Microchip Class B Safety Software Help

PIC18F Family

Table of Contents

1 Introduction

This document describes the Class B Safety Software Library routines that detect the occurrence of Faults in a single channel CPU. These routines have been developed in accordance with the IEC 60730 standard to support the Class B certification process. These routines can be directly integrated with the end user's application to test and verify the critical functionalities of a microcontroller without affecting the end user's application. The Class B Safety Software Library routines can be called at start-up or periodically during run-time to test the following components:

- CPU Registers
- CPU Program Counter
- Invariable Memory
- Variable Memory
- Clock

2 Software License Agreement

MICROCHIP IS WILLING TO LICENSE THE ACCOMPANYING SOFTWARE AND DOCUMENTATION TO YOU ONLY ON THE CONDITION THAT YOU ACCEPT ALL OF THE FOLLOWING TERMS. TO ACCEPT THE TERMS OF THIS LICENSE, CLICK "I ACCEPT" AND PROCEED WITH THE DOWNLOAD OR INSTALL. IF YOU DO NOT ACCEPT THESE LICENSE TERMS, CLICK "I DO NOT ACCEPT," AND DO NOT DOWNLOAD OR INSTALL THIS SOFTWARE.

NON-EXCLUSIVE SOFTWARE LICENSE AGREEMENT

This Nonexclusive Software License Agreement ("Agreement") is a contract between you, your heirs, successors and assigns ("Licensee") and Microchip Technology Incorporated, a Delaware corporation, with a principal place of business at 2355 W. Chandler Blvd., Chandler, AZ 85224-6199, and its subsidiary, Microchip Technology (Barbados) II Incorporated (collectively, "Microchip") for the accompanying Microchip software including, but not limited to, Graphics Library Software, IrDA Stack Software, MCHPFSUSB Stack Software, Memory Disk Drive File System Software, mTouch(TM) Capacitive Library Software, Smart Card Library Software, TCP/IP Stack Software, MiWi(TM) DE Software, Security Package Software, and/or any PC programs and any updates thereto (collectively, the "Software"), and accompanying documentation, including images and any other graphic resources provided by Microchip ("Documentation").

1. Definitions. As used in this Agreement, the following capitalized terms will have the meanings defined below:
a. "Microchip Products" means Microchip microcontrollers and Microchip digital signal controllers.
b. "Licensee Products" means Licensee products that use or incorporate Microchip Products.
c. "Object Code" means the Software computer programming code that is in binary form (including related documentation, if any), and error corrections, improvements, modifications, and updates.
d. "Source Code" means the Software computer programming code that may be printed out or displayed in human readable form (including related programmer comments and documentation, if any), and error corrections, improvements, modifications, and updates.
e. "Third Party" means Licensee's agents, representatives, consultants, clients, customers, or contract manufacturers.
f. "Third Party Products" means Third Party products that use or incorporate Microchip Products.
2. Software License Grant. Microchip grants strictly to Licensee a non-exclusive, non-transferable, worldwide license to:
a. use the Software in connection with Licensee Products and/or Third Party Products;
b. if Source Code is provided, modify the Software; provided that Licensee clearly notifies Third Parties regarding the source of such modifications;
c. distribute the Software to Third Parties for use in Third Party Products, so long as such Third Party agrees to be bound by this Agreement (in writing or by "click to accept") and this Agreement accompanies such distribution;
d. sublicense to a Third Party to use the Software, so long as such Third Party agrees to be bound by this Agreement (in writing or by "click to accept");
e. with respect to the TCP/IP Stack Software, Licensee may port the ENC28J60.c, ENC28J60.h, ENCX24J600.c, and ENCX24J600.h driver source files to a non-Microchip Product used in conjunction with a Microchip ethernet controller;
f. with respect to the MiWi (TM) DE Software, Licensee may only exercise its rights when the Software is embedded on a Microchip Product and used with a Microchip radio frequency transceiver or UBEC UZ2400 radio frequency transceiver which are integrated into Licensee Products or Third Party Products.
For purposes of clarity, Licensee may NOT embed the Software on a non-Microchip Product, except as described in this Section.
3. Documentation License Grant. Microchip grants strictly to Licensee a non-exclusive, non-

transferable, worldwide license to use the Documentation in support of Licensee's authorized use of the Software

4. Third Party Requirements. Licensee acknowledges that it is Licensee’s responsibility to comply with any third party license terms or requirements applicable to the use of such third party software, specifications, systems, or tools. This includes, by way of example but not as a limitation, any standards setting organizations requirements and, particularly with respect to the Security Package Software, local encryption laws and requirements. Microchip is not responsible and will not be held responsible in any manner for Licensee’s failure to comply with such applicable terms or requirements.

