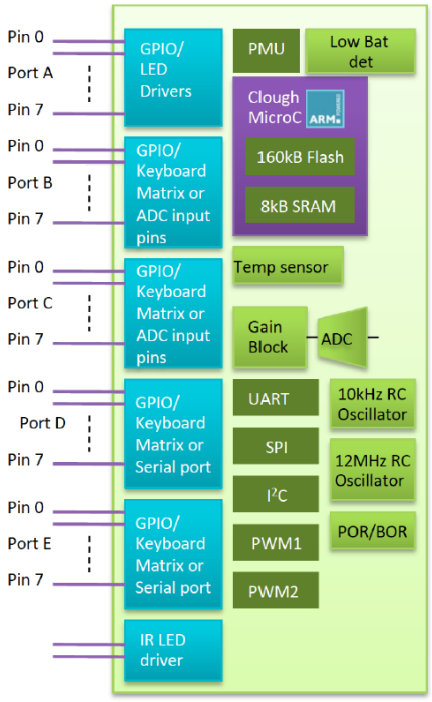
iND80205 “Kamcho”

Kamcho mini EVKit starter guide

11/12/15 **Application Note**

1.0 TABLE OF CONTENT

2.0 LIST OF FIGURES



3.0 KAMCHO HARDWARE INTRODUCTION

3.1 KAMCHO IC

Kamcho integrates an ARM Cortex-M0 low cost 32-bit microcontroller containing 160kB of flash program memory and 8kB of SRAM. It implements several general-purpose peripherals. Its main features are:

CPU Architecture:

- ARM Cortex-M0 processor running at 12MHz (Internal RC), 32.768kHz RTC, or 10kHz (Internal auxiliary RC)
- System Tick Timer (SysTick – 24 bits, interruptible)
- Serial Wire Debugger
- Built-in Nested Vectored Interrupt Controller (NVIC)
- Programmable Watch-Dog Timer

Memory:

- 160kByte of Flash Program Memory
- 8kByte of SRAM

Peripherals:

- 39 General purpose I/O ports, 8 of which have LED current sink capability
- An ADC (8-bit), total of 41 input channels, with selectable input references and input gain block
- Temperature Sensor
- Low Battery Detection
- 2 x PWM (12-bit)
- Wake-up Timer
- 32.786kHz RTC; 10kHz RC oscillator; 12MHz, 1% RC oscillator
- UART Interface
- SPI Interface
- I^2C Interface
- 500mA IR LED driver

Package:

- 7x7 48 pin QFN package

Application:

Kamcho provides abundant GPIO pins, necessary hardware protocol interfaces, and enough memory for small applications. Kamcho is designed to be an attractive solution for designers are looking for a basic MCU with 32-bit performance which has comparable pricing to 8-bit MCUs.

Figure 1 shows the device block diagram. For more information, please check the Kamcho Datasheet [1]

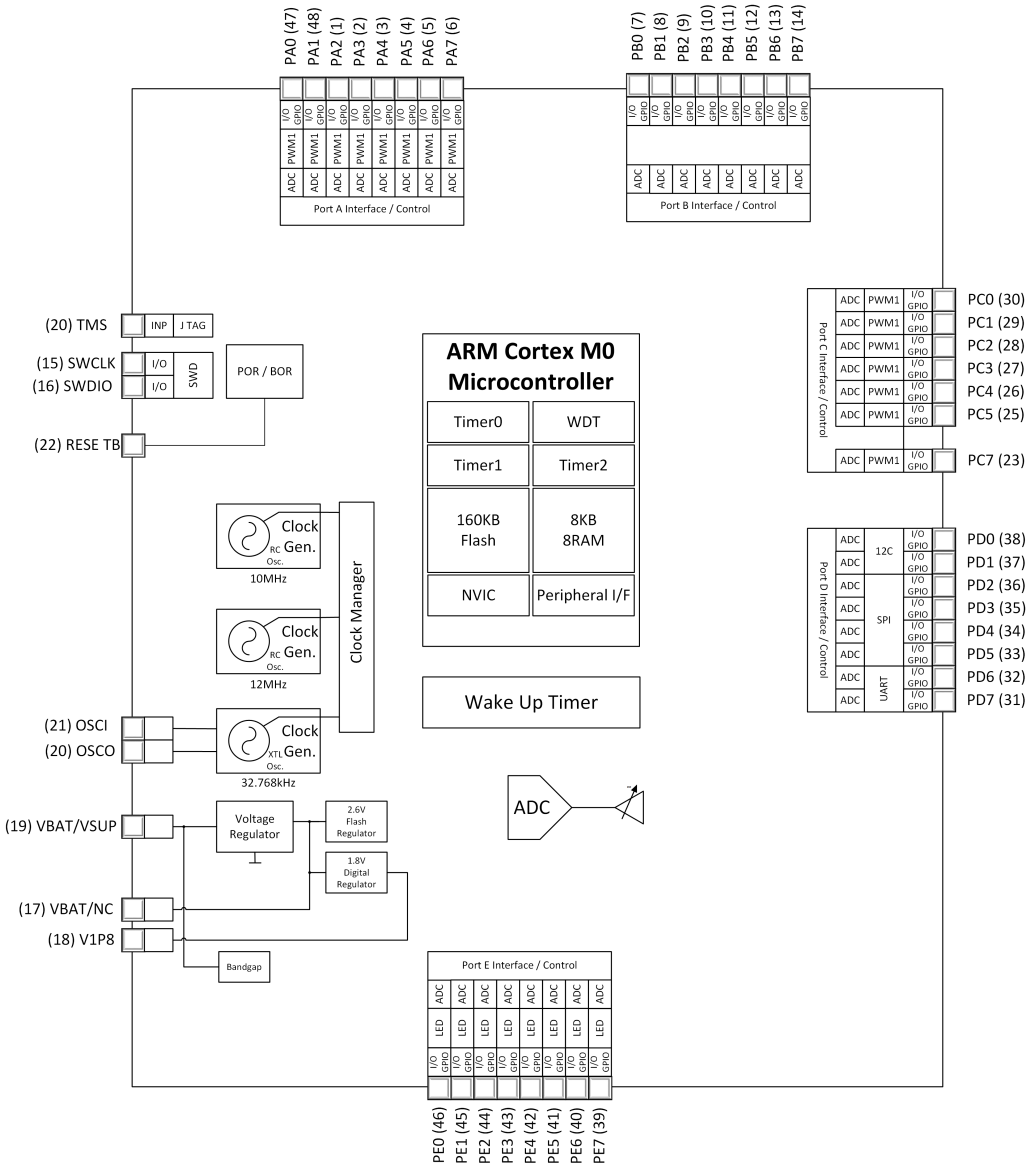


Figure 1 Kamcho block diagram with Port functions

Indie offers two support kits for software development on Kamcho: the Kamcho development board which includes more features and peripherals, and the Kamcho mini EVKit described here, which is cheaper and very easy to use, while still offering all functionality necessary for software development.

3.2 KAMCHO MINI EVKIT

Kamcho can be evaluated using a mini starter kit. This is an easy and fast way to program Kamcho by connecting a board directly to the USB port of your laptop and launching IAR for project development.

The Kamcho mini starter kit also include sensors and related programming so it can be used as a starting point for the user's own software development

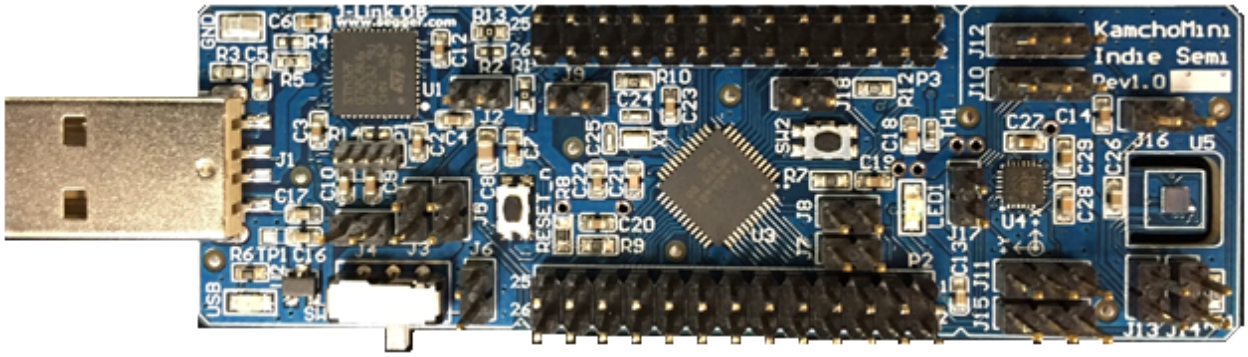


Figure 2 Kamcho Mini EVKit board

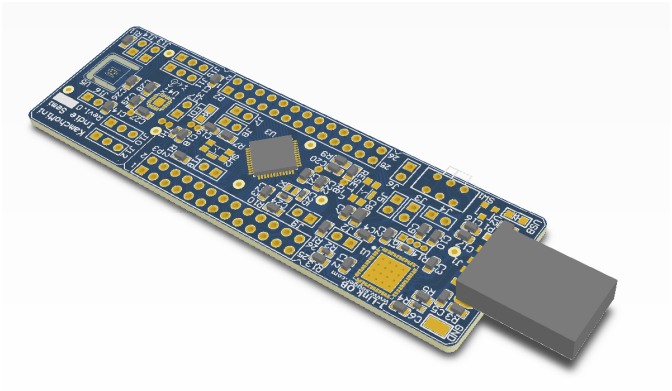


Figure 3 Kamcho mini EVKit (layout)

3.2.1 Kamcho mini EVKit Features

Kamcho mini EVKit provides the following features:

- J-LINK-OB:

J-LINK-OB is the Segger low cost, space-saving solution as an onboard alternative to their standard J-LINK programmer/debugger. J-LINK-OB is designed with STM32F072 (Cortex-M). J-LINK-OB supports SWD (Serial Wire Debug) + SWO interface. J-LINK-OB software is compatible with the regular J-LINK, such as JLINK Commander, J-Flash, etc. J-LINK-OB can work with IAR Embedded Workbench, Keil MDK, and other popular IDEs.

