ICU-X0201 HELLO CHIRP EXAMPLE

USER GUIDE

InvenSense, a TDK Group Company
1745 Technology Drive, San Jose, CA 95110 U.S.A
+1(408) 988–7339
invensense.tdk.com

Document Number: AN-000329
Revision: 2.3
Release Date: 02/10/2023

INTRODUCTION

This example program demonstrates how to build and run a simple ultrasonic sensing application using the Chirp SonicLib API and driver for ICU-x0201 sensors. The application runs on the SmartSonic2 board, and it can operate with either a single sensor or multiple sensors connected to the board, and it is recommended as the first example for developers to follow when integrating ICU-x0201 sensors in an embedded product.

REQUIRED EQUIPMENT

- TDK SmartSonic2 board
- TDK Ultrasonic ToF EVK board
- ICU-10201 or ICU-20201 sensor module(s) with flex cable(s)
- Micro-USB cable(s)

REQUIRED SOFTWARE PACKAGES

- **Invn.chirpmicro.smartsonic2.icux0201-hellochirp-example.*X_X_X*.zip** (actual file name will include version number), includes:
  - "Hello Chirp" application source files
  - Chirp SonicLib sensor API and driver files. The application requires SonicLib v3.16.0 or later.
  - Sensor firmware image files
  - Board support package files for SmartSonic2 board
  - MPLAB X project files to build the application
- MPLAB X integrated development environment – download from Microchip.com
- Terminal emulator of your choice (for example TeraTerm or PuTTY)

TABLE OF CONTENTS

LIST OF FIGURES

1 INSTALLATION / PREPARATION

- Follow instructions from the SmartSonic2 Programming guide.
- Optionally, download and install the **ICU-x0201 Evaluation Kit EVK** Application. The EVK is not required to use this example, but you can use the tool to experiment with different measurement configurations, then export the settings to use with Hello Chirp.
- Install terminal emulator.
- *Note: This example application only requires one cable, because the Debug Port is used for both the IDE connection and the serial output from the program. Other applications, notably the ICU-x0201 Evaluation Kit (EVK), may also require a second cable connected to the Applications Port, as shown in Figure* 1.

2 BUILDING AND PROGRAMMING THE EXAMPLE APPLICATION AND SONICLIB

- Please refer to the SmartSonic2 Programming Guide for building and programming instructions.
- The Hello Chirp project source files are organized in three sub-directories under **source**:
    - **application** – contains **src** and **inc** directories with the Hello Chirp application
        - The **application/icux0201-hellochirp-example/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 **application/icux0201-hellochirp-example/inc/app_config.h** file contains various definitions that control the behavior of the application and ultrasonic sensors. These include sensor configuration values such as sensor refresh sample rate and maximum range. **Modify the settings in this file and rebuild to change the configuration.**
    - **drivers/invn-soniclib/invn/soniclib** – contains the SonicLib API and driver files, sensor firmware modules, and other distribution files from InvenSense.
        - The directory contains header files that must be included when building applications with SonicLib. In particular, the **soniclib.h** file contains the interface definitions for the SonicLib API.
        - The **drivers/invn-soniclib/invn/soniclib** directory must be in the include path when building. This is already set in the supplied project files.
    - **board** – contains support files for the SmartSonic2 board.
        - The required board support package interface routines, as defined in **drivers/invn-soniclib/invn/soniclib /chirp_bsp.h**, are in the **board/smartsonic2/HAL/src/chbsp_chirp_samg55.c** file.

