D3 Engineering

Define | Design | Deploy

Phase 2 Radar Void Detection Firmware Design Description

Document Number: 900-325

Document Author: Andrew Redman

Customer: MTi Group

Project: 336-002

Date: 2023/03/21

Revision: 0.7

150 Lucius Gordon Dr,
West Henrietta, NY 14586
Phone: 585.429.1550
Fax: 585.429.1551
D3Engineering.com

Table of Contents

Table of Figures

150 Lucius Gordon Dr,
West Henrietta, NY 14586
Phone: 585.429.1550
Fax: 585.429.1551
D3Engineering.com

Table of Tables

Introduction

1.1 Overview

This document details the firmware design for the radar void detection sensor package. The main goal for this package is to measure the depth of a mining borehole. The package will be held by hand at the top of the borehole and will not be lowered into the hole.

1.2 Applicable Documents

Scope BLASTSCOUT Void.pdf

900-217 Radar Void Detection System Specification-Rev3_0.pdf

1.3 References

Scope Out-Of-Hole BLASTSCOUT.pdf

2.0 Design Details

The firmware is based on TI MMWAVE SDK version 3.5.0 out of the box demo. The firmware has the following features:

- The firmware is triggered by a wake-up message from the host through CAN.
- Uses a single chirp profile with the desired radar performance for a downward facing radar sensor.
- The sensor transmits range profile and detected points to the host via CAN.
- The primary communication interface is CAN 2.0. The physical layer of CAN 2.0 is a 5V differential bus with 2x 120 ohm termination. One set of terminating resistors is supplied by the radar sensor. The opposite end of the bus must be terminated externally.
- A 500 kbps data rate is used which should offer good signal integrity on a CAN bus up to 100 m in length.
- During testing, the following CAN settings were used to receive data using a PEAK PCAN adapter and PCAN-View.
    - Bitrate: 500 kbit/s
    - Sample Point: 75%
    - Time Quanta: 8
    - TSEG 1: 5
    - TSEG 2: 2
    - Bit Rate Prescaler: 5
    - Sync Jump Width: 1
    - Clock Frequency: 20 Mhz

150 Lucius Gordon Dr,
West Henrietta, NY 14586
Phone: 585.429.1550
Fax: 585.429.1551
D3Engineering.com

- The following CAN settings are defined in the radar firmware.
  - Bitrate: 500 kbit/s
  - Bitrate Prescaler: 5
  - nomPropSeg: 6
  - Pseg1 5
  - Pseg2 4
  - Sync Jump Width: 1

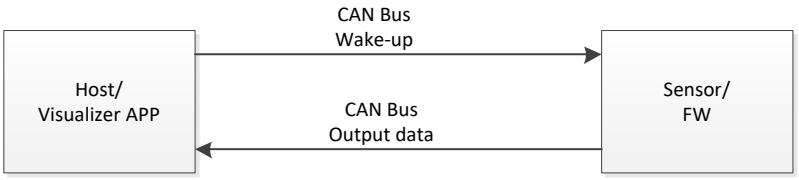


Figure 1 Interface between Firmware and Host

2.1 Chirp Configuration

2.1.1 Overview

The downward facing sensor uses a chirp profile with a maximum unambiguous range of 51.21 m. The firmware transmits all 512 available range bins. This gives the range bin a size of 0.1 m and a range resolution of 0.111 m.

Chirp configuration settings are described in the User Guide included with mmWave SDK version 3.5.0.

The chirp configuration is defined at compile time and cannot be modified after the sensor is deployed. The chirp configuration is defined in ***/18xx_mmwave_sdk/src/mss/cli.c***. An array of string values containing the chirp settings is stored in the array ***hardCodedChirpConfiguration[]***. This array is assigned different values depending on the variables defined at compile time. This allows for different sensor configurations (down-hole, side radar) to be stored in the same code base. Compiler settings are used to select which configuration is included in the build. These settings can either be changed using define statements included in the source code or by updating using the predefined symbols in code composer[1].

The final command of a hardcoded configuration is ***!!!END_OF_HARD_CODED_COMMANDS***. The leading `***!***` symbol is used to indicate to the final command.

The chirp configuration is sent to the RF front end in the ***CLI_task()*** function. For UART control, this function takes commands in the form of a string from the UART input using the ***UART_read()*** function. To quickly enable CAN commands in this codebase, an array of strings is stored in ***activeChirpConfiguration[]***. When the sensorStart CAN command is received, commands are copied from ***hardCodedChirpConfiguration[]*** to ***activeChirpConfiguration[]*** in the ***can_sensor_start()*** function and sent to the RF front end.