J-LINK-OB is also providing a Virtual COM UART (VCOM) port. This VCOM is connected to Kamcho micro to provide UART communication to the host PC through USB. Users can use terminal programs or a self-designed communication utility to interface with Kamcho Mini EVKit. The PC driver for VCOM is readily available as plug-n-play driver. The VCOM uses no hardware handshaking (CTS and RTS are not implemented). Only RXD and TXD are needed for the serial communication. Software handshaking is recommend for high speed communication.

J-LINK-OB USB can provide power for the entire PCB without any external supply. Refer to Power Management section.

- Kamcho 32-bit Cortex M0

Indie Semiconductor's Kamcho 32-bit ARM Cortex M0. There is a 32.768KHz crystal designed on board to provide RTC function.

- Power Management

Kamcho mini EVKit can be powered through J-LINK-OB USB. There is a 3.3V LDO regulator to power Kamcho and other circuits. The 3.3V LDO also powers the JLINK-OB STM32F072 circuit. The Kamcho system can also powered by an external 3.0V battery through a header (J6). A slider switch (SW1) is used to select between on-board 3.3V LDO and external battery. A jumper (J5) can be used to break the power supply path for the convenience of measuring system current consumption.

- Reset

Kamcho mini EVKit has a reset button to reset Kamcho micro. Press and hold for a short time to ensure a good reset. The reset does not affect JLINK-OB function (i.e. it will not disconnect the USB from PC).

- User Switch

Kamcho mini EVKit has a tactile switch that is connected to port A pin PA1. PA1 is capable of detecting pin state change to generate interrupt. This can be user programmed for his/her own application. Kamcho has an internal pull-up resistor. Therefore an external resistor is not needed.

- LED indicator

Kamcho mini EVKit has an on-board green color LED connected to Port E pin PE0, which has an integrated current-mirror LED driver. This can be user programmed as required for the application.

- Thermistor

Kamcho mini EVKit has a 10K Ohm NTC thermistor connected to Kamcho ADC8 (PB0). This can be used to monitor PCB temperature

- Motion Sensor

The Kamcho mini EVKit implements an MPU-9250 9-axis motion sensor that includes a 3-axis gyroscope, a 3-axis accelerometer, and a 3-axis magnetometer. MPU-9250 is interfaced with Kamcho on either SPI or I2C bus. This sensor is provided as a basic exercise for SPI communication. Full featured motion algorithm is not provided and beyond the scope of this evaluation. There are many open-source algorithms for motion control with this sensor. Users are welcomed to try some of them with Kamcho mini EVKit.

- Thermopile Sensor

The Kamcho mini EVKit has a fully-integrated micro electro mechanical system (MEMS) thermopile sensor TMP007 that measures the temperature of an object without direct contact. The TMP007 is interfaced with Kamcho on the I2C bus. This sensor is provided as a basic exercise for I2C communication. A full-featured algorithm is not provided and beyond the scope of this evaluation. Users can check its manufacture's website for additional reference code/algorithm information.

- Extension Connectors

The Kamcho mini EVKit has two extension headers that provide ground, supplies and all Kamcho pins for users to connect their system or to probe the Kamcho pins.

Kamcho can be programmed/debugged with on board J-LINK-OB. The connector also provides the necessary SWD interface wires for the user to use their own programmer. In this case, R2 and R13 should be removed to isolate the J-LINK-OB from Kamcho.

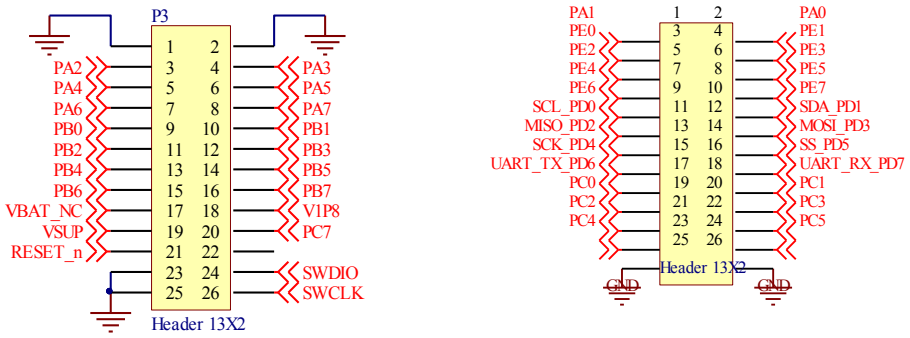


Figure 4 Kamcho mini EVKit Extension Connectors mapping

- Configuration Jumpers

Kamcho mini EVKit has jumpers to configure the connection and setup. Please refer to 4.1.1 Jumper and Switch.

