Best Solution Library Architecture Introduction

Revision: V1.00 Date: November 10, 2022

www.bestsolution.com

Table of Contents

1 Library Architecture Introduction

1.1 MAIN_PROGRAM Introduction

The MAIN_PROGRAM is the core control program of the library, which can implement device hardware initialisation, execute all the external functions and control the device working and sleep processes. Users can add custom functions, such as touch key function, A/D sampling function, serial communication function and so on, to the MAIN_PROGRAM. The MAIN_PROGRAM executes each function in turn through the process control of the software.

1.2 External Function Introduction

The V512 library includes five external functions. Each external function performs a different function. Users can add custom functions to the following five external functions according to the actual requirements:

① EXTEND_FUNCTION_INITIAL( )

This is the external initialisation function, which is executed only once.

② EXTEND_FUNCTION( )

This is the external work function. If the power-saving mode has been disabled, the program will execute this function repeatedly.

③ EXTEND_FUNCTION_HALT_PREPARE( )

The function is executed before entering the standby mode.

④ EXTEND_FUNCTION_HALT_WAKEUP( )

The function is executed after the device has been woken up from the standby mode by an interrupt or an I/O port.

⑤ EXTEND_FUNCTION_RETURN_MAIN( )

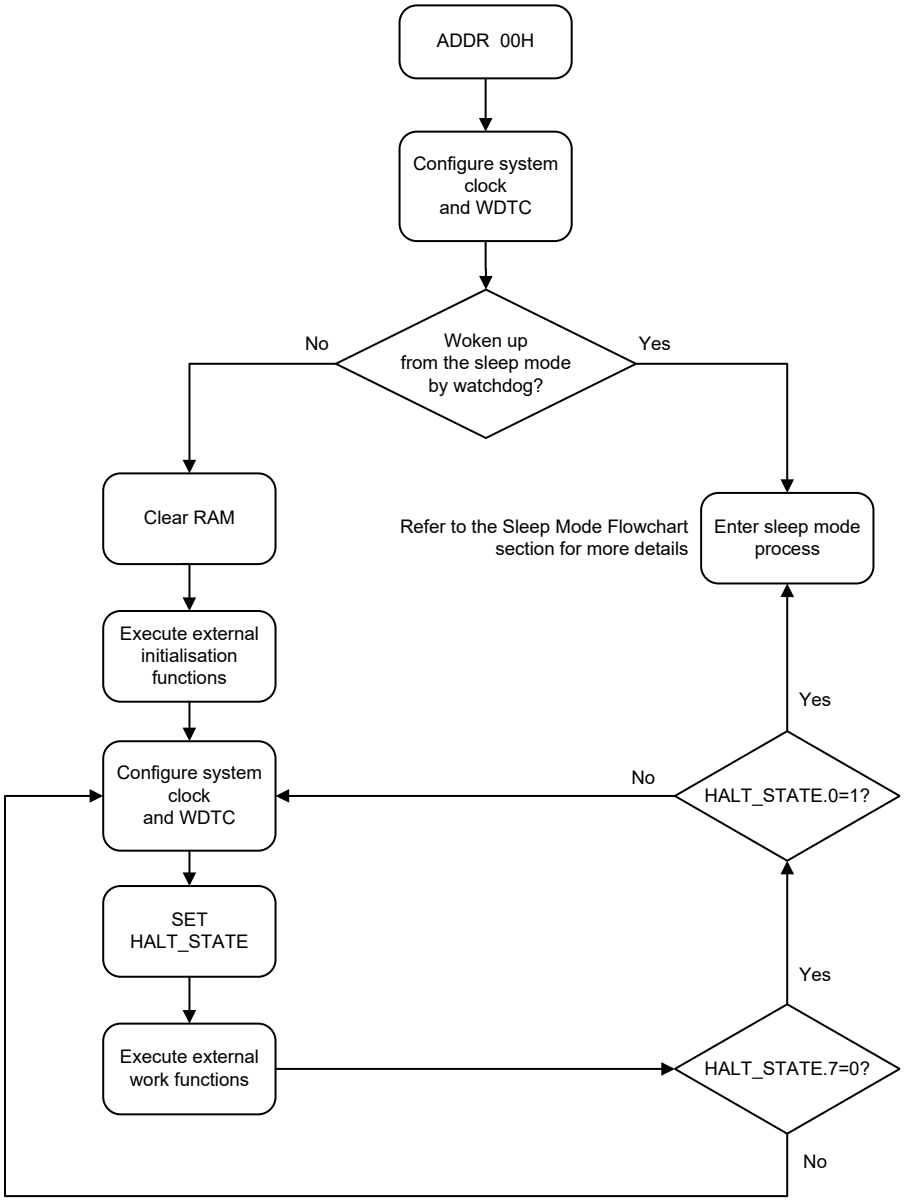
The function is executed when the program turns back to the working mode from the standby mode.