2.1.2 Specific chirp parameters

An overview of all chirp configuration parameters is available in the TI mmWave SDK User Guide. Full details are available in the mmWave Radar Interface Control Document (ICD) located in the mmWave DFP.

RX Gain: Receiver gain is set in the argument of ***profileCfg***. Valid values are even numbers from 24-52. These represent the RX gain in dB.

Frame Period: The frame period is defined in ***frameCfg***. The fifth argument represents the frame period in ms. This value can accept floating point values from 0.3 to 1342000 ms. Valid values may cause the radar to stop working if enough time isn't allotted for frame processing or data transfer.

2.2 Existing data format

The sensor outputs several CAN data messages to transmit data for every radar frame. The sensor sends a header message, a fixed number of range profile messages and a variable number of detected object messages per a radar frame, as shown below in Figure 2.

[1] Predefined symbols can be viewed and modified by navigating to **Project >> Show Build Settings… >> Predefined Symbols**.

150 Lucius Gordon Dr,
West Henrietta, NY 14586
Phone: 585.429.1550
Fax: 585.429.1551
D3Engineering.com

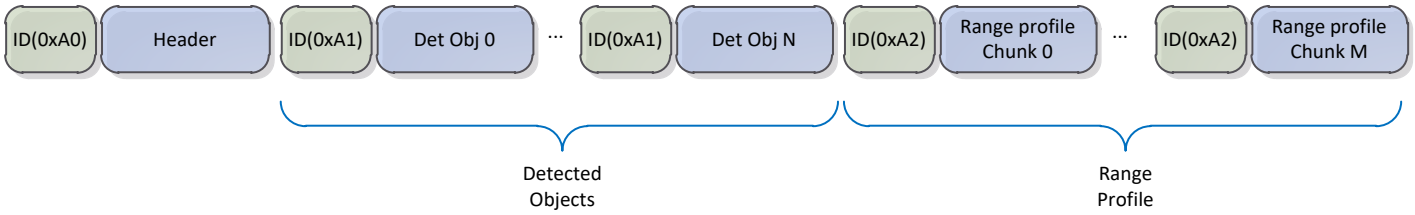


Figure 2 CAN Message Format per radar Frame

2.2.1 Header Message

Message identifier (arbitration Id): **down sensor 0xA0**

The header message data has a fixed size of 8 bytes. The table below shows the header data structure.

Name	Size (bytes)	Type (C)	Description
totalPacketLen	4	uint32_t	Total packet length including header in bytes
frameNumber	4	uint32_t	Frame number

2.2.2 Range Profile Message

Message identifier (arbitration Id): **down sensor 0xA2**

The sensor sends a fixed number of range bins for each radar frame, the long-range chirp configuration sends 512 bins. Each range bin is 2 bytes (uint16_t). The sensor transmits the range profile bins in chunks of 4 bins (8 bytes). Each chunk is transmitted as a separate CAN data message with same the same arbitration id (0xA2).

The range profile is an array of profile points at 0th Doppler (stationary objects). It provides the intensity of the signal received by the radar reflected from stationary objects. The values represent the sum of log2 magnitude of the received antennas expressed in Q8 format. The values will be converted to dB scale before visualization.

$$\text{Intensity (Linear)} = FFT_{SCALE} * 2^{\frac{R}{256}*\left(\frac{2^{ceiling(\log_2 nTx*nRx)}}{nTx*nRx}\right)}$$

Where $FFT_{SCALE} = \frac{32}{Range\ Bins}$, R is the range profile received from the sensor, nTx is the number of transmit antennas and nRx is the number of receiver antennas. The FFT_{SCALE} is applied when range FFT is performed to avoid overflow. The value is divided by 256 to convert the fixed point (Q8) representation to a floating-point value. The range value is multiplied by a factor of $\frac{2^{ceiling(\log_2 nTx*nRx)}}{nTx*nRx}$ to compensate for the scaling applied when the intensity value from each virtual antenna are summed together. The intensity value is converted to dB before it is plotted on the visualizer.

$$\text{dB} = 20 * \log_{10}(FFT_{SCALE} * 2^{\frac{R}{256}*\left(\frac{2^{ceiling(\log_2 nTx*nRx)}}{nTx*nRx}\right)}$$

$$dB = 20 * \log_{10} FFT_{SCALE} + 20 * \log_{10} 2 * \frac{R}{256} * \left(\frac{2^{ceiling(\log_2 nTx*nRx)}}{nTx * nRx}\right)$$

150 Lucius Gordon Dr,
West Henrietta, NY 14586
Phone: 585.429.1550
Fax: 585.429.1551
D3Engineering.com

2.2.3 Detected Object Message

Message identifier (arbitration Id): **down sensor 0xA1**

A detected object has a size of 8 bytes. The table below shows the data structure of an object (point). The sensor sends a CAN message for each detected point. There can be a maximum of 500 detected points per frame.

