SmartSonic *Hello Chirp* Application Hands-on Exercise

INTRODUCTION

This exercise demonstrates how to build and run a basic ultrasonic sensing application using the Chirp SonicLib sensor API.

Hello Chirp is a simple C application that demonstrates how the SonicLib functions are used to initialize, configure, and operate one or more ultrasonic sensors. It is intended to be a developer's first exposure to the Chirp sensor and the SonicLib API functions.

The application discovers what sensors are connected to the board, programs and configures them, and then triggers and displays range (distance) measurements through a serial connection.

In this example, the application is built using Atmel Studio 7 for the Chirp SmartSonic evaluation board, which features an Atmel SAMG55 microcontroller. The SmartSonic board uses sensor daughter boards that support up to four ultrasonic sensors (one mounted to the board, and others connected using flex cables). The Hello Chirp application can detect and run with either a single sensor or multiple sensors connected to the board.

The **main.c** file is the main file in the application. It contains extensive comments explaining how the SonicLib interfaces are used.

In addition, SonicLib includes descriptions of all functions and structures in browsable HTML pages, located in **chirpmicro/html**. Open the **index.html** file in that directory to get started.

REQUIRED EQUIPMENT

- SmartSonic evaluation board
- Chirp sensor daughter board for CH101 or CH201
- Micro-USB cable
- *Optional:* Additional Chirp sensor modules with flex cables, for different physical arrangement or multi-sensor operation

REQUIRED SOFTWARE PACKAGES

1. **SmartSonic_HelloChirp_v*X_X_X*.zip** (actual file name will include version number), includes:
    - Hello Chirp application files
    - Chirp SonicLib sensor API and driver files
    - Sensor firmware image files
    - Board support package files for Chirp SmartSonic board
    - Atmel Studio 7 project files to build the application
2. Atmel Studio 7 integrated development environment – download from Microchip.com
3. Terminal emulator of your choice (for example, PuTTY or TeraTerm)

Chirp Microsystems
2560 Ninth Street, Ste 220, Berkeley, CA 94710 U.S.A
+1(510) 640–8155
www.chirpmicro.com

Document Number: AN-000154
Revision: 1.2
Release Date: 08/05/2020

TABLE OF CONTENTS

LIST OF FIGURES

1 INSTALLATION / PREPARATION

1. Download and install the Atmel Studio 7 IDE.
2. Install terminal emulator program, if necessary.
3. Download and install (unzip) the SmartSonic_HelloChirp application to a project directory of your choice. The examples in this document show C:\Chirp as the base directory, but you are free to place it wherever is convenient.
4. Connect the Chirp sensor daughter board to the SmartSonic board. Be careful to align the white arrows.
5. *Optional:* Using flat flex cables, attach additional off-board sensor(s) to the connectors on the daughter board. If an off-board sensor is connected to the Sensor 0 connector (J6), you must set the switch on the side of the daughter board to use the off-board sensor as Sensor 0 instead of the sensor on the daughter board (U15).
6. Connect the SmartSonic board to a Windows PC with the micro-USB cable attached to the EDBG port. Configure the jumpers on the board as shown in the following photo. If you are only using the single USB cable attached to the EDBG port, jumper J1 should have pins 3 and 4 connected (shorted).