3 RUNNING THE EXAMPLE APPLICATION

- Start the terminal emulator program and open/configure the COM port assigned to the SmartSonic2 board (as found earlier using Windows Device Manager):
  - **921600 baud**
  - **8 bits data, no parity, 1 stop bit**
  - **New-line sequence = Line Feed only (no carriage return)**
- Reset the board using the board's reset button.
- Status messages from the application will appear on the terminal output, followed by summary data from the sensor initialization (device frequency, measurement configuration, etc.).
- Range measurement data from the sensor device(s) will be output in a continuous loop. Note that distance measurements are expressed in **millimeters**.

```
Hello Chirp! - SonicLib Example Application for ICU Sensors
    Compile time:  Oct 25 2022 14:47:17
    Version: 2.18.0    SonicLib version: 3.16.5

Initializing sensor(s)... starting group... OK

Dev   Type       ID     Freq Hz  B/W Hz   RTC Cal    CPU Freq    Firmware
 0  ICU-10201  071005J  177116    4184   2984@100ms  40.35MHz  gpt_1.3.3-rc.0

Configuring Device 0...
  Initializing measurement 0... OK
  Adding tx segment... OK
  Adding rx segment 0... OK
  Adding rx segment 1... OK
  Writing meas queue... OK
  Setting max range to 750 mm... OK
  Enabling in CH_MODE_TRIGGERED_TX_RX mode... OK
Device 0: Configuration OK

Device 0: Measurement 0 Configuration
  Total Samples = 97  (751 mm max range)        Ringdown samples = 20
  Active Segments = 3   Rate = CH_ODR_FREQ_DIV_8
   Seg 0  TX                  640 cycles  Pulse width =  3  Phase = 8
   Seg 1  RX      1 sample    128 cycles  Gain reduce = 24  Atten = 1
   Seg 2  RX     96 samples  12288 cycles  Gain reduce =  0  Atten = 0  Done Int
   Seg 3  EOF                   0 cycles
  Detection Thresholds:
     0  start sample:   0  =    0 mm    level: 1600
     1  start sample:  40  =  309 mm    level: 800
     2  start sample:  70  =  542 mm    level: 400
     3  start sample: 100  =  774 mm    level: 200
     4  start sample: 160  = 1239 mm    level: 140
     5  start sample: 200  = 1549 mm    level: 95
     6  start sample: 240  = 1859 mm    level: 50

Initializing sample timer for 100ms interval... OK

Starting measurements...
Dev 0:  Range: 218.2 mm (sample 29) amp= 4170
Dev 0:  Range: 218.3 mm (sample 29) amp= 4247
Dev 0:  Range: 218.4 mm (sample 29) amp= 4272
Dev 0:  Range: 218.3 mm (sample 29) amp= 4293
```

Figure 1. Example Application Output

4 SENSOR CONFIGURATION SETTINGS

The **app_config.h** header file contains symbolic definitions for parameters that affect the sensor measurements and application execution. You can change these definitions to adjust the overall operation.

> ***Note:*** You should get the Hello Chirp application running on your board using the default configuration **BEFORE** experimenting with configuration settings!

The values defined in **app_config.h** whose names begin with "CHIRP_" are used as parameters to SonicLib API function calls in **main.c**. In particular, the *configure_sensors()* routine in that file uses many of these values to apply different configuration settings and options during the initialization sequence.

Please refer to the usage and comments in **main.c** to see how the SonicLib API is called based on these symbols. See *AN-000175 SonicLib Programmer's Guide* and the included SonicLib HTML documentation for more information

4.1 CHIRP_FIRST_SENSOR_MODE

Specifies the operating mode for the first sensor (lowest numbered) that is present. If only one sensor is attached, this value must be either CH_MODE_TRIGGERED_TX_RX or CH_MODE_FREERUN.

This value is used as a parameter to *ch_set_mode()* for the first detected sensor.

4.2 CHIRP_OTHER_SENSOR_MODE

Specifies the mode for all other sensors (i.e. except the first) that are present.

For typical Pitch-Catch operation using two or more sensors, set CHIRP_FIRST_SENSOR_MODE to CH_MODE_TRIGGERED_TX_RX and set CHIRP_OTHER_SENSOR_MODE to CH_MODE_TRIGGERED_RX_ONLY.

This application assumes that multiple sensors in a pitch-catch configuration will be physically separated and facing each other. Therefore, the range value from the receive-only sensor(s) will indicate the one-way, direct path distance between the sensors (CH_RANGE_DIRECT). See the *AN-000254 SonicLib Programmer's Guide* document for more information on multi-sensor configurations.

Pitch-Catch operation is currently not supported on ICU-20201 sensors. Therefore, ICU-20201 sensors should not be configured to use CH_MODE_TRIGGERED_RX_ONLY.

This value is used as a parameter to *ch_set_mode()* for the second and subsequent sensors.

4.3 CHIRP_TRIGGER_TYPE

Sets the trigger type for sensors in CH_MODE_TRIGGERED_TX_RX or CH_MODE_TRIGGERED_RX_ONLY mode.

Typically, sensors are triggered by applying a signal to the sensor's trigger input INT pin (hardware triggering). To trigger a sensor by generating pulses on the trigger line, set CHIRP_TRIGGER_TYPE to CH_TRIGGER_TYPE_HW.

