Creating and Linking External C Library Functions

Document Version 1.0

CONTENT

1 GENERAL 3

2 STEPS FOR CREATING AN EXTERNAL C-LIBRARY 4

3. EXTERNAL LIBRARY FUNCTIONS IN THE RUN-TIME SYSTEM 8

CHANGE HISTORY 9

1 General

CoDeSys has an interface for linking user-defined functions and function blocks which are implemented in ANSI C. These so-called external functions can be collected into a library and linked into an IEC 1131 program.

This interface thus allows existing C libraries (e.g. regulator blocks, communications drivers,...) to be re-used.

The development and including of the C-function depends on the type of the CPU-platform. This document describes the necessary Steps in case of a 32-bit-platform.

A decision must be made in the case of external libraries as to whether the functions are to be located directly in the external library or in the run-time system.

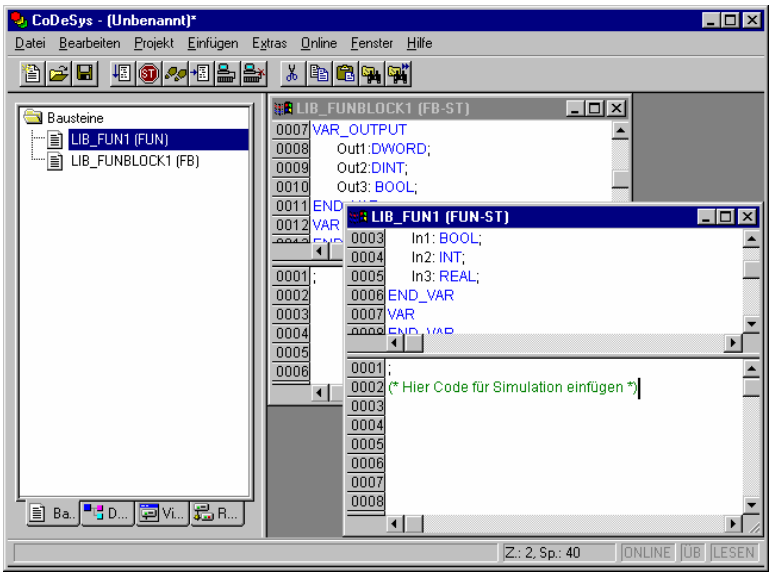
Section 2 describes the procedure for the creation of an external C library whose functions are programmed in the library.

Section 3 describes the creation of an external library whose functions are located in the run-time system. In those cases the library functions only describe the interfaces.

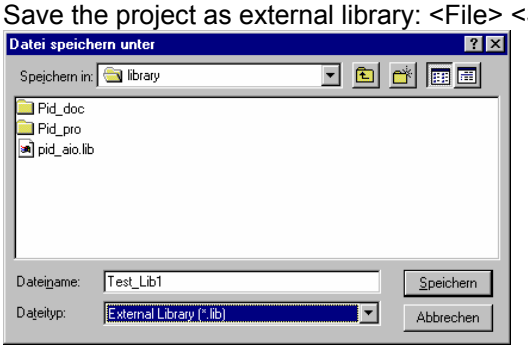
Note: CoDeSys does not support the call of external functions which return a structure. If such a function is defined in an external library, a pointer on the return value is expected. If this is not appropriate due to the re-entrance problematic, use an IN_OUT variable or a pointer, that is passed as input and filled by the function.

2 Steps for creating an external C-library

1. Generate a new project in CoDeSys. Create a POU, in which you declare the necessary input and output variables for your C functions or function blocks. This declaration is used as interface for the IEC 61131-3 development system. In the implementation part insert the code which is to be executed in simulation mode.

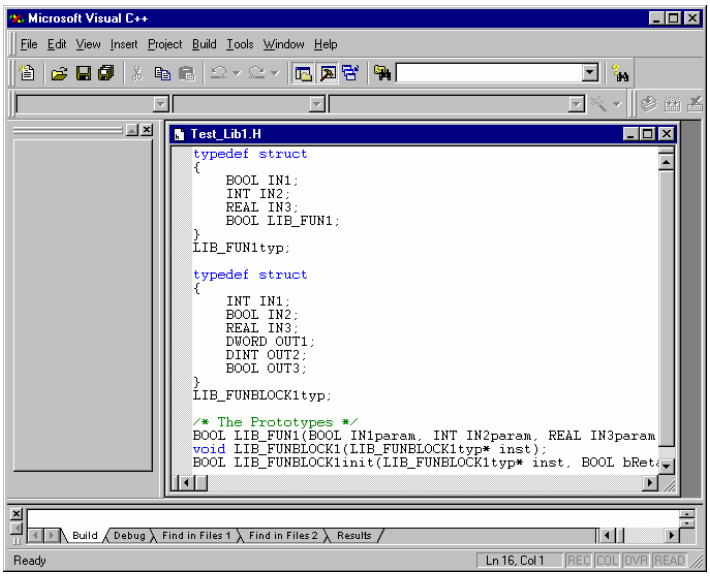


2. Save the project as external library: <File> <Save as...>, file type <External library>.



   Save it also as a normal project, in order to be able to maintain the library in the future, or to link in more functions.