1.3 External Function Work Introduction

In MAIN_PROGRAM, external functions are executed sequentially according to the order, from EXTEND_FUNCTION_1A_XXXX to EXTEND_FUNCTION_2F_XXXX. Here 1A~2F are different entry points for the external functions. Users can set different entry points for custom functions in MAIN_PROGRAM.INC. If the library executes an external function, the relevant functions will be executed according to their order. The V512 touch library is also a kind of custom function, which is optional.

2 Library Flowchart

2.1 Work Flowchart



2.2 Work Flowchart Introduction

2.2.1 Initialisation Process Introduction

When the device is powered on or reset, the library will execute programs starting from 00H. After the system clock and watchdog overflow period have been configured, the program will check the PDF and TO flags. If these two flags are both 1, this means that the device is woken up from the sleep mode by a watchdog overflow.

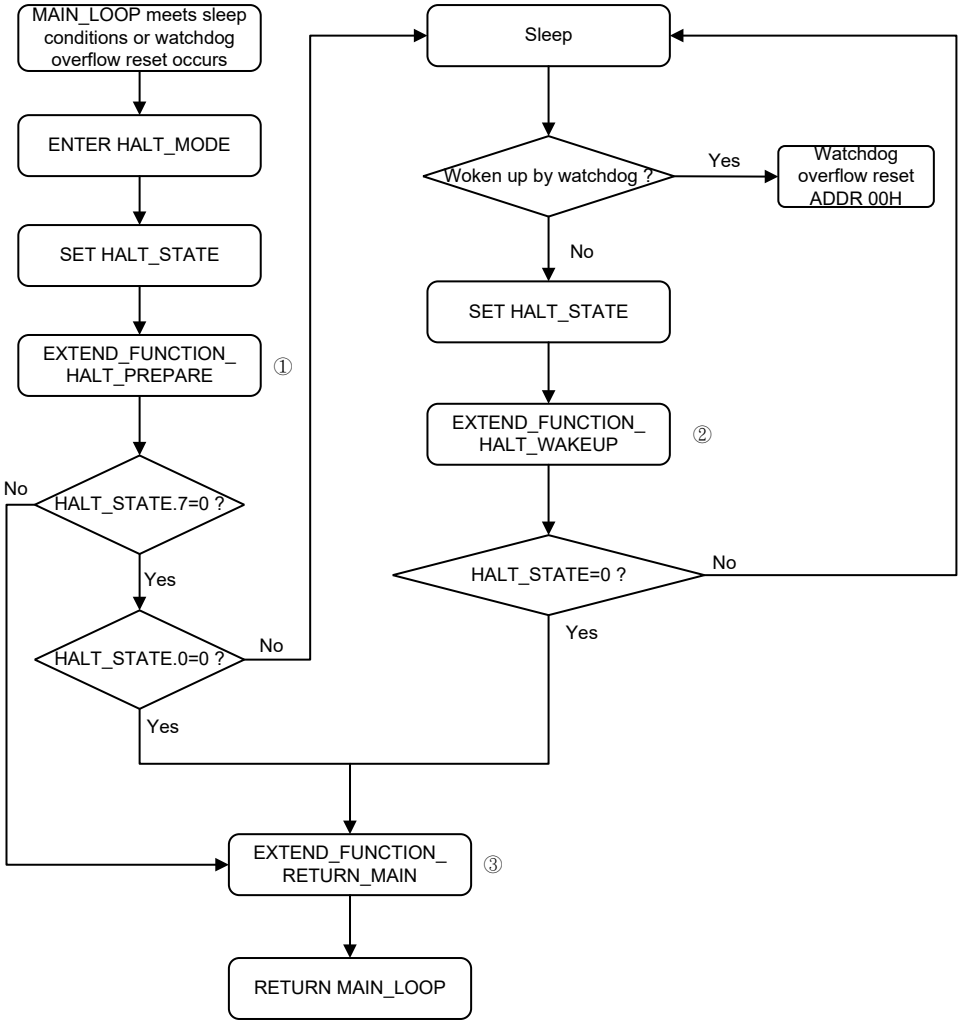
If the device is woken up from the sleep mode by the watchdog, the program will go to the right path to enter the sleep process, otherwise the program will go to the left initialisation path if the device reset is caused by power-on or other conditions. After clearing the RAM and executing all the initialisation functions, the program will enter the work process.

