

Ultrasonic Multi-Transducer Pair Software Design Specification

Leo Estevez & Srinivas Lingam

MSP430 Ultrasonic Applications

ABSTRACT

This specification document discusses the enhancements required to the Ultrasonic water flow meter example project for single transducer to support multiple transducer pairs based on MSP430FR604x devices. It also includes conceptual sequence diagrams.

Table of Contents

1	Introduction	2
2	#defines	2
3	USS Library APIs	2
4	Data Structures	2
5	Functions and APIs	3
6	Call Sequences	4
6.1	Initialization.....	4
6.2	Execution.....	5
7	Conceptual Sequence Diagrams	6
7.1	Steady State Multi-channel Sequencing	6
7.2	Alternate Steady State Multi-channel Sequencing	6
7.3	Sequencing of Measurement K of Channel N	7
7.4	Sequencing of Transducer Excitation and Capture of each Channel N	8
7.5	Channel Configuration vs. Reconfiguration	8
7.6	Algorithm <i>AbsToF</i> Search vs. Track	8
8	Using the Design Center GUI.....	9
9	References	10
	Revision History	11

1 Introduction

This document describes how the USS Application Software that is part of the UltrasonicWaterFR604x' USS_Water_Demo example project will be modified to support multiple transducer pairs. References to the existing variables and data structures can be found in the latest water demo project at ti.com from the **Water Metering** section of <http://www.ti.com/tool/MSP-ULTRASONIC-DESIGN-CENTER>. The enhancements are intended to work with the existing Ultrasonic Design Center.

The Ultrasonic Design Center configuration panels can be used to specify the parameters for each channel in a sequential manner. Similarly, the waveform and ADC display panels can be used to display the DToF, AbsToF, VFR and ADC for a specified channel number or in an interleaved manner.

2 #defines

USS_NUMBER_OF_TRANSDUCER_PAIRS: # of Transducer pair channels

USS_ALG_OBJECT_MAX_LENGTH: The Maximum size or length in bytes required or allocated for algorithm object inside the ultrasonic library

Param10 in the Ultrasonic Design Center configuration is used to specify the channel (multi-transducer pair) number while configuring the parameters.

3 USS Library APIs

The Ultrasonic SW Library is expected to have the following new APIs to support multi-transducer pairs. These will be used by the application. More information can be found in `ussSwLib.h` and the Ultrasonic SW Library documentation.

```
uint8_t* USS_getAlgorithmObjPtr(void);
```

```
uint16_t USS_getAlgorithmObjLength(void);
```

4 Data Structures

The following data structures shall be implemented:

USS_Measurement_Configuration**ussMeasurementConfigPtr[USS_NUMBER_OF_TRANSDUCER_PAIRS+1]**

This is an array of pointers to an independent number of measurement configurations. One of the configurations is reserved for holding the configuration state of the GUI while the other configurations are specific to each transducer pair.

USS_Capture_Configuration **ussCaptureConfigPtr[USS_NUMBER_OF_TRANSDUCER_PAIRS+1]**

This is an array of pointers to an independent number of capture configurations. One of the configurations is reserved for holding the configuration state of the GUI while the other configurations are specific to each transducer pair.

USS_Algorithms_User_Configuration **ussAlgConfigPtr[USS_NUMBER_OF_TRANSDUCER_PAIRS+1]**

This is an array of pointers to an independent number of algorithm configurations. One of the configurations is reserved for holding the configuration state of the GUI while the other configurations are specific to each transducer pair.

USS_Meter_Configuration **ussMeterConfigPtr[USS_NUMBER_OF_TRANSDUCER_PAIRS+1]**

This is an array of pointers to an independent number of meter configurations. One of the configurations is reserved for holding the configuration state of the GUI while the other configurations are specific to each transducer pair.

uint8_t gUssAlgObjectPtr[USS_NUMBER_OF_TRANSDUCER_PAIRS][USS_ALG_OBJECT_MAX_LENGTH]

This is an array of bytes to an independent number of algorithm objects. These objects are specific to each transducer pair.

5 Functions and APIs

The following functions shall be implemented:

USS_storeGUIConfiguration(uint8_t transducerPairNumber)

This function will update the `ussMeasurementConfigPtr[transducerPairNumber]` and `ussCaptureConfigPtr[transducerPairNumber]` values with the values stored in `gUssSWConfig.measurementConfig` and `gUssSWConfig.captureConfig` where `transducerPairNumber = gCommandHandler.user_param10`.