5. Open Source Components. Notwithstanding the license grant in Section 1 above, Licensee further acknowledges that certain components of the Software may be covered by so-called "open source" software licenses ("Open Source Components"). Open Source Components means any software licenses approved as open source licenses by the Open Source Initiative or any substantially similar licenses, including without limitation any license that, as a condition of distribution of the software licensed under such license, requires that the distributor make the software available in source code format. To the extent required by the licenses covering Open Source Components, the terms of such license will apply in lieu of the terms of this Agreement. To the extent the terms of the licenses applicable to Open Source Components prohibit any of the restrictions in this Agreement with respect to such Open Source Components, such restrictions will not apply to such Open Source Component.

6. Licensee Obligations. Licensee will not: (a) engage in unauthorized use, modification, disclosure or distribution of Software or Documentation, or its derivatives; (b) use all or any portion of the Software, Documentation, or its derivatives except in conjunction with Microchip Products, Licensee Products or Third Party Products; or (c) reverse engineer (by disassembly, decompilation or otherwise) Software or any portion thereof. Licensee may not remove or alter any Microchip copyright or other proprietary rights notice posted in any portion of the Software or Documentation. Licensee will defend, indemnify and hold Microchip and its subsidiaries harmless from and against any and all claims, costs, damages, expenses (including reasonable attorney's fees), liabilities, and losses, including without limitation: (x) any claims directly or indirectly arising from or related to the use, modification, disclosure or distribution of the Software, Documentation, or any intellectual property rights related thereto; (y) the use, sale and distribution of Licensee Products or Third Party Products; and (z) breach of this Agreement.

7. Confidentiality. Licensee agrees that the Software (including but not limited to the Source Code, Object Code and library files) and its derivatives, Documentation and underlying inventions, algorithms, know-how and ideas relating to the Software and the Documentation are proprietary information belonging to Microchip and its licensors ("Proprietary Information"). Except as expressly and unambiguously allowed herein, Licensee will hold in confidence and not use or disclose any Proprietary Information and will similarly bind its employees and Third Party(ies) in writing. Proprietary Information will not include information that: (i) is in or enters the public domain without breach of this Agreement and through no fault of the receiving party; (ii) the receiving party was legally in possession of prior to receiving it; (iii) the receiving party can demonstrate was developed by the receiving party independently and without use of or reference to the disclosing party's Proprietary Information; or (iv) the receiving party receives from a third party without restriction on disclosure. If Licensee is required to disclose Proprietary Information by law, court order, or government agency, License will give Microchip prompt notice of such requirement in order to allow Microchip to object or limit such disclosure. Licensee agrees that the provisions of this Agreement regarding unauthorized use and nondisclosure of the Software, Documentation and related Proprietary Rights are necessary to protect the legitimate business interests of Microchip and its licensors and that monetary damage alone cannot adequately compensate Microchip or its licensors if such provisions are violated. Licensee, therefore, agrees that if Microchip alleges that Licensee or Third Party has breached or violated such provision then Microchip will have the right to injunctive relief, without the requirement for the posting of a bond, in addition to all other remedies at law or in equity.

8. Ownership of Proprietary Rights. Microchip and its licensors retain all right, title and interest in and to the Software and Documentation including, but not limited to all patent, copyright, trade secret and other intellectual property rights in the Software, Documentation, and underlying technology and all copies and derivative works thereof (by whomever produced). Licensee and Third Party use of such