3. When saving as an external library, a header file named <Libname>.h and a library file <Libname>.lib are created. The header file is the basis for the development of the C function.



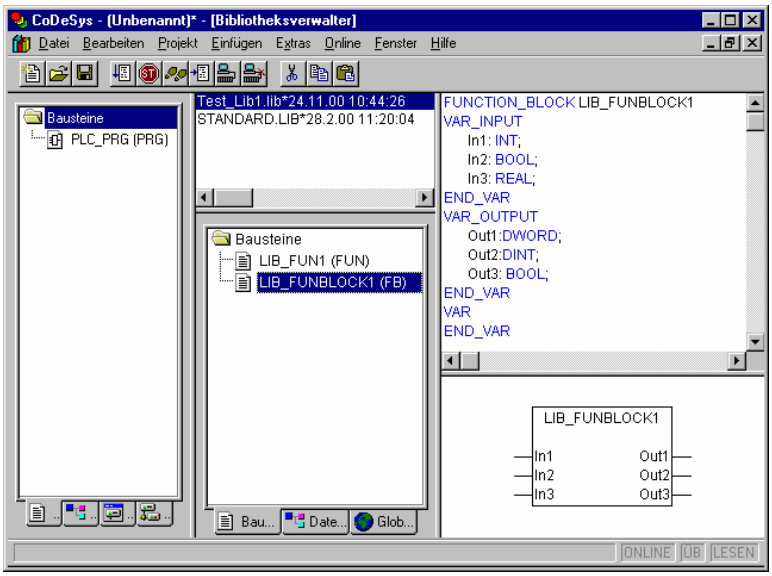
4. Create a C file from the header file <Libname>.h.
   For a function block, an associated initialisation function is generated, in addition to the library function itself. You must initialise the data for the function block in this.
   Note that the sequence of functions in the C file must be the same as in the project (alphabetical order). The sequence of functions in the header file must remain unchanged!

   Type definitions, such as BOOL, which are used in the header file are found in the file lzstyp.h. Note that this can be in conflict with, for instance, windows.h! You should not include both header files. If that cannot be avoided: a BOOL parameter in a CoDeSys library function must be a char. If necessary, write char explicitly!
   Please not that all referenced functions must be present in the C file.
   Initialising of global declared C-Variablen with Constants is possible:
   int i = 5;
   However, the initialising with expressions is not:
   int pi = &i; /* not possible */
   Initialisations that need a expression must therefore be made in a init-Function.
5. Program the library functions, placing any help functions at the end of the file.
6. With Visual C++ (or your C compiler) create a project for a Windows or console application with the name of the library (<Libname>.mdp), and insert the library's C file into the project.

   Important: The following compiler settings are necessary in Visual C++:

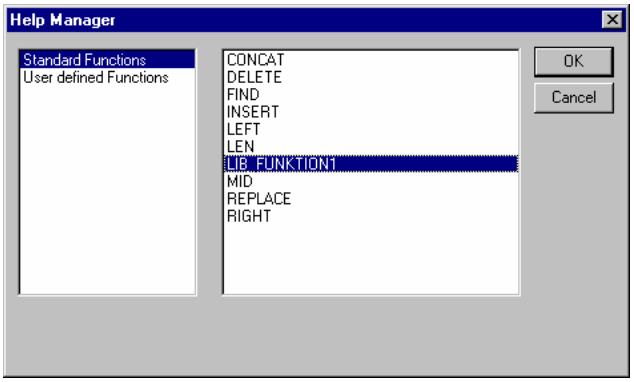
   - Project / Settings / C/C++ / Customize: Enable function-level linking deactivate
   - Project / Settings / C/C++ / Optimization: Customize: Global Optimizations + Generate Intrinsic Functions + Favor small code activate
   - Project / Settings / C/C++ / Code Generation: Struct member alignment = 1 byte (80x86) resp. 4 bytes (RISC-processors: ARM, PowerPC)
   - If you want to use any ANSI C standard library functions you must activate the following selection in the project settings:
     Build / Settings / C/C++ / Category: Optimizations : Customize: Generate Instrinsic Functions
     In this way the simplest ANSI C library functions are implemented in-line (strcpy(),...). Otherwise no libraries may be used.
   - For Visual 6.0:
     Project / Settings / C/C++ / General: Debug Info: Program Database (not: Program Database for Edit and Continue)

7. After this, compile the project. If the library is correct, all that appears is the error message: „Unresolved external symbol WinMain“ (or „main“).
   An object file for linking into your IEC 1131 project (<Libname>.obj) exists after this.
8. Copy the object file (<Libname>.obj) which has been created and the library (<Libname>.lib) into the CoDeSys library directory.
9. Link the library into your IEC 31131 application program using the CoDeSys library manager (Window/Library manager, Insert/Library).