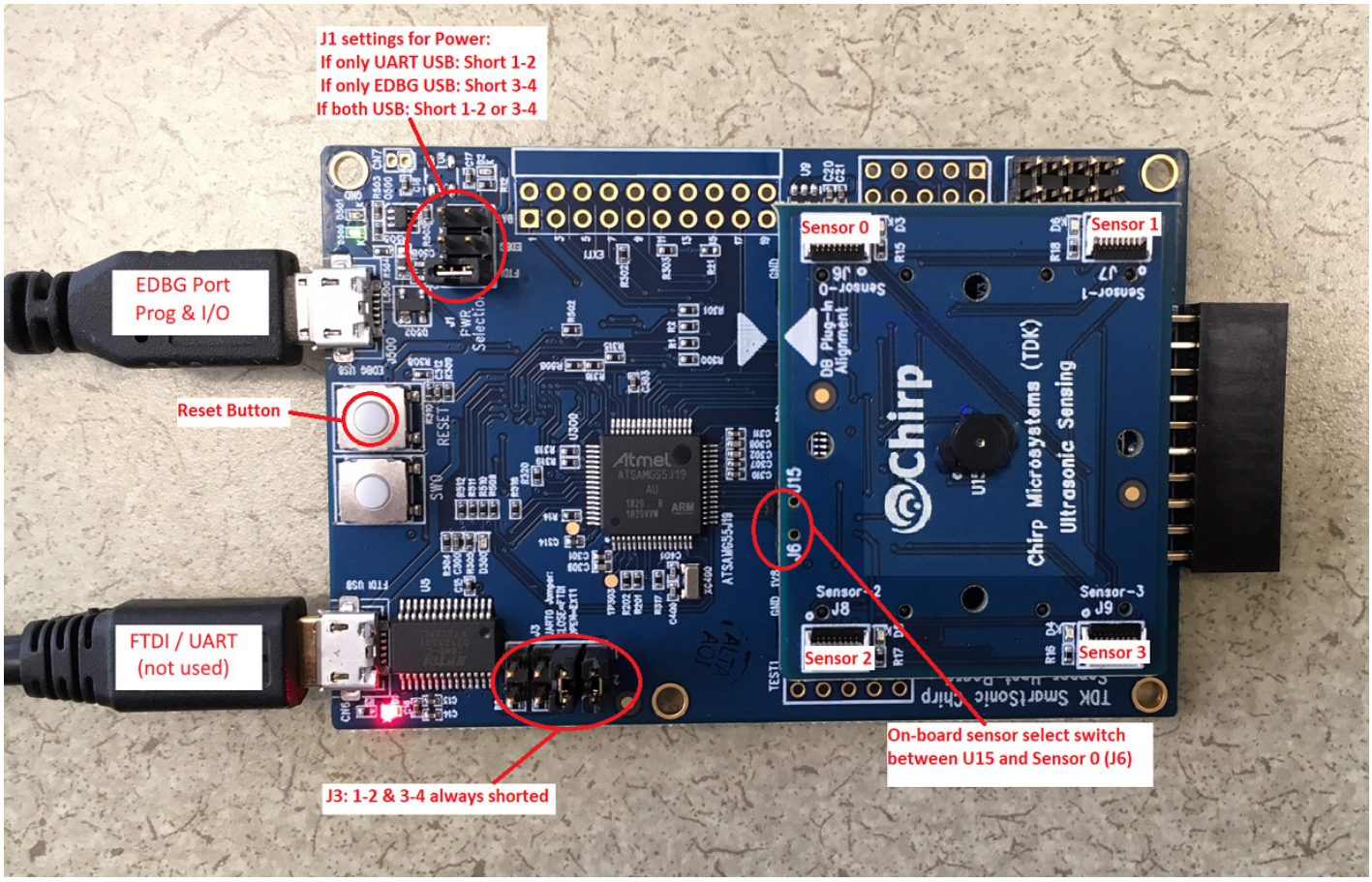

Figure 1. SmartSonic with CH101 Daughter Board

7. Open Windows Device Manager, open the Ports (COM & LPT) list, and identify the COM port number(s) assigned to the SmartSonic board.

   There are two possible ports associated with the SmartSonic board: "**EDBG Virtual COM Port**" and "**USB Serial Port**".

   - The **EDBG Virtual COM Port** is the connection that will be used in this application. It is used to connect to the on-board debugger for programming the board. It also is used for regular serial I/O to or from an application running on the board (like Hello Chirp). You will need to **specify this port number when opening the terminal emulator** to display output from the program when it runs.
   - The **USB Serial Port** entry will appear if a second USB cable is connected to the FTDI / UART Serial Port. This port is not used in this application.