modifications and derivatives is limited to the license rights described in this Agreement.
9. Termination of Agreement. Without prejudice to any other rights, this Agreement terminates immediately, without notice by Microchip, upon a failure by Licensee or Third Party to comply with any provision of this Agreement. Upon termination, Licensee and Third Party will immediately stop using the Software, Documentation, and derivatives thereof, and immediately destroy all such copies.
10. Warranty Disclaimers. THE SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY, TITLE, NON-INFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE. MICROCHIP AND ITS LICENSORS ASSUME NO RESPONSIBILITY FOR THE ACCURACY, RELIABILITY OR APPLICATION OF THE SOFTWARE OR DOCUMENTATION. MICROCHIP AND ITS LICENSORS DO NOT WARRANT THAT THE SOFTWARE WILL MEET REQUIREMENTS OF LICENSEE OR THIRD PARTY, BE UNINTERRUPTED OR ERROR-FREE. MICROCHIP AND ITS LICENSORS HAVE NO OBLIGATION TO CORRECT ANY DEFECTS IN THE SOFTWARE.
11. Limited Liability. IN NO EVENT WILL MICROCHIP OR ITS LICENSORS BE LIABLE OR OBLIGATED UNDER ANY LEGAL OR EQUITABLE THEORY FOR ANY DIRECT OR INDIRECT DAMAGES OR EXPENSES INCLUDING BUT NOT LIMITED TO INCIDENTAL, SPECIAL, INDIRECT, PUNITIVE OR CONSEQUENTIAL DAMAGES, LOST PROFITS OR LOST DATA, COST OF PROCUREMENT OF SUBSTITUTE GOODS, TECHNOLOGY, SERVICES, OR ANY CLAIMS BY THIRD PARTIES (INCLUDING BUT NOT LIMITED TO ANY DEFENSE THEREOF), OR OTHER SIMILAR COSTS. The aggregate and cumulative liability of Microchip and its licensors for damages hereunder will in no event exceed $1000 or the amount Licensee paid Microchip for the Software and Documentation, whichever is greater. Licensee acknowledges that the foregoing limitations are reasonable and an essential part of this Agreement.
12. General. THIS AGREEMENT WILL BE GOVERNED BY AND CONSTRUED UNDER THE LAWS OF THE STATE OF ARIZONA AND THE UNITED STATES WITHOUT REGARD TO CONFLICTS OF LAWS PROVISIONS. Licensee agrees that any disputes arising out of or related to this Agreement, Software or Documentation will be brought exclusively in either the U.S. District Court for the District of Arizona, Phoenix Division, or the Superior Court of Arizona located in Maricopa County, Arizona. This Agreement will constitute the entire agreement between the parties with respect to the subject matter hereof. It will not be modified except by a written agreement signed by an authorized representative of Microchip. If any provision of this Agreement will be held by a court of competent jurisdiction to be illegal, invalid or unenforceable, that provision will be limited or eliminated to the minimum extent necessary so that this Agreement will otherwise remain in full force and effect and enforceable. No waiver of any breach of any provision of this Agreement will constitute a waiver of any prior, concurrent or subsequent breach of the same or any other provisions hereof, and no waiver will be effective unless made in writing and signed by an authorized representative of the waiving party. Licensee agrees to comply with all import and export laws and restrictions and regulations of the Department of Commerce or other United States or foreign agency or authority. The indemnities, obligations of confidentiality, and limitations on liability described herein, and any right of action for breach of this Agreement prior to termination, will survive any termination of this Agreement. Any prohibited assignment will be null and void. Use, duplication or disclosure by the United States Government is subject to restrictions set forth in subparagraphs (a) through (d) of the Commercial Computer-Restricted Rights clause of FAR 52.227-19 when applicable, or in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013, and in similar clauses in the NASA FAR Supplement. Contractor/manufacturer is Microchip Technology Inc., 2355 W. Chandler Blvd., Chandler, AZ 85224-6199.

If Licensee has any questions about this Agreement, please write to Microchip Technology Inc., 2355 W. Chandler Blvd., Chandler, AZ 85224-6199 USA. ATTN: Marketing.

3 Getting Started

The folder structure of the Class B Safety Software Library is shown below:

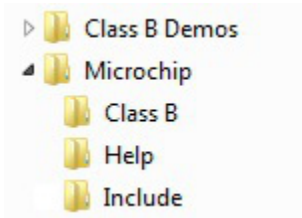


Figure 1: Folder Structure

- The …\Class B Demos folder will contain the library demo projects.
- The …\Microchip folder will contain the library components.
- The …\Class B sub-folder will contain all C files related to the library.
- The …\Help sub-folder will contain this document.
- The …\Include sub-folder will contain all Header files related to the library.

3.1 Class B Library Files

Common	
CLASSB.h	This file contains all of the function declarations for all routines in the library. This file must be included in the main application project.
CLASSB_Config.h	This file contains the user-defined configurations for the Class B routines. This file must be included before CLASSB.h in the main application project.
CLASSB_Types.h	This file contains the types used in the library.
CPU Registers	
CLASSB_CPURegistersTest.h	Supporting file for Register test.
CLASSB_CPURegistersTest.c	Routines for CPU Registers test.
CPU Program Counter	
CLASSB_CPUPCTest.h	Supporting file for Program Counter test.
CLASSB_CPUPCTest.c	Routines for CPU Program Counter test.
Invariable Memory	
CLASSB_CRCEEPROMTest.h	Supporting file for EEPROM CRC test.
CLASSB_CRCEEPROMTest.c	Routines for CRC of EEPROM test.
CLASSB_CRCFlashTest.h	Supporting file for Flash CRC test.
CLASSB_CRCFlashTest.c	Routines for CRC of Flash test.
Variable Memory	

CLASSB_RAMMarchCTest.h	Supporting file for March C memory test.
CLASSB_RAMMarchCTest.c	Routines for RAM March C memory test.
CLASSB_RAMMarchCStackTest.h	Supporting file for March C stack test.
CLASSB_RAMMarchCStackTest.c	Routines for RAM March C stack test.
Clock	
CLASSB_ClockTest.h	Supporting file for Clock accuracy test.
CLASSB_ClockTest.c	Routines for testing CPU clocks against known external clock source.
CLASSB_ClockLineFreqTest.h	Supporting file for Clock accuracy using line frequency test.
CLASSB_ClockLineFreqTest.c	Routines for testing CPU clocks against external line frequency.