3.2.2 Kamcho mini EVKit Block Diagram

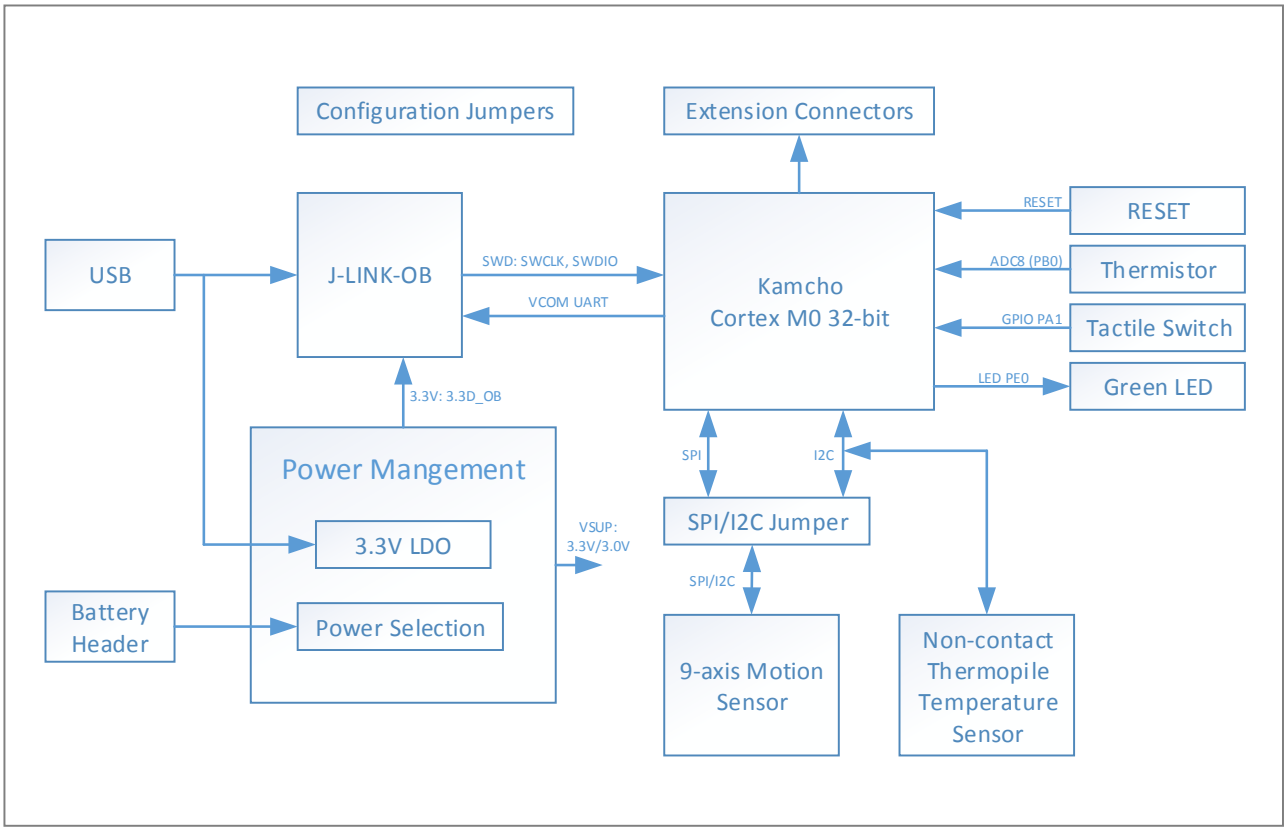


Figure 5 Kamcho mini EVKit System Block Diagram

4.0 RUNNING DEMO

4.1 KAMCHO MINI EVKIT BOARD CONFIGURATION

4.1.1 Jumper and switch check

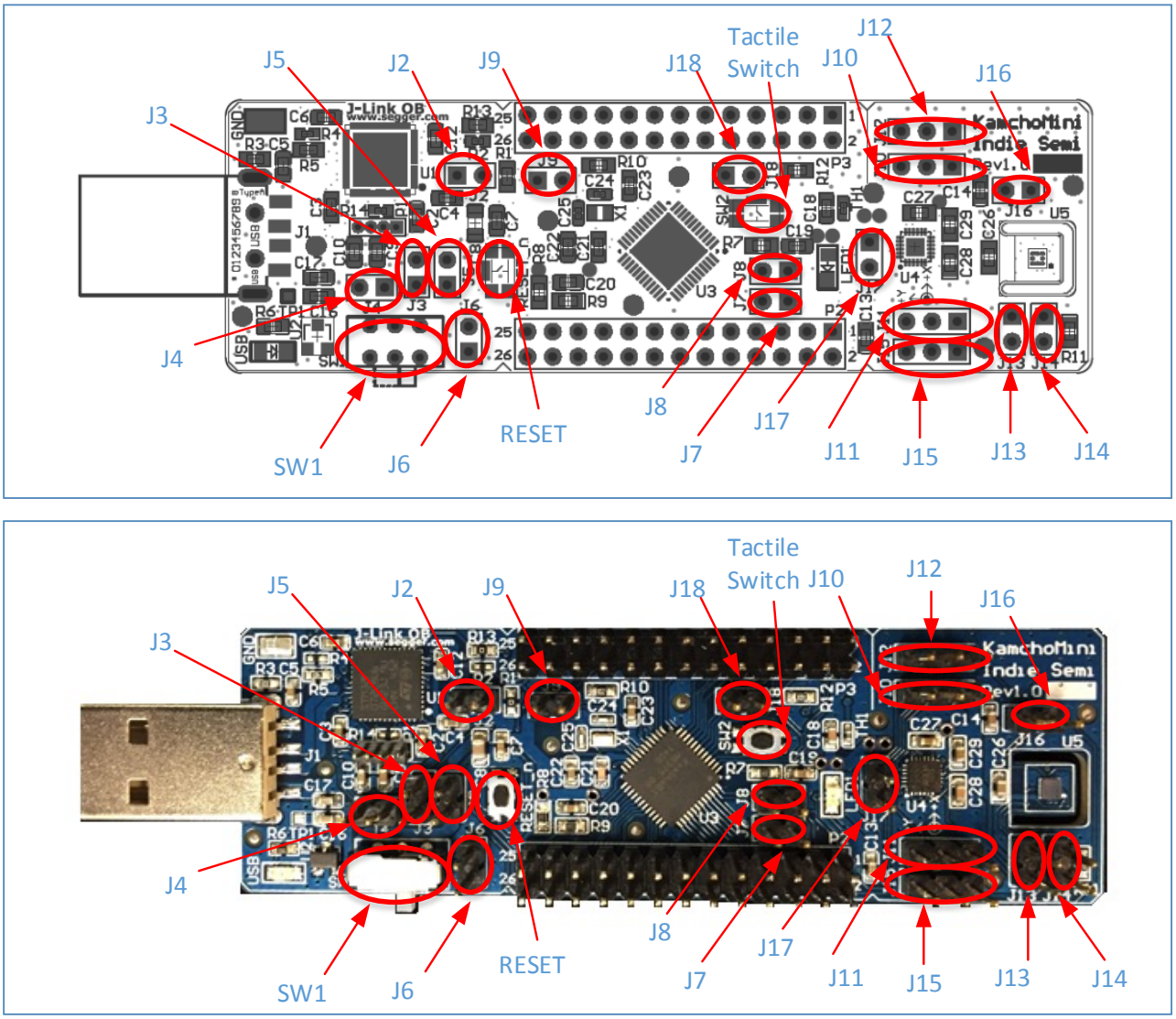


Figure 6 Kamcho mini EVKit Jumpers and Switches

Table 1: Jumper and Switch Configuration					
	Description	**On**	**Off**	**Left Position**	**Right Position**
J2	J-LINK-OB Reset	Disable J-LINK-OB	Normal	-	-
J3	VCOM RXD on/off	Enable VCOM RXD to Kamcho	Kamcho TXD is available on P2-17 for UART or GPIO	-	-
J4	VCOM TXD on/off	Enable VCOM TXD to Kamcho	Kamcho RXD is available on P2-18 for UART or GPIO	-	-
J5	VSUP Current Measure	Normal Operation	Use current meter across jumper pin 1/2 to measure Kamcho system current. J-LINK-OB current is excluded	-	-
J6	3.0V Battery Input	Do not short with jumper shunt	Connect battery terminals here Pin1: Battery (+) (Bottom Position) Pin2: Battery (-) (Top Position)	-	-
J7	On-board LED enable	On-board green LED is connected to PE0	PE0 is available to extention connector P2-3	-	-
J8	On-board Switch Enable	On-board tactile switch is connected to PA0	PA0 is availabel to extention connector P2-2	-	-
J9	Factory Test Mode	Test Mode- Do not use!	Normal Operation	-	-
J10	Motion Sensor U4 SPI/II^{2}C Setting	See "Left/Right Position"	PD5 is available on extension connector P2-16	Pin 2/3 short SPI is selected	Pin 1/2 short I2C is selected I2C Slave Address:0x68
J11	Motion Sensor U4 SPI/I2C Clock Selection	See "Left/Right Position"	PD4 is available on extension connector P2-15 PD0 is available on extension connector P2-11 if J13 is off	Pin 2/3 short SPI SCK is used for SPI PD0 is available on extension connector P2-11 if J13 is off	Pin 1/2 short I2C SCL is used for I2C PD4 is available on extension connector P2-15
J12	Motion Sensor U4 SPI/I2C Data in Selection	See "Left/Right Position"	PD3 is available on extension connector P2-14 PD1 is available on extension connector P2-12 if J16 is off	Pin 2/3 short SPI MOSI is used for SPI PD1 is available on extension connector P2-12 if J16 is off	Pin 1/2 short I2C SDA is used for I2C PD3 is available on extension connector P2-14
J13	Thermopile U5 I2C SCL	PD0 is used for I2C for U5	PD0 is available on extension connector P2-11 if Motion sensor is not using I2C - (J11 is either off or at Left position: Pin 2/3 short)	-	-
J14	Thermopile U5 Interrupt Enable	Thermopile sensor interrupt is connected to PA1 *** Note: J14 and J17 can not be both "on" at the same time	Thermopile sensor interrupt is not used. PA1 is available on extension connector P2-1	-	-

	Description	On	Off	Left Position	Right Position
J15	Motion Sensor U4 SPI Data out /I2c Address Selection	See "Left/Right Position"	PD2 is available on extension connector P2-13	Pin 2/3 short SPI MISO is used for SPI	Pin 1/2 short I2C Address set to 0x68
J16					
J17	Motion Sensor U4 Interrupt Enable	Motion sensor interrupt is connected to PA1 *** Note: J14 and J17 can not be both "on" at the same time	Motion sensor interrupt is not used. PA1 is available on extension connector P2-1	-	-
J18	On-board Thermistor Enable	Thermistor is connected to ADC8/PB0	ADC8/PB0 is available on extension connector P3-9	-	-
SW1	Kamcho Power Selection	-	-	Kamcho is powered by on-board 3.3V LDO, derived from USB 5V power	Kamcho is powered by external 3V battery through J6
RESET	Kamcho RESET	Reset Kamcho system. J-LINK-OB is not affected	Normal Operation		
Tactile Switch	User Switch	PA0 is driven low if J8 is On. If J8 is off, no effect. *** This can be used for user's application	PA0 remains high if J8 is On. If J8 is off, no effect.		

Demo Setting:

USB power, Motion Sensor SPI (interrupt), Thermopile I2C (no interrupt), LED enabled, Tactile Switch Enabled, Virtual Com enabled

a. SW1: Left
b. J4: On
c. J3: On
d. J5: On
e. J18: On
f. J8: On
g. J7: On
h. J17: On
i. J12: Left
j. J10: Left
k. J11: Left
l. J15: Left
m. J16: On
n. J13: On

4.1.2 PC connection

Connect the Kamcho mini EVKit directly to a USB port of the computer or use a USB cable/extension cable.

4.2 KAMCHO MINI EVKIT PC COMMUNICATION

Kamcho mini EVKit can be used with Windows. Terminal programs like HyperTerminal, Tera Term, etc. can be used to communicate with the Kamcho mini EVKit.

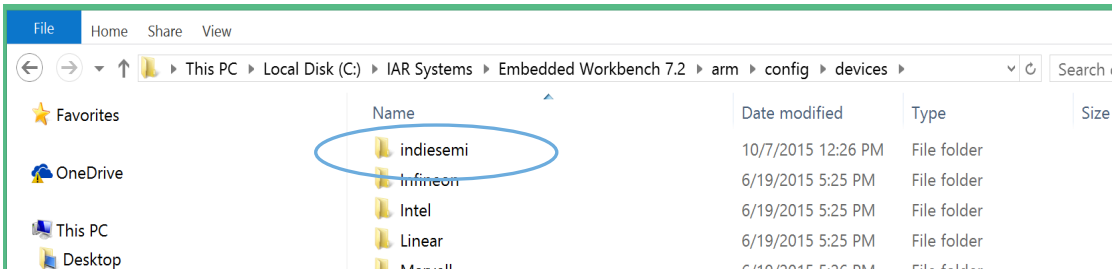
The setting is:

- Baud rate: 57600
- Data: 8 bit
- Parity: None
- Stop: 1 bit
- Flow control: None

5.0 LAUNCHING DEMO CODE (IAR): STARTING AND CONFIGURATION

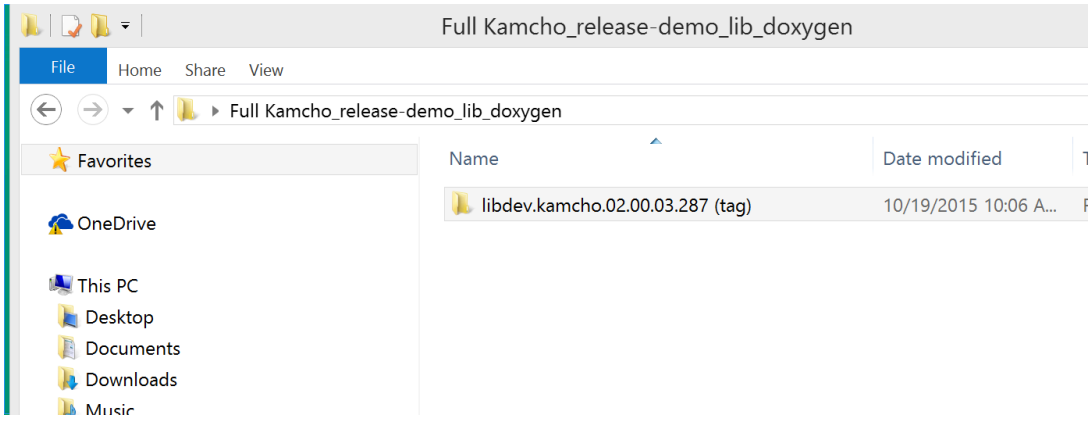
- The Kamcho mini EVKit demo software works with IAR open (free) version. The free version allows the creation of downloadable code limited to 16kBytes. All demos can be individually selected and the compiled code fits within the 16kB limit. If more complex programs are compiled it may be necessary to license the full version of IAR.
- Before launching IAR, it is necessary to add the 2 Kamcho config product files within the ARM config directory.

