SmartSonic *CH201 Static Target Rejection* Example User's Guide

INTRODUCTION

This example project shows how to build and run an ultrasonic Static Target Rejection (STR) application using the Chirp CH201 sensor and the SonicLib API. STR is a special processing mode in which the sensor ignores constant reflections from static (non-moving) objects but will still report data for moving objects.

The STR example is a simple C application that demonstrates the use of motion detection to determine the presence of people and objects. The application uses Chirp SonicLib API functions to initialize, configure, and operate a CH201 ultrasonic sensor. It uses special sensor firmware and API functions which specifically support STR.

The application discovers the sensor connected to the board, programs and configures it, and then captures and displays ultrasonic measurement results through a serial connection. When a moving object is detected, a measurement result is displayed. Each measurement result includes a range (distance) to the detected object, based on the STR algorithm.

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 daughterboards that support up to four ultrasonic sensors (one mounted to the board, and others connected using flex cables). The STR example application operates with a single sensor connected to the board.

The source code for the application may be found in **source/application/smartsonic-str-example**. The **main.c** file is the main file in the application. It contains extensive comments explaining how the SonicLib interfaces are used. The **app_config.h** file contains various configuration settings that can be changed to adjust the sensor and application operation.

REQUIRED EQUIPMENT

- SmartSonic evaluation board
- Chirp sensor daughter board. If the on-board sensor will be used, the daughter board must be equipped a CH201 sensor. Optionally, an off-board CH201 sensor may be used with a flex cable.
- One Micro-USB cable

REQUIRED SOFTWARE PACKAGES

- **invn.chirpmicro.smartsonic.str-example.X.X.X.zip** (actual file name will include version number), includes:
  - STR application source files
  - Chirp SonicLib sensor API and driver files. The STR example application requires SonicLib v3.2.0 or later.
  - Sensor firmware image files
  - Board support package files for Chirp SmartSonic board
  - Atmel Studio 7 project files to build the application
- Atmel Studio 7 integrated development environment – download from Microchip.com
- Terminal emulator of your choice (for example PuTTY or TeraTerm)

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-000261
Revision 1.1
Release Date: 05/05/2022

TABLE OF CONTENTS

LIST OF FIGURES

1 INSTALLATION / PREPARATION

- Download and install the Atmel Studio 7 IDE.
- Download and install (unzip) the SmartSonic STR application to a project directory of your choice.
- Install terminal emulator.
- Connect the Chirp sensor daughterboard to the SmartSonic board. Be careful to align the white arrows.
- *Optional:* Using flat flex cables, attach additional off-board CH201 sensor(s) to the connectors on the daughterboard. If an offboard sensor is connected to the Sensor 0 connector (J6), you must set the switch on the side of the daughterboard to use the off-board sensor as Sensor 0 instead of the sensor on the daughterboard.
- Connect the SmartSonic board to a Windows PC with the USB cable. Configure the jumpers on the board as shown in the following photo with J1 in EDBG USB mode (short 3-4)
- The second USB connector (UART/UBS for Data) is not used in this application.