3.2 Class B Library Configuration

The Class B Library requires a few user-defined configurations located in CLASSB_config.h. This is where the tests used in the end-user application will be defined, as well as a few part-specific features that will need to be enabled or disabled.

This is shown here:

```
/**************************************************************************
* CLASSB_DEVICE_FLASH_SIZE_KB
* Defines the size of the flash in K words.
* Used in CLASSB_CPUPCTest
* Valid sizes include: 2, 4, 8, 16, 32, 64, 128
**************************************************************************/
#define CLASSB_DEVICE_FLASH_SIZE_KB 8


/**************************************************************************
* MARCHCMINUS
* Defines if user wants to run the MarchCMinus test.
* Used in CLASSB_MarchCTest and CLASSB_MarchCStackTest.
* Comment out to run redundant March C routine.
**************************************************************************/
```

```
/*****************************************************************************
* CLASSB test function descriptions and enables.
* Default is to enable all tests!
* Comment out each test not used!
*****************************************************************************/
#define CLASSB_MARCHB_TEST_ENABLED
#define CLASSB_MARCHC_TEST_ENABLED
#define CLASSB_MARCHC_STACK_TEST_ENABLED
#define CLASSB_CHECKER_TEST_ENABLED
#define CLASSB_CRCFLASH_TEST_ENABLED
#define CLASSB_CRCEEPROM_TEST_ENABLED
#define CLASSB_CPUREG_TEST_ENABLED
#define CLASSB_CPUPC_TEST_ENABLED
#define CLASSB_LINE_TEST_ENABLED
#define CLASSB_CLOCK_TEST_ENABLED
```

3.3 Class B Library Memory Usage and Tests Execution Time

PRO Compiler (XC8 v1.12)					
Function:	**# tested**	**Cycles**	**Time**	**RAM(bytes)**	**FLASH(words)**
MarchB	80 bytes	83370	20.8425 ms	16	950
MarchC	80 bytes	89744	22.436	16	1094
MarchCMinus	80 bytes	77908	19.477 ms	16	996
MarchCStack	NA	x	x	x	x
Checkerboard	10 bytes	1492	373 us	23	300
CRCFlash	8K addresses	840167	840.167 ms	21	176
CRCEEPROM	254 addresses	17122	4.2805 us	14	162
CPU Reg	19 registers	238	59.5 us	8	498
CPU PC	NA	31	7.75 us	0	46
Clock_LineFreq	NA	NA	USER	x	x
Clock	NA	NA	USER	x	x

Total (allTestsDemo) - MarchCMinus		
Time	**RAM**	**FLASH**
x	x	x

Notes:
1. Processor speed is 16 MHz.
2. All times are estimates based on the MPLABX simulator.
3. All RAM and FLASH usage is test-only (does not include main file).
4. RAM usage is estimated based on MAP file.

4 Class B Library API

This section describes the Class B Library API. The API is setup so that all test functions either return the related information or an industry standard 0 for pass, 1 for fail. This is handled in the CLASSB_Types.h header file and described below.

The function definitions are listed here and described in greater detail below:

- `CLASSBRESULT CLASSB_CPURegistersTest();`
- `CLASSBRESULT CLASSB_CPUPCTest();`
- `uint16_t CLASSB_CRCEEPROMTest(uint16_t startAddress, uint16_t length, uint16_t crcSeed);`
- `uint16_t CLASSB_CRCFlashTest(uint32_t startAddress, uint32_t length, uint16_t crcSeed);`
- `CLASSBRESULT CLASSB_RAMCheckerboardTest(uint8_t* startAddress, uint16_t length, uint8_t* bufferAddress);`
- `CLASSBRESULT CLASSB_RAMMarchBTest();`
- `CLASSBRESULT CLASSB_RAMMarchCTest();`
- `CLASSBRESULT CLASSB_RAMMarchCStackTest();`
- `CLASSBRESULT CLASSB_ClockTest(uint32_t clockFrequency, uint32_t referenceFrequency, size_t msec, uint8_t tolerance);`
- `void CLASSB_ClockLineFreqTest(uint32_t clockFrequency, uint8_t lineFrequency, uint8_t tolerance);`

4.1 CLASSBRESULT

Enumeration:
CLASSBRESULT

Description:
This enumeration is used by the class B test functions to return the results:

CLASSB_TEST_PASS = 0	-	the test finished successfully,
CLASSB_TEST_FAIL	-	the test is failed,
CLASSB_TEST_TIMEOUT	-	the test is failed because a timeout was detected,
CLASSB_TEST_INPROGRESS	-	the test is still in progress.

4.2 CPU Registers Test

Function:
CLASSBRESULT CLASSB_CPURegistersTest()

Description:
Tests CPU registers.