To instead trigger sensors using the software (SPI) interface, set CHIRP_TRIGGER_TYPE to CH_TRIGGER_TYPE_SW.

Software triggering is not recommended for multi-sensor pitch-catch operation, because there is a time lag between triggering the sensors.

This value is used as a parameter to *ch_set_trigger_type()*.

4.4 SETTING THE MAXIMUM RANGE

Specifies how long the sensor "listens" for an ultrasound signal.

The value set here provides a convenient way to specify the length of a measurement in real-world units. It will be applied after the measurement is defined and may therefore change the number of Rx samples specified below in 4.12 Defining Measurement Segments). Note that the maximum possible range may vary depending on sensor model and sensor firmware type. If the value specified here is greater than the maximum possible range, the maximum possible range will be used.

Using a longer maximum range setting will increase the time required for each measurement (and therefore increase the power consumption) and will generate more raw sample data, if used. So, you should select a range value that is sufficient for your actual sensing needs but is not excessively long.

The appropriate value for the detected sensor type is used as a parameter to *ch_set_max_range()*.

CHIRP_ICU10201_MAX_RANGE_MM	Sets the maximum detection range for ICU10201 sensors, in millimeters. Typical max range is 1250 mm or less.
CHIRP_ICU20201_MAX_RANGE_MM	Sets the maximum detection range for ICU20201 sensors, in millimeters. Typical max range is 5000 mm or less.

4.5 CHIRP_STATIC_REJECT_SAMPLES

Specifies if static target rejection (STR) will be used.

If CHIRP_STATIC_REJECT_SAMPLES is non-zero, STR will be enabled and will apply to the specified number of samples at the beginning of a measurement. The sensor will only report targets that are non-stationary. Targets whose reflections do not change between measurements will be ignored.

To enable STR filtering for the entire measurement, set this value to **ICU_MAX_NUM_SAMPLES**.

It may be necessary to lower the detection threshold levels in the sample ranges where STR is used to get the desired sensitivity (see section 4.10 below).

STR is often combined with target interrupt filtering to reduce the number of data ready interrupts, for power conservation.

This value is used as a parameter to *ch_set_static_range()*.

4.6 CHIRP_TARGET_INT_FILTER

Enables or disables target interrupt filtering.

This setting controls whether the sensor will generate an interrupt after every measurement, or only if a target is detected. By default, a data ready interrupt is generated after every measurement whether a target object was detected or not. (Note that the "Done Interrupt" must be enabled for the final receive segment in the measurement to generate a "data ready" interrupt.)

Set CHIRP_TARGET_INT_FILTER to non-zero to enable target interrupt filtering. The sensor will only generate a data ready interrupt if a target was detected during the measurement.

This feature is often combined with static target rejection to reduce the number of data ready interrupts, for power conservation.

This value is used as a parameter to *ch_set_target_interrupt()*.

4.7 CHIRP_RX_HOLDOFF_SAMPLES

Controls the Receive Holdoff feature.

This value specifies a certain number of samples at the start of each measurement that will be ignored during target detection. Although data from these samples will be measured (and reported if amplitude or I/Q output is enabled), no target will be reported within this range, regardless of its signal amplitude. This may be useful for ignoring objects close to the sensor or handling difficult acoustic situations.

For ICU sensors, the Rx Holdoff setting works by modifying the detection thresholds for the specified range of samples. To return to the original threshold values (and undo the Rx Holdoff setting), use the *ch_set_thresholds()* function and pass in the original threshold definition again.

Set CHIRP_RX_HOLDOFF_SAMPLES to non-zero to specify the number of samples to ignore, or zero to disable.

This value is used as a parameter to *ch_set_rx_holdoff()*.

4.8 CHIRP_RX_PRETRIGGER_ENABLE

Enables receive sensor pre-triggering.

This value specifies if receive-only sensor pre-triggering will be used. This setting only applies if more than one sensor is used and at least one sensor is operating in CH_MODE_TRIGGERED_RX_ONLY.

Receive pre-triggering improves performance in pitch-catch operation at short distances, by triggering the receive-only sensor(s) slightly before the transmitting sensor. However, this setting will reduce maximum range of Rx-only sensors approximately 200mm, relative to the CHIRP_MAX_RANGE_MM setting, above.

Set RX_PRETRIGGER_ENABLE to non-zero to enable receive pre-triggering, or zero to disable.