2.2.2 Work Process Introduction

After the library has been initialised, it will enter the MAIN_LOOP. If the power-saving mode has been disabled, the library will execute the MAIN_LOOP repeatedly. After configuring the system clock and the watchdog overflow period, the sleep flag HALT_STATE will be set high and then all the external work functions will be executed.

After the external work functions have been completely executed, the library will determine whether the program meets the sleep conditions according to the sleep flag HALT_STATE. If the sleep mode has been enabled, HALT_STATE will be configured by the touch layer to allow the library to enter the sleep mode when the program meets the sleep conditions. If the sleep mode has not been enabled, users need to configure the HALT_STATE flag in the external functions in order to enter the sleep mode. Otherwise, the library will execute the workflow repeatedly.

2.3 Sleep Mode Flowchart



2.4 Sleep Mode Flowchart Introduction

2.4.1 Sleep Mode Flag

If the V512 library has enabled the power-saving mode, the program will determine whether to enter the sleep process according to the HALT_STATE flag. The following is the explanation of the lowest and highest bits of the sleep flag:

1. HALT_STATE.0 (read/write)

This bit determines whether the program can enter the sleep mode. If the bit is 1, this means that the program meets the sleep conditions and then will enter the sleep mode. If the bit is 0, this means that the program does not meet the sleep conditions, and then will return to the working process.

2. HALT_STATE.7 (read/write)

This bit indicates whether it is valid to operate the sleep flag RAM. If the bit is 0, the sleep flag is valid and the library will determine whether to enter sleep mode according to HALT_STATE.0. If the bit is 1, the sleep flag is invalid and the library will not control the process according to the sleep flag bit. Therefore, HALT_STATE.7 must be cleared to 0 in order to modify the sleep flag.

2.4.2 Sleep Mode Function Introduction

After the library enters the sleep process, three external sleep functions will be executed according to the flowchart. How the each external sleep function is executed is shown below:

① EXTEND_FUNCTION_HALT_PREPARE( )

External sleep mode preparation function: It is executed after the device enters the sleep process.

② EXTEND_FUNCTION_HALT_WAKEUP( )

External sleep mode wake-up function: It is executed after the device is woken up by an interrupt or an I/O port.

③ EXTEND_FUNCTION_RETURN_MAIN( )

External sleep-return-to-work function: It is executed after the device returns to the working mode from the sleep mode.

2.4.3 Sleep Mode Process Introduction

Before the external sleep mode preparation and the wake-up function have been executed, the library will execute the “SET HALT_STATE” instruction to set HALT_STATE.7 and HALT_STATE.0 to 1. Users can operate on the HALT_STATE flag in the external custom sleep functions. The library determines whether to enter the sleep process according to the HALT_STATE after the external functions have been executed.

If the HALT_STATE has not been modified in the EXTEND_FUNCTION_HALT_PREPARE function, the library will not enter the sleep mode. It will return to the external work function MAIN_LOOP directly. If the HALT_STATE has not been modified in the EXTEND_FUNCTION_HALT_WAKEUP function, the device will continue to enter the sleep mode after executing all the EXTEND_FUNCTION_HALT_WAKEUP functions.

3 Custom Library Program Example

This chapter will provide an example of adding a custom A/D sampling program. In this program, user can create a library with the A/D sampling function according to the following steps of creating C/ASM, EXT and INT files.

