



Buzibachstrasse 31
6023 Rothenburg
Switzerland

Phone +41(0)41 541 50
40
info@zub.ch
www.zub.ch

ZbComp Library

Software Reference

Version 7.00.102

Address zub machine control AG
Buzibachstrasse 31
CH-6023 Rothenburg
Switzerland
Phone +41-41-541 50 40
Fax +41-41-541 50 49
<http://www.zub.ch>
<http://www.aposs.ch>
info@zub.ch

Copyright © zub machine control AG

zub machine control AG reserves the right to make modifications to the software, documentation, and product as it sees fit without prior or subsequent notice to users.

No part of this publication may be reproduced or distributed in any form or by any means (e.g. photocopy, microfilm, electronic data exchange) or stored in a database or retrieval system, without the prior written permission of zub machine control AG.

While every precaution has been taken in the preparation of this manual, zub machine control AG assumes no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.

Trademark Many of the designations that are used by manufacturers and sellers to distinguish their products are claimed as trademarks. We point out that such software and hardware designations, brand names, and trademarks used in this publication are protected by law.

In particular, this includes the following:

APOSS is a registered trademark of zub machine control AG.

Microsoft, Windows 7, Windows 10 and Windows 11 are either registered trademarks or trademarks of the Microsoft Corporation in the USA and other countries.

None of the registered trademarks are marked in this publication. This, and the fact that the "R" symbol is missing, does not mean that the trademark is a free trademark.

Table of Contents

1.	Imprint	1
2.	Introduction to ZbComp	3
3.	Function Descriptions	4
3.1.	ZbCompFreecode	4
3.2.	ZbCompFromFile	4
3.3.	ZbCompApossCFromFile	4-5
3.4.	ZbCompGetCodesize	5
3.5.	ZbCompGetError	5-6
3.6.	ZbCompGetErrorCount	6
3.7.	ZbCompGetMinFirmware	6
3.8.	ZbCompGetVersion	6
3.9.	ZbCompOptions	6-7
3.10.	ZbCompReadcode	7-8
4.	Index	9

2 Introduction to ZbComp

The ZbComp Library allows Window developers to create their own applications that make direct calls to the ApossIDE compiler to compile application programs. These programs may then be downloaded to the controller using either the **ZbMocProgramLoad (on-line documentation)** or **ZbMocProgramLoadBinary (on-line documentation)** routine in the **ZbMoc Library ('Introduction to ZbMoc' in the on-line documentation)**.

The ZbComp Library is a Dynamic Link Library (DLL) available for 32-bit Windows (WIN32). Application development depends on the development environment used by your application program. However, if Microsoft Visual Studio is used, then it is sufficient to include the header "ZbComp.h" and link the base library (ZbComp.lib) in order to use all functions of the library. The corresponding 'ZbComp.dll' must be placed somewhere in the search path which is used by the program.

If the ApossIDE software has been installed, then the ZbComp Library files will be installed in the "Development\ZbComp" subdirectory under the ApossIDE install directory. The "ZbComp.dll" file itself will be installed directly in the ApossIDE install directory, along with the other ApossIDE DLL files.

This is how the ApossIDE compiler is typically used:

Step	Routine	Description
1	ZbCompOptions	Before calling any other routines, call ZbCompOptions to set the compiler options.
2	ZbCompFromFile	Call ZbCompFromFile to compile the Aposs source code file.
3	ZbCompApossCFromFile	Call ZbCompApossCFromFile to compile the ApossC source code file.
4	ZbCompGetErrorCount ZbCompGetError	Check if there has been an error using the function ZbCompGetErrorCount . This will return 0 if no error has occurred during compilation. If there have been errors, you can retrieve information about the errors using the function ZbCompGetError .
5	ZbCompGetMinFirmware	Call ZbCompGetMinFirmware to get the minimum firmware version required to execute the code produced by the compiler. This should be checked against the controller's firmware version before attempting to download the compiled code to the controller.
6	ZbCompGetCodesize	Call ZbCompGetCodesize to determine how many bytes of executable code have been produced by the compiler.
7	ZbCompReadcode	Call ZbCompReadcode to retrieve each individual byte of the executable code.
8	ZbCompFreecode	After retrieving the executable code, call ZbCompFreecode to free the compiler's internal memory buffer used to store the executable code.