This value is used as a parameter to *ch_set_rx_pretrigger()*.

4.9 CHIRP_MAX_TARGETS

Sets maximum number of targets to report for a single measurement.

The specifies the maximum number of separate target objects for which the sensor will report range and amplitude values. For the standard ICU_GPT sensor firmware used in this example, the maximum number of targets that may be reported is 5.

This value is used to initialize the *meas_config* (**ch_meas_config_t**) structure defined in **main.c**, which is passed to *ch_meas_init()*.

4.10 SETTING TARGET DETECTION THRESHOLDS

These definitions set the sensor's target detection thresholds. These thresholds specify how large a signal must be received, and at what point in the measurement, for the sensor to indicate that a target was detected and calculate range, etc. These settings are ignored if IMPORT_MEASUREMENT is defined (see below).

Each threshold consists of a starting sample number within the measurement and the corresponding amplitude level that must be reached. A threshold extends until the starting sample number of the next threshold, if any. At each sample point, the observed amplitude is compared against the threshold level for that sample number.

Lower threshold levels will increase sensitivity for detecting objects, but they can also increase the rate of “false positives.” So, some experimentation is usually required to optimize the thresholds for a particular use.

Not all thresholds must be used, if your sensing environment and application do not require 8 different threshold values. Unused threshold entries may be set to zero, and the last threshold will be used for the remainder of the measurement.

These values are used to initialize the *chirp_detect_thresholds* (**ch_thresholds_t**) structure defined in **main.c**, which is passed to *ch_meas_init()*.

CHIRP_THRESH_0_START	Threshold 0 (closest) starting sample.
CHIRP_THRESH_0_LEVEL	Threshold 0 level.
CHIRP_THRESH_1_START	Threshold 1 starting sample.
CHIRP_THRESH_1_LEVEL	Threshold 1 level.
CHIRP_THRESH_2_START	Threshold 2 starting sample.

CHIRP_THRESH_2_LEVEL	Threshold 2 level.
CHIRP_THRESH_3_START	Threshold 3 starting sample.
CHIRP_THRESH_3_LEVEL	Threshold 3 level.
CHIRP_THRESH_4_START	Threshold 4 starting sample.
CHIRP_THRESH_4_LEVEL	Threshold 4 level.
CHIRP_THRESH_5_START	Threshold 5 starting sample.
CHIRP_THRESH_5_LEVEL	Threshold 5 level.
CHIRP_THRESH_6_START	Threshold 6 starting sample.
CHIRP_THRESH_6_LEVEL	Threshold 6 level.
CHIRP_THRESH_7_START	Threshold 7 (farthest) starting sample.
CHIRP_THRESH_7_LEVEL	Threshold 7 level.

4.11 IMPORT_MEASUREMENT

This definition controls whether the example application will use a measurement definition imported from a file (for example, one exported from the ICU-x0201 EVK evaluation kit) instead of using the settings defined below. If a measurement is imported, it will also set the target detection thresholds.

If you set IMPORT_MEASUREMENT to non-zero, the application will use the measurement defined in the example **measurement_config.c** file.

See Section 7 Optional: Using a Measurement from the ICU-x0201 EVK for information on using the EVK to generate a measurement definition that may be imported here.

4.12 DEFINING MEASUREMENT SEGMENTS

The following definitions specify the settings for the transmit and receive segments within a measurement. These settings are ignored if IMPORT_MEASUREMENT is defined (see above).

This example uses one transmit segment and two receive segments.

These values are used as parameters to *ch_meas_add_segment_tx()* and *ch_meas_add_segment_rx()*.

CHIRP_TX_SEG_CYCLES	Transmit segment length (duration), in cycles. In general, a pulse using a longer transmit length will have a higher amplitude when the echo is received.
CHIRP_TX_SEG_PULSE_WIDTH	Transmit pulse width within each cycle. This is the length of the positive pulse of the TX waveform. Making it shorter will reduce the effective drive strength. Valid values are from 0 to 4. Each unit of pulse_width is 1/8th of a cycle.
CHIRP_TX_SEG_PHASE	Phase of the transmit waveform relative to the internal clock generator. Valid values are from 0 to 15. Each unit of phase is 1/16th of a cycle.
CHIRP_TX_SEG_INT_EN	Whether to interrupt at end of transmit segment (non-zero = generate interrupt). Normally not enabled.