1) First, create a "/indiesemi/" folder within the IAR ARM devices area:



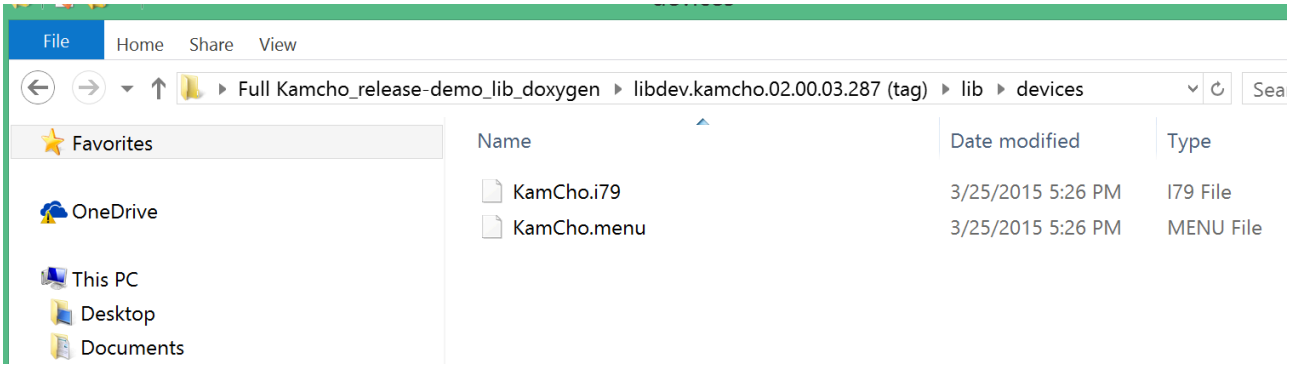
2) Copy the indie Kamcho Software Development Kit into your working area:

Unzip the provided file - libdev.kamcho.02.00.03.287.zip (or similar / later revision) anywhere in your working environment.

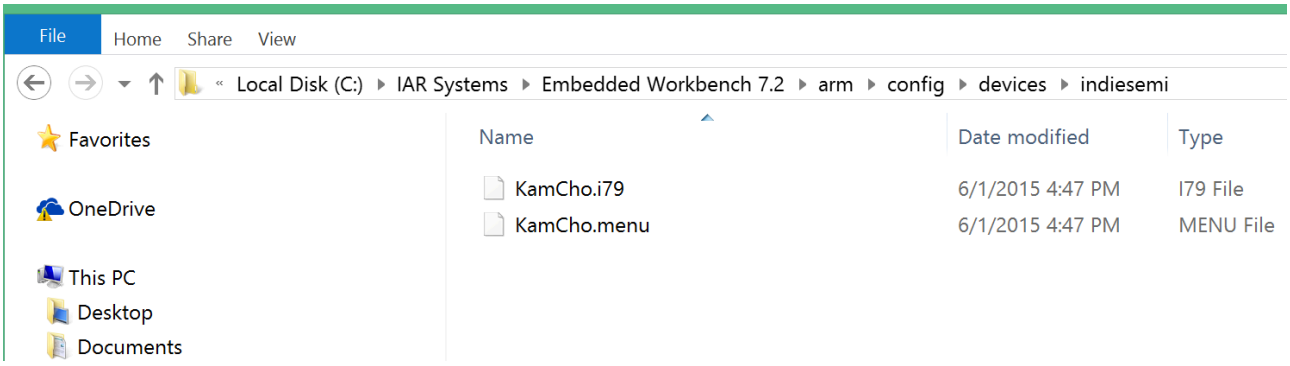


3) Locate the two Kamcho config files in the indie Kamcho subfolders:

The two files (Kamcho.i79 and Kamcho.menu) are provided within the Kamcho software development kit. They are located under the following directory: \libdev.kamcho.02.00.03.287\lib\devices\

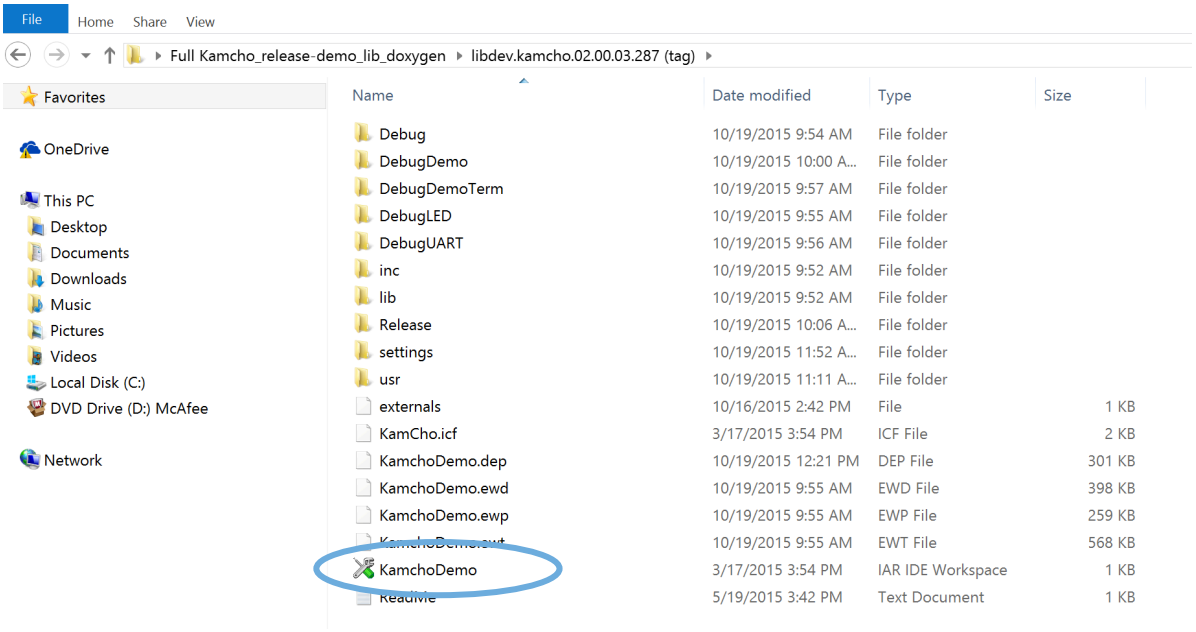


4) Copy these two files and paste them into the /indiesemi directory created in 1)

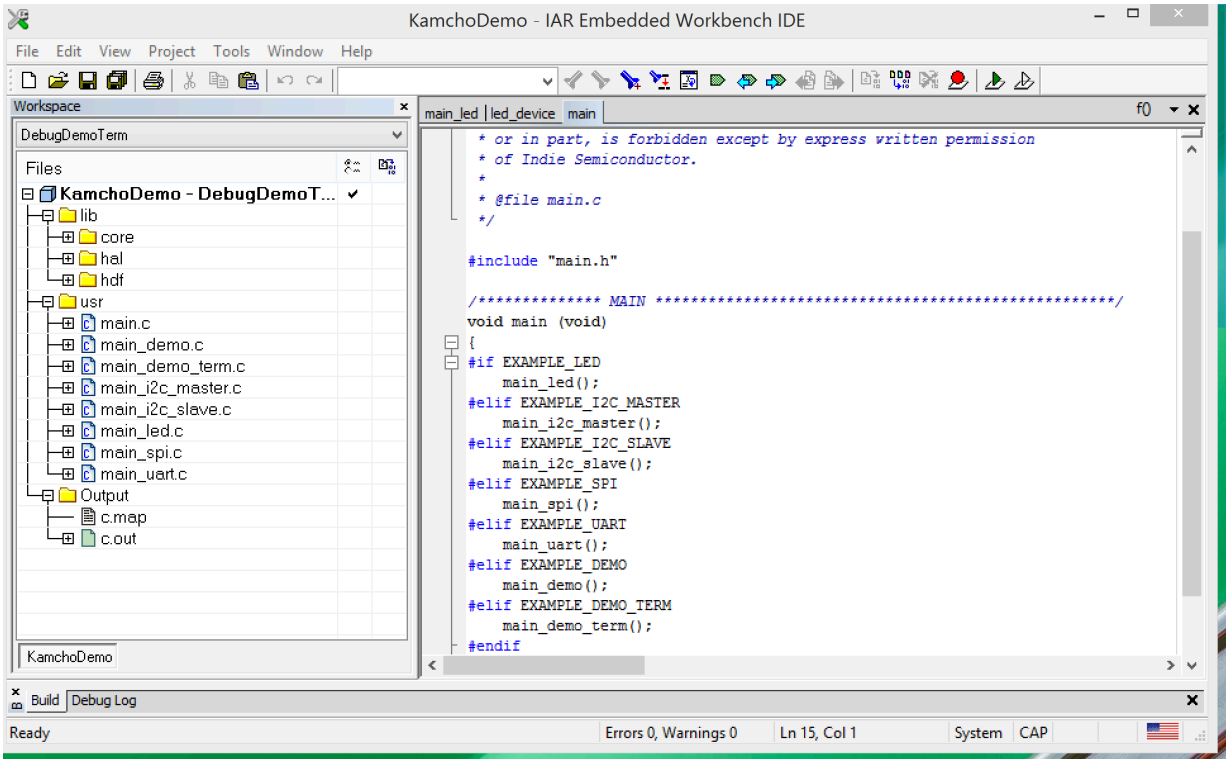


You are now ready to launch IAR and the indie Kamcho demo code:

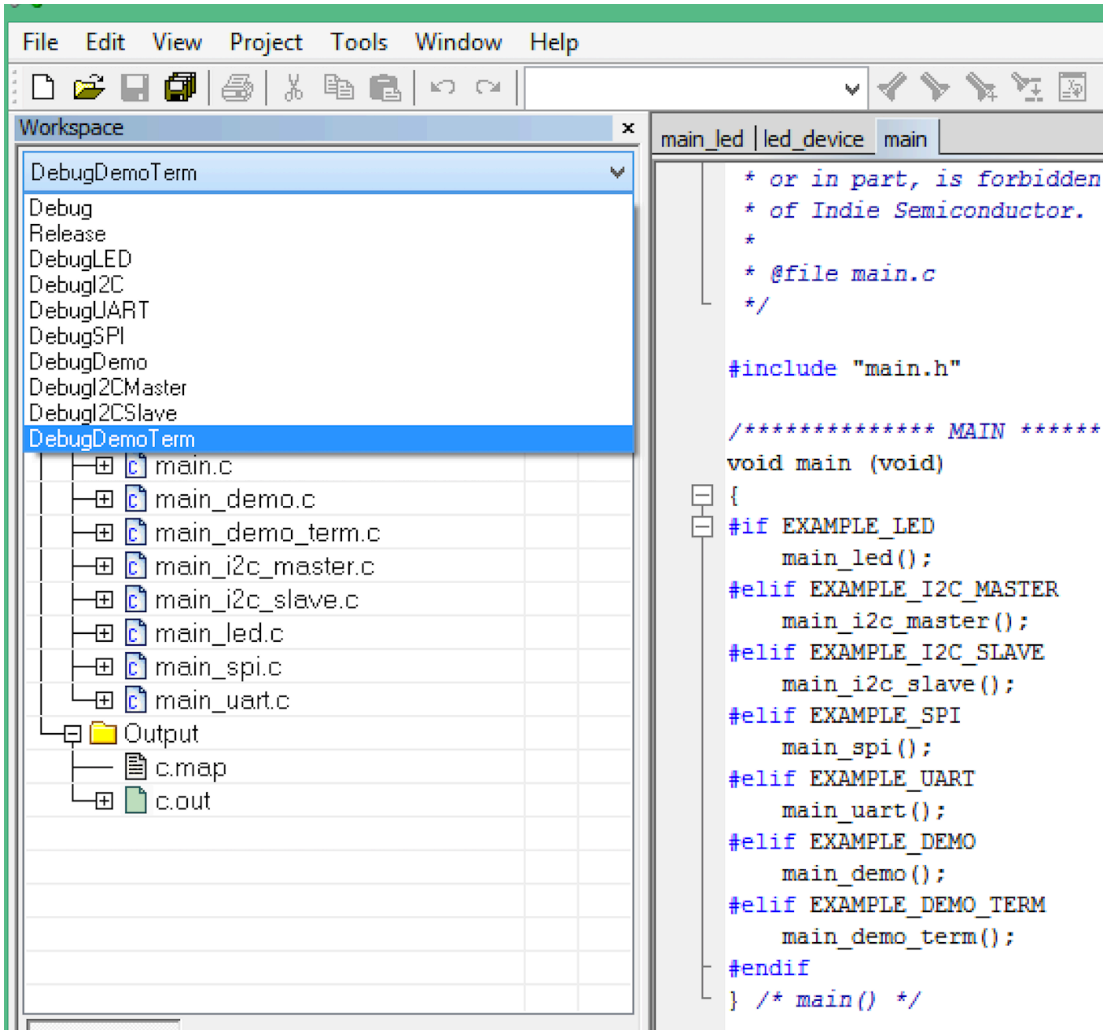
5) Go back to the indie Kamcho SDK directory and launch KamchoDemo IAR project (IAR IDE Workspace)



The KamchoDemo project will open with all the options:



You can now select the type of demo you would like to upload into the Kamcho mini EVKit:

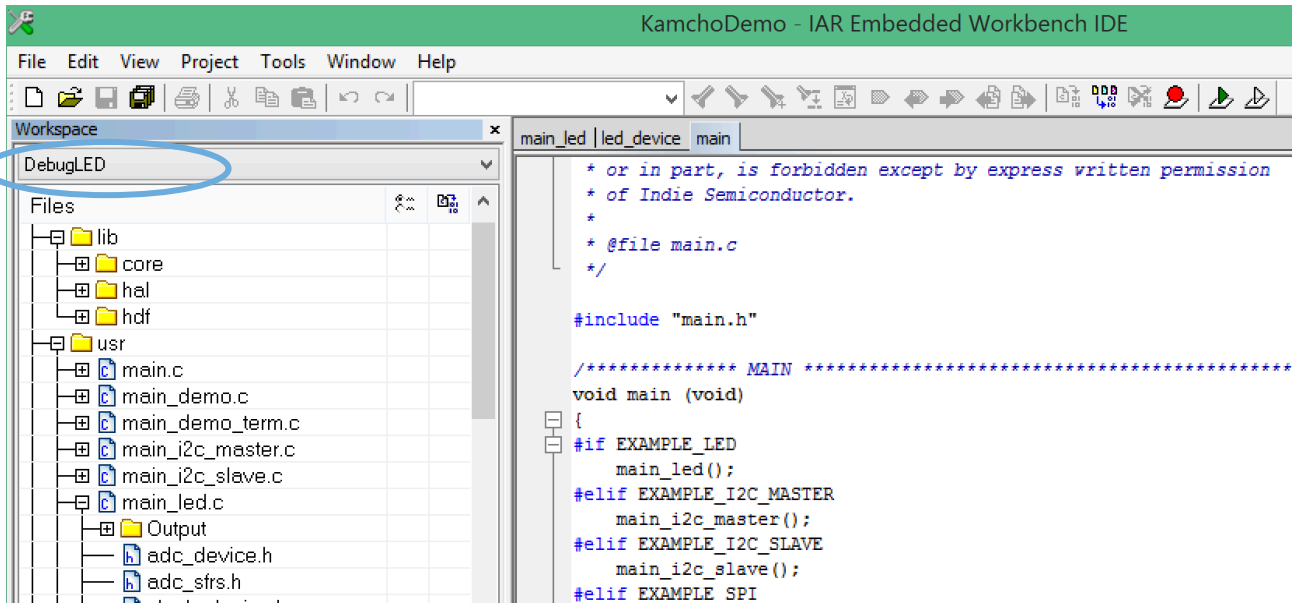


There are 9 options that can be run:

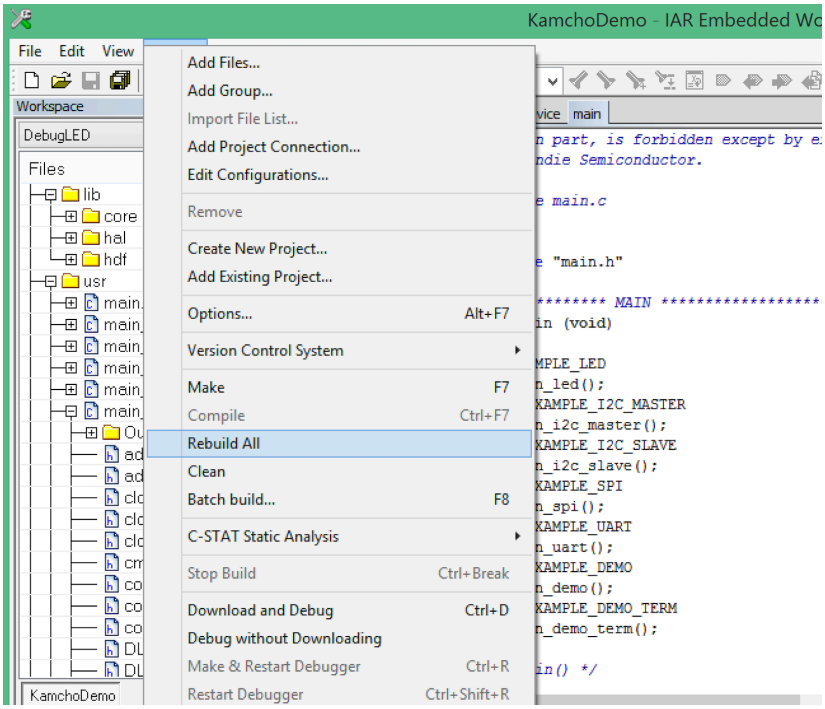
1. Debug: Allows users to write their own code in debug mode.
2. Release: Users write their own code in release mode
3. DebugLED: Example code to work with Timer, GPIO interrupt and LED
4. DebugUART: Example code to use UART driver with Windows Communication Terminal
5. DebugSPI: Example code to use SPI driver with SPI/I2C tester
6. DebugI2CMaster: Example code to use SPI driver with SPI/I2C tester
7. DebugI2CSlave: Example code to use SPI driver with SPI/I2C tester
8. DebugDemo: Example code to demo SPI/I2C with on-board SPI/I2C slave sensors, sending out binary message through UART
9. DebugDemoTerm: Example code to demo SPI/I2C with on-board SPI/I2C slave sensors, sending out text message through UART

Select the demo example program you would like to run:

➔ Let's start with the DebugLED

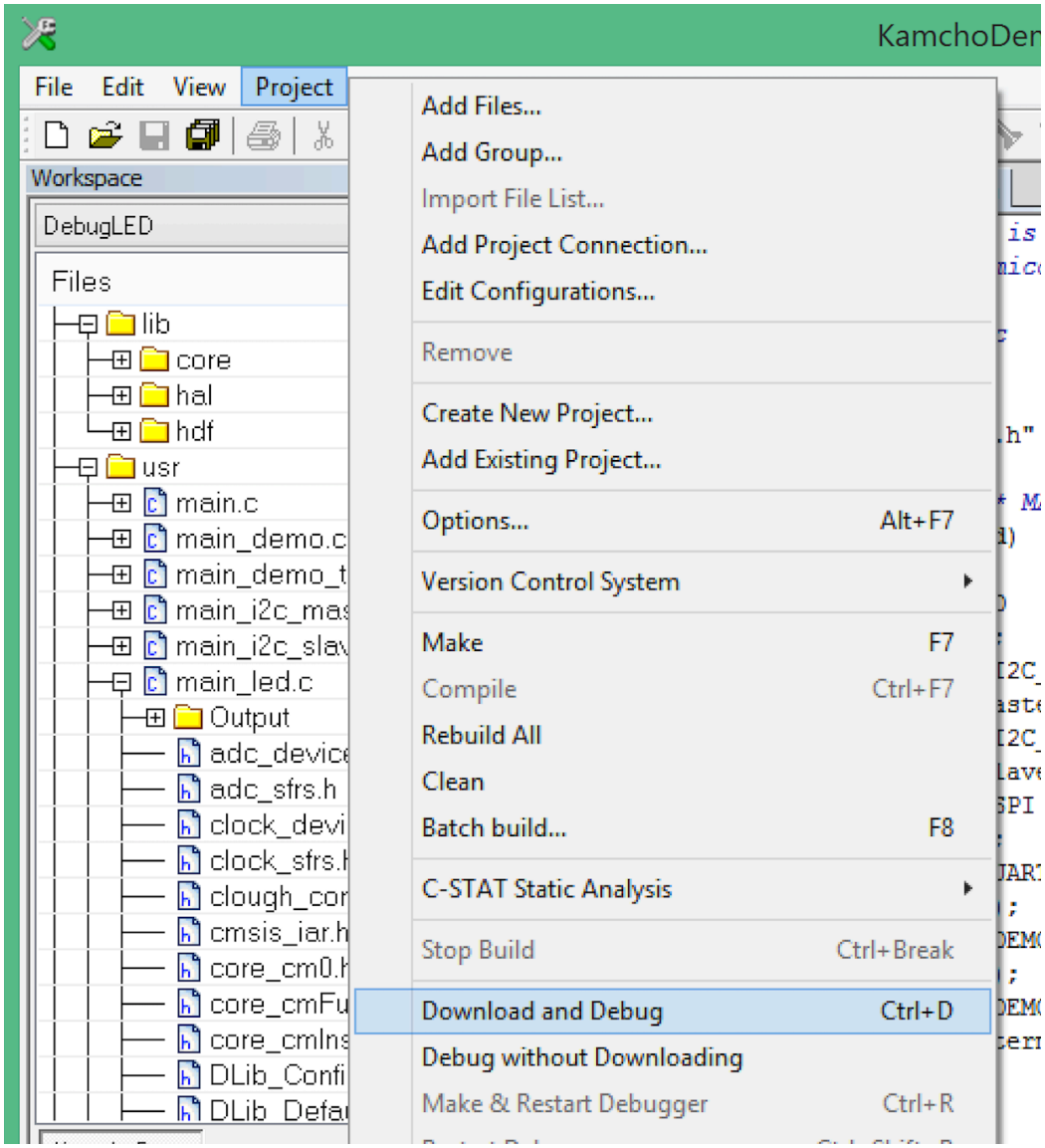


Now you can build the demo code:



There may be a list of Warnings and these can be ignored.

The code is now ready to be uploaded into the Kamcho microcontroller:



- Once the code has been uploaded into the MCU, you are now able to run it (DEBUG\GO)
- As an example, the DemoLED code demonstrates the setup of a GPIO controlling LED 2 on the Kamcho mini EVKit. Once the button is pressed, the LED will turn on for 500ms.

6.0 AN EXAMPLE OF DEMO CODE

The following example explains how to modify the existing code and demonstrates how to build your own project.

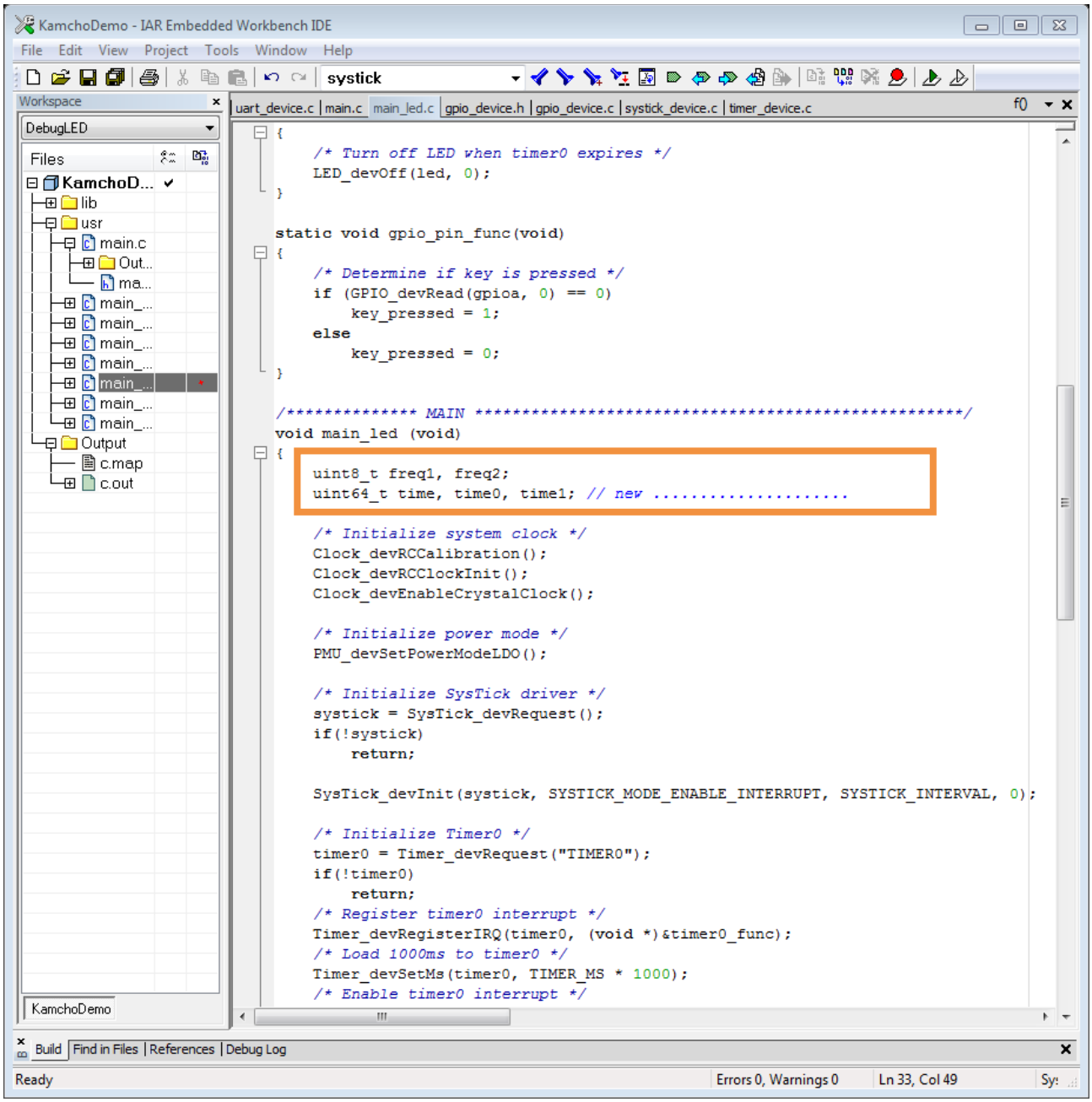
The proposes example starts from the DebugLED:.

- Open the demo code project, select the DebugLED option. This will demo the LED and switch input.
- Double click on “main_led.c” and open it.

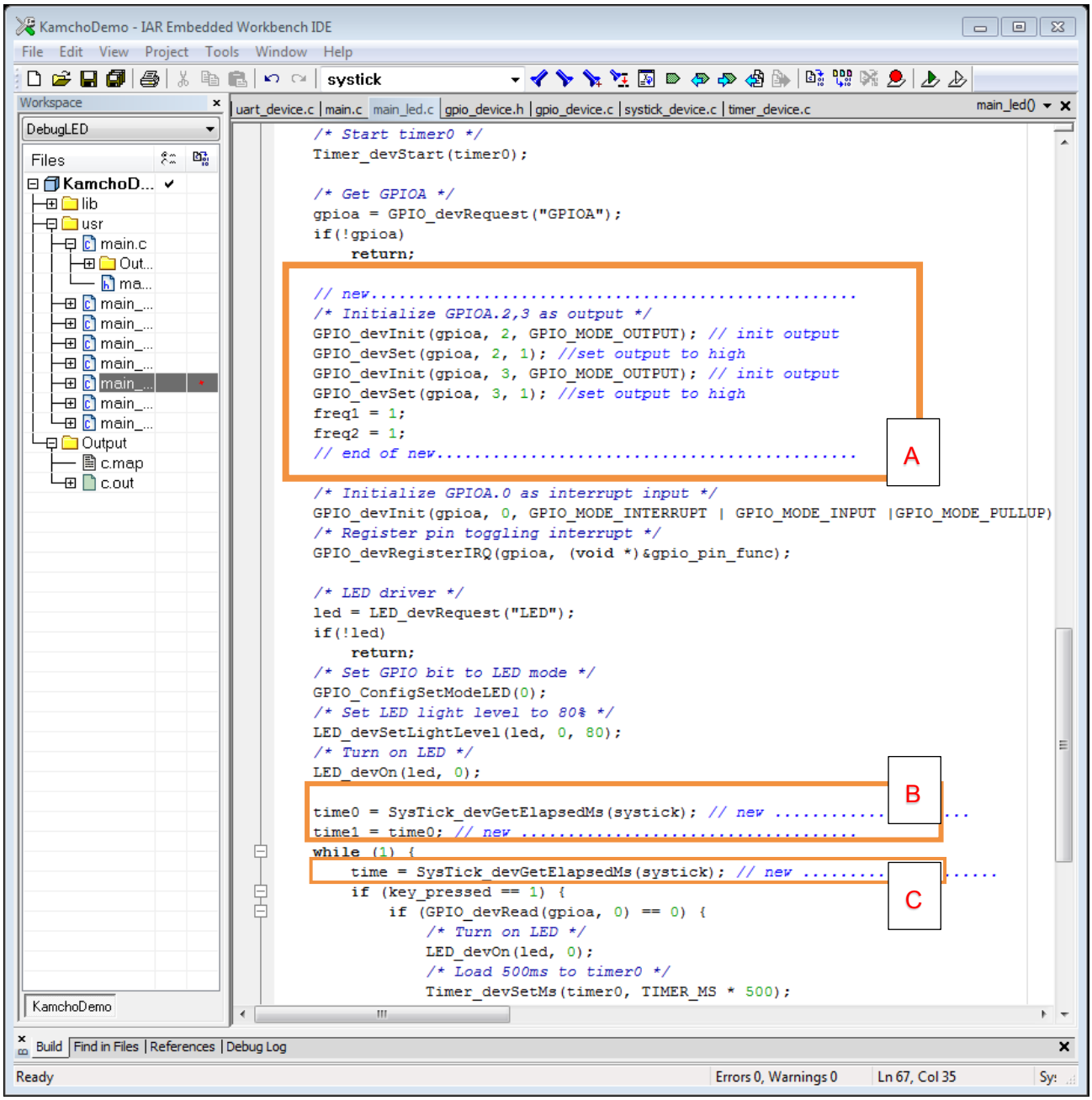
The current demo will turn on the on-board LED for 0.5 sec once the tactile switch (SW2) is pressed. The switch is connected PA0.