3.1 Creating a C/ASM Program File

When creating a C/ASM file and writing an A/D sampling program, it should be noted that the initialisation function and the work function should be programed separately. When the library has been powered on and executed the initialisation function, it will execute all the work functions repeatedly. The example codes of C language are shown below:

```
#include "AD_FUNCTION.INC"
//================================================
//************************************************
//================================================

unsigned char  AD_VALUE_L = 0;
unsigned char  AD_VALUE_H = 0;

void AD_FUNCTION_INITIAL()
{
// A/D//
    _sadc0 = 0b00010000;  //設置輸入通道為AN0
    _sadc1 = 0b00001010;  //外部通道,VDD,fsys/4
    _ace0  = 1;           //设置PC0为AN0通道
    _adcen = 1;
}

//================================================
//************************************************
//================================================

void AD_FUNCTION_WORK()
{
    _start = 0;
    _start = 1;
    _start = 0;
    while(_adbz);
    AD_VALUE_L = _sadol;
    AD_VALUE_H = _sadoh;
}
```

3.2 Writing an EXT File

In order to allow the functions and parameters of the functional files in the library to be used by other files, each C/ASM file comes with a corresponding EXT file. MAIN_PROGRAM.ASM, USER_PROGRAM.ASM (C) and BSXXXXXC_CTOUCH.OBJ all have their corresponding EXT files, where the external declarations of their functions and variables are stored. If the header files have included these EXT files, the relevant functions and variables can be directly used.

When adding a custom function, an EXT file also should be created in order to allow other files to use the custom functions and parameters through this EXT file. The way of declaring external functions and variables for EXT files is shown below:

Declare an assembly function: EXTERN _XXX : near

Declare an assembly parameter: EXTERN _A : BYTE

Declare a C language function: extern void XXX( );

Declare a C language parameter: extern unsigned char A;

In order for the EXT files to identify both assembly and C language declarations, ;;;/* and ;;;*/ signs are used to divide the area of C language programs and assembly programs. When encountering the assembly programs, /* and */ signs will be annotated so that the programs between /* and */ can be compiled. Otherwise, the contents between /* and */ will be skipped. A program or a variable can have external declarations of assembly and C language simultaneously.

The way to declare the initialisation function, work function and two global variables which store A/D values in the EXT file is shown below with red rectangles: It is allowed to copy from other EXT files and modify the XXX_DEF, the function names and the parameter names.

```
;;;;;/*
ifndef AD_FUNCTION_DEF
    EXTERN  _AD_FUNCTION_INITIAL    :NEAR
    EXTERN  _AD_FUNCTION_WORK       :NEAR

    EXTERN  _AD_VALUE_H             :BYTE
    EXTERN  _AD_VALUE_L             :BYTE
endif
;;;;;*/
```

```
;;;;;///* ---------------------------------------- */
;;;;;///* C exported function,register             */
;;;;;///* ---------------------------------------- */
;;;;;/*
ifdef _INCLUDE_C_
;;;;;*/
;;;;;//-----------------------------------------
#ifndef AD_FUNCTION_DEF
    extern void  AD_FUNCTION_INITIAL();
    extern void  AD_FUNCTION_WORK();

    extern unsigned char  AD_VALUE_H;
    extern unsigned char  AD_VALUE_L;
#endif
;;;;;//-----------------------------------------
;;;;;/*
endif
;;;;;*/
```

3.3 Configuring an INC File

All the EXT files of Best Solution library will be included in GLOBE_INCLUDE.INC and meanwhile the corresponding MCU hardware resources are stored in GLOBE_VARIES.INC. Hence, if all the INC files of the library include these two header files, the library will include all the EXT files and corresponding MCU hardware resources.

However, this will also cause the program to include its own EXT file and thus report an error for external declarations that call its own functions. Therefore, the ifndef/endif precompilation should be applied in the EXT file to avoid this problem. Firstly, XXX_DEF should be defined as a mark in the corresponding INC file, and then the ifndef/endif precompilation can be used to identify whether the INC file is its header file. If yes, the external declaration is omitted to avoid this mistake. The configuration steps are shown below:

STEP 1: Define AD_FUNCTION_DEF in the AD_FUNCTION.INC, which includes two global header files, GLOBE_VARIES.INC and GLOBE_INCLUDE.INC, in order to call the global variables.

```
#define AD_FUNCTION_DEF
#include "..\GLOBE\GLOBE_VARIES.INC"
#include "..\GLOBE\GLOBE_INCLUDE.INC"
```