2 FILE ORGANIZATION

The file organization used in Hello Chirp follows the same structure used for other Chirp example applications. The directories are organized so that multiple applications can co-exist and share common SonicLib and BSP components.

The Hello Chirp project source files are organized in three sub-directories under **source** in the project directory:

- **source/application/smartsonic-hellochirp-example** – contains **src** and **inc** directories with the Hello Chirp application.
  - The **src/main.c** file contains the entry point for the application along with various routines that demonstrate how to read and manage the Chirp sensor(s). See the comments in that file for detailed information about the operation of the application.
  - The **inc/app_config.h** file contains various definitions that control the behavior of the application and Chirp sensors. These include sensor configuration values such as operating mode and maximum range and the measurement interval for the application.
- **source/drivers/chirpmicro** – contains the SonicLib API and driver files, sensor firmware modules, and other distribution files from Chirp.
  - The **chirpmicro/src** directory contains the source code for the SonicLib API and driver, sensor firmware interface files, and sensor firmware images.
  - The **chirpmicro/inc** directory contains header files that must be included when building applications with SonicLib. In particular, the **soniclib.h** file contains the key definitions for the SonicLib API.
  - The **chirpmicro/html** directory contains HTML documentation for SonicLib. Open the **index.html** file in a web browser to get started.
- **source/board** – contains support files for the Chirp SmartSonic board and the Atmel SAMG55 microcontroller.
  - The main board support package routines, which implement the interfaces defined in the **chirp_bsp.h** file, are in the **board/HAL/src/chbsp_chirp_samg55.c** file.
  - The **chirp_board_config.h** file, required by SonicLib, is in the **board/config** directory. This file contains definitions for the number of possible devices and I^2C buses on the board, which are used for static allocation of arrays.

3 BUILDING THE HELLO CHIRP APPLICATION

1. Open Atmel Studio 7
2. Open the Hello Chirp project:
    - Open **File** menu
    - Select **File > Open > Project/Solution**
    - Locate and select the **atmelstudio/smartsonic-hellochirp-example/smartsonic-hellochirp-example.atsln** file in the project directory.
    - Click **Open**. The program should find the project files and display the name of the project.
3. If you are using a CH201 sensor, you will have to change the sensor firmware selection from the default, by changing the definition of **CHIRP_SENSOR_FW_INIT_FUNC** in the **app_config.h** header file. See Section 6, Choosing the Sensor Firmware Image, below.
4. Build the project:
    - Select **Build > Build Solution**

    The project should build successfully. The default build configuration is “Debug” so the build output files will be placed in the **atmelstudio/smartsonic-hellochirp-example/Debug** directory.