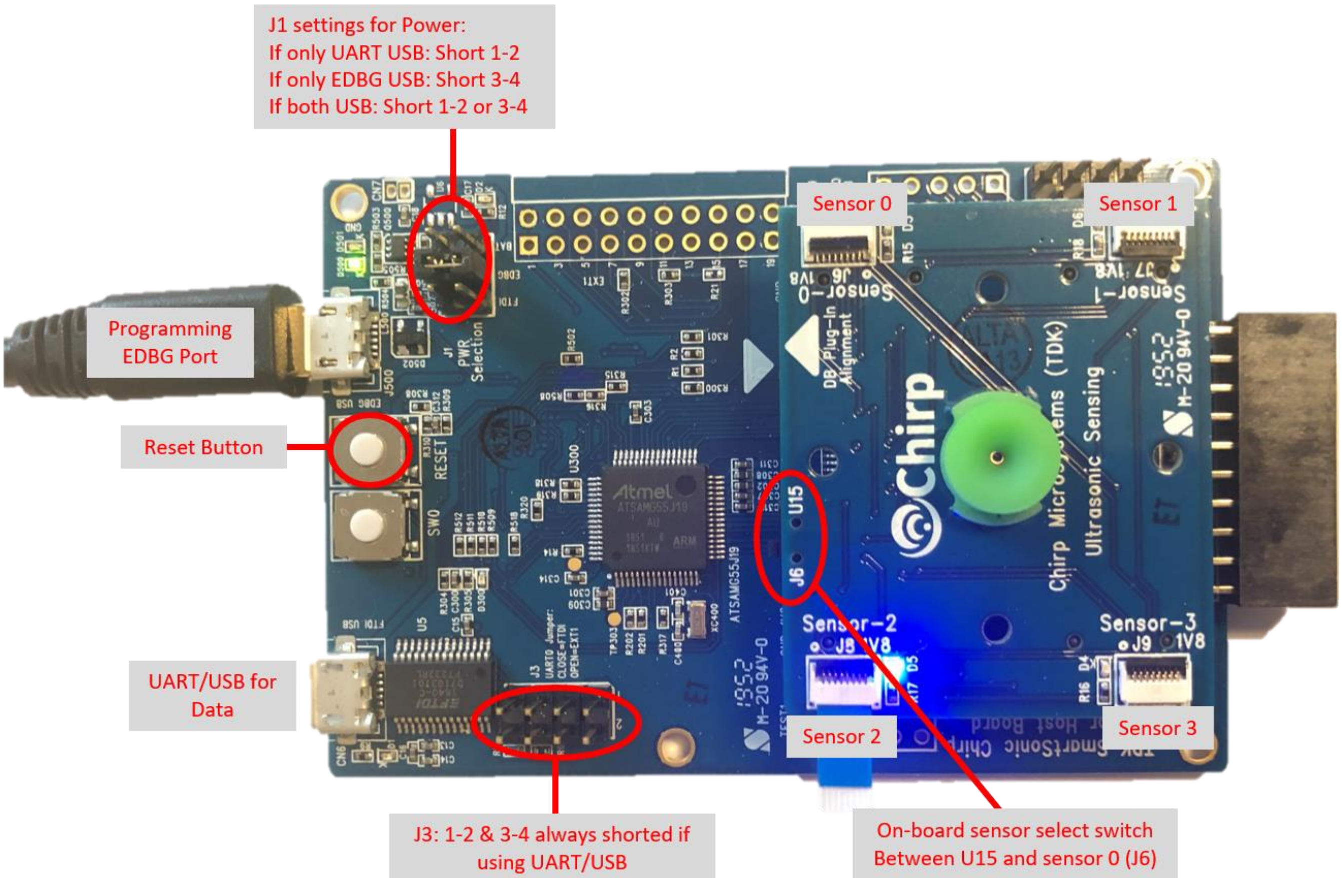

Figure 1- SmartSonic with CH201 Daughterboard

- Open Windows Device Manager, open the Ports (COM & LPT) list, and identify the COM port number assigned to the SmartSonic board. There will be one port associated with the SmartSonic board called "EDBG".
    - The EDBG port is used to connect to the on-board debugger and is programming the board as well as to display output from the program when it runs.

2 BUILDING THE STR APPLICATION

- Open Atmel Studio 7
- Open the STR project:
  - Open **File** menu
  - Select **File > Open > Project/Solution…**
  - Select the **atmelstudio/smartsonic-str-example/smartsonic-str-example.atsln** file in the project directory.
  - Click **Open**. The program should locate the project files and display the name of the project.

- The STR project files are organized in three top-level sub-directories under **source**:
  - **application** – contains **src** and **inc** directories with the STR application
    - The **application/smartsonic-str-example/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/smartsonic-str-example/inc/app_config.h** file specifies various configuration parameters affecting the sensor operation and the application behavior. These parameters and options are described later in this document.
  - **board** – contains support files for the Chirp SmartSonic board.
    - The main board support package routines, as defined in **drivers/chirpmicro/inc/chirp_bsp.h**, are implemented in the **board/HAL/src/chbsp_chirp_samg55.c** file.
    - The **board/config/chirp_board_config.h** file is required by SonicLib. This file contains definitions for the number of possible devices and I^2C buses on the board and is used for static allocation of arrays.
  - **drivers/chirpmicro** – contains the SonicLib API and driver files, sensor firmware modules, and other distribution files from Chirp.
    - The **drivers/chirpmicro/inc** directory contains header files that must be included when building applications with SonicLib. In particular, the **drivers/chirpmicro/inc/soniclib.h** file contains the key definitions for the SonicLib API.
    - The **drivers/chirpmicro/sensor_fw/ch201/ch201_gprstr_fw.c** file is the sensor firmware that contains the STR algorithm.
    - The **drivers/chirpmicro/html** directory contains HTML documentation for the SonicLIb and BSP interfaces. Open the **index.html** file to get started.
- 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 **Debug** sub-directory.