CHIRP_RX_SEG_0_SAMPLES	First receive segment length, in samples. This example uses a very short (1 sample) segment with reduced gain for very close to the sensor.
CHIRP_RX_SEG_0_GAIN_REDUCE	First receive segment gain reduction, in dB. Possible values are 0 (no reduction in gain) to 31 (maximum reduction), but values 28-31 all result in the same reduction of approximately 28dB
CHIRP_RX_SEG_0_ATTEN	First receive segment attenuation, to avoid saturation. Valid values are 0 (no attenuation) to 3 (maximum attenuation). Each increment (starting from zero) will drop the effective receiver gain by a factor of 8.
CHIRP_RX_SEG_0_INT_EN	Whether to interrupt at end of first receive segment (non-zero = generate interrupt). Normally not enabled.
CHIRP_RX_SEG_1_SAMPLES	Second receive segment length, in samples.
CHIRP_RX_SEG_1_GAIN_REDUCE	Second receive segment gain reduction, in dB (0 to 31).
CHIRP_RX_SEG_1_ATTEN	Second receive segment attenuation (0 to 3).
CHIRP_RX_SEG_1_INT_EN	Whether to interrupt at end of second receive segment (non-zero = generate interrupt). This interrupt is subject to target interrupt filtering, if enabled.

4.13 CHIRP_RINGDOWN_FILTER_SAMPLES

Sets the ringdown filtering range.

The ICU sensor performs "ringdown cancellation" filtering to remove artifacts from the received signal in the first samples in a measurement. This symbol specifies how many samples at the beginning of the measurement will be subject to ringdown filtering.

Because the ringdown artifacts are generally constant, the ringdown cancellation uses a difference filter. The unchanging ringdown signal is removed; however, any constant reflections within the ringdown filter range will also be removed. As a result, non-moving objects within the ringdown filtering range will generally not be detected, because their signals will be filtered out. However, close objects that are moving can still be detected.

This value is used to initialize the *meas_config* (**ch_meas_config_t**) structure defined in **main.c**, which is passed to *ch_meas_init()*.

4.14 CHIRP_SENSOR_ODR

Sets the sample output data rate.

This symbol specifies the sample output data rate (ODR) for the sensor, i.e. how often a sample is taken within each measurement. This allows more precise measurements at close distances, by increasing the rate at which internal samples are taken. Conversely, the number of samples in each measurement can be reduced (perhaps to conserve data storage) by decreasing the ODR.

The rate is set relative to the ultrasonic transducer (PMUT) operating frequency, which is typically in the range of 70-95 kHz for ICU-20201 and 170-200 kHz for ICU-10201. The default ODR (CH_ODR_FREQ_DIV_8 also known as CH_ODR_DEFAULT) generates one sample for every 8 PMUT cycles (sample rate = PMUT freq / 8).

Faster sampling (i.e. more samples for a given distance) can be enabled by setting the ODR to CH_ODR_FREQ_DIV_4 (sample rate = PMUT freq / 4) or CH_ODR_FREQ_DIV_2 (sample rate = PMUT freq / 2). Slower sampling (i.e. fewer samples for a given distance) can be enabled by setting the ODR to CH_ODR_FREQ_DIV_16 (sample rate = PMUT freq / 16) or CH_ODR_FREQ_DIV_32 (sample rate = PMUT freq / 32).

The ODR setting will affect how many samples are taken during the ringdown phase of the measurement. You may need to increase the value of CHIRP_RINGDOWN_FILTER_SAMPLES when using a faster ODR.

This value is used to initialize the *meas_config* (**ch_meas_config_t**) structure defined in **main.c**, which is passed to *ch_meas_init()*.

4.15 CHIRP_MEAS_OPTIMIZE

Enables measurement optimization.

The following symbol enables optimization of the measurement segments to improve performance at close distances that could otherwise be affected by sensor ringdown. Additional transmit segments will be inserted into the original sequence to actively dampen ringdown artifacts. These added segments will be visible when the measurement configuration is displayed during startup.

Set CHIRP_MEAS_OPTIMIZE to non-zero to enable measurement optimization. This setting applies whether the measurement definition was created locally using the SonicLib API or was imported from a file.

This value is used to control whether *ch_meas_optimize()* is called.

4.16 CHIRP_SENSOR_FW_INIT_FUNC

Selects which sensor firmware to use.

