rooted by **100-Initialization**.
 - All processing responsible for creating the reserved-word list belongs to the subtree rooted by **200-Build-Keyword-Table**.
 - The subtree rooted by **300-Produce-Source-Listing** is responsible for generating the program source listing
 - All source-code parsing takes place in the subtree rooted by **400-Tokenize-Source**.
 - The production of the cross-reference listing occurs in the subtree rooted by **500-Produce-Xref-Listing**.
5. Those sections of code that could be **PERFORMED** by multiple procedures, maybe even from multiple subtrees, are numbered **0nn**.
6. All procedures are coded in sequence of the numeric component of their names.
7. Each non-trivial program contains visual documentation of their **PROCESS TREE**, showing what procedures exist and what other procedures they **PERFORM**. Please keep these current as you make any changes.

8. Each procedure contains documentation stating that procedure's role. These too should be kept current.

[TOP](#)

Testing GCic

Should you decide to make changes to GCic or any of it's subprograms, you may find the built-in debugging capabilities useful. This feature will allow you to add debugging switches to the GCic command-line, **after** the name of the file you are compiling. Debugging information will then be written to SYSERR (or STDERR) for you to review. To activate these features, you will need to either:

1. Recompile using the command `cobc -x -fdebugging-line GCic.cbl`, or...
2. Recompile GCic using GCic, setting the **Debugging** feature to the **Compile 'D' Lines** option.

With that done, the command used to debug your GCic changes would be:

```
path\GCic yourprogfile switch-1 [ switch-n ]... 2>filename
```

All debugging output will be piped to *filename*; each line of output will be prefixed with the name of the switch that caused it. This is especially useful if your text editor has the ability to hide lines matching (or not matching) a mask!

Command-Line Switch Name	Switch Data-Item Name	Program(s)	Description
ALL	None	All	Turns on every debugging switch.
INFO	INFO-Sw	GCINFO	Displays the "thought process" as GCINFO acquires the information it seeks.
SOURCE	SOURCE-Sw	LISTER	Displays the handling of program source during the process of generating the source code listing.
SPLIT	SPLIT-Sw	LISTER	Displays the splitting of the cobc output file into the formatted sourcencode and symbol table files.
SUB	SUB-Sw	GCic	Displays the "thought process" as GCic determines if the to-be-compiled program is a subroutine or not (020-Is-Prog-A-Subprogram).
TRACE	TRACE-Sw	All	Generates a procedure-entry trace of all procedures in all programs.
USER	USER-Sw	GCic	Logs all mouse-clicks, action key-presses (Enter, Esc, PgUp, PgDn), and the contents of the input fields.
XALL	None	LISTER	Turns on all "X..." switches.
XFAST	XFAST-Sw	LISTER	Displays Files-And-Statuses-Tbl (FAST) entries as they are saved.
XFSM	XFSM-Sw	LISTER	Traces the state-by-state, character-by-character operation of the finite state machine (FSM) that parses GnuCOBOL programs while a cross-reference listing being generated.
XPARSE	XPARSE-Sw	LISTER	Produces diagnostic displays of information documenting the "thought process" the parser is going through while identifying and properly recognising the user-defined data names found in the FSM output.
XRAFT	XRAFT-Sw	LISTER	Displays Records-And-Files-Tbl (RAFT) entries as they are saved.
XREAD	XREAD-Sw	LISTER	Displays program expanded source code as it is being read and having its contents prepared for parsing. The records will be dumped AFTER the transformation they undergo to be prepared for parsing has taken place.
XTOKEN	XTOKEN-Sw	LISTER	Displays the recognized syntactical tokens that are generated by the finite state machine and parser.
XWORDS	XWORDS-Sw	LISTER	Displays the Reserved-Word-Tbl once it's been loaded, sorted, and tailored.