3 PROGRAMMING THE SMARTSONIC BOARD

- Setup SmartSonic board Jumper J1 as in Figure 1.
- Connect the SmartSonic board to a Windows PC with EDBG USB cable.
- In Atmel Studio 7 select **Tools > Device Programming**. The Device Programming screen will appear. 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.**

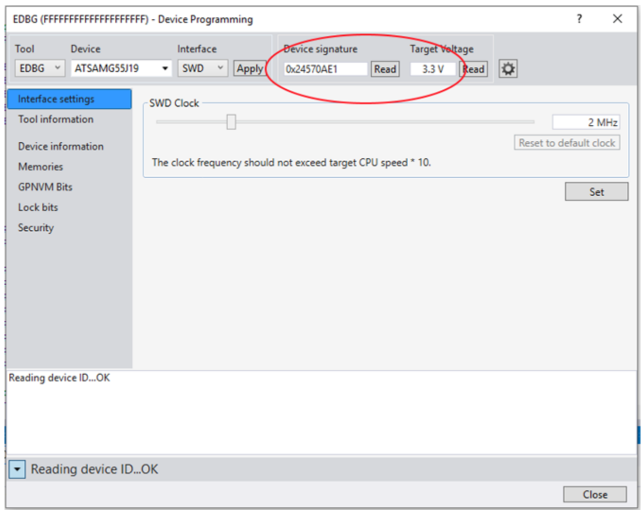

Figure 2- Device Programming Screen

- The Device Programming screen may prompt you to set the programming clock frequency. Leave the clock frequency unchanged (use the default). Select **Read** near the **Device signature** field. The device signature bytes should be read and should not generate any error messages. The Device programming menu should look as follows:

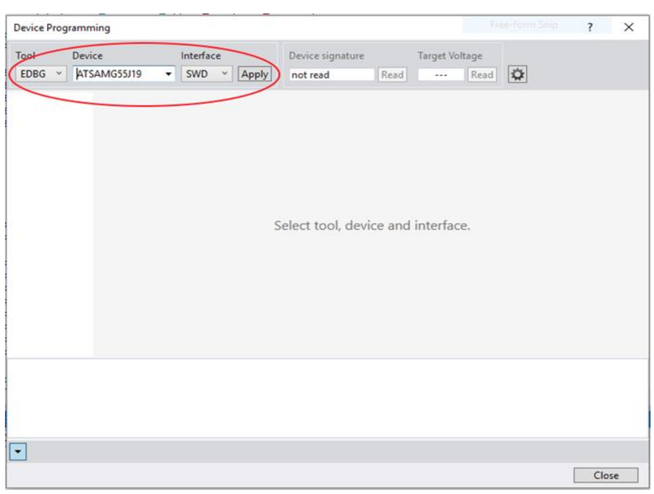

Figure 3- Device Signature & Target Voltage

- Select **Memories** on the Device Programming menu. The Device Programming menu will prompt for the name of the file to program. Navigate to the project's Debug directory and select the **smartsonic-str-example.hex** file.
    - Note: Alternately, you may use the **smartsonic-str-example.elf** file, which contains symbolic debug information. (Both files are generated when you build the application. If you are simply running the STR application, and do not need to use the Atmel Studio 7 debugging features, it does not matter which file you use.)

- Select **Program**. Your SmartSonic board is successfully programmed when the Device Programming screen displays the "OK" messages shown below on the bottom left:

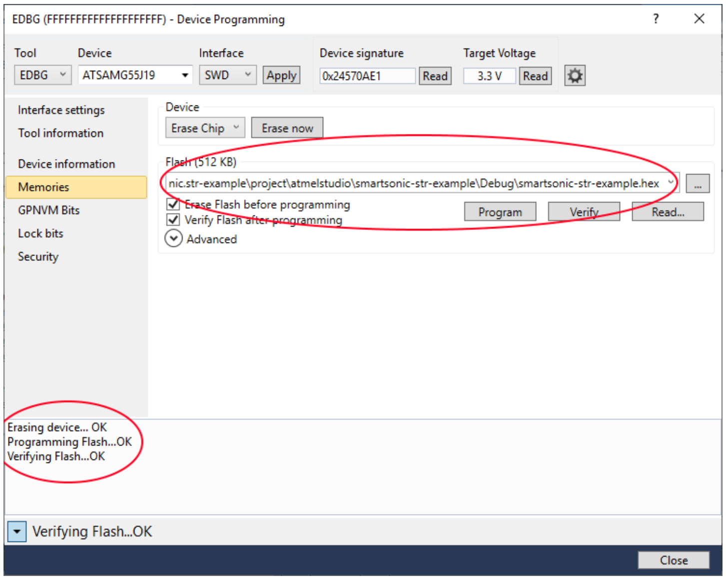

Figure 4- Successful Programming

4 RUNNING THE STR APPLICATION

- Start the terminal emulator program and open/configure the COM port assigned to the SmartSonic board "EDBG":
  - **1000000 baud**
  - **8 bits data, no parity, 1 stop bit**
  - **New-line sequence = Line Feed only (no carriage return)**
    - PuTTY: Open **Terminal** configuration. Select **"Implicit CR in every LF"**.
    - TeraTerm: Open **Setup > Terminal**. Under **"New-line"** set **"Receive"** to **"LF"**.
- Reset the SmartSonic board using the board's reset button (next to the Programming EDBG 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 (distance) data from the sensor device will then be output in a continuous loop. By default, the sensor operates in "target interrupt" mode, and it will only interrupt when a target object has been detected. A range measurement, expressed in both millimeters and the sample number within the measurement, is displayed each time the sensor interrupts.
- Along with the range indication, there is an amplitude value of the received ultrasound for the target being reported. Higher amplitude values indicate stronger reflections from the target. The internal units (LSBs) for the amplitude value correspond to those used to set the detection thresholds, as discussed below in Section 6.

COM6 - Tera Term VT

File Edit Setup Control Window Help

```
Chirp sensor 0 found
Chirp sensor 1 not found
Chirp sensor 2 not found
Chirp sensor 3 not found
Chirp sensor Idd: 168 uA


TDK InvenSense CH-201 STR Example
    Compile time:  Apr 27 2022 17:12:01
    Version: 2.7.0    SonicLib version: 3.2.0

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

Sensor  Type      Freq           B/W    RTC Cal        Firmware
0       CH201   84129 Hz        5047 Hz 2917@100ms      gprstr_str-201_v17

Configuring sensor(s)...
Sensor 0:       max_range=4000mm (246 decimated samples)        mode=FREERUN  st
atic_range=246 samples
                threshold_0 = 500       threshold_1 = 100
                rx_holdoff = 0          rx_low_gain = 60        tx_length = 30
                Target interupt filter: ANY

Initializing sample timer for 100ms interval... OK
Starting measurements
Port 0: Range: 505.5 mm   Amplitude: 243
Present ===========
Present ==========
Port 0: Range: 489.2 mm   Amplitude: 491
Present ===========
Present ==========
Present =========
Present ========
Present =======
Present ======
Present =====
Present ====
Present ===
Present ==
Present =
```

Figure 5- Typical Application Output

5 STR ALGORITHM DESCRIPTION

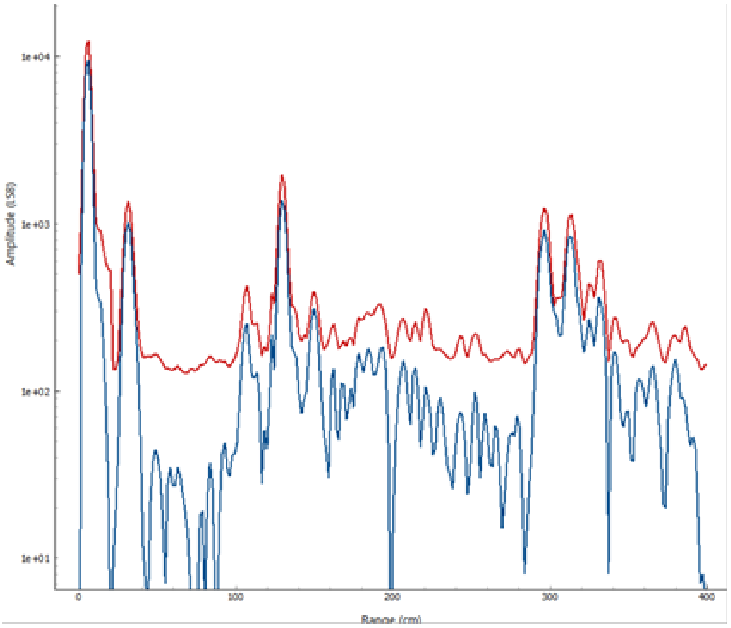

Figure 6: Received echo signal amplitude vs. range (blue) and STR threshold (red)

The on-chip static target rejection algorithm works by calculating the running average (1st order IIR) echo amplitude returned from each distance, adding a small configurable threshold to it, and using this as a detection threshold. Refer to the plot in Figure 6; the threshold is shown in red and the latest measurement shown in blue. The y-axis is the log-scale echo amplitude and the x-axis is the range in cm. As you can see, the threshold is raised above the stationary targets that were in the FOV at 0.3m, 1.2m, and 3m. When an object approaches, the running average won't respond fast enough to cover the echo of the user, and the object will be detected, and the range to it estimated.