The sensor firmware type is specified during the call to *ch_init()*, by giving the name (address) of the firmware initialization function that will be called. The CHIRP_SENSOR_FW_INIT_FUNC symbol is used to specify the init routine for the sensor firmware to be used.

To use a different sensor firmware type (for example, a new distribution from TDK InvenSense), simply define CHIRP_SENSOR_FW_INIT_FUNC to be the name of the init routine for the new firmware.

This value is used as a parameter to *ch_init()*.

5 APPLICATION CONFIGURATION SETTINGS

The **app_config.h** header file also contains several definitions to control the behavior of the application itself. These include the measurement timing, data storage, and the optional reading and output of sample data.

5.1 MEASUREMENT_INTERVAL_MS

Defines how often the application will get a new sample from the sensor(s) This symbol defines the sensor measurement interval, in milliseconds.

For sensors in triggered mode (CH_MODE_TRIGGERED_TX_RX or CH_MODE_TRIGGERED_RX_ONLY), the application will use a periodic timer to trigger a sensor measurement each time this period elapses. This value is used as a parameter to the *chbsp_periodic_timer_init()* function in the board support package.

For sensors in free-running mode (CH_MODE_FREERUN), the application will set this period as the sensor's internal sample interval. This value is used as a parameter to *ch_set_sample_interval()*.

5.2 APP_DATA_MAX_SAMPLES

Defines how many samples per measurement are expected by this application. Maximum value is ICU_MAX_NUM_SAMPLES.

This value is used to allocate storage in the application-specific **chirp_data_t** structure defined in main.c. That structure contains arrays for individual data values (either amplitude or I/Q) from the raw samples within an ultrasound measurement.

5.3 ENABLING FULL SAMPLE DATA OUTPUT

The following build options control if and how the full set of values for all internal samples within an ultrasound measurement will be read and displayed. This data is separate from the standard range and simple target amplitude values that are normally output.

The measurement data can be read and displayed either as net amplitude values or as individual I and Q components.

Note that reading the full data set is not required for most basic sensing applications - the reported range value, possibly combined with the simple target amplitude value, is typically all that is required. However, the full set of sample values may be read and analyzed for more advanced sensing or data capture needs.

These values are used to enable, read, and output the appropriate sample data. This includes use of the *ch_get_amplitude_data()* and *ch_get_iq_data()* functions.

OUTPUT_AMP_DATA_CSV

Set OUTPUT_AMP_DATA_CSV to 1 (non-zero) to enable output of the amplitude data via the serial port, as a series of comma separated values all on one line. This allows easy cut/paste into a spreadsheet program to generate an "A-Scan" chart. This kind of summary graph can be very helpful to understand what the sensor is actually observing.

OUTPUT_AMP_LOG

Set OUTPUT_AMP_LOG to 1 (non-zero) to enable output of the amplitude data in InvenSense Red Swallow log format. This is not a common setting – Red Swallow logs typically contain I/Q data, see OUTPUT_IQ_LOG below.

OUTPUT_IQ_DATA

Set OUTPUT_IQ_DATA to 1 (non-zero) to enable output of the I/Q pair values. Each I/Q sample appears as a comma separated pair (Q, I) on its own line. Having the separate I and Q values allows more complex analysis of signal phase, etc.

OUTPUT_IQ_LOG

Set OUTPUT_IQ_LOG to 1 (non-zero) to enable output of the I/Q data in InvenSense Red Swallow log format. This data format includes header information and the full I and Q data as separate arrays, with one output line per measurement. The Red Swallow log format is also used by the ICU-x0201 Evaluation Platform tools.

Note: OUTPUT_AMP_DATA_CSV, OUTPUT_AMP_LOG, OUTPUT_IQ_DATA, and OUTPUT_IQ_LOG cannot be used together.

5.4 READ_DATA_NONBLOCKING

Selects blocking vs. non-blocking I/O when reading sample data. This setting only applies if OUTPUT_AMP_DATA_CSV, OUTPUT_AMP_LOG, OUTPUT_IQ_DATA, or OUTPUT_IQ_LOG has been selected to enable sample data output.

By default, this application will read sample data in blocking mode (i.e. READ_DATA_NONBLOCKING is zero by default). The amplitude or I/Q data will be read from the device and placed in the corresponding data array field in the application's **chirp_data_t** structure. Because the data is read in blocking mode, the calls to *ch_get_amplitude_data()* or *ch_get_iq_data()* will not return until the data has actually been copied from the device.