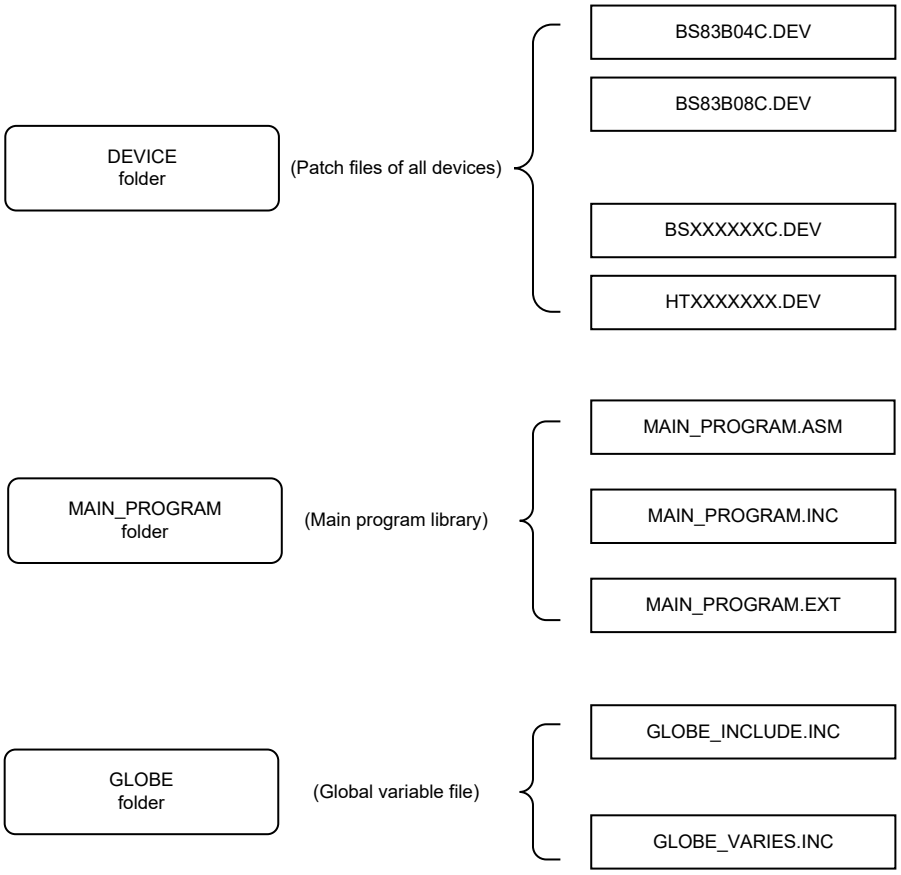
STEP 2: Use ifndef/endif precompilation in AD_FUNCTION.EXT to avoid the compilation error because of the external declarations of self-functions and variables.

```
;;;;;/*
ifndef AD_FUNCTION_DEF
    EXTERN  _AD_FUNCTION_INITIAL    :NEAR
    EXTERN  _AD_FUNCTION_WORK       :NEAR

    EXTERN  _AD_VALUE_H             :BYTE
    EXTERN  _AD_VALUE_L             :BYTE
endif
;;;;;*/


;;;;;///* ---------------------------------------- */
;;;;;///* C exported function,register             */
;;;;;///* ---------------------------------------- */
;;;;;/*
ifdef _INCLUDE_C_
;;;;;*/
;;;;;//-----------------------------------------
#ifndef AD_FUNCTION_DEF
    extern void  AD_FUNCTION_INITIAL();
    extern void  AD_FUNCTION_WORK();

    extern unsigned char  AD_VALUE_H;
    extern unsigned char  AD_VALUE_L;
#endif
;;;;;//-----------------------------------------
;;;;;/*
endif
;;;;;*/
```