When an object is moving in the FOV and is detected, this is termed a positive detection. When an object is moving in the FOV and is not detected, this is termed a false negative detection. When no object is moving in the FOV and the sensor detects an event, this is termed a false positive detection.

The STR algorithm has some limitations. It is good at rejecting static targets when the air around the sensor is also static. If the air is moving, or the temperature of the air changing, the static target echo will also tend to fluctuate, leading to false detections. Usually, these fluctuations will be short lived, and the CH_TGT_INT_FILTER_COUNTER feature, discussed below, can be used to reduce the false positive rate.

The customer can tune the detection by changing the parameters covered in Section 6.

6 CHANGING THE STR APPLICATION SETTINGS

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

In general, the values defined in **app_config.h** whose names begin with "CHIRP_SENSOR" are used as parameters to SonicLib API functions calls in **main.c**.

Some symbols use zero as a special value within this application only, often to indicate that a sensor setting should be left at its default value. Please refer to **main.c** to see how the SonicLib API is actually called. See *AN-000175 SonicLib Programmer's Guide* and the included SonicLIb HTML documentation for more information.

CHIRP_SENSOR_FW_INIT_FUNC

Specifies the sensor firmware initialization function. This definition is where the specific sensor firmware type for the CH201 device is selected. The default value of **ch201_gprstr_init** selects the standard CH201 GPRSTR firmware. This is appropriate for all use and should normally not be changed.

An alternate firmware build with an on-sensor watchdog timer enabled is also available by specifying **ch201_gprstr_wd_init** as the value. The sensor must complete a measurement cycle at least once every ~900 ms, otherwise the sensor will raise and hold the INT line to indicate an error condition. The watchdog-enabled version is not recommended for development or evaluation use.

CHIRP_SENSOR_MODE

Sets the operating mode of the sensor. Because this example is a single-sensor application, the two appropriate choices are CH_MODE_FREERUN (internal timing by the sensor) or CH_MODE_TRIGGERED_TX_RX (external timing by MCU board).

This example uses CH_MODE_FREERUN by default, which is typical for isolated sensors that use STR and target-interrupt modes. This combination allows the host MCU to enter a low-power state and be awakened when a moving object has been detected. The measurement interval is maintained internally by the sensor's own timer.

However, externally triggered operation can also be demonstrated in this example. External triggering is strongly recommended if multiple independent sensors will be operating close enough to each other that one sensor may "hear" another. By using accurate timers and a common measurement interval for triggering, multiple sensors can minimize their interference. If two sensors do overlap in their active measurement phases, the natural STR operation will work to filter out such interference, because the foreign sensor's ultrasound signal will appear as a constant "object."

To avoid interference between multiple sensors, all triggering devices must maintain a consistent, accurate time base. A crystal clock source is usually required.

If CH_MODE_TRIGGERED_TX_RX is specified, the periodic timer in the SmartSonic BSP will be used to trigger the sensor. There is no significant difference in the observable behavior of the application in either mode.

CHIRP_SENSOR_TARGET_INT

Controls under what conditions the sensor will interrupt following a measurement. The value of this symbol will be used as a parameter to the *ch_set_target_interrupt()* API function.

By default, the symbol is set to CH_TGT_INT_FILTER_ANY. The sensor will interrupt if any target is detected, but it will not interrupt after measurements in which no target is detected.

If the symbol is set to CH_TGT_INT_FILTER_COUNTER, the sensor will only interrupt if a certain number of recent measurements detected a target. These parameters are set by the CHIRP_SENSOR_TARGET_INT_HIST and CHIRP_SENSOR_TARGET_INT_THRESH values, below. The current measurement is always included, along with results stored in the filter history. An interrupt will be generated if CHIRP_SENSOR_TARGET_INT_THRESH target detections occur within the most recent (CHIRP_SENSOR_TARGET_INT_HIST + 1) measurements.

With proper tuning of CHIRP_SENSOR_TARGET_INT_HIST and CHIRP_SENSOR_TARGET_INT_THRESH values while CH_TGT_INT_FILTER_COUNTER is enabled, the false positive rate can be reduced significantly, while only having a small impact on the false negative rate and latency. This is because when an object is approaching the sensor, the sensor will tend to generate a series of detections within a short timespan. On the other hand, false positive events are usually somewhat uncorrelated to one another, meaning many false positive events are unlikely to happen in a short timespan.