Precondition:
Interrupts must be disabled.

Parameters:
None.

Returns:
Returns zero if successful. Non zero means - failed.

Remarks:
Interrupts must be disabled during the test.
Only CPU specific registers are tested.
Peripheral registers are not tested.

4.3 CPU Program Counter Test

Function:
CLASSBRESULT CLASSB_CPUPCTest()

Description:
Tests CPU program counter.

Precondition:
None.

Parameters:
None.

Returns:
Returns zero if successful. Non zero means - failed.

Remarks:
The device flash memory size must be set using parameter CLASSB_DEVICE_FLASH_SIZE in CLASSB_config.h.

4.4 EEPROM CRC Test

Function:
uint16_t CLASSB_CRCEEPROMTest(startAddress, length, crcSeed)

Description:
Tests the EEPROM memory using the Cyclic Redundancy Check (CRC).

Parameters:
startAddress - the first address of the tested EEPROM memory.
length - the number of EEPROM locations tested.
crcSeed - the initial value of the CRC calculation.

Returns:
Returns the final CRC result.

4.5 Flash CRC Test

Function:
uint16_t CLASSB_CRCFlashTest(startAddress, length, crcSeed)

Description:
Tests the flash memory using the Cyclic Redundancy Check (CRC).

Parameters:
startAddress - the first address of the tested flash memory.
Length - the number of flash locations tested.
crcSeed - the initial value of the CRC calculation.

Returns:
Returns the final CRC result.

4.6 RAM Checkerboard Test

Function:
CLASSBRESULT CLASSB_RAMCheckerboardTest(startAddress, length, bufferAddress)

Description:
Tests the RAM memory region using checker board pattern.

Precondition:
Interrupts must be disabled.

Parameters:
startAddress - the first address of the tested RAM memory,
length - the byte length of the tested RAM memory,
bufferAddress - the first address of the RAM memory to save user data.

Returns:
Returns zero if successful. Non zero means - failed.

Remarks:
The length must be even.
Interrupts must be disabled during the test.

4.7 RAM March B Test

Function:
CLASSBRESULT CLASSB_RAMMarchBTest()

Summary:
Tests the RAM memory using March B algorithm.

Precondition:
Interrupts must be disabled.

Parameters:

CLASSB_MarchstartAddress	- the first address of the tested RAM memory,
CLASSB_MarchLength	- the byte length of the tested RAM memory.
CLASSB_MarchbufferAddress	- the first address of the location in RAM to save user data

Returns:
Returns zero if successful. Non zero means - failed.

Remarks:
Interrupts must be disabled during the test.
If CLASSB_MarchbufferAddress is NULL the test is destructive.
The test uses 7 bytes of RAM for global variables used in test.

4.8 RAM March C Test

Function:
CLASSBRESULT CLASSB_RAMMarchCTest()

Description:
Tests the RAM memory region using March C algorithm.

Precondition:
Interrupts must be disabled.

Parameters:

CLASSB_MarchstartAddress	- the first address of the tested RAM memory,
CLASSB_MarchLength	- the byte length of the tested RAM memory.
CLASSB_MarchbufferAddress	- the first address of the location in RAM to save user data
MARCHCMINUS	- if the parameter is TRUE the "minus" algorithm is used.

Returns:
Returns zero if successful. Non zero means - failed.

Remarks:
Interrupts must be disabled during the test.
If bufferAddress is NULL the test is destructive.
The test uses 7 bytes of RAM for global variables used in test.

4.9 RAM March C Stack Test

Function:
CLASSBRESULT CLASSB_RAMMarchCStackTest()

Description:
Tests the RAM memory region associated with the stack using March C algorithm.

Precondition:
Interrupts must be disabled.

Parameters:

CLASSB_MarchstartAddress	- the first address of the tested RAM memory,
CLASSB_MarchLength	- the byte length of the tested RAM memory.
CLASSB_MarchbufferAddress	- the first address of the location in RAM to save user data
MARCHCMINUS	- if the parameter is TRUE the "minus" algorithm is used.

Returns:
Returns zero if successful. Non zero means - failed.

Remarks:
Interrupts must be disabled during the test.
To run the stack test a minimum of 33 bytes must be allocated for the bufferAddress.

4.10 Clock Test

Function:
CLASSBRESULT CLASSB_ClockTest(clockFrequency, referenceFrequency, msec, tolerance)

Summary:
Tests the system clock using an external crystal into Timer1.

Precondition:
An external oscillator is connected to Timer1.

Parameters:

clockFrequency	- system clock frequency.
referenceFrequency	- reference clock frequency.
msec	- the time in milliseconds to run the test.
tolerance	- the tolerance level of the system oscillator.

Returns:
Returns zero if successful. Non zero means - failed.

Remarks:
Recommend a 20 ms test time for best results.
Will not work with a reference frequency higher than 2MHz.