3 Function Descriptions

3.1 ZbCompFreecode

SIGNED16 ZbCompFreecode();

Summary Free the data buffers that have been allocated internally during a compilation.

Return Value 0 - Success

Remarks This routine may be called after the generated executable code is no longer needed.

3.2 ZbCompFromFile

**SIGNED16 ZbCompFromFile(const char *fname, const char *logicalfname,
const char *incpath, SIGNED16 mode, UNSIGNED32 flags,
const char *reffname);**

Summary Execute the compiler using the Aposs source code provided in the given file.

Parameter fname - Name of an ASCII file which contains the Aposs application source code.
logicalfname - File name which is inserted into the error messages. This is truncated to CMP_LOGICALNAMELEN characters, if it is too long.
incpath - Directories (separated by ';') to be searched for include files. This can be NULL.
mode - Type of compilation:
CMP_NORMAL - Normal code generation.
CMP_DEBUG - Debug code generation, including debug information.
CMP_DBGINFO - Normal code generation, but including debug information.
flags - Compiler flags (see "ZbComp.h").
reffname - Cross reference filename ("" if none).

Return Value 0 - Compile completed. Use [ZbCompGetErrorCount](#) to check for compile errors.
1 - Compiler errors. Use [ZbCompGetError](#) to retrieve compile errors.
2 - Could not open file being compiled.
4 - Memory allocation error in compiler.
6 - Internal compiler error.

Remarks [ZbCompOptions](#) must be called before this routine can be called. After compilation is complete, call [ZbCompGetErrorCount](#) to check if any errors occurred during compilation.

3.3 ZbCompApossCFromFile

**SIGNED16 ZbCompApossCFromFile(const char *fname, const char *logicalfname,
const char *incpath, SIGNED16 mode, UNSIGNED32 flags,
const char *reffname);**

Summary Execute the compiler using the ApossC source code provided in the given file.

Parameter fname - Name of an ASCII file which contains the ApossC application source code.
logicalfname - File name which is inserted into the error messages. This is truncated to CMP_LOGICALNAMELEN characters, if it is too long.
incpath - Directories (separated by ';') to be searched for include files. This can be NULL.
mode - Type of compilation:
CMP_NORMAL - Normal code generation.
CMP_DEBUG - Debug code generation, including debug information.
CMP_DBGINFO - Normal code generation, but including debug information.
flags - Compiler flags (see "ZbComp.h").
reffname - Cross reference filename ("" if none).

Return Value 0 - Compile completed. Use [ZbCompGetErrorCount](#) to check for compile errors.
1 - Compiler errors. Use [ZbCompGetError](#) to retrieve compile errors.
2 - Could not open file being compiled.
4 - Memory allocation error in compiler.
6 - Internal compiler error.

Remarks [ZbCompOptions](#) must be called before this routine can be called. After compilation is complete, call [ZbCompGetErrorCount](#) to check if any errors occurred during compilation.

3.4 ZbCompGetCodesize

UNSIGN16 ZbCompGetCodesize(void);

Summary Return the size in number of bytes of the executable code produced by the compiler.

Return Value Number of executable code bytes

Remarks The returned number of bytes can be used to read the compiled executable code by calling [ZbCompReadcode](#).

3.5 ZbCompGetError

```
typedef struct ZbCompErrorTag
{
    int    Number;
    char   Text[256];
    int    Line;
    int    Column;
    char   Filename[CMP_LOGICALNAMELEN];
} ZbCompError;
```

ZbCompError* ZbCompGetError(int n);

Summary Return a pointer to a structure containing the error information corresponding to the given error number from the last compilation.

Parameter n = Number of the error (0 ... n-1).

Return Value Pointer to a structure containing error information. NULL will be returned if the error number does not correspond one of the errors generated during the last compilation. This pointer is TEMPORARY and must not be saved for later use.