However, if READ_IQ_NONBLOCKING is non-zero, the I/Q data will be read in non-blocking mode. The *ch_get_amplitude_data()* or *ch_get_iq_data()* calls will return immediately, and a separate callback function will be called to notify the application when the read operation completes.

This value is used for conditional compilation of sections supporting non-blocking I/O. This build-time selection is used so that the board support package (BSP) is not required to provide the non-blocking functions if they are not needed.

5.5 DISPLAY_AMP_VALUE

Controls output of the amplitude value for detected targets.

if non-zero, the reported amplitude for detected target(s) is shown along with the range. (This option is enabled by default.)

5.6 DISPLAY_SAMPLE_NUM

Controls output of the amplitude value for detected targets.

if non-zero, the sample number in which a target was detected is shown along with the range. (This option is enabled by default.)

5.7 DISPLAY_MULTI_TARGET

Controls display of multiple detected targets.

By default (if DISPLAY_MULTI_TARGET is zero), only the range and other data from closest detected target is displayed.

If DISPLAY_MULTI_TARGET is non-zero, the range and other data from multiple detected targets will be displayed, up to the limit set by CHIRP_MAX_TARGETS (see above). The number of detected targets may vary from one measurement to another – only the successfully detected targets will be displayed.

6 OPTIONAL: USING A MEASUREMENT FROM THE ICU-X0201 EVK

By default, this example application configures the sensors based on the options described in the **app_config.h** file. However, it is possible to import the settings from an external set of data structures instead. This section describes how to use the ICU-x0201 Evaluation Kit (EVK) to define a measurement and create a file that will be imported by this application (if IMPORT_MEASUREMENT is non-zero).

Experimenting with the ICU-x0201 EVK can speed up the process of deciding what features and settings are needed by your application. The EVK provides interactive controls for many of the same measurement configuration options that may be set in this example, including:

- Reception (Rx) time (listening for a longer time allows further away targets to be detected, but increases the power consumption)
- Threshold levels (raising the threshold reduces the impact of small, unwanted echoes on the sensor performance, but reduces the maximum range)
- Transmission (Tx) time (using a shorter pulse allows closer range targets to be detected with higher fidelity, but limits the maximum range)
- Ringdown cancellation samples (extent of filtering for internal noise in the sensor at close distances - below this value, a non-moving target cannot be detected)
- Receive gain reduction and attenuation (using lower gain at close distances can increase the range fidelity when the echo signal is very large)
- Interrupt pin behavior, sensor triggering mode, and sensor measurement rate (when in free-running mode)
- Number of separate “targets” detected by the sensor during a measurement (with independent range values)

In most cases, the best way to begin defining a custom measurement is to use one of the four built-in “Quick Start” configurations. These are pre-defined measurements that optimize for different behaviors (standard, short-range, long-range, or static target rejection), and they can be used as a starting point for modifying and tuning the settings to fit your situation.

1. Open the ICU-x0201 EVK application and follow the instructions in the **ICU-x0201 EVK User Manual** to connect the ICU-x0201 sensors via the Ultrasonic ToF EVK Board. If you have been running Hello Chirp on your SmartSonic2 board, you will have to re-program the board with the EVK firmware (under Device, select Flash Default Firmware).
2. Click the **Configure Sensor** button in the **Widgets** panel to open the **ICU-x0201 GPT Configuration Utility**, in which you can specify the measurement parameters.
3. There are four “Quick Start” templates provided with the ICU-x0201 GPT Configuration Utility (see **Figure 2**):
    - **Default**: Default settings for general purpose rangefinding.
    - **Short Range**: Settings optimized for short range measurements. This special configuration provides more measurement resolution at close distances. However, the maximum range for the sensor is reduced significantly (by a factor of 4).
    - **Long Range**: Settings optimized for long range measurements.
    - **STR**: Settings configured for static target rejection.

    Choose the template that seems closest to your needs, and click it to load as your starting settings.

