MPLAB Harmony Prebuilt Libraries Help

MPLAB Harmony Integrated Software Framework

Prebuilt Libraries

This section provides information on the MPLAB X IDE projects that are provided to build binary (i.e., .a file) versions of key MPLAB Harmony libraries.

Introduction

This section describes the MPLAB X IDE projects that are provided to build binary (i.e., `.a` file) versions of key MPLAB Harmony libraries.

Description

Source code is provided for most of the libraries included in the MPLAB Harmony installation. However, some MPLAB Harmony libraries are provided in prebuilt binary format (`.a` file) because source code is not released or to provide optimal performance for users of the free version of the compiler. The MPLAB X IDE projects used to build these libraries are provided so that you can rebuild these libraries with different build parameters, optimization settings, and/or debug symbols if desired.

The Build Projects that are provided in MPLAB Harmony are located in the following folder: `<install-dir>/build/framework`.

Note: Building these libraries with high optimization settings requires a fully licensed version of the necessary MPLAB XC compiler.

Supported Libraries

This topic provides information on the supported libraries that are provided in MPLAB Harmony.

DSP Fixed-Point Math Library

This section describes MPLAB X IDE projects used to build the MPLAB Harmony DSP math libraries.

Description

The DSP Fixed-Point Math Library consists of routines that are optimized in assembly to take advantage of the microAptiv™ core of PIC32MZ devices. The library operates on fixed point integers, which are scaled to represent floating point numbers. Many functions are available in both 16-bit and 32-bit numerical formats.

The library contains functions for mathematical operations on a vector (or array) of values, complex scalar values and matrixes. Operations include add, subtract, multiply, power, data generation, and in the cases of vectors more complex functions including statistics.

The DSP Math Library also contains a full set of digital filtering functions which include FIR and IIR primitives, as well as more complex architectures (e.g., cascade and parallel bi-quad structures).

Finally the library contains a number of transform functions include FFT and inverse FFT, as well as a number of windowing functions.

Some functions in the DSP Math Library require the use of the LibQ Fixed Point Math library as well. In those cases the LibQ library must also be installed with your project. Details on dependent functions are found in the DSP Fixed-Point Library section within the remarks section for that specific function.

LibQ Fixed-Point Math Library

This section describes MPLAB X IDE projects used to build the MPLAB Harmony fixed-point math LibQ libraries.

Description

The LibQ Fixed-Point Math Library consists of routines in optimized assembly for performing advanced mathematical operations on a scalar. This is similar in construction to the floating point equivalent `math.h` file that is included with the compiler. All functions are performed using fixed-point integer operations and generally available in both high precision and low precision variants. Lower precision functions generally perform faster and may be suited for time critical operations.

PIC32 Bluetooth Stack Library

This section describes MPLAB X IDE projects used to build the MPLAB Harmony PIC32 Bluetooth Stack libraries.

Description

The PIC32 Bluetooth Stack Library is provided only in binary form. It consists of a large number of routines that enable the interface of a PIC32 system to a Bluetooth radio via a Hardware Communication Interface (HCI) controller and a UART port. The communication is enabled by a Simple Secured Pairing (SSP) and data is transmitted through the Bluetooth Serial Port Profile (SPP).

The SPP-only library, `lib_bluetooth_spp_only_<version>.a`, is located in the directory `<install-dir>/bin/framework/bluetooth`. Support for audio is provided as well, as part of a premium package, in the library, `lib_bluetooth_<version>.a`, which is located in `<install-dir>/bin/framework/bluetooth/premium/audio`.

Premium Bluetooth Audio applications also need at least one audio decoder. The Bluetooth specification requires that every Bluetooth audio application support a Sub-Band Coding (SBC) decoder. Support for a SBC decoder is provided in the library, `lib_sbc_decoder_<version>.a`, which is located in the same directory as the Premium Bluetooth Stack Library: `<install-dir>/bin/framework/bluetooth/premium/audio`.

The library interface allows for an external header file to control the port via Service Discovery Protocol (SDP), so customized interfaces are possible through the API. All API functions are exposed in the header file group, which is located within the `<install-dir>/framework/bluetooth/cdbt` folder of your MPLAB Harmony installation.

Assembly Language Issue

There is an assembly language issue in the SBC Decoder Library that misaligns the stack pointer. Contrary to the MIPS programming specification, this error is benign except when the application is built for a device with a hardware Floating Point Unit (FPU), such as the PIC32MZ EF family of devices. In this case, the sub-routine context-saving operations will produce an exception and stop the application. A work around exists for this issue, which involves disabling FPU context-saving for Interrupt Service Routines (ISRs).