4.11 Clock Line Frequency Test

Function:
  void CLASSB_ClockLineFreqTest(clockFrequency, lineFrequency, tolerance)

Summary:
  Tests the system clock using the line frequency into the Capture Module (CCP).

Precondition:
  Zero Cross detection circuit is the input to one of the CCP inputs.

Parameters:
  clockFrequency - system clock frequency.
  referenceFrequency - reference clock frequency.
  tolerance - the tolerance level of the system oscillator.

Returns:
  None

Result:
  The result of this test can be accessed two ways:
  1. Through a Function pointer declared above as *ClockLineFreqTestFail or...
  2. Using the ClockLineFreqTestFlag which will follow the definition of CLASSBRESULT

Remarks:
  This is a timing critical test.
  Changes to TMR1 during the progress of this test will cause the test to fail.
  This test takes one second.

5 Using the API

The Class B Library does not require modifications of the linker script to help to ensure compatibility with end-user applications. To use the library functions, the header files CLASSB_Config.h and CLASSB.h must be included in the application source file. Unless otherwise stated, interrupts must be disabled during the Class B Library tests. Failure to disable interrupts can result in memory corruption and false results from the tests.

The following information will help guide the user through the process of adding the Class B Library tests to the end application:

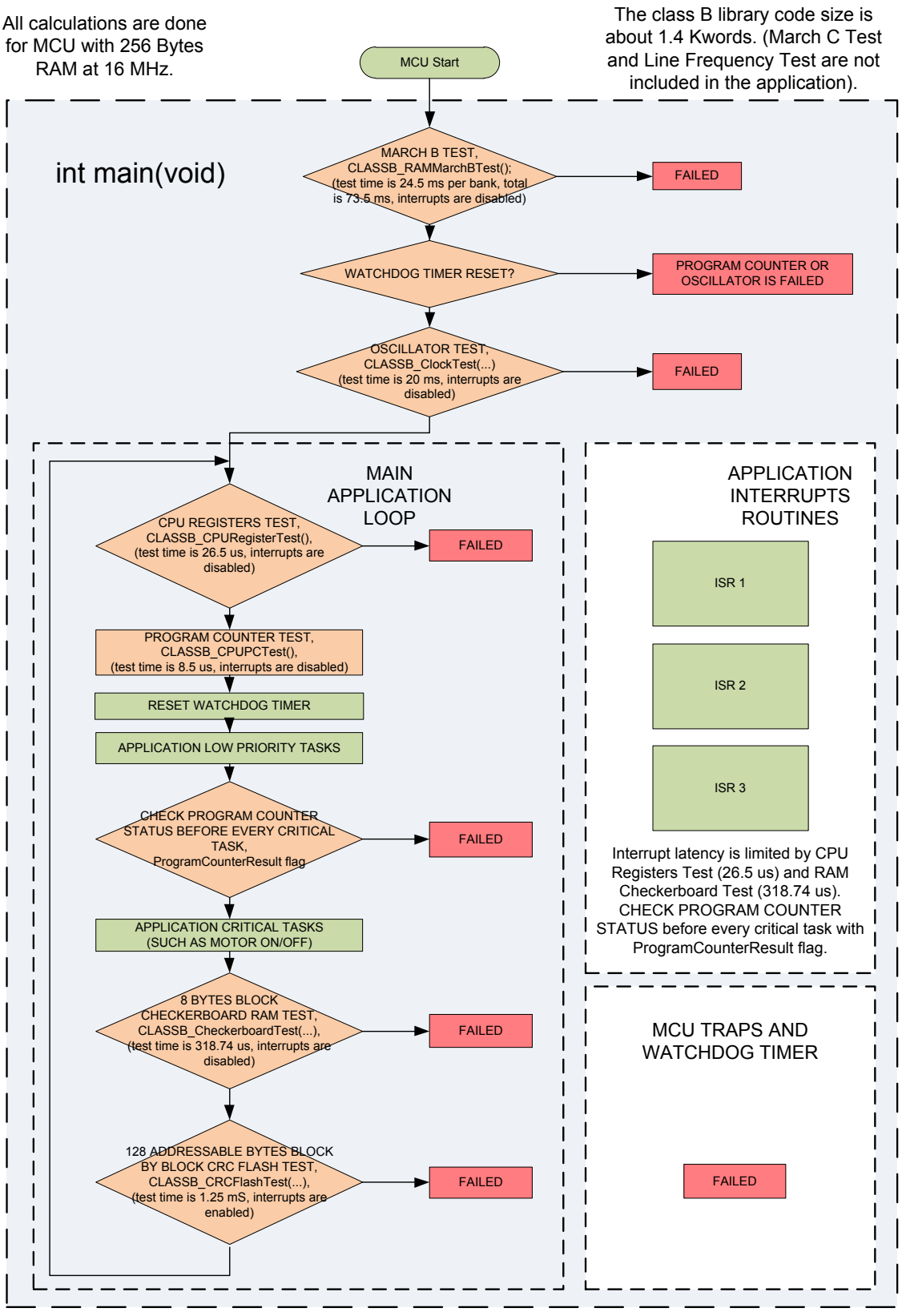
- Typical application flow chart using Class B functions.
- Useful MPLABX Tools.
- Code examples for using the Class B functions.