These switches are defined in the `gcic-setup.cpy` proc via the following structure:

```
DEBUG D01 Debug-Switches.
```

DEBUG D	05	INFO-Sw	PIC 9(1).
DEBUG D	05	SOURCE-Sw	PIC 9(1).
DEBUG D	05	SPLIT-Sw	PIC 9(1).
DEBUG D	05	SUB-Sw	PIC 9(1).
DEBUG D	05	TRACE-Sw	PIC 9(1).
DEBUG D	05	USER-Sw	PIC 9(1).
DEBUG D	05	X-OPTIONS.	
DEBUG D	10	XFAST-Sw	PIC 9(1).
DEBUG D	10	XFSM-Sw	PIC 9(1).
DEBUG D	10	XPARSE-Sw	PIC 9(1).
DEBUG D	10	XRAFT-Sw	PIC 9(1).
DEBUG D	10	XREAD-Sw	PIC 9(1).
DEBUG D	10	XREF-Sw	PIC 9(1).
DEBUG D	10	XTOKEN-Sw	PIC 9(1).
DEBUG D	10	XWORDS-Sw	PIC 9(1).

[TOP](#)

Summary of Documentation Changes

September, 2022

Original publication

[TOP](#)

Summary of Software Changes

The source code to GCic has grown to over 6000 lines of code, between all seven programs. In effort to reduce "clutter" in the code, the summary of documentation changes will now be placed in this document.

Legend to initials used:

GLC - Gary L Cutler

VBC - Vincent B Coen

GC0922 - V2.0 (RC2) - GLC - September, 2022

1. All GCic components are now named entirely in lower-case. So, "GCic.cbl" becomes "gcic.cbl", "GCic-SETUP.cpy" becomes "gcic-setup.cpy", and "GCic-README.html" becomes "gcic-readme.html".
2. The GCINFO subroutine has been enhanced to auto-detect the operating environment of GCic. The **OS** configuration constant still exists, but now expects a numeric integer value in the range 0 to 5. A setting of 0 (the default setting) indicates auto-detection should be performed while 1-5 defines a specific environment of Native Windows, Windows/Cygwin, *NIX, MacOS, and Windows/MinGW, respectively. These values of 1-5 should be resorted to only if auto-detection fails to identify the environment.
3. The **resize** command requires the installation of X11, and the usage of an xterm-compliant terminal/console emulator. While not a problem for MacOS users, where X11 is standard, this does pose a problem for *NIX users that have chosen not to install X11 on their systems. A new configuration constant - **X11** - has been introduced into **gcic-setup.cpy** so that *NIX users may turn off the attempt to use **resize**.
4. Related to the previous point, another configuration constant - **RESIZECMD** - has been introduced into **gcic-setup.cpy** so that a user-specified alternative command (or series of commands, separated by ";", "||", or "&&") may be specified to perform the terminal/console window resizing. The **\$SET** statement for this may not exceed the GnuCOBOL statement length in effect when **gcic-setup.cpy** is COPYed.
5. GCic temporary files formerly named "gcic\$xxx" are now named "gcic-xxx".
6. A bug that prevented PROGRAM-IDs of programs from appearing in page headings of GCic-generated source listings was corrected. The first page of a program will NOT have a PROGRAM-ID ON THE FIRST PAGE OF its listing unless the PROGRAM-ID statement is the first line of the program, but all subsequent pages will.

V2.0 (RC1) - GLC - June-September, 2022

A nearly complete re-write to utilize the latest features of GnuCOBOL V3+:

1. Fixed a problem where unnecessary "could not delete files" messages were being issued.
2. The GnuCOBOL version and package date is now determined automatically at runtime - no need to update GCic just for these things (see subroutine GCINFO).