A detected object frame contains the range (in meters) and SNR (in dB) values as shown the table below. The range is mapped to a range bin on the range profile plot. The SNR represents the signal intensity at the detected peak in the range profile divided by the average noise power from the neighboring bins which is an output of the peak detection algorithm.

Title	Format	Size (bytes)	Value
range	float	4	Range
snr	float	4	Intensity

2.2.4 Status Message

Message identifier (arbitration Id): **down sensor 0xA3**

The sensor sends a stock message in response to the sensorStatus command to indicate the state it is in.

Title	Format	Size (bytes)	Value
Status	Int	4	0x1: Sensor Ready 0x2: Sensor Chirping 0x3: Sensor Stopped

2.3 CAN Output Interface

CAN messages can be scheduled by calling the ***Can_Transmit_Schedule()*** function. This function takes 3 arguments, Message ID, Message Data as a pointer, the size of data to send. This function is declared in ***mmw_can.c*** and accepts data of any length which will be broken into 8 byte packets before transmission. Data is broken into an appropriate number of 8 byte packets which are all sent with the given message ID.

2.4 Commands

A CAN message is composed of a 11-bit message (arbitration) identifier followed by 0 to 8 bytes of data. The message identifier will be used to address a specific sensor on the bus. CAN commands must be addressed to the CAN ID of the specific sensor it affects. A message identifier of 0x80 will be used to address the down sensor. This can be configured by updating the ***CAN_START_MESSAGE_ID*** in the project's predefined symbols.

A CAN command will use 8 bytes in the data field to transfer the command identifier along with its arguments. The first byte in the CAN message data is used for the command identifier. The remaining 7 bytes will be used to pass arguments for the command. The following commands will be supported.

Table 1. Supported Commands

Command Name	Command Identifier	Arguments	Action
sensorStart	0x00	None	Begin chirping and posting data to CAN bus
sensorStop	0x01	None	Stop chirping
calibDcRangeSig	0x02	None	Start sensor calibration based on chirp profile

150 Lucius Gordon Dr,
West Henrietta, NY 14586
Phone: 585.429.1550
Fax: 585.429.1551
D3Engineering.com

setPower	0x03	TX_Power_BackoffCode	Adjust power settings in chirp configuration based on new backoff code settings (change requires sensorStart)
sensorStatus	0x04	None	Query the sensor to see if it is ready
setDetectionThreshold	0x05	Detection threshold scale in dB	Sets new detection threshold for range CFAR. The default value is 15. This command can only be used after the sensor is started. The value passed in to the CAN command must be an integer, however the firmware does allow floating point values to be configured using the compiled chirp configuration or the serial port.
enableSpreadSpec	0x06	Spread spectrum enable flag 0 – disable 1 – enable	Enables/disables the PMIC spread spectrum mode. This command can only be used after the sensor is started.
selectChirpProfile	0x07	The chirp profile number 0 - 50m maximum range 1 - 100m maximum range	This command is used to choose a chirp profile. The sensor has to be restarted for the change to take effect.

3.0 Functional Algorithm

3.1 Processing in the MSS

The system execution flow in MSS and DSS is given below in Figure 3. The functional algorithm can be implemented in the MSS before data is output over the CAN bus. This is indicated as '*MSS Algorithm*' in flow diagram. Data is output from the MSS in the function ***MmwDemo_transmitProcessedOutput***. This function outputs both the CAN data and UART data and is the last opportunity to make changes to the data before it is transmitted. The data is generated in the DSS and transferred to the MSS through shared memory (HSRAM). Examples of multiple data types are shown being accessed in this function include ***result->detMatrix.data*** which is the range profile and ***result->objOut*** which is an array of detected points.

The ***CLI_write()*** function is defined in ***clic.c*** and prints a string argument to the serial UART. This can be useful for debugging during the software development period.

The ***System_printf()*** function is a function defined in the xdc tools package which prints an output to the debugger. These print statements will not be visible unless a JTAG debugger is connected to the unit.

3.2 Processing in the DSS

The functional algorithm can also be implemented in the DSS. More processing power is available in the DSS and should be used for more complex and cycle-intensive algorithms. Data computed in the DSS will need to be saved in shared memory which is also accessible to the MSS.