4 PROGRAMMING THE SMARTSONIC BOARD

1. Connect the **EDBG** port of the SmartSonic board to a Windows PC using micro-USB cable.

2. In Atmel Studio 7 select **Tools > Device Programming**. The Device Programming screen will appear:

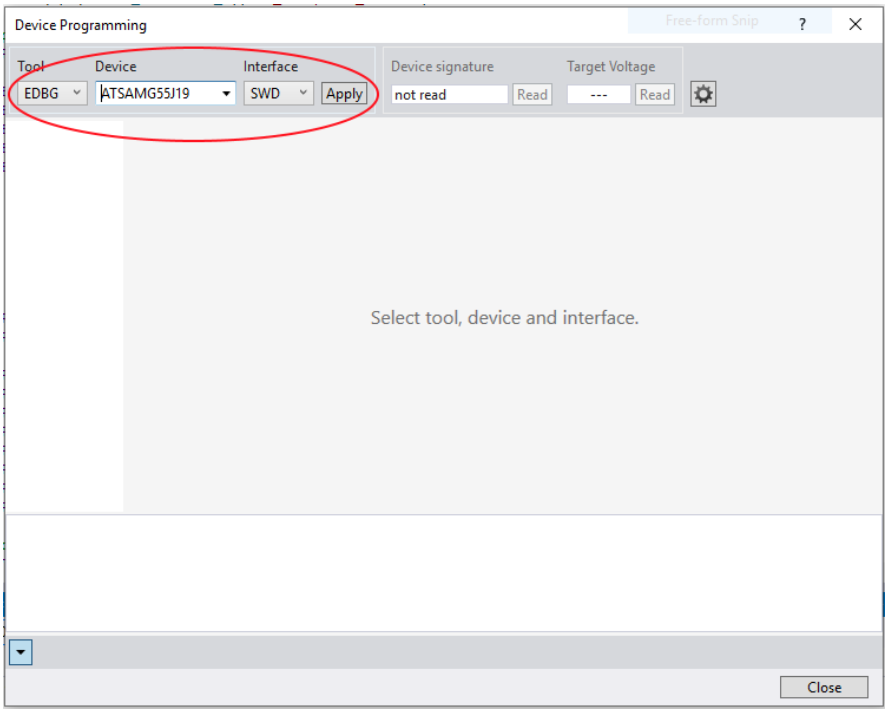

Figure 2. Device Programming Screen

3. Verify that the tool is **EDBG**, device is **ATSAMG55J19**, and interface is **SWD**. Select **Apply**.

    - **Note: Atmel Studio 7 may require you to update the EDBG debug interface firmware on the SmartSonic board before continuing. Follow the on-screen instructions to update the EDBG firmware.**

4. The Device Programming screen will prompt you to set the programming clock frequency. Leave the clock frequency unchanged (use the default).

5. Select **Read** near the **Device signature** field. The device signature bytes and target voltage should be read and should not generate any error messages. The Device programming menu should look as follows:

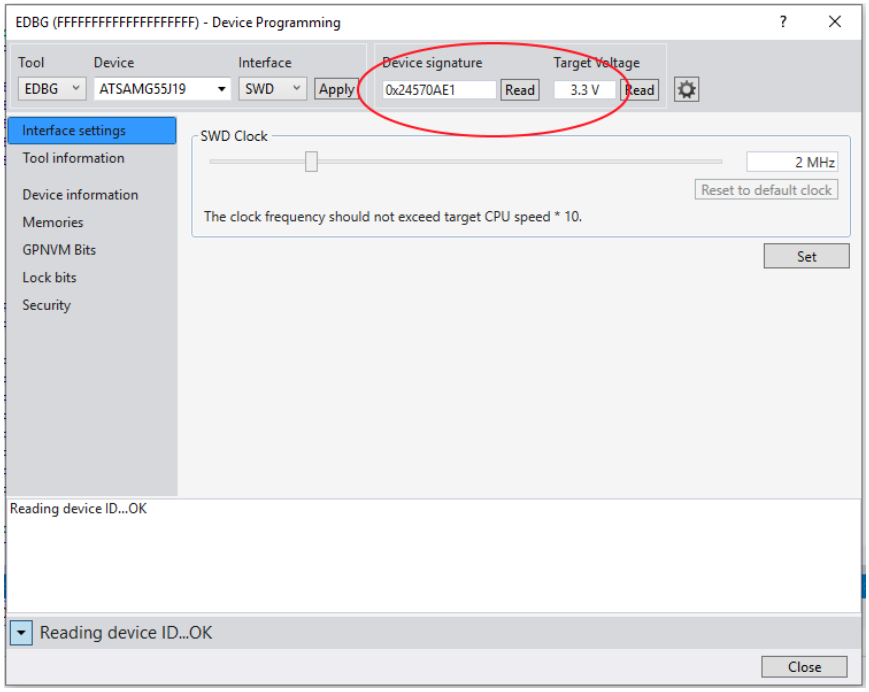

Figure 3. Device Signature and Target Voltage

6. Select **Memories** on the Device Programming menu. The Device Programming menu will prompt for the name of the file to program:

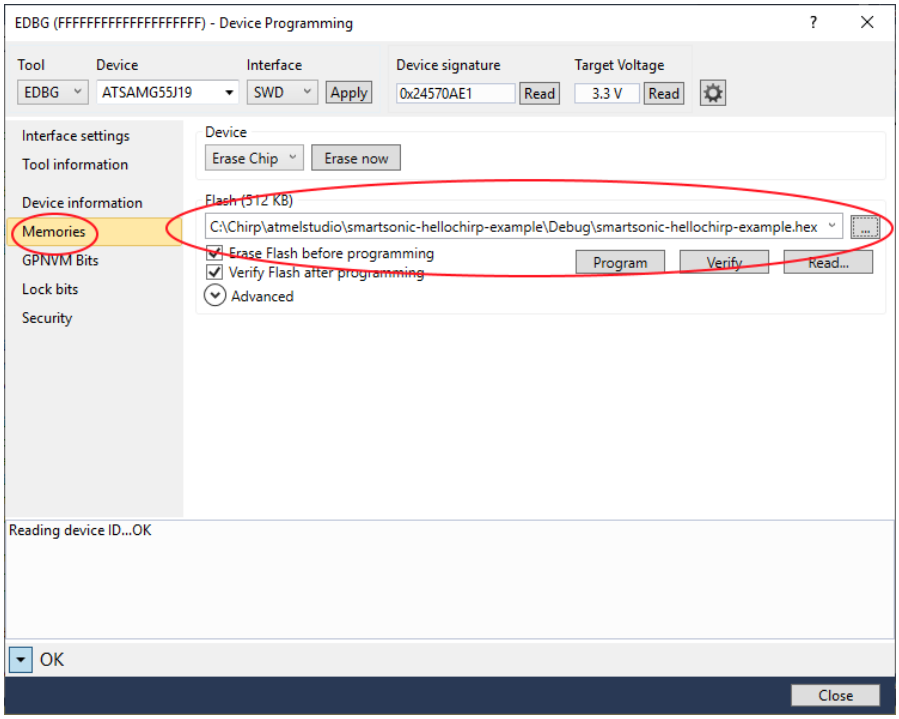

Figure 4. Programming Hex File

7. Navigate to the project's Debug directory and select the **smartsonic-hellochirp-example.hex** file.
    - Note: Alternately, you may use the **smartsonic-hellochirp-example.elf** file, which contains symbolic debug information. (Both files are generated when you build the application. If you are simply running the Hello Chirp application, and do not plan to use the Atmel Studio 7 debugging features, it does not make a difference which file you use.)
8. Select **Program**. The application binary will be programmed into the SmartSonic board's memory. Your SmartSonic board is successfully programmed when the Device Programming screen displays the "OK" messages shown below on the bottom left:

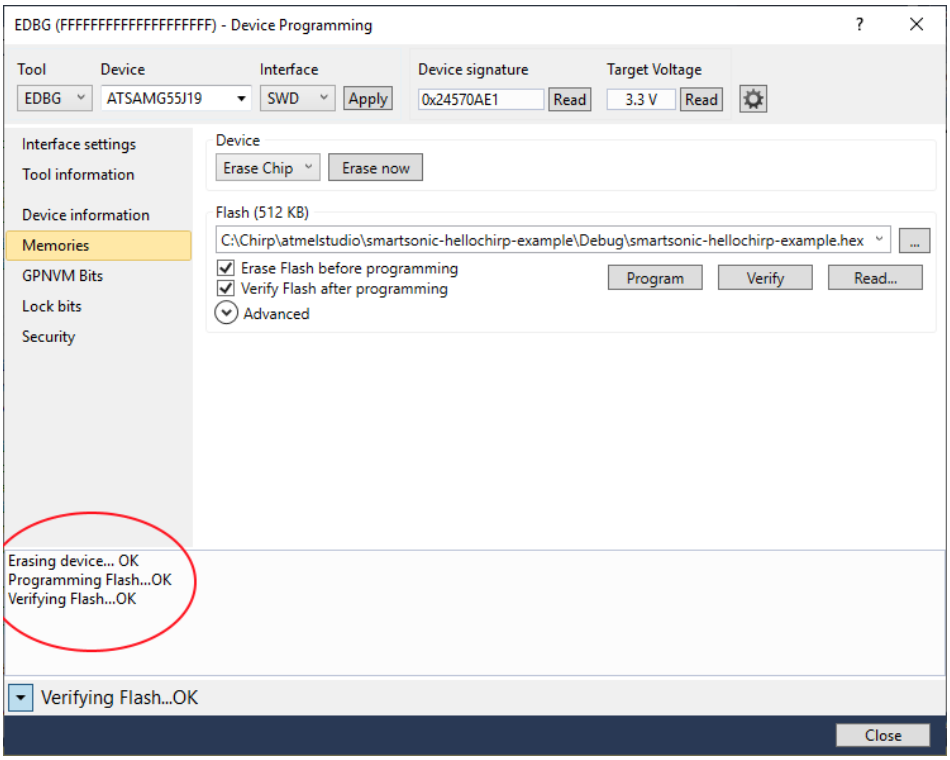

Figure 5. Successful Programming

5 RUNNING THE HELLO CHIRP APPLICATION

- Start the terminal emulator program and open/configure the COM port assigned to the SmartSonic board "EDBG Virtual COM Port":
    - **1000000 baud**
    - **8 bits data, no parity, 1 stop bit**
    - **New-line sequence = Line Feed only (no carriage return)**
- Reset the SmartSonic board using the board's reset button (next to the EDBG USB connector).
- Status messages from the application will appear on the terminal output, followed by summary data from the sensor initialization (device frequency, etc.) and configuration (maximum range, etc.).
- Range measurement and amplitude data from the sensor device(s) will then be output in a continuous loop, as shown in Figure 6, below. If two sensors are being used (e.g. in a Pitch-Catch configuration), each line will include data from both sensors. For all other configurations, each sensor's data is output on a separate line, in sequence.