5.1 Application Flow Chart

The following is a typical application flow chart using the Class B Library functions. This shows an application like motor control with minimal interruptions for Class B testing. Some functions can be called periodically during normal run-time without interrupting the normal flow of the application significantly. Other functions take significantly longer and should be run only at start-up or in small chunks.

It is important to note that in most cases it is not necessary to run both the March B and March C tests, only one needs to be chosen.

The clock test that best fits the application should be chosen. If the application already has access to a zero-cross detection circuit, the ClockLineFreqTest can be chosen without interfering greatly with the



f MPLABX, there are a few useful tools that can be quite helpful for developing an application with he Class B Library.

One such tool is the MAP file. The MAP file will indicate the used memory regions (the start address nd length). The MAP file's default location is:

..\ApplicationProject.X\dist\default\production

Figure 2: Application Flow Chart

The other useful tool for developing Class B applications is the stopwatch. The stopwatch can give the ser important information about the specific timing requirements for the application. This will define when the Class B functions can be called.

The location of the Stopwatch is shown here:

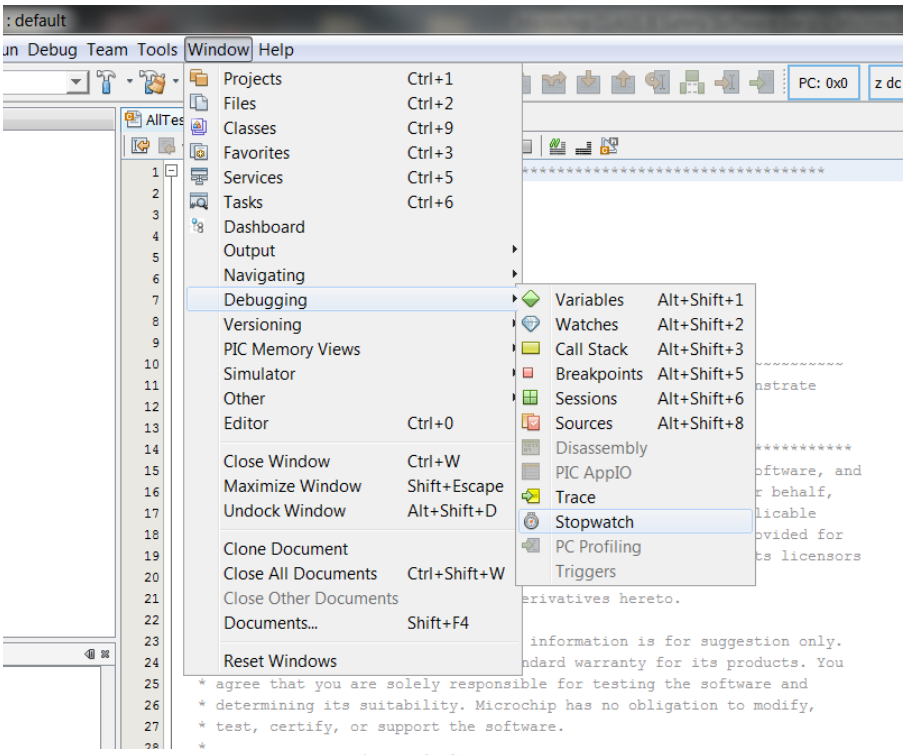


Figure 3: Stopwatch

Additional code examples can also be found in AN_testMain.c in CLASSB_AN_testDemo.X.

```
/*****************************************************************************
  Program Counter Test.
  CPU Registers Test.
*****************************************************************************/
CLASSBRESULT ProgramCounterResult = CLASSB_TEST_INPROGRESS;
//Do a check to see if the Program Counter did not error.
ProgramCounterResult = CLASSB_CPUCTest();
if (ProgramCounterResult == CLASSB_TEST_FAIL)
{
    ErrorMode();
}
if (ProgramCounterResult == CLASSB_TEST_PASS)
{
testResult = CLASSB_CPURegistersTest();
    asm("nop");           /* Test pass */
}
if (testResult == CLASSB_TEST_PASS)
else
{
{
    asm("nop");           /* Test pass */
    ErrorMode();          /* Test not pass */
}
}
else
{
    ErrorMode();          /* Test not pass */
}
```

```
/**********************************************************************
Clock Test:
***********************************************************************/
uint32_t clockFrequency = 16000000; //system clock: 16MHz
uint32_t referenceFrequency = 32768; //reference clock: 32.768kHz
uint8_t msec = 20; //Test time: 20ms
uint8_t tolerance = 5; //Tolerance level: 5%

testResult = CLASSB_ClockTest(clockFrequency, referenceFrequency, msec,
  tolerance);

if (testResult == CLASSB_TEST_PASS)
{
    asm("nop");    /* Test pass */
}
else
{
    ErrorMode();    /* Test not pass */
}
```

```
/**********************************************************************
Clock Line Frequency Test:
***********************************************************************/
uint32_t clockFrequency = 16000000; //system clock: 16MHz
uint32_t lineFrequency = 60; //reference clock: 60Hz zero cross
uint8_t tolerance = 5; //Tolerance level: 5%

CLASSB_ClockLineFreqTest(clockFrequency, lineFrequency, tolerance);
//For this demo the code will wait for the test to finish.
//In application, code can be placed here as long as it is ok to be
//interrupted!
while (ClockLineFreqTestFlag == CLASSB_TEST_INPROGRESS);