The code run on the DSS for object detection is stored in ***objectdetection.c***. New functional algorithm can be added before the result is copied to the shared memory which is shown as 'DSS Algorithm' in the flow diagram. The ***DPC_ObjectDetection_ExecuteResult*** structure specifies output data that is produced in the DSS and shared to the MSS. The result data structure must be extended to include new data types generated by the new algorithm. The definition of the ***DPC_ObjectDetection_ExecuteResult*** data structure can be found in the mmWave SDK, *C:/ti/mmwave_sdk_<version>/datapath/dpc/objectdetection/objdethwa/objectdetection.h.*

150 Lucius Gordon Dr,
West Henrietta, NY 14586
Phone: 585.429.1550
Fax: 585.429.1551
D3Engineering.com

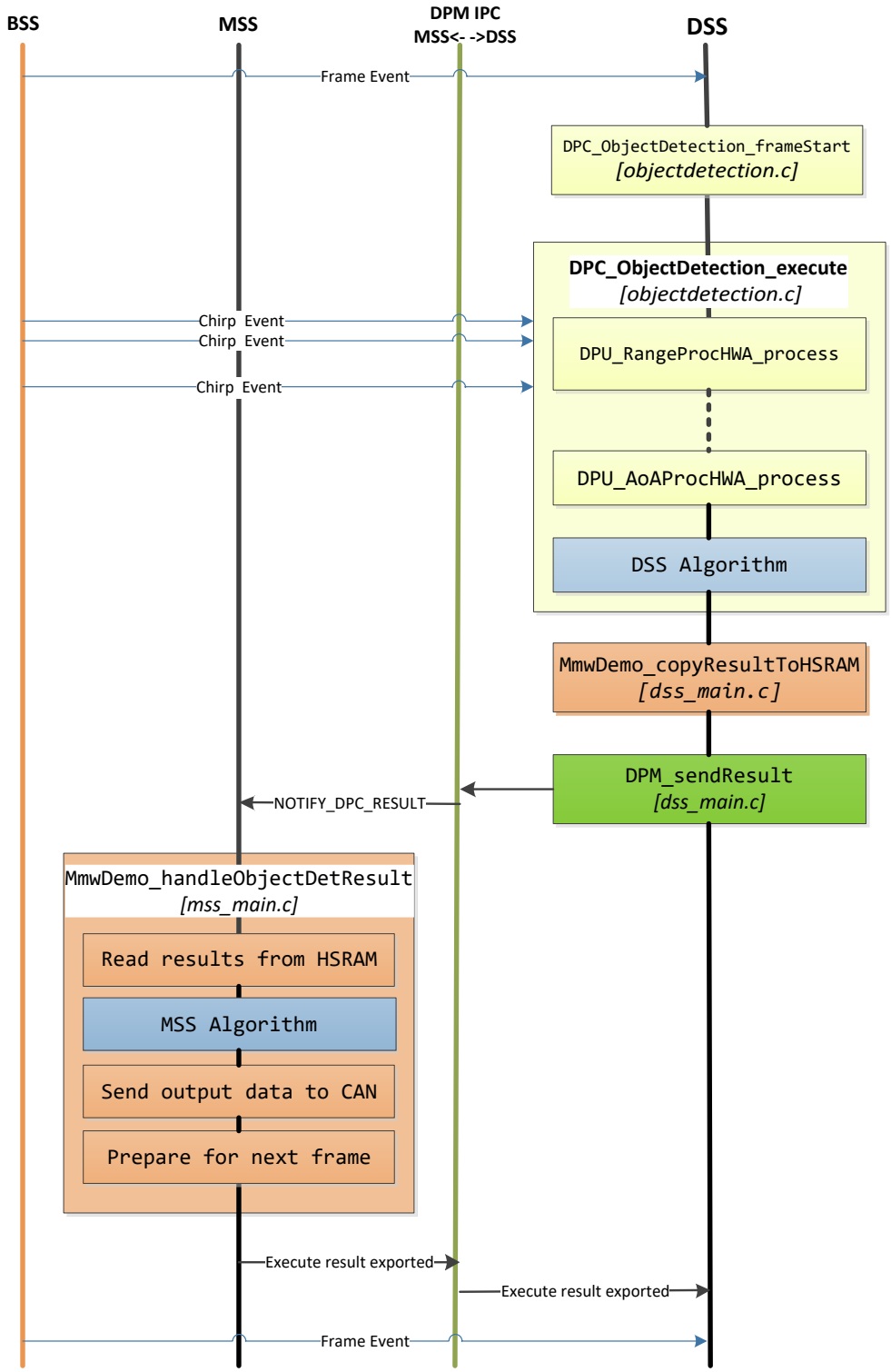


Figure 3 System Execution Flow

150 Lucius Gordon Dr,
West Henrietta, NY 14586
Phone: 585.429.1550
Fax: 585.429.1551
D3Engineering.com

4.0 Software Environment

4.1 Dependencies

The following dependencies are required to build the firmware for the down-hole sensor.