From here, we will modify the code to control the LED through a GPIO port, instead of SW2.

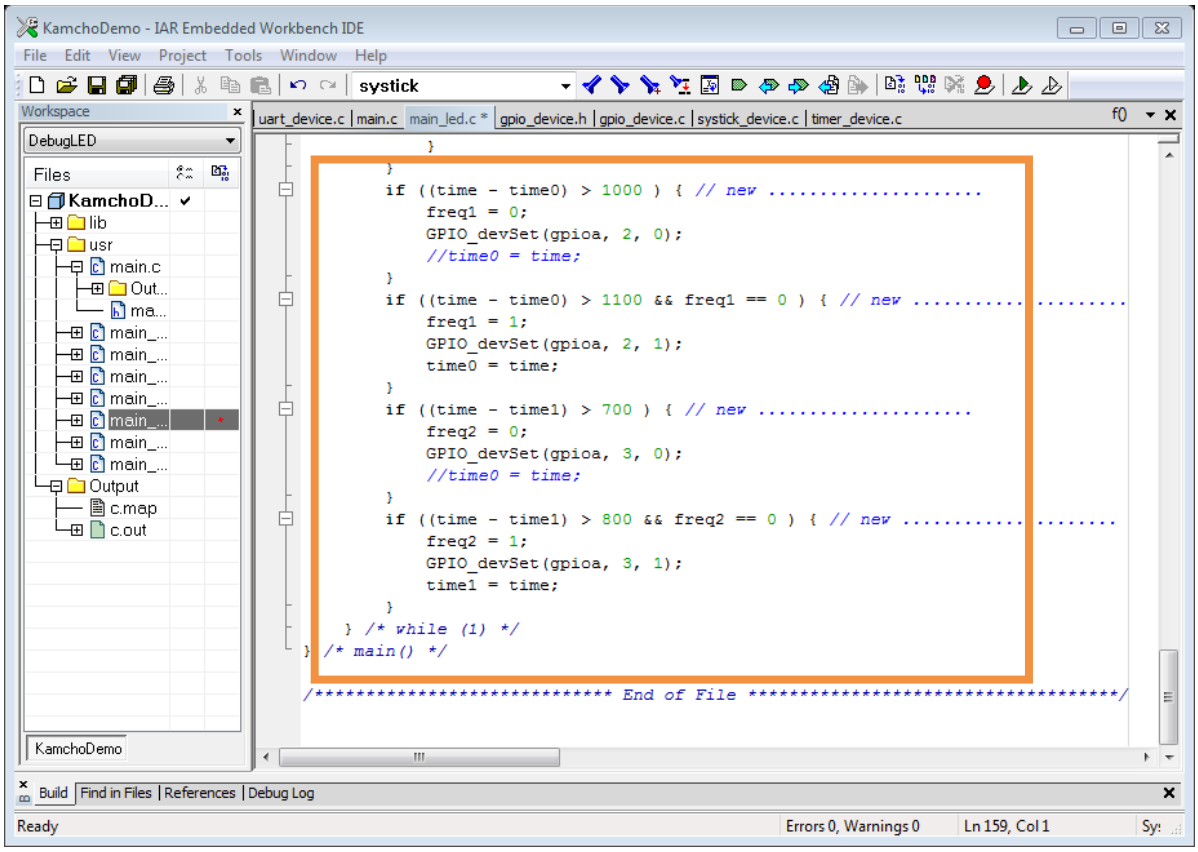
- First, we need remove Jumper J8. This will disconnect PA0 from SW2.
- Next, we need prepare a short jumper wire (both ends are female header). Locate Pin 2 of Jumper J8 (the pin closer to U3) and PA2 and PA3 on extension connector P3 (PA2 is P3-3, and PA3 is P3-4).
- The goal is to output low on PA2 for 100ms for every 1 sec; output low on PA3 for 100ms for every 0.7 sec. Since the existing code will detect PA0 going low then turn on LED for 0.5 sec, we can loop back PA2 or PA3 to PA0 to show the action of PA2 or PA3.
- We only need to add a few lines to code to achieve the function.
- See the following the pictures. We only show the changes made to the demo project.



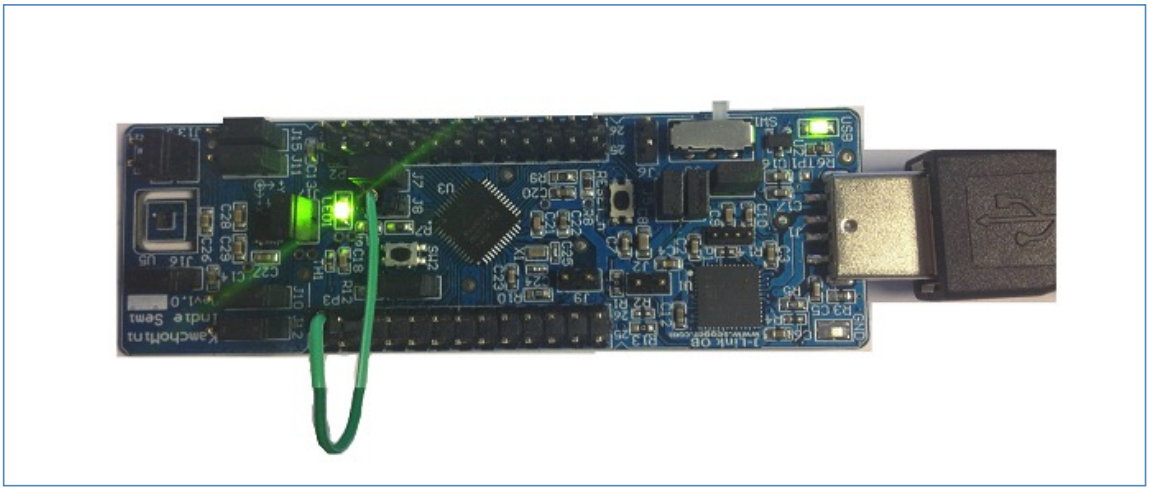
- Add two variables to control the output frequency: freq1, freq2
- Add a few time variables for timing schedule: time, time0, time1.



- A: Initialize PA2 and PA3 to be outputs. Drive them high. Initialize freq1 and freq2 to 1
- B: Initialize timing variables.
- C: Get the SysTick time in the loop.



- Add timing schedule for every 1000 ms and 700ms. Once the time is reached, output corresponding port to low. This will trigger the LED event and turn on LED for 500ms. After the port is set to low, the corresponding port will be set back to logic high and timing is reset.

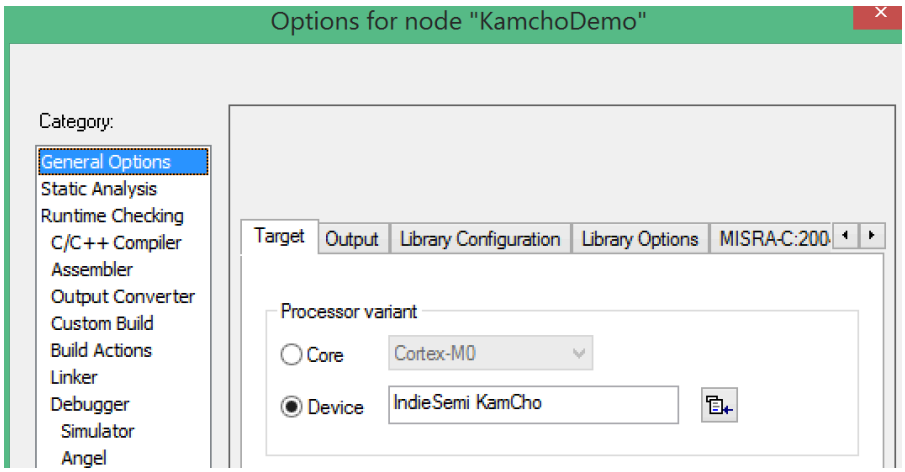


- Remove Jumper J8 and connect J8-2 to P3-3 or P3-4 with a jumper wire. For P3-3, the LED will flash 0.5 sec every 1 sec. For P3-4, the LED will flash 0.5 sec every 0.7 sec.

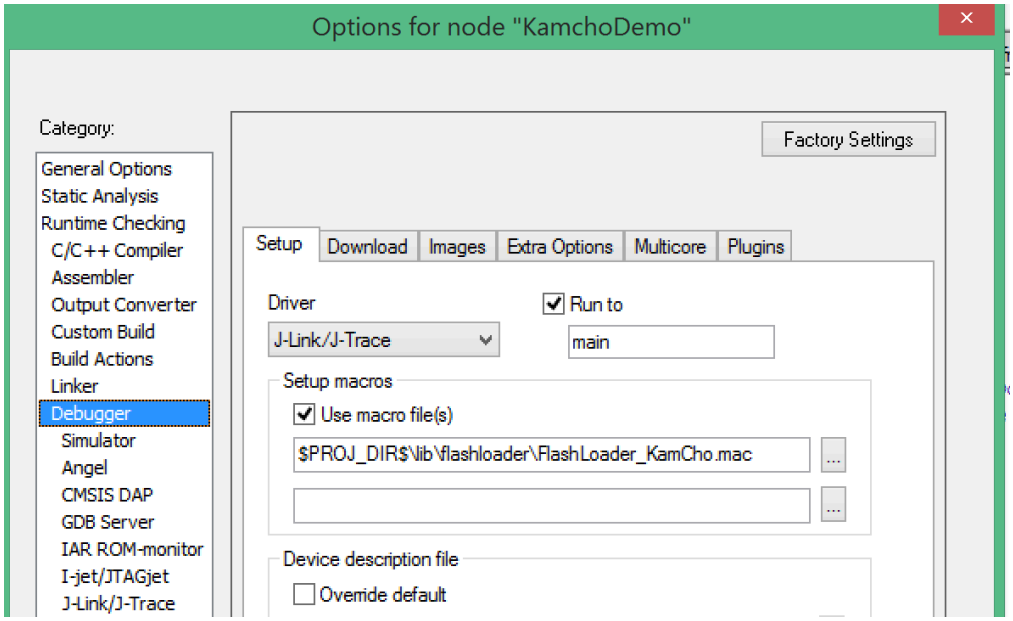
7.0 ANNEX: OVERVIEW OF IAR CONFIG

The IAR environment should be configured for Kamcho demo and Flash programming:

➔ General Options within the IAR Project should indicate the indiesemi Kamcho device



➔ The links to debugger and flash programmer are within the indie Kamcho SDK:



You can now re-Flash the Kamcho evaluation board with the code, and explore its structure.