4 Library Architecture Diagram

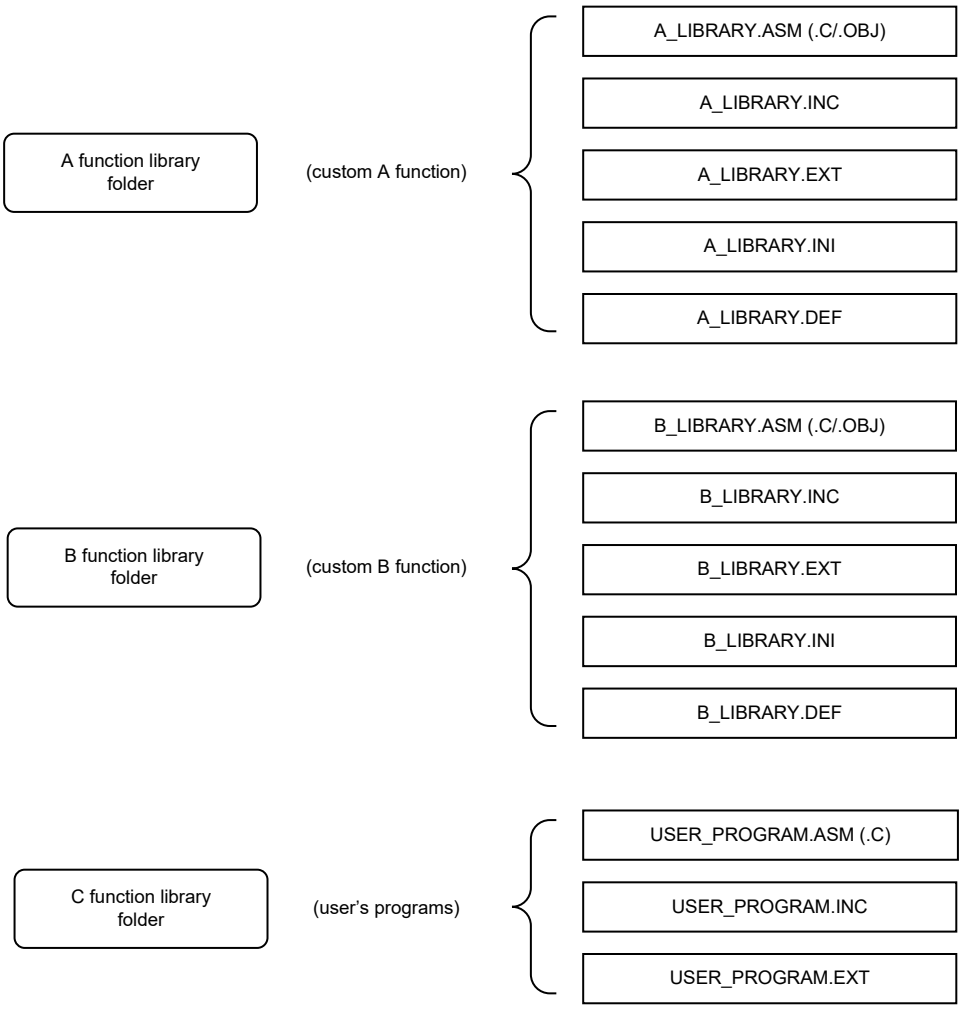
4.1 Best Solution Library Folder Architecture

The V512 library folder is divided into two parts. The first are library architecture related folders, which store essential files maintaining normal operation of library. And the other are custom program folders, as shown below.



Library Architecture Related Folders

1. DEVICE folder: It is used to store the corresponding DEV file of the device. The internal device resources are declared in the DEV file to ensure that the DEV file in this folder has been modified when using different devices.
2. MAIN_PROGRAM folder: It is used to store the main program of the library. Executing the programs in MAIN_PROGRAM.ASM can run the library. Users can add external custom functions to the main program through the MAIN_PROGRAM.INC. The EXT file is used to make external declarations for flags used in the main program.
3. GLOBE folder: It is used to store global variables of the library. It has two .INC files. The first one is GLOBE_INCLUDE.INC which is used to include all the EXT files of the library and the other is GLOBE_VARIES.INC, which includes the MCU hardware declaration DEV file and this means that the GLOBE_VARIES.INC includes all the corresponding MCU hardware information.



Custom Program Folders

1. Functional library folder: It is used to store the user functions, such as touch key library. As the touch key library file is an .OBJ file, its variables must be modified through the INI file indirectly. Note that the MAIN_PROGRAM.INC must also include the corresponding INI file. After configuring the INC and EXT files according to the 3.2 and 3.3 sections, the custom functions can be added to the library.