3. The reserved word list (needed for the cross-reference listing) is now determined automatically at runtime - no need to update GCic for new GnuCOBOL releases unless new verbs capable of changing the contents of a data-item or file are introduced.
4. Introduced a built-in "Help" feature (which opens this document in your default web browser).
5. No more function keys! use the mouse now to click on buttons to change compilation options, trigger compilations, quit without compiling, and invoke help.
6. Introduced the ability to specify **COBCPY COPY Libraries** and additional arguments (for multiple source files, C files, object code files).
7. Added new choices to some already existing options.
8. Introduced the ability to easily change color schemes.
9. The "-save-temps[=folder]" option, if specified in the "Extra Options" field now is safe to use even if you are generating the source/xref listing.
10. Updated the source listing to include COPY proc contents *retaining comments and formatting*.
11. The "const-set-1.cpy" proc has been renamed to "**gcic-setup.cpy**".
12. Added cross-reference support for JSON PARSE and JSON GENERATE.
13. Added cross-reference support for XML PARSE and XML GENERATE.
14. Added cross-reference support for READ/RETURN... INTO (this was missing from the first version of GCic).
15. Updated the cross-reference listing to accommodate 63-character user-defined names.
16. You can now configure the characters used in cross-reference listings to flag definitions, updates, and references - see the **gcic-setup.cpy** proc.
17. A built-in debugging feature (NOT A GENERAL-PURPOSE DEBUGGER) is now included - see the [Testing GCic](#) topic and the documentation for subroutines DBGCOL, DBGKWV, and DBGTXT for details. This feature is intended for "tinkerers" who might like to make their own enhancements to GCic.
18. The format of GCic source code is now VARIABLE.
19. Because of the addition of so many new options (optimization, data division postmortem dumps, **COBCPY COPY Libraries**, run-time error checking, and a new user interface paradigm), it became necessary to raise the screen size from 80x24 to 35x106.
20. GCic now automatically resizes its window to fit the new screen size (Windows cmd.exe and Windows/MinGW cmd.exe); the code to do the same for Cygwin, MacOS, and *NIX has been written, and is included, but is currently untested. This will probably NOT work with builds of GnuCOBOL that utilize the so-called "wingui" builds of PDCurses (since there's no cmd.exe) but you are welcome to try.
21. GCic no longer terminates if you get compilation errors, but rather informs you the compilation failed and invites you to correct the errors and click (or press the Enter key) to recompile. This will continue until you correct the last error and get a clean compilation or until you decide to click the button (or press the Esc key). All input-field data you've entered and option settings you've selected are remembered until you quit GCic; you are welcome, of course, to change any of them if you wish.
22. Comments such as ***> COBC Switches: -facucomment -fwrite-after -fsingle-quote**, which make great documentation for someone who might need to compile a program, can now be recognized by GCic and used to put the specified switches on the **cobc** command. See the [Program-Specified Switches](#) topic for more information.

VC0820 - VBC - August, 2020

Updated compiler to 3.1 July2020 and the copyright.

VC1217 - VBC - December, 2017

Update compiler version to v3.0 24DEC2017, and copyright to 2018 (in 3 places).

VC0717 - VBC - July, 2017

Replaced compile param 'intrinsic=all' with 'intrinsics=all'. Changed mod detail inits for Gary from GCL to GLC. Update version printed to 2.2 20JUL2017. Should really get this from the compiler if avail?

VC0617 - VBC - June, 2017

Remove the Blinking in menu screen as uncomfortable. Update version printed to 2.2 30JUN2017. Move the system constant settings to a copy file named "const-set-1.cpy" in case GCic is updated. Added SET ENVIRONMENT "COB_EXIT_WAIT" TO "0" to 100-Initialization section.

GC0314 - GLC - March, 2014

Fixed a problem where 1st char of 1st token on a line is lost if >>SOURCE MODE IS FREE is in effect and the 1st character is non-blank.

GC0114 - GLC - January, 2014

Introduce a "Press ENTER to Close" action after running the compiled program in the compiler window (F4).

GC1213 - GLC - December, 2013

Updated for 23NOV2013 version of GnuCOBOL 2.1; Stop showing INTRINSIC functions as if they were identifiers in the XREF listing; Flag all CALL argument references with a "C" rather than "*" because they aren't necessarily altered; Fixed assorted formatting bugs; DOWNWARD COMPATIBLE WITH GNUCOBOL 2.0 SYNTAX.

