How to Use the MPLAB Harmony v3 Debug System Service

Abstract

This document describes how to use the Debug System Service in MPLAB® Harmony v3 with an example application. It also covers the architecture, configuration options, and APIs of the Debug System Service.

To get started with MPLAB Harmony v3, and to understand its architecture and usage, refer to the Reference section.

1. Overview

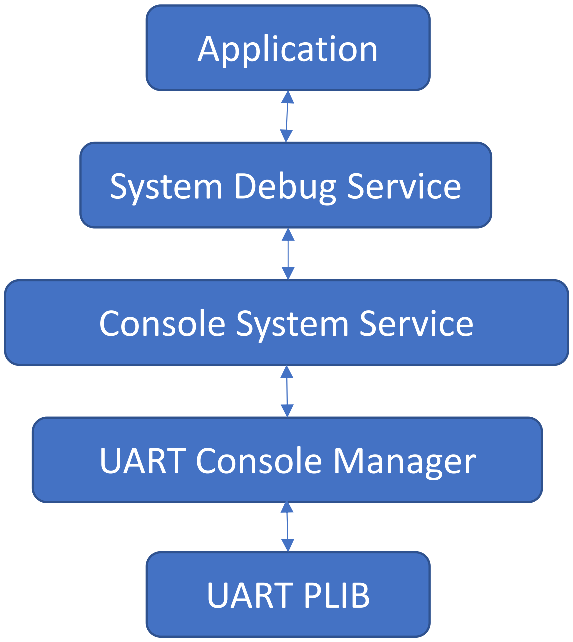
The MPLAB Harmony v3 software framework provides system service libraries to support common functionality and manage resources that are shared between software modules. The MPLAB Harmony v3 software framework offers the following the system service libraries: System Time, DMA, File System, System Debug, System Console, and Cache.

The System Debug and System Console libraries that help to add debug capability to the application code. The debug capabilities help the application in printing various levels of debug messages through the debug console port.

1.1 Architecture

The debug messages from the application are sent to the System Debug and Console Services and then routed to a console manager, which in turn communicates with the underlying hardware.

Figure 1-1. Debug Service Block Diagram



Application
System Debug Service
Console System Service
UART Console Manager
UART PLIB

The application calls the function macros from the system debug module. The user can send formatted messages through the System Debug module. The Debug System Service function calls are routed through the Console System Service layer and a Console Manager layer. The Console System Service consists of core and device (console manager) layers.

The core layer handles module initialization, system functions, and provides common APIs for use in the application and middleware. The device layer contains the functional implementation of the core layer APIs. Each Console device may have a different implementation, but the behavior of the API is uniform across different devices. The Instance and Queue management are handled by the device layer. The Console device layer interacts with the peripheral libraries (PLIBs) to communicate with the hardware peripheral.

1.2 Application Program Interface (APIs)

The following APIs are available in System Debug Service library to handle and output debug messages. These APIs are implemented as a ‘C’ preprocessor directive called macros. For example, the default implementation of these

Macros maps to nothing. This helps the developer to release the code with debug messages without affecting code size and performance.

```
/* When No console device is mapped */
#ifndef SYS_DEBUG_MESSAGE
    #define SYS_DEBUG_MESSAGE(level, message)
#endif

/* When console device is mapped */
#ifndef SYS_DEBUG_MESSAGE
    #define SYS_DEBUG_MESSAGE(level, message)   _SYS_DEBUG_MESSAGE(level, message)
#endif
```

To use the APIs, the user must be aware of System Debug levels defined in the Debug System Service. The user can specify a debug level parameter along with a debug message when calling the `SYS_DEBUG_MESSAGE()` function macro. The following table shows the debug levels defined in the Debug System Service. Once the Debug System Service module is added to the project, configured and the code is generated, the Debug System Service initialization is done as a part of the `SYS_Initialize()` function. While initializing the Debug System Service, the user-defined debug level is mapped as system error level.

The messages with the same or higher priority (other than the system error level) will be printed THROUGH the console.

Tip: The available generic APIs print a debug message without checking the error level.