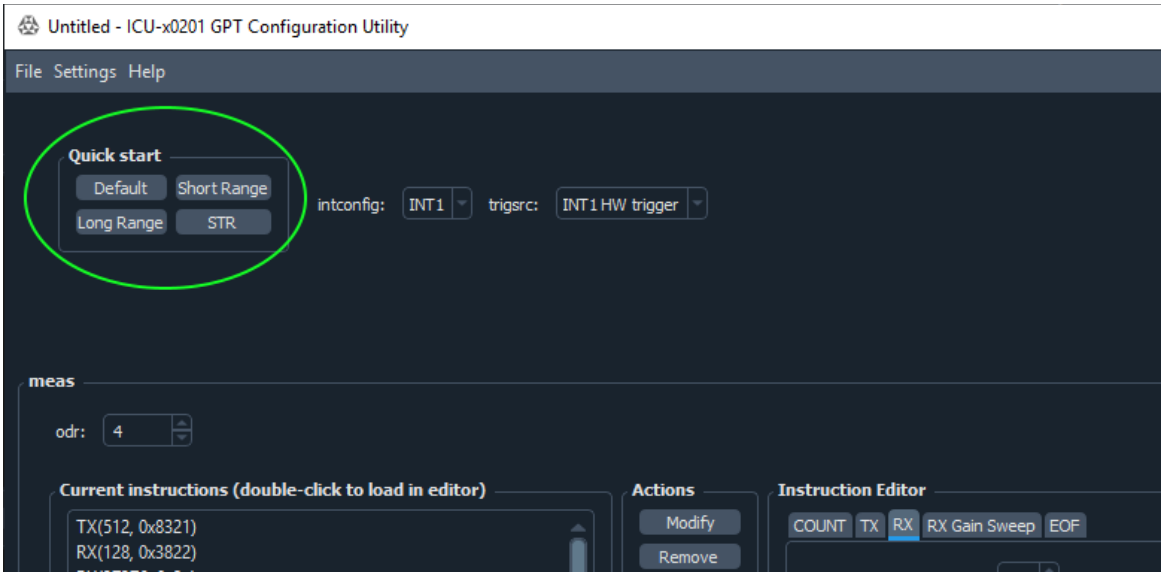

Figure 2. ICU-x0201 GPT Configuration Utility Quick Start Options

4. In the **meas** panel, modify the measurement by editing the set of sensor instructions to fit your application needs. Mouse over each button or spin box to access the online help. For more information, see the **ICU-x0201 EVK User Manual**.
5. In the **gpt algo** panel (see **Figure 3. ICU-x0201 GPT Configuration Utility gpt algo Panel**), set the number of ranges (targets) and the ringdown and static target filter sample counts. (In the Hello Chirp application, the iq_output_format setting will be over-written based on whether amplitude or I/Q data output is configured in app_config.h. So, it can be left as the default Q,I.)

   Click on the **Thresholds** button to open and modify the target detection thresholds.

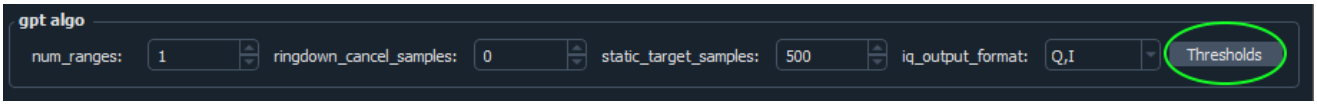

Figure 3. ICU-x0201 GPT Configuration Utility gpt algo Panel

6. When you are ready, use **File > Export to C** to save the measurement configuration in a C structure format that SonicLib can import using *ch_meas_import()* or *ch_meas_optimize()*. Note that this will also save the configuration to a JSON format file.
7. Replace the **measurement_config.c** file in the Hello Chirp example directory (**source/application/icux0201-hellochirp-example/src/ measurement_config.c**) with the new exported C file.
8. Define **IMPORT_MEASUREMENT** as 1 (non-zero) in **app_config.h**.
9. Follow the steps in Sections 1, 2 and 3 of this document to rebuild the example application, program the SmartSonic2 board, and run the example application with the new measurement settings.

7 REVISION HISTORY

Revision Date	Revision	Description
December 3, 2021	1.0	Initial Release (beta)
July 25, 2022	1.1	Updated for example full release. Added configuration descriptions.
October 25, 2022	1.2	Added logging output options.
December 2, 2022	2.0	Updated for SmartSonic2 board.
January 20, 2023	2.1	Add range of APP_DATA_MAX_SAMPLES.
January 30, 2023	2.2	Pitch-catch not supported on ICU-20201.
February 10, 2023	2.3	Clarified pitch-catch setup, added reference to AN-000254. Note: All versions previous to 2.3 are zipped and attached to ECO-020062 for future reference.

©2023 InvenSense. All rights reserved.