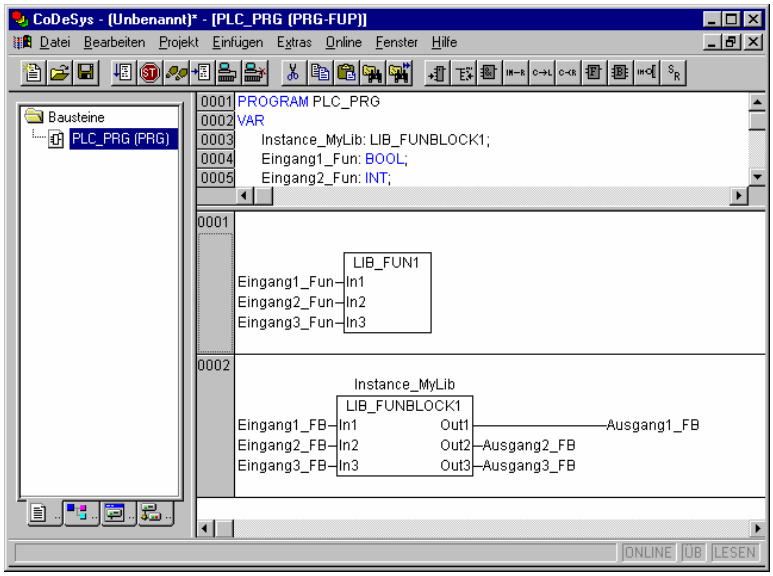


Build the application project with the external library in order to check that you have correctly executed the procedure described above.

10. The new libraries will be displayed in the Help-Manager (Input assistant dialogue) in the category standard functions / function blocks and can be called from the IEC 61131-3 code.



Now you can access all the functions in your library from your project.


CoDeSys - (Unbenannt)* - [PLC_PRG (PRG-FUP)]
Datei Bearbeiten Projekt Einfügen Extras Online Fenster Hilfe
Bausteine
PLC_PRG (PRG)
0001 PROGRAM PLC_PRG
0002 VAR
0003 Instance_MyLib: LIB_FUNBLOCK1;
0004 Eingang1_Fun: BOOL;
0005 Eingang2_Fun: INT;
0001
LIB_FUN1
Eingang1_Fun In1
Eingang2_Fun In2
Eingang3_Fun In3
0002
Instance_MyLib
LIB_FUNBLOCK1
Eingang1_FB In1 Out1 Ausgang1_FB
Eingang2_FB In2 Out2 Ausgang2_FB
Eingang3_FB In3 Out3 Ausgang3_FB
ONLINE ÜB LESEN

3. External Library Functions in the Run-Time System

This section describes how an external library is created whose functions are located in the run-time system. The library functions here only describe the interfaces to the functions (prototypes).

1. Use CoDeSys to create a new project which contains the library functions (functions and function blocks). Define the interface, and program the function, which will only be used in the simulation (see Section 2., Point 1.).
2. Save this project as a library with the <File><Save as library...> menu, selecting the <External library> button in the dialog box. Save it also as a normal project, in order to be able to maintain the library in the future, or to link in more functions.
   When saving as an external library, files named <Libname>.h and <Libname>.lib are created. The header file contains the interface definition for the functions in the run-time system.
3. The integration of the library functions into the SoftPLC is done in the module RtsCst.c. The process is illustrated with the example of a library function (CstDummy):

   a) Implementing the library function in the module RtsCst.c

   b) enlarging the reference table (CstExtRefTable) with the entry of the new library function
      The first structure component has to contain the name of the library function, how it is defined in the IEC library (see upper description). The name length is limited to a maximum of 31 characters.
      The second entry has to deliver the function pointer to the function.

   c) with the function CstGetExtRefTable a pointer to the reference table has to be returned

   In the following an excerpt of the concerning code is illustrated:

```
static void CstDummy(void)
{
    /* TODO: Implement your library-function */
    return;
}

static ExtRef CstExtRefTable[] =
{
    /*{Function-name (max. 31 chars), Function-pointer}*/
    {"Dummy", (void (*)())CstDummy},
    {"", NULL}
};

ExtRef* CstGetExtRefTable(void)
{
    return CstExtRefTable;
}
```

Change History

Version	Description	Date
0.4	Issued	04.02.2003
1.0	Note added (#4331) + general extension	10.10.2006
1.0	Review and Rework (Generalization for 32-Bit-platforms, document renamed from C_IN_386_E.doc to C_IN_32Bit_E.doc)	20.10.2006
1.0	Release	20.10.2006