if (ClockLineFreqTestFlag == CLASSB_TEST_PASS)
{
    asm("nop");    /* Test pass */
}
else
{
    ErrorMode();    /* Test not pass */
}
```

```
   March B Test. (This example tests BANK 0)
   March C Test. (This example tests Bank 0)
*******************************************************************************/
*******************************************************************************/
   CLASSB_MarchstartAddress = (char*)0x20;
   CLASSB_MarchstartAddress = (char*)0x20;
   CLASSB_MarchbufferAddress = (char*)0xB0;
   CLASSB_MarchbufferAddress = (char*)0xB0;
   CLASSB_MarchLength = 0x10;
   CLASSB_MarchLength = 0x30;

   testResult = CLASSB_RAMMarchBTest();
   testResult = CLASSB_RAMMarchCTest();

   if (testResult==CLASSB_TEST_PASS)
   if (testResult==CLASSB_TEST_PASS)
   {
   {
       asm("nop");      /* Test pass */
       asm("nop");      /* Test pass */
   }
   }
   else
   else
   {
   {
       ErrorMode();     /* Test not pass */
       ErrorMode();     /* Test not pass */
   }
   }
```

```
   March C Stack Test.
   Initial EEPROM CRC test.
*****************************************************************************/
*****************************************************************************/
   CLASSB_MarchbufferAddress = (char*)0x27;
   uint8_t EEPROMAddress = 0x01;
   CLASSB_MarchLength = 33;
   uint8_t length = 254;
   uint16_t crcSeed = 0xFFFF;
   testResult = CLASSB_RAMMarchCStackTest();
   volatile unsigned int CLASSB_EEPROMtestResult;

   if (testResult==CLASSB_TEST_PASS)
   CLASSB_EEPROMtestResult = CLASSB_CRCEEPROMTest(EEPROMAddress, length,
   {
     crcSeed);
       asm("nop");      /* Test pass */

   }
*****************************************************************************
   else
   Initial Flash CRC test.
   {
*****************************************************************************/
       ErrorMode();     /* Test not pass */
   uint16_t FlashAddress = 0x00;
   }
   uint16_t Flashlength = 0x2000;
   crcSeed = 0xFFFF;
   volatile unsigned int CLASSB_FlashtestResult;


   CLASSB_FlashtestResult = CLASSB_CRCFlashTest(FlashAddress, Flashlength,
     crcSeed);

/The Check Values for the Flash and EEPROM can now be saved for future
/comparison or compared to a known constant, depending on the application
```

```
   Checkerboard Test
*************************************************************************/
   uint8_t* CheckerStartAddress;
   uint8_t* CheckerBufferAddress;
   uint8_t CheckerLength;

/In this example we show the checkerboard test used on RAM locations
/0x50 - 0x57. The Checkerboard should be used as a faster alternative
/to the March Tests during run-time. Safety specific RAM locations
/should be tested with this method.
   CheckerStartAddress = (char*)0x50;
   CheckerBufferAddress = (char*)0x58;
   CheckerLength = 8;

   testResult = CLASSB_RAMCheckerboardTest(CheckerStartAddress,
     CheckerLength, CheckerBufferAddress);

   if (testResult==CLASSB_TEST_PASS)
   {
       asm("nop");      /* Test pass */
   }
   else
   {
       ErrorMode();     /* Test not pass */
   }
```

o get more information about the Class B Safety Software Library visit:

ttp://www.microchip.com/pagehandler/en-us/technology/homeAppliance/classbsafetysoftware.html

he following Articles are also available:

- AN1229 – Class B Safety Software Library for PIC® MCUs and dsPIC® DSCs
- Future AN - Using Hardware or Software CRC with Enhanced Core PIC16F1xxx in Class B Applications

/10/2014……First release of Help File.