Remarks The error message text will always have the format "#nnn Message" where "nnn" is the 3-digit error number and "Message" is the actual message text. The number of errors can be retrieved by calling [ZbCompGetErrorCount](#).

3.6 ZbCompGetErrorCount

```
int ZbCompGetErrorCount(void);
```

Summary Return the number of errors which occurred during the compilation.

Return Value Number of errors found during the last compilation.

Remarks Call this routine after each compilation to retrieve the number of errors. Detailed information on each error can be retrieved by calling [ZbCompGetError](#).

3.7 ZbCompGetMinFirmware

```
UNSIGNED16 ZbCompGetMinFirmware(void);
```

Summary Return the minimum controller firmware version required to execute the compiled code.

Return Value Minimum controller firmware version (e.g. 70129 for version 7.01.29).

Remarks [ZbCompFromFile](#) must be called before this routine will return a valid version number. As well, the compile must complete without errors (warnings are acceptable). Call [ZbCompGetErrorCount](#) to check the error count.

3.8 ZbCompGetVersion

```
const char *ZbCompGetVersion(void);
```

Summary Return a pointer to the compiler's version string.

Return Value Pointer to a \0-terminated character string containing the version information.

Remarks A typical version string is similar to this:

ApossIDE Compiler 6.13.69 (2019/02/19)

3.9 ZbCompOptions

```
typedef struct
{
    SIGNED32    length;           // Length of structure (in bytes).
    UNSIGNED32  fversion;         // Firmware version.
    SIGNED32    maxvars;          // Maximum number of variables available.
    SIGNED32    stksize;          // Stack size.
    SIGNED16    maxaxes;          // Maximum number of axes available.
    SIGNED32    controller;       // Controller type.
    SIGNED32    hwopt;            // Hardware options.
    SIGNED32    language;         // Language: 0 - English, 1 - Deutsch.
    SIGNED32    maxerrs;          // Maximum number of compiler errors.
    SIGNED32    maxwarn;          // Maximum number of compiler warnings.
} ZbCompParams;

SIGNED16 ZbCompOptions(ZbCompParams *params);
```

Summary Define all necessary parameters for the next compilation.

Parameter params = Pointer to parameter structure. See "ZbComp.h" for a more detailed description of the compiler options.

Return Value 0 - Success
255 - Error (e.g. invalid number of variables, axes, etc.).

3.10 ZbCompReadcode

```
UNSIGNED8 ZbCompReadcode(UNSIGNED32 indx);
```

Summary Return the byte of executable code corresponding to the given index.

Parameter indx = Index of the executable byte (0...n-1).

Return Value The byte of executable code corresponding to the index.


Remarks [ZbCompFromFile](#) must be called before this routine will return valid data. As well, the compile must complete without errors (warnings are acceptable). Call [ZbCompGetErrorCount](#) to check the error count.

This routine must be called "n" times with index values from 0 to n-1, where "n" is the total number of bytes. This can be retrieved by calling [ZbCompGetCodesize](#). If the index is out of range, then 127 is returned. This corresponds to the end-of-file command in executable code.

After all bytes have been read and there is no longer any need for the generated code, then [ZbCompFreecode](#) may be called to free the internal buffers allocated during the compilation.

The executable code may be downloaded to the controller using either the **ZbMocProgramLoad (on-line documentation)** or **ZbMocProgramLoadBinary (on-line documentation)** routine in the **ZbMoc Library ('Introduction to ZbMoc' in the on-line documentation)**.

Summary Return the byte of executable code corresponding to the given index.

 **ZbCompReadcode** is designed to be passed directly to **ZbMocProgramLoad** ([on-line documentation](#)) as an argument.

4 Index

Imprint, 1

Introduction to ZbComp, 3

ZbComp Cover Page, -1-0

ZbCompApossCFromFile, 4-5

ZbCompFreecode, 4

ZbCompFromFile, 4

ZbCompGetCodesize, 5

ZbCompGetError, 5-6

ZbCompGetErrorCount, 6

ZbCompGetMinFirmware, 6

ZbCompGetVersion, 6

ZbCompOptions, 6-7

ZbCompReadcode, 7-8