If the symbol is set to CH_TGT_INT_FILTER_OFF, no target interrupt filtering is performed. The sensor will always interrupt after each measurement, whether or not a target object is detected.

CHIRP_SENSOR_TARGET_INT_HIST

Sets the number of measurements kept in history by the target interrupt filter when using counter filter mode. This definition is only used if CHIRP_SENSOR_TARGET_INT is set to CH_TGT_INT_FILTER_COUNTER (see above). The value of this symbol will be used as a parameter to the *ch_set_target_int_counter()* API function.

The sensor will keep the specified number of measurement results in its filter history, to compare the number of target detections (including the current result) with the threshold value (CHIRP_SENSOR_TARGET_INT_THRESH, see below).

The default value of CHIRP_SENSOR_TARGET_INT_HIST is 5, which will cause the 6 most recent measurements (5 in history plus the current result) to be included in comparisons against the threshold.

The maximum value of CHIRP_SENSOR_TARGET_INT_HIST is 15.

CHIRP_SENSOR_TARGET_INT_THRESH

Sets the threshold number of successful target detections that must occur for an interrupt to be generated when using counter filter mode. This definition is only used if CHIRP_SENSOR_TARGET_INT is set to CH_TGT_INT_FILTER_COUNTER (see above). The value of this symbol will be used as a parameter to the *ch_set_target_int_counter()* API function.

After each measurement, the sensor will compare the specified number with the total number of target detections observed during the current measurement plus the measurements in the filter history.

The default value of CHIRP_SENSOR_TARGET_INT_THRESH is 3. The maximum value is 15.

CHIRP_SENSOR_MAX_RANGE_MM

Sets the maximum detection range for the sensor, in millimeters. The default setting is 4000 mm.

Longer distances require more listening time, so the range setting controls how long each measurement will take to complete. Because the sensor samples at a fixed rate within each measurement, the range also determines how many individual samples will be part of each complete measurement.

Increasing the maximum range will increase the power consumption of the sensor roughly linearly. This is because the receiver has to listen for a longer time, and the sensor processor has to process a longer raw data from the transceiver.

Increasing the maximum range will also increase the false positive rate.

Generally, set the max range as low as possible while still meeting the application requirements.

CHIRP_SENSOR_THRESHOLD_0
CHIRP_SENSOR_THRESHOLD_1

Set the target detection threshold values.

The CH201 GPRSTR firmware provides two detection threshold values that may be set by an application. The detection threshold is the minimum amplitude required for a target detection to be reported. Threshold 0 is used at close distances (< 25 cm) and Threshold 1 applies to normal distances (> 25 cm). These may be tuned to adjust the balance of good detection vs. false positive performance in the STR application.

The threshold value is an integer (in internal units, or LSBs), typically between 50 and 1000. A value of 50 is very sensitive, and 1000 is much less sensitive. In CH201 GPRSTR firmware, the default value of the short range threshold (Threshold 0) is 500, and the default value of the normal range threshold (Threshold 1) is 100.

In this example application, zero is a special value to indicate that the default threshold should not be changed. Any non-zero value will be used as the detection threshold.

CHIRP_SENSOR_RX_HOLDOFF

Sets the receive (Rx) holdoff sample count. The Rx holdoff count is an optional setting which specifies a number of samples at the beginning of a measurement (i.e. closest reflections) that will be ignored for the purpose of detecting a target. This provides another way to “tune” the sensor’s response for specific application needs, and can be helpful if reflections from objects close to the sensor are an issue.

By default, this value is zero and no Rx holdoff is applied (all samples are included in target detection).

CHIRP_SENSOR_RX_LOW_GAIN

Specifies the number of samples within the measurement that use a lower receive gain setting than farther samples. The CH201 GPRSTR firmware uses two different gain settings for samples within the overall measurement. A lower receive gain is used for the earlier samples (closer reflections) because of their naturally higher amplitude. A second, higher gain is applied to the later samples (farther reflections) in the measurement.

This setting controls the sample offset at which the transition from lower to higher gain occurs. It does not control the actual gain levels – both the low and high gain values are fixed. The transition to higher gain naturally results in higher amplitude values for the following samples. Modifying this parameter will change the distance from the sensor where the transition occurs and can be used to adjust for the physical operating environment expected by your application.

If this value is set to zero, the sensor's default number of samples for the low-gain range will not be changed. Otherwise, it will be set to the specified number of samples.