4.1.1 Code Composer Studio

Code Composer Studio is a free, proprietary IDE developed by Texas Instruments based on Eclipse. It is available from Texas Instruments website[2]. This package is offered as a web installer or offline installer for Windows, Linux, and Mac under "Download Options". The offline installer is much more reliable. During installation the user is prompted to select which packages are desired. Only the mmWave packages are required to develop on the AWR1843AOP chip and others in the mmWave family.

4.1.2 mmWave SDK

The mmWave SDK includes libraries required to develop on the mmWave family of chips. This is available from the TI website[3]. This code was developed using mmWave SDK version 03.05.00.04. This can be downloaded as a zip file and should be installed in the default location (C:/ti).

4.1.3 D3 Software Release

Code for both the R4F arm core and C6000 DSP core is included in the D3 project release. The project includes two projectspec files which describe the project structure to Code Composer. These files can be unzipped and stored in any location accessible to Code Composer. It will be imported into the Code Composer workspace location, usually ***C:/users/<username>/Code Composer***, in a later step,

4.2 Project Import

The D3 codebase can be imported to Code Composer using the following steps

1. Launch Code Composer
2. Create a new workspace
3. Selecting ***Project >> Import CCS Project…***

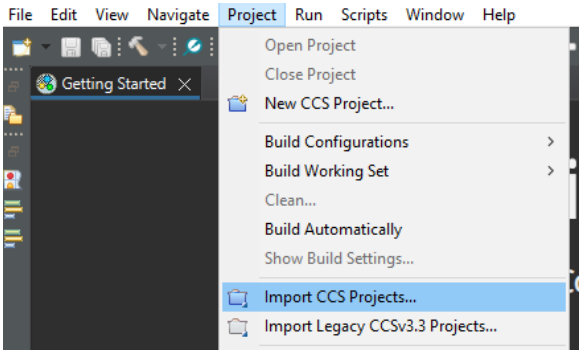


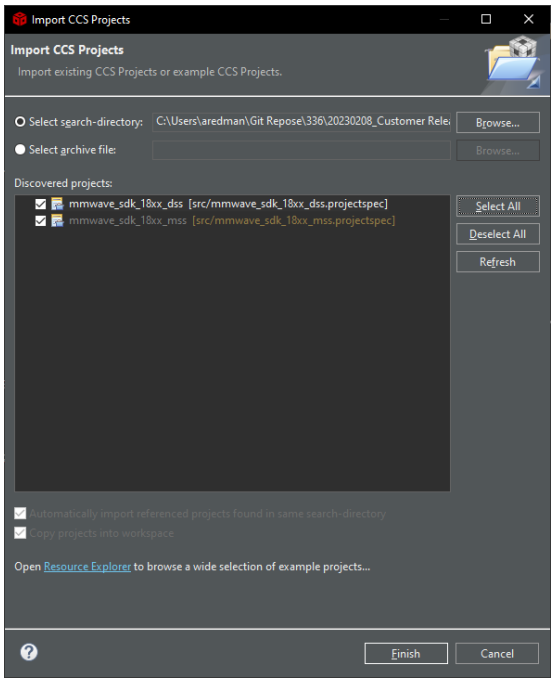
Figure 4. Code Composer Project Import

4. Select ***Browse*** and navigate to the folder containing the D3 firmware release. Code Composer will search the directory tree for the projectspec files.
5. Select all projects and click finish

[2] https://www.ti.com/tool/CCSTUDIO

[3] https://www.ti.com/tool/download/MMWAVE-SDK

150 Lucius Gordon Dr,
West Henrietta, NY 14586
Phone: 585.429.1550
Fax: 585.429.1551
D3Engineering.com



4.3 Build Process

Projects can be built by selecting them from the project explorer menu and either clicking the ***Build*** (hammer) button or right-clicking on the project name (***mmWave_sdk_18xx_dss*** or ***mmWave_sdk_18xx_mss***) and selecting ***Build Project*** or ***Rebuild Project***. The DSS project must be built before the MSS project. After both projects are built, the binary file is saved in the ***aop_down*** folder under the project folder.

5.0 Programming

The radar sensor can be programmed over the UART header by using the XMODEM serial interface through TeraTerm[4]. The programing interface is shown in Figure 5.