Table 1-1. System Debug Levels

DEBUG LEVELS (From Higher Priority to Low)	DESCRIPTION
`SYS_ERROR_FATAL`	Errors having possible chances of crashing the system.
`SYS_ERROR_ERROR`	Errors those are deviations from expected behavior.
`SYS_ERROR_WARNING`	Errors which cause unexpected behavior and have possible side effects.
`SYS_ERROR_INFO`	Information about the system parameters and so on.
`SYS_ERROR_DEBUG`	Debug messages are helpful for testing and debugging.

Table 1-2. System Debug Service APIs

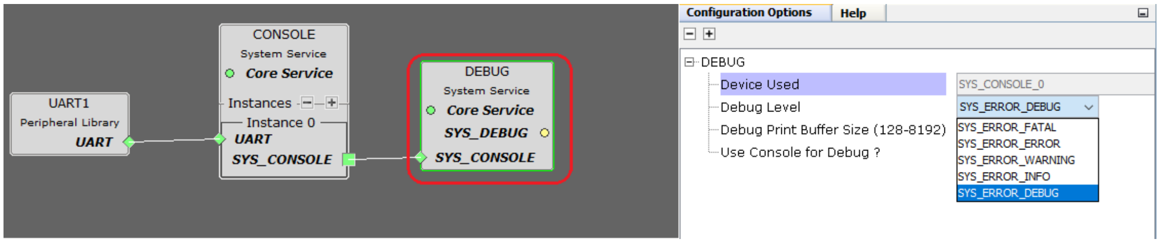
NAME	DESCRIPTION
`SYS_MESSAGE()`	Prints a message to the console regardless of the system error level.
`SYS_DEBUG_MESSAGE()`	Prints a debug message if the system error level is defined at or lower than the level specified. This API accepts `error level` and `message` as parameters. Specified level is the one which is passed as a parameter.
`SYS_PRINT()`	Formats and prints an error message with variable number of arguments, regardless of the system error level.
`SYS_DEBUG_PRINT()`	Formats and prints an error message if the system error level is defined at or lower than the level specified.
`SYS_DEBUG_ErrorLevelGet ()`	Returns the global system error reporting level.
`SYS_DEBUG_ErrorLevelSet ()`	Sets the global system error reporting level.

2. Configuration and Usage

2.1 Configuration Options

The Debug System Service library is configured through the MPLAB Harmony Configurator (MHC).

Figure 2-1. MHC UI Option for Debug System Service



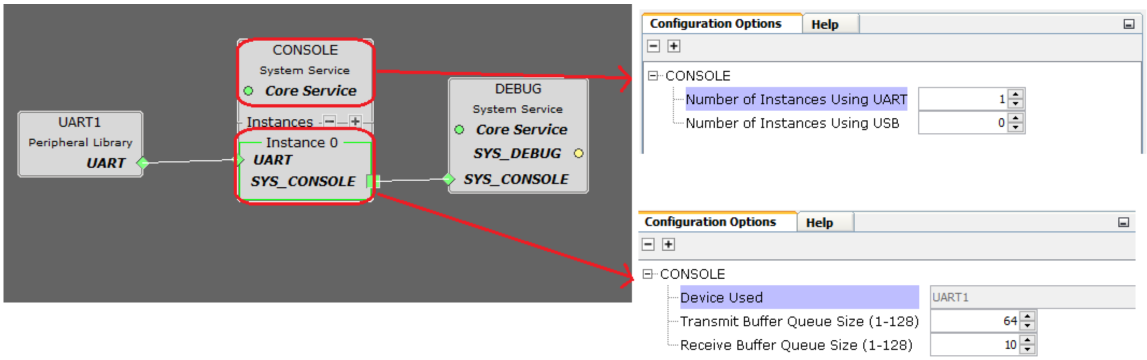
Device Used: SYS_CONSOLE_0
Debug Level: SYS_ERROR_DEBUG
Debug Print Buffer Size (128-8192)
Use Console for Debug ?