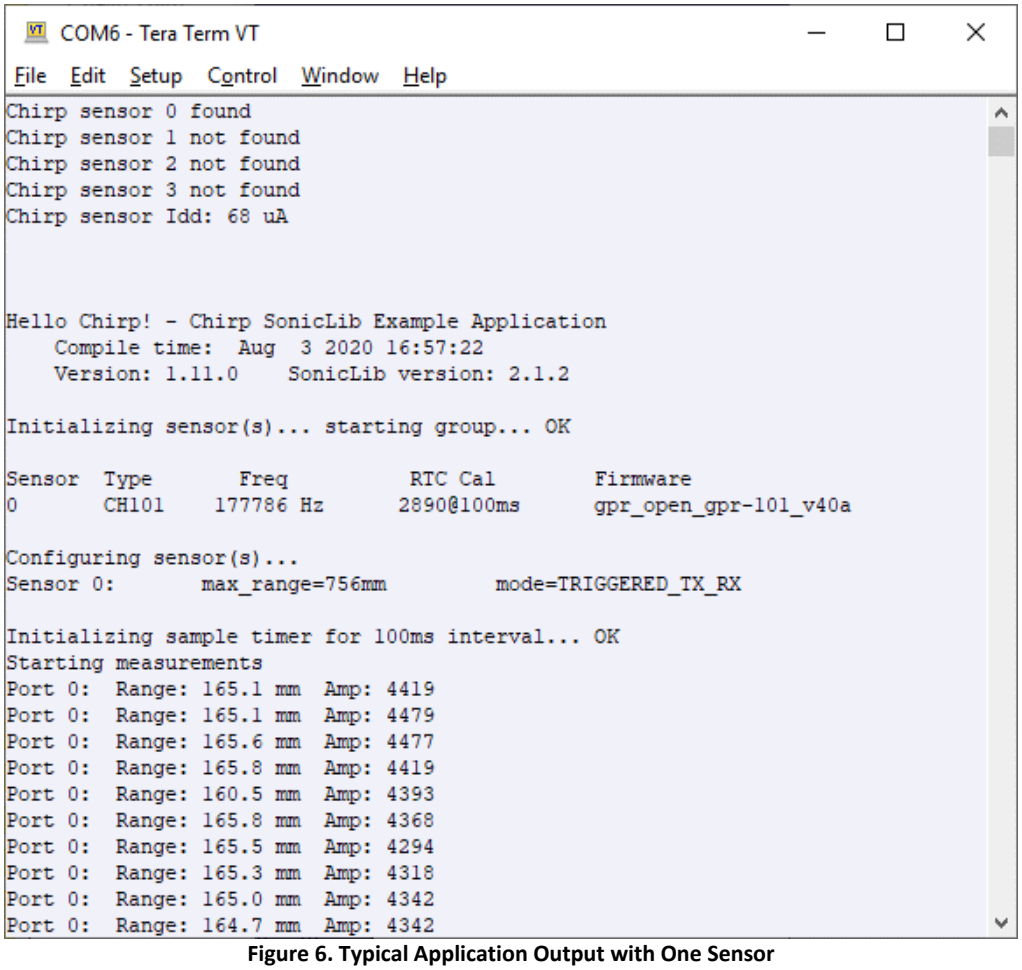

Figure 6. Typical Application Output with One Sensor

6 CHOOSING THE SENSOR FIRMWARE IMAGE

Chirp sensors are fully programmable, and during initialization they must be loaded with a specific firmware image to operate. Different sensor firmware types can provide different features or performance characteristics.

By default, the Hello Chirp example application uses one of the standard Chirp "GPR" (General Purpose Rangefinding) sensor firmware types. These firmware versions provide good performance over various distances under most conditions.

The firmware type (and therefore the sensor model, CH101 or CH201) is selected in the **app_config.h** file by the **CHIRP_SENSOR_FW_INIT_FUNC** symbol. This definition specifies which firmware initialization routine will be passed to the ***ch_init()*** function when each sensor is initialized in the Hello Chirp ***main()*** function, so that the selected firmware image will be programmed into the device.

In this example project, three types of sensor firmware are typically used, two for CH101 devices (CH101 GPR Open and CH101 GPR Short-range Open) and one for CH201 devices (CH201 GPRMT). The "Sensor Firmware Selection" section of the **app_config.h** header file contains a definition of **CHIRP_SENSOR_FW_INIT_FUNC** for each of these variants. You should uncomment the one line that corresponds to the firmware you want to use:

- The **CH101 GPR Open** firmware can be used for sensing up to one meter away. This is the default firmware type selected by the **app_config.h** file. Uncomment the line reading:

  ```
  #define CHIRP_SENSOR_FW_INIT_FUNC ch101_gpr_open_init
  ```

- The **CH101 GPR Short-range Open** firmware is optimized for short-range performance. This special firmware provides more measurement resolution and operates at closer distances. The resolution of the sensor is increased by a factor of four, however the maximum operating range for the sensor is therefore reduced by a factor of 4. Uncomment the line reading:

  ```
  #define CHIRP_SENSOR_FW_INIT_FUNC ch101_gpr_sr_open_init
  ```

  When using the special short-range firmware, you should consider changing the maximum range setting for the sensor, also defined in **app_config.h**, from the default (750 mm) to 250 mm or less. If left unchanged, the sensor will use the maximum possible range for this firmware.