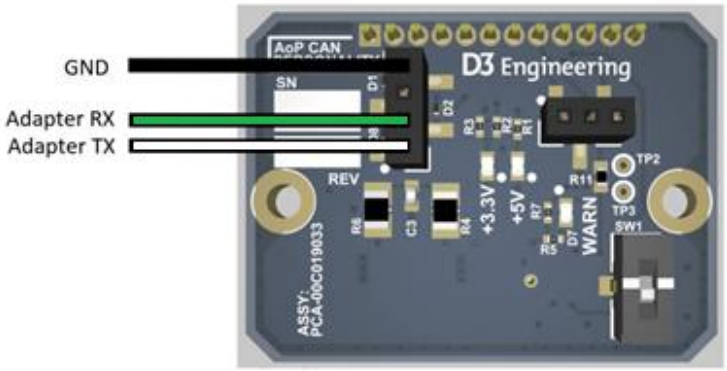


Figure 5. AOPC Programming Interface

Using an 3.3 V FTDI adapter, connect the serial interface to a host PC using USB. Connect using the following parameters.

[4] https://ttssh2.osdn.jp/index.html.en

150 Lucius Gordon Dr,
West Henrietta, NY 14586
Phone: 585.429.1550
Fax: 585.429.1551
D3Engineering.com

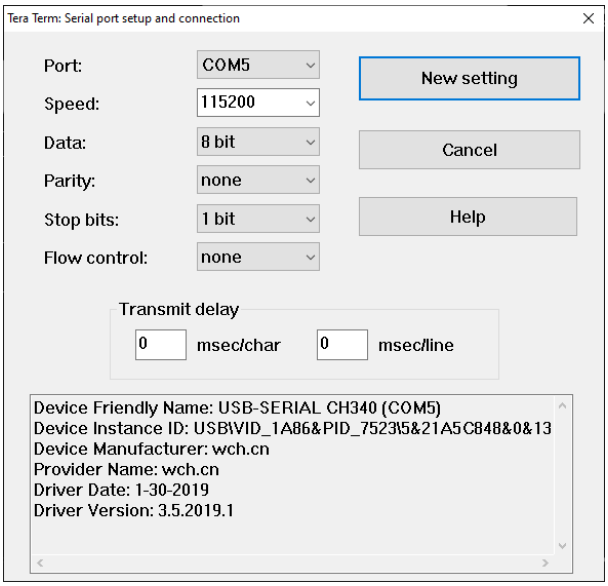


Figure 6. TeraTerm Serial Port Settings

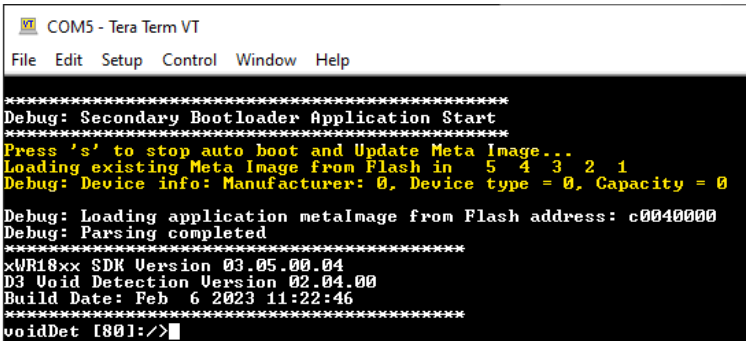


Figure 7. Serial Output for board booting successfully

When connecting to the sensor the secondary bootloader will offer a 5 second waiting period before loading the flashed program from memory. To reflash the board this process must be stopped by sending the '*s*' character in TeraTerm. If the board is properly interrupted there will be a brief period where the flash memory is erased followed by the character '*C*' output while waiting for new firmware.