In CH201 GPRSTR firmware, the default is 60 samples, corresponding to approximately 1 meter. Generally, if a sensor module has lower sensitivity than the standard 45 degree field of view (FOV) CH201 evaluation module provided by TDK (e.g. due to added ingress protection or wider FOV), it may be appropriate to reduce the CHIRP_SENSOR_RX_LOW_GAIN value in order to improve the sensitivity to targets in the range slightly below 1m. On the other hand, if the module has higher sensitivity (e.g. due to narrower FOV) it may be appropriate to slightly increase CHIRP_SENSOR_RX_LOW_GAIN in order to ensure that the receive data does not saturate in the presence of a large target in the range slightly above 1m.

CHIRP_SENSOR_TX_LENGTH

Specifies the duration of the ultrasound pulse that is generated by the sensor at the start of a measurement. The units are the number of internal PMUT (transducer) cycles per pulse. In general, a pulse using a longer transmit length will have a higher amplitude when the echo is received.

However, there is a good reason not to increase the TX length too much. The mechanism that creates a lot of false positives is interference between stationary echoes that exist in the environment. In order for the echoes to interfere, they must overlap in time/range, meaning if e.g. you have static object A at 1m and static object B at 1.1m range from the sensor, and the pulse is e.g. 100 cycles/200mm in duration (1 cycle = 2mm in range), then the echoes from objects A and B will overlap, and have an interference pattern which is a function of the range to the two targets. Small changes in the propagation time to object A which doesn't affect the propagation time to B in exactly the same way will cause the phase of the echo returning from A to change relative to that of B, for example, when the air around the sensor is turbulent or changing temperature, the interference pattern will start changing rapidly (especially when the echoes are about the same amplitude) and the STR algorithm is unfortunately sensitive to that.

If this value is set to zero, the sensor's default transmit length will not be changed.

For CH201 GPRSTR firmware, the default is 30 cycles, which means the TX pulse will have a length of about 60mm in the range axis.

CHIRP_SENSOR_STR_RANGE

Specifies the static target rejection sample range, in samples. The STR range is the count of samples in each measurement that will have the static target filtering and will therefore have stationary objects "ignored." Beyond this range, stationary targets will be detected by the sensor. The count of samples always begins at the start of the measurement data (i.e. reflections closest to the sensor).

Default = 0 (STR is enabled for the entire measurement). In this example application, zero is a special value used to indicate that the sensor's entire measurement range will have STR enabled. In the *main()* routine, the application will replace this value with the correct sample count for the entire range.

Minimum Allowed STR Range

Chirp sensors naturally exhibit "ringdown" in the sensor, which causes a small number of samples at the beginning of a measurement to have relatively fixed, high amplitude values. The CH201 STR firmware does not have separate ringdown cancellation filtering, so it relies on the STR behavior to "ignore" this ringdown signal. Therefore, there is a minimum STR range that should be set – the ringdown signal may be incorrectly reported as a target if the STR range is set too low.

The time-domain bandwidth (***BW***) of the sensor is used to estimate the duration of the ringdown and therefore the minimum static range that will give good results. If we assume 3 time constants settling for TX and 6 tau settling for the ringdown, then

$$t_{tx} = \frac{3}{\pi BW}$$

$$t_{ringdown} = \frac{6}{\pi BW}$$

$$StaticRange_{min} = \frac{c}{2} * \left(t_{tx} + t_{ringdown}\right) = \frac{343\,\frac{m}{s}}{2}\frac{9}{\pi BW}$$

Example: If the bandwidth of a CH201 is 3kHz,

$StaticRange_{min}$= *343 / 2 · 9 / (3.14 · 3000)* = 0.16 m = 16 cm

The bandwidth of the sensor can be obtained by using the *ch_get_bandwidth()* API function in SonicLib.

The SonicLib *ch_set_static_range()* and *ch_set_config()* API functions will automatically increase the STR range to the minimum recommended number of samples if too low a value is specified. For CH201 GPRSTR firmware, the minimum STR range is 14 samples (approx 20 cm). Non-moving objects closer to the sensor than this minimum will not be reported.

SHOW_PRESENCE_INDICATORS

Controls whether "Present" status indicators are displayed in the application serial output. If this symbol is defined, a series of "Present" messages with a graphic indicator will be displayed after each successful target detection.

Comment out this line to disable the Present indicator messages. Default = defined (enabled).

PRESENCE_HOLD_SECONDS

Specifies a minimum time for a change in presence to be reported, in seconds. Default = 1 s.