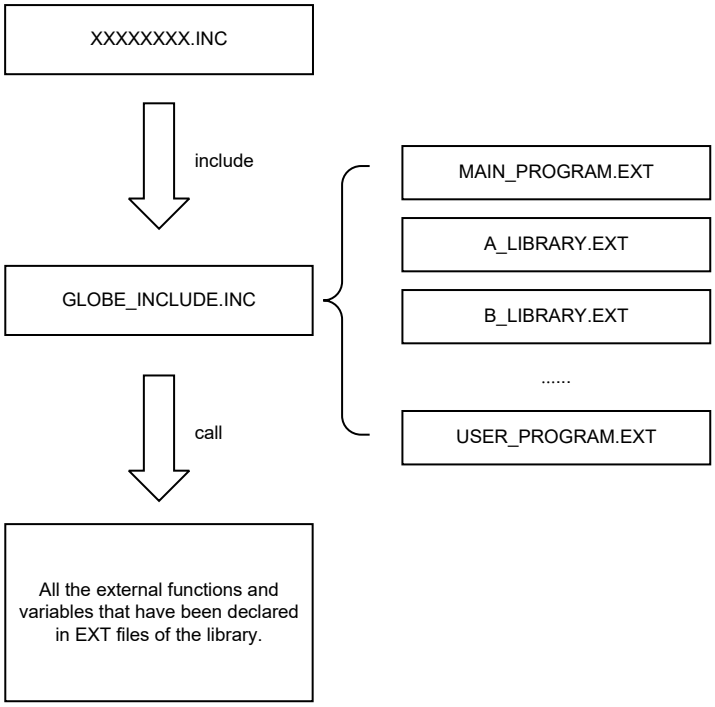
   DEF file is the extended file for the programs. For example, when an IIC program has been created to obtain the data of 0X10, it is required to reconfigure the IIC program in order to obtain the data of 0X12. If these two programs are packed as a DEF file respectively, users can use the desired DEF file according to the requirements. Whether to add this file is up to the user's needs.

2. User function folder: It can be used to store the user programs. Users can write programs in the USER_PROGRAM.ASM (.C) and directly use the functions and variables of other files.

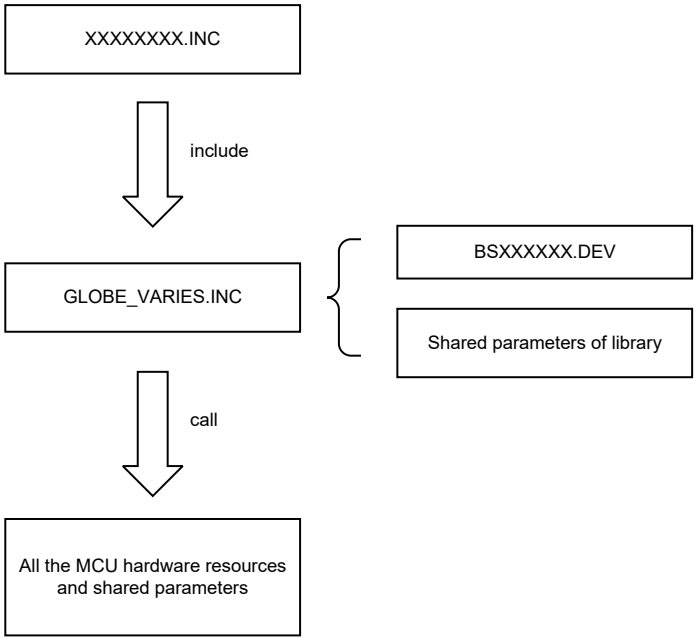
4.2 V512 Library Reference Diagram

The GLOBE_INCLUDE.INC and GLOBE_VARIES.INC files of the GOLBE folder in the V512 library are considered as a link. By including these two header files, the user can implement references and link to various files in the library.