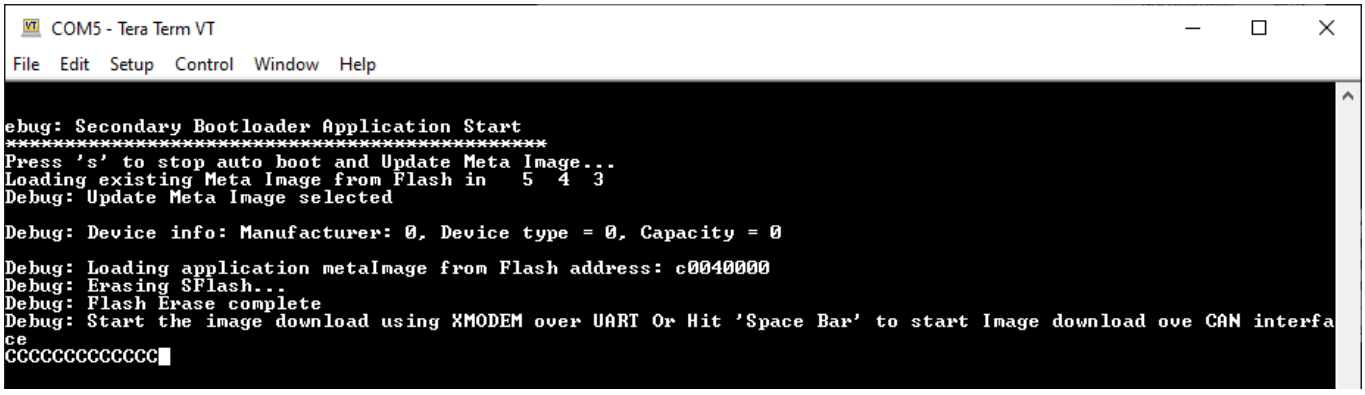


Figure 8. Serial output for board waiting for firmware update

150 Lucius Gordon Dr,
West Henrietta, NY 14586
Phone: 585.429.1550
Fax: 585.429.1551
D3Engineering.com

To send new firmware select ***File >> Transfer >> XMODEM >> Send*** from the TeraTerm menu as shown in Figure 9. When the firmware transfer has started, the file transfer progress bar will be shown.

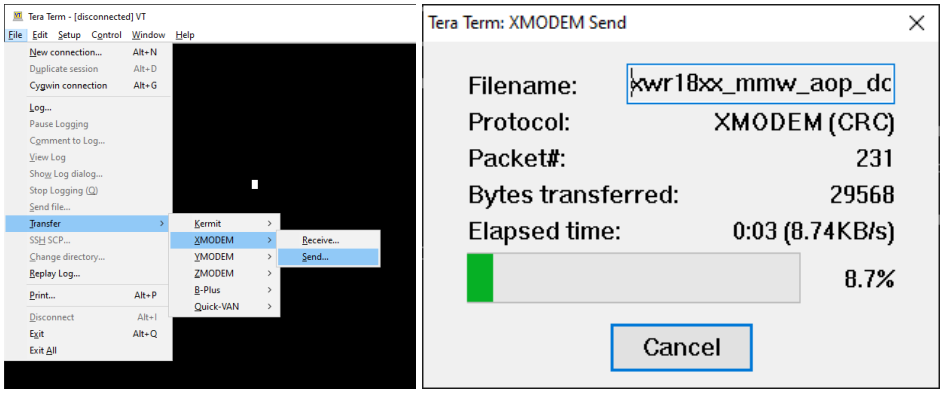


Figure 9. TeraTerm XMODEM File Transfer Menu (left) and Progress Bar (right)

Once the firmware has been fully loaded, the sensor can be restarted and the new firmware will run after the secondary bootloader waiting period has expired.

6.0 Recommendations

6.1 Wet Hole

During field testing, MTi Group observed that multiple reflections originate from the bottom of holes with water at the bottom. Field testing suggests that this is a reliable method of detecting water at the bottom of the hole. The PMIC in use is creating a spur which is detected in the data. It was also noticed in testing that a second spur is detected in holes with water in the bottom. Because the spur is caused by unwanted and uncharacterized interference from the PMIC and is likely to be removable in future hardware releases, it is not a reliable method for determining if a hole has water at the bottom.

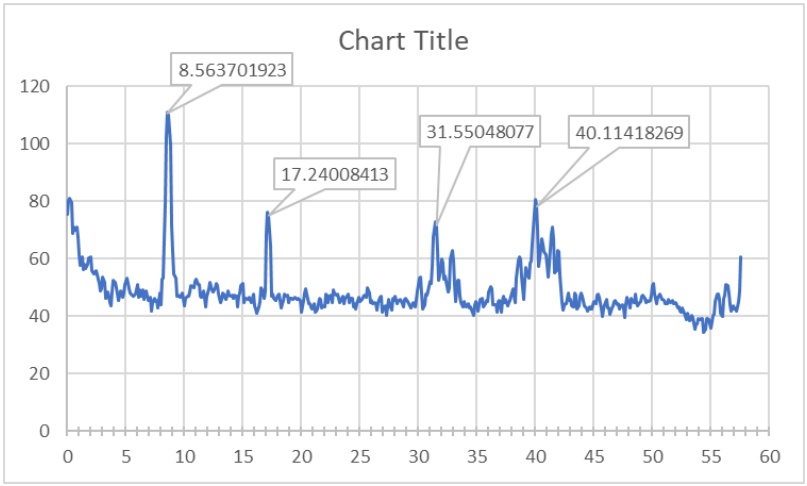


Figure 10. Wet Hole Data (Annotated)

There are some key features that we think are going to be useful in developing an algorithm for classifying wet holes. In holes that have water in the bottom, the primary reflection is much stronger when compared to holes without water in the bottom. This is likely caused by level water offering a very good reflector. When this signal returns to the top of the hole it is often reflected down into the hole a second time and the returning signal is detected by the sensor as a peak in the range FFT with twice the path-length distance.