The work around is to convert all ISRs from:

```
void __ISR(vector,ipl) MyInterruptHandler(void)
```

and change them to:

```
void __attribute__((vector),interrupt(ipl),nomips16,no_fpu)) MyInterruptHandler(void)
```

where, `vector` and `ipl` are replaced with the actual interrupt vector and interrupt priority level.

So as long as no FPU operations occur in ISRs and the routines are called by ISRs, this work around will prevent the misaligned stack pointer from causing an exception. If FPU operations result from an ISR with no_fpu enabled, a general exception will result.

Peripheral Libraries

This section describes MPLAB X IDE projects used to build the MPLAB Harmony peripheral libraries.

Description

The MPLAB Harmony peripheral libraries are implemented almost entirely using C language inline functions. This is done for efficiency so that the compiler can generate a few simple instructions in place of multiple layers of function calls when function parameters are passed as constants. However, if the project that uses the peripheral library is built with low optimization settings, the compiler may generate a function call instead of "inlining" the function implementation. Unfortunately, this will cause an undefined symbol error at link time if the linker cannot find an actual implementation of the function to which it can link the call.

To satisfy the linker (and to provide optimized PLIB operation, even when low compiler optimization settings are used in your project), the MPLAB Harmony peripheral libraries must be prebuilt as `.a` file, linkable libraries. The appropriate MPLAB X IDE peripheral library `.a` file for the processor in use must be added to the MPLAB X IDE project so that the linker can use them.

Note: Prebuilt binary `.a` files are provided for all supported processors in the MPLAB Harmony installation in the following folder: `<install-dir>/bin/framework/peripheral`.

The MPLAB Harmony peripheral library `.a` file build projects are provided so that you may rebuild the binary peripheral library `.a` files using any desired optimization settings. However, if you do this, you must have the appropriate XC compiler and you must copy the resultant `.a` file to your project folder and add it to your MPLAB X IDE project.

Build Configurations

There are two MPLAB X IDE configurations from which to choose when building peripheral libraries for MPLAB Harmony:

- device_with_ethernet - Select this configuration if the device for which the PLIB `.a` file is to be created *has* an Ethernet module. The default device for this configuration is PIC32MX795F512L.
- device_without_ethernet - Select this configuration if the device for which the PLIB `.a` file is to be created *does not* have an Ethernet module. The default device for this configuration is PIC32MX450F256L.

The default Optimization level for both configurations is O3.

Key Configuration Options

To build the peripheral library binary `.a` files, there are two key configuration options that must be defined, as follows:

```
#define PLIB_INLINE_API extern
#define PLIB_INLINE static inline
```

These two configuration items are defined in the source file (`peripheral.c`), which then directly includes the peripheral library implementation headers.

Note: This has already been done for you. This information is provided strictly for reference.

During normal peripheral library use, both of these options are defined as "extern inline" so that the compiler can choose between generating a function call or generating inline code, directly in the calling function (as described previously). However, to build the binary library `.a` files the peripheral library Application Program Interface (API) functions must be exposed as global "external" symbols. The "PLIB_INLINE_API" attribute is placed as an attribute on all PLIB API function implementations. So, defining it as "extern" allows the compiler to expose these functions within the library `.a` file so that the linker can find them. The VREG and other "internal" functions prefixed by the "PLIB_INLINE" attribute are then defined as "static inline" functions for efficiency.

Rebuilding the Prebuilt Libraries

This topic provides information on how to use the MPLAB X IDE projects provided in MPLAB Harmony to rebuild the prebuilt libraries.

Description

Prerequisites

To build MPLAB Harmony libraries, you must have the following installed on your development workstation:

- MPLAB X IDE
- MPLAB XC32 C/C++ Compiler
- Appropriate compiler license for your desired optimization settings

Note: Refer to the Release Notes for the specific version numbers. A PDF copy of the release notes is provided in the `<install-dir>/doc` folder of your installation.

Using the MPLAB X IDE Projects

Do the following in MPLAB X IDE:

1. Select the appropriate configuration that is applicable for your device.
2. Select the desired processor in the project properties in MPLAB X IDE.
3. Select the desired optimization levels, debug, capabilities, etc.
4. Build the project to generate the `.a` file.

A post-build step in the project will copy the output (`.a`) of the build process to the `<install_dir>/bin` folder.

Note: Refer to the MPLAB X IDE Help for detailed information on using the Integrated Development Environment.

Index