GC1113 - GLC - November, 2013

Edited to support the change of "OpenCOBOL" to "GnuCOBOL"

GC0313 - GLC - March, 2013

Expand the source code record from 80 chars to 256 to facilitate looking for "LINKAGE SECTION" in a free-format file. Fixed XREF problem where the first procedure name defined in the PROCEDURE DIVISION lacks a "Defined" line number.

GC0712 - GLC - July, 2012

Replaced all switches with configuration settings; Tailored for 11FEB2012 version of GNU COBOL 2.0; Reformatted screen layout to fit a 24x80 screen rather than a 25x81 screen and to accommodate shell environments having only F1-F12 (like `terminal` in MACOS); Fully tested with MACOS (required a few alterations); Expanded both extra-options and runtime-arguments areas to 2 lines (152 chars total) each; Added support for MF/IBM/BS2000 listing-control directives EJECT,SKIP1,SKIP2,SKIP3 (any of these in copybooks will be ignored).

GC0711 - GLC - July, 2011

Tailored for 29APR2011 version of OpenCOBOL 2.0

GC1010 - GLC - October, 2010

Corrected several problems reported by Vince Coen:

1. Listing/Xref wouldn't work if "-l" additional cobc switch specified.
2. Programs coded with lowercase reserved words did not get parsed properly when generating listing and/or xref reports.
3. Reliance on a TEMP environment variable caused errors when generating listing and/or xref reports in a session that lacks a TEMP variable. As a result of this change, GCic no longer runs a second "cobc" when generating listing and/or xref reports. Instead, a "-save-temps" (without "=dir") specified in the EXTRA options field will be ignored. A "-save-temps=dir" specified in the EXTRA options field will negate both the XREF and SOURCE opts, if specified.

GC0710 - GLC - July, 2010

Handle duplicate data names (i.e. "CORRESPONDING" or qualified items) better; ignore "END PROGRAM" recs so program name doesn't appear in XREF listing.

GC0410 - GLC - April, 2010

Introduced the cross-reference and source listing features. Also fixed a bug in EXTRA switch processing where garbage will result if more than the EXTRA switch is specified.

GC0310 - GLC - March, 2010

Virtualized the key codes for S-F1 thru S-F7 as they differ depending upon whether PDCurses or NCurses is being used. [EDITORIAL NOTE: Somehow, this got lost down the line...it got permanently fixed in V2.0, but then became irrelevant when V2.0 dropped the use of F-keys]

GC0909 - GLC - September, 2009

Updated to work on Cygwin/Linux as well as MinGW.

GC0809 - GLC - August, 2009

Add a SPACE in front of command-line args when executing the just-compiled program. Add a SPACE after the "-ftraceall" switch when building cobc command.

GC0709 - GLC - July, 2009

When "EXECUTE" is selected, a "FILE BUSY" error will still cause the (old) executable to be launched. Also, the "EXTRA SWITCHES" field is being ignored. Changed the title bar to lowlighted reverse video & the message area to highlighted reverse-video.

GC0609 - GLC - June, 2009

Don't display compiler messages file if compilation is successful. Also don't display messages if the output file is busy (just put a message on the screen, leave the OCic screen up & let the user fix the problem & resubmit.

GC0609 - GLC - June, 2009

Initial release.

Thank You

I would like to thank you for downloading GCic and giving it a look. I hope you find it useful.

I would also like to thank the GnuCOBOL team for developing and supporting such an amazing product.

Finally, special thanks go to Vince B. Coen, who's "cobxref" tool has proven so valuable to so many of us. Vince - thank you too for the suggestions you made during the development of V2.0 of GCic.

If you have any questions, comments, or would like to report a bug, you can contact me through the [Sourceforge GnuCOBOL support site](#).

[TOP](#)