The Kamcho demo includes all the libraries needed to check all peripherals as well as all communications.

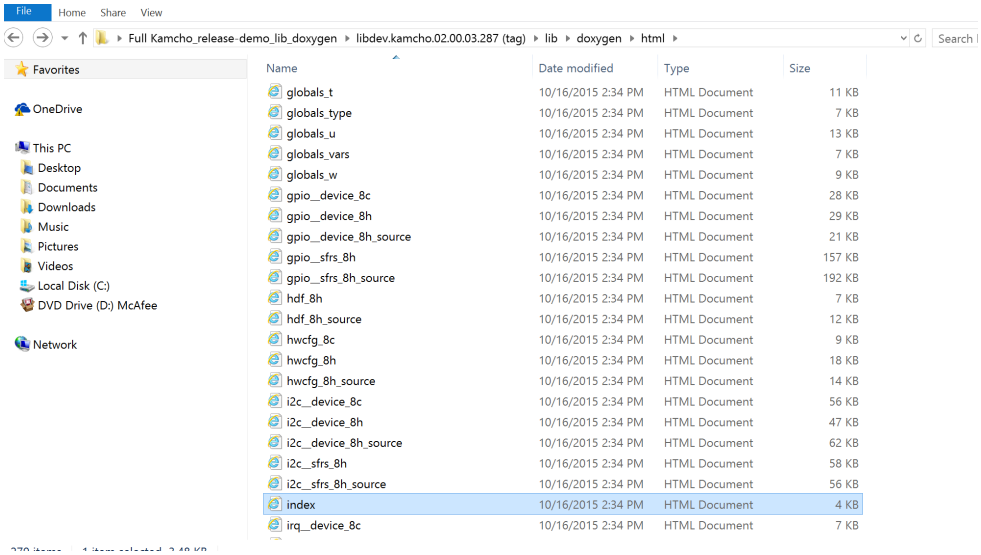
8.0 INDIE KAMCHO LIBRARIES AND DOCUMENTATION

indie provides all the software to access and control Kamcho peripherals. The software architecture is built through multiple dedicated header files with specific functions (SPI, ADC, etc.). Every library has specific set of functions to control the mentioned peripherals, configure the device, or run the demo.

There is detailed documentation included in the indie Kamcho Software Development Kit, located in the

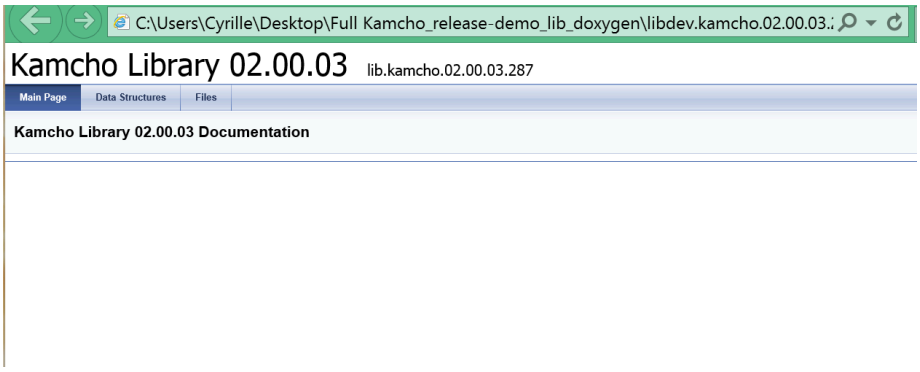
\libdev.kamcho.02.00.03.287(tag)\lib\doxygen\html

To launch the documentation, double click on index.html as shown below:



Kamcho library documentation is created using doxygen. Doxygen is a documentation generator, a tool for writing software reference documentation. The documentation is written within code, and is thus relatively easy to keep up to date. Doxygen can cross reference documentation and code, so that the reader of a document can easily refer to the actual code. Doxygen is free software, released under the terms of the GNU General Public License.

After clicking on index.html, the full documentation page opens:



You can navigate through the all library details, data structures being used as well as the manual for all the files and functions:

All the header files used in the provided SDK and demo files are included.

Selecting one file will provide all details on functions included, as well as source code:

Kamcho Library 02.00.03 lib.kamcho.02.00.03.287

Main Page | Data Structures | Files

File List | Globals

kamcho › silicon › branches › firmware › release › rel_lib.kamcho.02.00.03.287 › hal › inc

led_device.h File Reference

```
#include <stdint.h>
#include "hdf.h"
```

Go to the source code of this file.

Data Structures

struct **LED_Device_t**
A structure to represent LED device. More...

Typedefs

typedef const struct **LED_Device_t** **LED_Device_t**
A structure to represent LED device. More...

Functions

int32_t **LED_devInit** (**LED_Device_t** *dev, uint8_t pos)
Init LED mode for a pin. More...

int32_t **LED_devDeInit** (**LED_Device_t** *dev, uint8_t pos)
DeInit LED mode for a pin. More...

int32_t **LED_devOn** (**LED_Device_t** *dev, uint8_t pos)
Turn On a LED. More...

int32_t **LED_devOff** (**LED_Device_t** *dev, uint8_t pos)
Turn off LED. More...

int32_t **LED_devSetLightLevel** (**LED_Device_t** *dev, uint8_t pos, uint8_t level)
Set LED light level. More...

LED_Device_t * **LED_devRequest** (const char *name)
Request a LED Device. More...

Detailed Description

Copyright
2015 Indie Semiconductor.

Main Page | Data Structures | Files

File List | Globals

kamcho > silicon > branches > firmware > release > rel_lib.kamcho.02.00.03.287 > hal > inc

led_device.h

Go to the documentation of this file.

```
 1
12 #ifndef __LED_DEVICE_H__
13 #define __LED_DEVICE_H__
14
15 #include <stdint.h>
16 #include "hdf.h"
17
21 typedef const struct LED_Device_t {
22     void (*On)(uint8_t pos);
23     void (*Off)(uint8_t pos);
24     void (*SetLightLevel)(uint8_t pos, uint8_t level);
25     const char *Name;
26 } LED_Device_t;
27
35 int32_t LED_devInit(LED_Device_t *dev, uint8_t pos);
36
44 int32_t LED_devDeInit(LED_Device_t *dev, uint8_t pos);
45
53 int32_t LED_devOn(LED_Device_t *dev, uint8_t pos);
54
62 int32_t LED_devOff(LED_Device_t *dev, uint8_t pos);
63
72 int32_t LED_devSetLightLevel(LED_Device_t *dev, uint8_t pos, uint8_t level);
73
80 LED_Device_t *LED_devRequest(const char *name);
81
82 #endif /* __LED_DEVICE_H__ */
```

9.0 REFERENCES

[1] iND80205 data sheet

10.0 REVISION HISTORY

Rev #	Date	Action	By
0.1	10/20/2015	Kamcho mini EVKit starting guide initial draft	CR
0.2	10/22/2015	Adding hardware info and make code modification paragraph	TL
1.0	10/26/2015	Final review before release	
1.1	11/4/15	Minor errors corrected	PH

11.0 CONTACTS

United States
32 Journey
Aliso Viejo, California 92656, USA
Tel: +1 949-608-0854
sales@indiesemi.com

China
232 Room, Donghai Wanhao Plaza,
South Hi-tech 11th Road, Hi-tech Industry Park,
Nanshan District, Shenzhen, China.
Tel: +86 755-86116939

Scotland
MWB Business Exchange
9-10 St. Andrew Square
Edinburgh EH2 2AF, Scotland
Tel: +44 131 718 6378

http://www.indiesemi.com/

Important Notice

indie semiconductor reserves the right to make changes, corrections, enhancements, modifications, and improvements to indie semiconductor products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on indie semiconductor products before placing orders. indie semiconductor products are sold pursuant to indie semiconductor's terms and conditions of sale in place at the time of order acknowledgement. Purchasers are solely responsible for the choice, selection, and use of indie semiconductor products and services described herein. indie semiconductor assumes no liability for the choice, selection, application assistance or the design of Purchasers' products.

No license, express or implied, to any intellectual property right is granted by indie semiconductor by this document.

The materials, products and information are provided "as is" without warranty of any kind, whether express, implied, statutory, or otherwise, including fitness for a particular purpose or use, merchantability, performance, quality or non-infringement of any intellectual property right. Indie semiconductor does not warrant the accuracy or completeness of the information, text, graphics or other items contained herein. indie semiconductor shall not be liable for any damages, including but not limited to any special, indirect, incidental, statutory, or consequential damages, including without limitation, lost of revenues or lost profits that may result from the use of the materials or information , whether or not the recipient of material has been advised of the possibility of such damage.

Unless expressly approved in writing by two authorized indie semiconductor representatives, indie semiconductor products are not designed, intended, warranted, or authorized for use as components in military, space, or aircraft, in systems intended to support or sustain life, or for any other application in which the failure or malfunction of the indie semiconductor product may result in personal injury, death, or severe property or environmental damage.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.