The following configuration options are available for users:

- Device Used: To have multiple Console System Services in the project and the user can associate a selected console with the Debug System Service. Currently, a single instance of the console is supported.
- System Debug level: System Error level can be configured by the user from the list of Error levels mentioned in System Debug Levels. For example, if `SYS_ERROR_FATAL` is set as a system error level, then the messages tagged as `SYS_ERROR_FATAL` messages only will be printed. If `SYS_ERROR_DEBUG` is set as the system error level, then all messages tagged from `SYS_ERROR_FATAL` to `SYS_ERROR_DEBUG` messages will be printed.
- Print buffer size: The debug print buffer size can be set from 128 to 8192 bytes. This option helps to fit the Debug Console module into smaller application runs on a device with small code memory.
- Use Console device for debug: Un-checking this option maps the debug macros to nothing. This option helps in keeping the application code uniform with and without the Debug System Service module.

The console system service module also has the following configuration options, as shown in the figure below:.

Figure 2-2. Console System Service Configuration Option



Number of Instances Using UART: 1
Number of Instances Using USB: 0

Device Used: UART1
Transmit Buffer Queue Size (1-128): 64
Receive Buffer Queue Size (1-128): 10

- Number of instances of the UART and USB are used. The number of instances can be changed through the UI option.
  Note: Currently, only the UART interface is supported.
- Each instance of the interface has the TX and RX buffer queue size configuration option.

Users can use the following options according to the resource availability and application requirement.

- The decision of which UART instance to be used for the debug messages is based on the free UART resources available in the customer application.
- Based on the free SRAM availability, the user can configure the print buffer size.
- The number of Transmit (TX) and Receive (RX) queues can come from the customer application requirement. This needs to be configured in the console system service module.

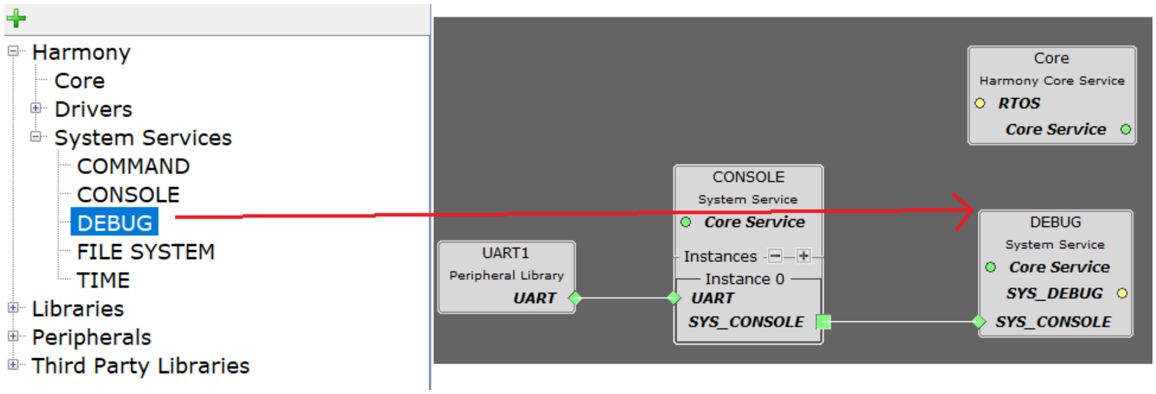
2.2 Example Usage with Existing Driver Demonstrations

The Debug System Service module is a part of the MPLAB Harmony v3 core repository. Typically, it is used in a driver-based application model.

Follow these steps to use the Debug System Service:

1. In MPLAB X IDE, open any application project from the core repository.
2. Open the MHC and add the system debug service to the project from the following location: *core* > *apps* > *driver* > *usart* > *async* > >*usart_echo* > *firmware*.
3. In the Available Components window, select *Harmony* > *System Services* > *DEBUG* and then add it to the project.
4. In the Project Graph section, notice the Debug System Service module has a dependency on the Console System Service and UART PLIB modules. Add all these modules to meet the dependency.
   Note: While adding the System Debug Service module, the RTOS module addition will pop-up. If the project is non-RTOS based, click **NO** to avoid adding the RTOS module into the project.