- The **CH201 GPRMT** firmware is used with the CH201 long-range sensor. It provides multiple programmable detection thresholds to allow more control over object detection at greater distances (up to 5m). Uncomment the line reading:

  ```
  #define CHIRP_SENSOR_FW_INIT_FUNC ch201_gprmt_init
  ```

It is also possible to use the Hello Chirp example application with other sensor firmware releases from Chirp. Define the **CHIRP_SENSOR_FW_INIT_FUNC** symbol to equal the name of the new firmware's initialization function, following the same pattern as the GPR firmware types. For more information on adding new sensor firmware types to an existing installation, see the ***AN-000175 SonicLib Programmer's Guide*** document.

7 CHANGING THE APPLICATION CONFIGURATION

The **app_config.h** file in the Hello Chirp example contains a set of definitions that control the sensing performed by the ultrasonic sensor. Three key controls are for the sensor operating mode, sensing range, and measurement timing.

SENSOR MODE

There are separate definitions specifying the sensor mode that will be used by the first (lowest numbered) sensor and the mode that will be used by all other sensors. This allows one device (the first sensor) to operate in Transmit/Receive mode while all others operate in Receive-only mode, for Pitch-Catch operation.

- **CHIRP_FIRST_SENSOR_MODE** – This symbol sets the mode that will be used by the first (lowest numbered) sensor. If only one sensor is attached, this value must be either CH_MODE_TRIGGERED_TX_RX or CH_MODE_FREERUN.
- **CHIRP_OTHER_SENSOR_MODE** – This symbol sets the mode that will be used by all other sensors (if any).

The default definitions in **app_config.h** are:

CHIRP_FIRST_SENSOR_MODE = CH_MODE_TRIGGERED_TX_RX

CHIRP_OTHER_SENSOR_MODE = CH_MODE_TRIGGERED_RX_ONLY

The default settings will work for either a single sensor or multiple sensors in Pitch-Catch operation.

MAXIMUM RANGE

The maximum range at which the sensor can detect objects can be changed by modifying the definition of **CHIRP_SENSOR_MAX_RANGE_MM**. This value of this symbol is the sensor's maximum range, in millimeters.

By default, the maximum range setting is 750 mm, but it can be adjusted to a shorter or longer distance.

The true maximum range that can be set for any given sensor will vary, due to differences in the sensor's operating frequency and other factors. The following are typical usable range settings for the basic sensor firmware types:

CH101 GPR Open	up to ~1000 mm
CH101 GPR SR Open (short-range)	up to ~250 mm
CH201 GPRMT	up to ~5000 mm

If the value specified for CHIRP_SENSOR_MAX_RANGE_MM is greater than the maximum possible range for the individual sensor and firmware, the sensor's maximum possible range will be used.

At startup, the Hello Chirp application displays the actual maximum range setting being used for each sensor. The maximum range that is reported may be slightly different than the value of CHIRP_SENSOR_MAX_RANGE_MM, due to the granularity of the samples that make up a whole measurement.

MEASUREMENT TIMING

In the application, the sensors continuously generate new measurement results. The **MEASUREMENT_INTERVAL_MS** symbol specifies how often a new measurement should occur, in milliseconds. The default interval in **app_config.h** is 100 ms (i.e. a 10 Hz sampling rate).

For sensors in triggered mode (CH_MODE_TRIGGERED_TX_RX or CH_MODE_TRIGGERED_RX_ONLY), the application will use a periodic timer in the SAMG55 microcontroller to trigger a sensor measurement each time this period elapses.

For sensors in free-running mode (CH_MODE_FREERUN), the application will set this period as the sensor's internal sample interval.

8 REVISION HISTORY

REVISION DATE	REVISION	DESCRIPTION
08/02/2019	1.0	Advance Draft Release
11/01/2019	1.1	Title changed from "CH-101 Example Driver Hands On" to "SmartSonic Hello Chirp Application Hands-on Exercise".
08/05/2020	1.2	Updated for SonicLib 2.1 and new application file structure.

TDK Chirp Microsystems