This value defines the Presence Hold Time and controls how long the "Present" status indicators will be displayed after the last measurement which successfully detected an object (if SHOW_PRESENCE_INDICATORS is also defined).

NUM_RANGE_HISTORY_VALUES

Specifies the number of history range values stored in the STR application. A median filter is then used for smoothing out the STR range output. In most the cases, the default value (7) is appropriate.

MEASUREMENT_INTERVAL_MS

Specifies how often a new measurement cycle will be performed, in milliseconds. Default = 100 ms (10Hz sample rate).

This value is used to program the sensor's internal timer if in CH_MODE_FREERUN (default). It is used to program a periodic hardware timer on the SmartSonic board MCU to trigger the sensor if in CH_MODE_TRIGGERED_TX_RX.

IQ_DATA_MAX_NUM_SAMPLES

Controls the amount of space used to hold I/Q data values from the sensor, as part of the example **chirp_data_t** structure. By default, this value is set to the maximum number of samples used by the CH201 GPRSTR firmware (290 samples).

To reduce memory requirements, this value can be lowered if your application does not operate the sensor at the maximum possible range setting (and will therefore not require the maximum number of samples).

OUTPUT_AMP_DATA_CSV

Specifies that the I/Q data read from the device will be converted to amplitude values for each sample, then output in ascii CSV (comma separated value) format. If this symbol is defined, the amplitude values for each sample in the measurement will be output on the same line that contains the regular target range information, separated by commas.

Output in this format allows easy import into spreadsheets or other analysis tools. Often, the amplitude values are plotted in what is called an A-scan Chart. This can provide very useful insights into the ultrasound environment and help tune the sensor performance.

The usual “Present” presence indicators are not output when this option is used.

Default = not defined (disabled).

OUTPUT_IQ_DATA_CSV

Specifies that the raw I/Q values read from the device will be output in ascii CSV (comma separated value) format. If this symbol is defined, each I/Q pair will be output on a separate line (Q value followed by I) with a comma separator.

The usual “Present” presence indicators are not output when this option is used.

Default = not defined (disabled).

READ_IQ_BLOCKING

READ_IQ_NONBLOCKING

These definitions control whether the raw I/Q sample data for the measurement will be read in blocking (synchronous) mode or non-blocking (asynchronous) mode. If READ_IQ_BLOCKING is defined, the I/Q data will be read in blocking mode. If READ_IQ_NONBLOCKING is defined, the data will be read in non-blocking mode. Only one of these symbols should be defined.

These settings only apply if OUTPUT_AMP_DATA_CSV or OUTPUT_IQ_DATA_CSV is defined.

Default = READ_IQ_BLOCKING defined, READ_IQ_NONBLOCKING not defined (blocking mode enabled).

7 TUNING THE POSITIVE DETECTION RATE AND THE FALSE POSITIVE RATE

This section briefly describes the tuning parameters which will affect the positive detection (PD) rate and the false positive (FP) rate. For details on individual tuning parameters, see Section 6.

Parameter	**Effect of *increasing* parameter on Positive Detection (PD) and False Positive (FP) rates**	**Description**
CHIRP_SENSOR_TARGET_INT_HIST*	PD↑ FP↑	More history means more chances of detection
CHIRP_SENSOR_TARGET_INT_THRESH*	PD↓ FP↓	Larger count of STR events required for interrupt means less chances of detection
CHIRP_SENSOR_MAX_RANGE_MM	PD↗ FP↑	Longer listening time will detect more distant targets (with diminishing returns) and raise FP rate
CHIRP_SENSOR_THRESHOLD_0 CHIRP_SENSOR_THRESHOLD_1	PD↓ FP↓	Larger threshold for STR detection means smaller changes in the echo trace may not be detected
CHIRP_SENSOR_RX_HOLDOFF	PD↓ FP↓	Ignoring the start of the echo trace will reduce the PD and FP from this portion of the trace
CHIRP_SENSOR_RX_LOW_GAIN	PD↓ FP↓	Increasing low gain range will tend to decrease PD & FP from the end of the low gain range
CHIRP_SENSOR_TX_LENGTH	PD↗ FP↑	Diminished returns on PD from increasing TX length. Increasing TX length may increase FP rate from static objects in environment with almost the same range to the sensor.

*when CHIRP_SENSOR_TARGET_INT is set to CH_TGT_INT_FILTER_COUNTER

8 REVISION HISTORY

Revision Date	Revision	Description
3/5/2021	1.0	Initial version.
5/5/2022	1.1	Updated with new options in Section 6. Added section 5 and 7.