Figure 2-3. System Debug Service Module Dependencies



5. Add and configure the UART PLIB module and regenerate the code. Ensure UART pins are configured in the Pin Manager before regenerating the code.
6. Use available debug APIs to add debug messages from the application. The `APP_Tasks()` function is typically a state machine implementation. The debug APIs can be called from `APP_tasks()` to output the debug messages.

```
case APP_STATE_RECEIVE_DATA:

    DRV_USART_ReadBufferAdd(Handle, readBuffer, APP_DATA_SIZE, bufferHandle);
    if (bufferHandle != BUFFER_HANDLE_INVALID)
    {
        appData.state = APP_STATE_WAIT_RECEIVE_COMPLETE;

        /*   Debug Message from Debug System Service */
        SYS_DEBUG_MESSAGE(SYS_ERROR_DEBUG,"Data is Received \r\n");
    }
    else
    {
        appData.state = APP_STATE_ERROR;
    }
    break;
```

2.3 Example Usage with Existing PLIB Demonstrations

The system debug service can also be used in a PLIB-based application model. Follow these steps to use the Debug System Service in a PLIB-based application model.

Note: Ensure the **core** repository is enabled while launching the MHC.

1. In MPLAB X IDE, open any PLIB application project from the *csp* repository. Open MHC and add the System Debug Service to the project. For example, the PLIB demo repository file structures are as follows: *csp* > *apps* > *systick* > *systick_periodic_timeout* > *firmware* > *Select board of user's choice*.
2. In the Available Components window, select *Harmony* > *System Services* > *DEBUG* and add it to the project.
   Note: Ensure the **core** repository is selected while launching the MHC.
3. In the Project Graph, add and connect the dependent (Console System Service and UART PLIB) modules of the Debug System Service.
   Note: While adding the system debug service module, the core module addition will pop-up. The user must click **Yes**, even if it is PLIB-based project (which does not use any middleware or the MPLAB Harmony v3 driver). The user must include the core module in the project.
4. Add and configure the UART PLIB module, and then regenerate the code. Ensure UART pins are configured in the Pin Manager before regenerating the code.
5. Use the available System Debug APIs to add debug messages from the application. The System Debug APIs can be called from `main()` in a PLIB-based application.

```
int main ( void )
{
    /* Initialize all modules */
    SYS_Initialize ( NULL );
    SYSTICK_TimerStart();
    IO_DEBUG_Set();

    printf("=== PLIB Example Project === \r\n");

    while ( true )
    {
        IO_DEBUG_Toggle();
        SYS_DEBUG_MESSAGE(SYS_ERROR_DEBUG, "\r\n You are reading a debug message !!");
        SYSTICK_DelayMs(1000);
    }

    /* Execution should not come here during normal operation */

    return ( EXIT_FAILURE );
```

2.4 Debug and Console System Service

Users can call the Debug System Service macros or the Console System Service functions from the application directly. Calling the macros from the Debug System Service has the following advantage:.

- Sending formatted messages is possible when using the Debug System Service Macros.
- Debug messages can be sent based on a system defined debug level parameter.
- Debug messages can be present in the application code, whether there is a debug interface associated with it or not.

The Console System Service has the following features.

- Provides APIs to receive the user input from the console in addition to providing APIs to print debug messages on the console
- The Console System Service provides APIs, which return a callback on the completion of the transfer or receive operation. This is useful for applications which need to take an action based on the completion of a transfer or receive operation. For example, an application may need to change the state to Sleep mode after a message is printed on the terminal console.

3. References

1. Microchip Developer Help:
   https://microchipdeveloper.com/harmony3:start
2. MPLAB® Harmony :
   https://www.microchip.com/mplab/mplab-harmony
3. MPLAB® Harmony GitHUB wiki :
   https://github.com/Microchip-MPLAB-Harmony/Microchip-MPLAB-Harmony.github.io/wiki
4. How to Setup MPLAB Harmony v3 Software Development Framework:
   http://ww1.microchip.com/downloads/en/DeviceDoc/How_to%20_Setup_MPLAB_%20Harmonyv3_%20Software_%20Development_Framework_DS90003232A.pdf

The Microchip Website

Microchip provides online support via our website at http://www.microchip.com/. This website is used to make files and information easily available to customers. Some of the content available includes:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip design partner program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Product Change Notification Service

Microchip's product change notification service helps keep customers current on Microchip products. Subscribers will receive email notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, go to http://www.microchip.com/pcn and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Embedded Solutions Engineer (ESE)
- Technical Support

Customers should contact their distributor, representative or ESE for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in this document.