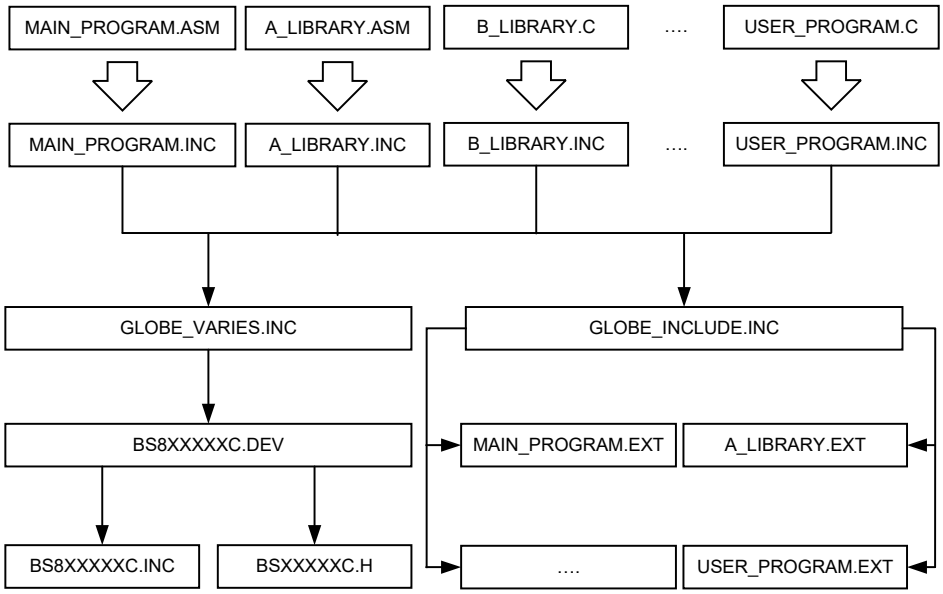
1. GLOBE_INCLUDE.INC includes all the EXT files. If the file in this library has included this header file, then it can call all the external functions and variables that have been declared, as shown below:



2. GLOBE_VARIES.INC includes DEV file with MCU resources. As long as a header file in this library has included GLOBE_VARIES.INC, then it will include the corresponding MCU hardware resources and shared parameters, as shown below:



3. When using the V512 library to add new function codes, users only need to configure the EXT file and INC file according to the steps in the 3.2 and 3.3 sections. After adding these two files to MAIN_PROGRAM, including these two header files of the GLOBE folder in the INC file will complete the function transplant without changing the originally configured header files, as shown below:



No matter whether the assembly programs or C language programs are added to the library, users only need to include the configured EXT header file into GLOBE_INCLUDE.INC. Then the original program can directly call the newly added functions or parameters. This makes it easier to transplant new functions or programs.

4.3 New Library Advantages

The Best Solution library not only has a complete program architecture, but also has a powerful portability. When users need to add a custom function with C or assembly language, it is not needed to make considerable changes to the original framework, just configure the EXT file and modify the key header files so the new function will be added to the library.

This version of the library has eliminated the limitations of the touch key library. It is not necessary to use the touch key functions, so that the library can be not only used in the touch control application areas, but also in other low-power application areas. In addition, this new library version has added three external sleep functions. Users can write programs according to the actual situations, which greatly enhances the operational space of the codes. The following is the description of new files added to this library version:

1. Description of .DEV File

The register names of different devices may be different. All registers in the old library are named and stored in the MAIN_PROGRAM.MCU file. If it is required to add a new device, the MAIN_PROGRAM.MCU files in all the previous libraries need to be updated.

The DEV file of the new library corresponds to a specific MCU type. For example, the BS83B04C, BS83B08C and BS83B12C libraries have their own independent DEV files. These files name the MCU registers for libraries to use. To add a new device, it is only needed to add a DEV file without requiring to modify the original library.

2. Description of .DEF File

This file is used to extend the library programs, especially for exclusive communication codes such as IIC or UART. For example, if it is required to use IIC to obtain the addresses of library different locations, just write a DEF file for specific addresses. This file is attached to the library when in use, and users can choose to attach any kind of DEF file to obtain different locations' data according to the actual requirements.

3. Description of .INI File

This file is used to assist with numerical modifications in the OBJ file and to solve the problem that the older version of libraries must load touch key library parameters in MAIN_PROGRAM. The benefits of this file are that it has moved most of the library functions to the INI file, allowing users to save ROM space by turning off unused functions in the INI file and to modify partial underlying touch key parameters.