150 Lucius Gordon Dr,
West Henrietta, NY 14586
Phone: 585.429.1550
Fax: 585.429.1551
D3Engineering.com

When measuring the second path length and searching for a peak at twice the distance, it is important to make sure the DC distance offset is accounted for. There is a sensor-specific DC offset which should be calibrated before testing. Information about this DC offset, the calibration process, and other available calibrations is available in a whitepaper from TI.[5]

Without the focus tube, there is an increased risk in detecting objects off-angle to the direction the radar is pointing such as vehicles or people. In order to reliably find a second peak down hole it is important to minimize these kinds of reflections or use other methods to validate that the secondary peaks are coming from down hole.

Once the intensity of the primary reflection has been measured, there are theoretical limits to the intensity of the received signal. The two-way radar equation can be used to set an upper limit to the secondary peak.

$$\mathrm{P_r} = P_t G_t G_r \cdot \frac{\boldsymbol{\sigma} c^2}{(4\pi)^3 f^2 \boldsymbol{R^4}}$$

Figure 11. The Two-Way Radar Equation in Free Space

The two-way radar equation shows that signal strength decreases as range increases and depends on the radar-cross section $(RCS, \boldsymbol{\sigma})$ of the bottom of the hole. Additionally, this equation assumes that the signal is travelling through free space. The geometry of the hole will change both of these measurements and data should be collected across a variety of use-cases to characterize the expected intensity of a secondary down-hole reflection.

The primary detection of the spur can be explained by recognizing that noise from the power supply is leaking into either the TX signal or IF[6] signal of the radar sensor. The fact that a second spur can be found in holes with a high RCS indicates that the noise is likely in the TX signal and being transmitted off-board. A high RCS target is able to reflect this signal back to the receiver, and it appears in the sampled data twice. Because this is undefined behavior of the PMIC and subject to change in future board revisions, algorithm development is not recommended based on this behavior. The PMIC spur can also be affected by chirp configuration changes and either moved outside the desired range of the sensor or moved lower in the range profile such that an undesired harmonic is detected that does not indicate a wet hole. Future board revisions are planned which offer the ability to control the PMIC clock speed and adjust or remove the spur location.

6.2 Peak Detection and Spur Rejection

The point detection algorithm being used is called CFAR (Constant False Alarm Rate). This is a radar-specific algorithm which is used to separate desired peaks in the output FFT from background noise. This algorithm is applied to the output data in both the range and doppler direction. There are multiple settings for this algorithm, but the most common is the threshold scale. This setting is exposed in the setDetectionThreshold command, but the rest of the settings can be tuned by issuing the whole command documented in the mmWave SDK within the chirp configuration. These settings are best adjusted while capturing data in the desired use-case.

It may be difficult to separate peak detection due to real targets from peaks created by spurs disrupting the data. One method for determining if peaks are caused by spurs or actual targets is to toggle spread spectrum mode using the enableSpreadSpec CAN command. Peaks crated by spurs will have become wide hills when spread spectrum mode is enabled and narrow peaks when spread spectrum mode is disabled. This can be toggled in real time to confirm that a peak is caused by spread spectrum interference.

The 100 meter chirp configuration available using the selectChirpProfile command with an argument of 1 has a spur peak that is beyond 100 meters and will not be detected by the point detection algorithm which has a limit of 100 meters. This limit is set in the cfarFovCfg command.

[5] https://www.ti.com/lit/an/spracf4c/spracf4c.pdf?ts=1680747172895

[6] Intermediate Frequency which is sampled by ADC

150 Lucius Gordon Dr,
West Henrietta, NY 14586
Phone: 585.429.1550
Fax: 585.429.1551
D3Engineering.com

7.0 Document Revision History

Revision	Comments
0.1	Preliminary Review
0.2	Second Review
0.3	Add CCS notes
0.4	Added notes on DSS functional algorithm
0.5	Added sensorStatus command
0.6	Added setDetectionThreshold and enableSpreadSpec commands
0.7	Added comments to address