Technical support is available through the website at: http://www.microchip.com/support

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Legal Notice

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE. Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Trademarks

ISBN: 978-1-5224-5661-2

Quality Management System

For information regarding Microchip's Quality Management Systems, please visit http://www.microchip.com/quality.

Worldwide Sales and Service

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://www.microchip.com/support
Web Address:
http://www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Austin, TX
Tel: 512-257-3370

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Novi, MI
Tel: 248-848-4000

Houston, TX
Tel: 281-894-5983

Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453
Tel: 317-536-2380

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608
Tel: 951-273-7800

Raleigh, NC
Tel: 919-844-7510

New York, NY
Tel: 631-435-6000

San Jose, CA
Tel: 408-735-9110
Tel: 408-436-4270

Canada - Toronto
Tel: 905-695-1980
Fax: 905-695-2078

ASIA/PACIFIC

Australia - Sydney
Tel: 61-2-9868-6733

China - Beijing
Tel: 86-10-8569-7000

China - Chengdu
Tel: 86-28-8665-5511

China - Chongqing
Tel: 86-23-8980-9588

China - Dongguan
Tel: 86-769-8702-9880

China - Guangzhou
Tel: 86-20-8755-8029

China - Hangzhou
Tel: 86-571-8792-8115

China - Hong Kong SAR
Tel: 852-2943-5100

China - Nanjing
Tel: 86-25-8473-2460

China - Qingdao
Tel: 86-532-8502-7355

China - Shanghai
Tel: 86-21-3326-8000

China - Shenyang
Tel: 86-24-2334-2829

China - Shenzhen
Tel: 86-755-8864-2200

China - Suzhou
Tel: 86-186-6233-1526

China - Wuhan
Tel: 86-27-5980-5300

China - Xian
Tel: 86-29-8833-7252

China - Xiamen
Tel: 86-592-2388138

China - Zhuhai
Tel: 86-756-3210040

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-3090-4444

India - New Delhi
Tel: 91-11-4160-8631

India - Pune
Tel: 91-20-4121-0141

Japan - Osaka
Tel: 81-6-6152-7160

Japan - Tokyo
Tel: 81-3-6880- 3770

Korea - Daegu
Tel: 82-53-744-4301

Korea - Seoul
Tel: 82-2-554-7200

Malaysia - Kuala Lumpur
Tel: 60-3-7651-7906

Malaysia - Penang
Tel: 60-4-227-8870

Philippines - Manila
Tel: 63-2-634-9065

Singapore
Tel: 65-6334-8870

Taiwan - Hsin Chu
Tel: 886-3-577-8366

Taiwan - Kaohsiung
Tel: 886-7-213-7830

Taiwan - Taipei
Tel: 886-2-2508-8600

Thailand - Bangkok
Tel: 66-2-694-1351

Vietnam - Ho Chi Minh
Tel: 84-28-5448-2100

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

Finland - Espoo
Tel: 358-9-4520-820

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Garching
Tel: 49-8931-9700

Germany - Haan
Tel: 49-2129-3766400

Germany - Heilbronn
Tel: 49-7131-72400

Germany - Karlsruhe
Tel: 49-721-625370

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Germany - Rosenheim
Tel: 49-8031-354-560

Israel - Ra'anana
Tel: 972-9-744-7705

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Italy - Padova
Tel: 39-049-7625286

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Norway - Trondheim
Tel: 47-72884388

Poland - Warsaw
Tel: 48-22-3325737

Romania - Bucharest
Tel: 40-21-407-87-50

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

Sweden - Gothenberg
Tel: 46-31-704-60-40

Sweden - Stockholm
Tel: 46-8-5090-4654

UK - Wokingham
Tel: 44-118-921-5800
Fax: 44-118-921-5820