USS_copyAlgorithmObject(uint8_t* srcAlgorithmObject ,uint8_t* destAlgorithmObject, uint8_t algorithmObjectLength)

This function will copy the contents of one algorithm object specific to transducerPairNumber into the library algorithm object.

USS_configTransducerPair(&gUssSWConfig, uint8_t transducerPairNumber)

This function will load the `ussMeasurementConfigPtr[transducerPairNumber]` and `ussCaptureConfigPtr[transducerPairNumber]` specific to a given `transducerPairNumber` into the `gUssSWConfig` structure. This function will also control three GPIO pins to enable the MUX and select the appropriate transducer pair for capture.

USS_reconfigureUltrasonicMeasurement(&gUssSWConfig, uint8_t transducerPairNumber)

This function will update the minimal number of registers in the SAPH and SDHS subsystems required to update the Number of Pulses, ADC Start Gap, and Gain setting for a given `transducerPairNumber`.

6 Call Sequences

The following call sequence shall be implemented:

6.1 Initialization

The initialization functions **USSLibGUIApp_Init ()** and **USSLibGUIApp_ReInitialize()** in the water demo project have to be enhanced to support multiple-transducer pairs functionality as suggested below.

```
// Initialization
```

```
gUssLibAlgObj = USS_getAlgObjPtr();
```

```
gUssLibAlgObjLength = USS_getAlgObjLength();
```

```
USS_configureUltrasonicMeasurement(&gUssSWConfig);
```

```
gUssTransducerPairNumber=USS_TP_1;
```

```
gCommandHandler.user_param10=0xFF;
```

```
for(transducerPairNumber=0;transducerPairNumber<USS_NUMBER_OF_TRANSDUCER_PAIRS;transducerPairNumber++){
```

```
    USS_configTransducerPair(transducerPairNumber);
```

```
USS_initAlgorithms(&gUssSWConfig);

    USS_copyAlgorithmObject(gUssLibAlgObj , gUssAlgObjectPtr[transducerPairNumber],
gUssLibAlgObjLength);
}
```

6.2 Execution

The main loop for the measurement and algorithm processing sequence also needs to be enhanced to support multiple-transducer pairs functionality as suggested below.

```
// Execution
```

```
while(1){

    USS_configTransducerPair(USS_GUI_TRANSDUCER_PAIR);

    HMI_PreMeasurement_Update();

    // Check if param10 is not 0xFF – if not, then copy the data from the GUI to the relevant transducer pair
    Measurement/Capture Configuration – except in case of interleave. Then reset back to 0xFF.

    USS_storeGUIConfiguration(gCommandHandler.user_param10)

    for(transducerPairNumber=0;transducerPairNumber<USS_NUMBER_OF_TRANSDUCER_PAIRS;transducerPairNumber++){

        USS_copyAlgorithmObject(gUssAlgObjectPtr[transducerPairNumber], gUssLibAlgObj ,
gUssLibAlgObjLength);

        USS_configTransducerPair(transducerPairNumber);

        USS_reconfigureUltrasonicMeasurement(&gUssSWConfig, transducerPairNumber);

        USS_startLowPowerUltrasonicCapture(&gUssSWConfig);

        if((gUssTransducerPairNumber&0x03)==transducerPairNumber || (gUssTransducerPairNumber
==0x24)

        {

            HMI_PostMeasurement_Update();

            USS_runAlgorithmsFixedPoint(&gUssSWConfig,&algResFixed);
```

```

        USS_copyAlgorithmObject(gUssLibAlgObj,
        gUssAlgObjectPtr[transducerPairNumber], gUssLibAlgObjLength);

        if ( (USS_message_code_valid_results != code) &&
            (USS_message_code_algorithm_captures_accumulated != code) )
        {
            USSLibGUIApp_send_error(COMMAND_HANDLER_ERROR_FAULT_ALG_ERROR,
                                   (uint16_t) code);
        }

        HMI_PostAlgorithm_Update(&algResFixed);
    }
}

```

7 Conceptual Sequence Diagrams

The following figures show the conceptual sequence diagrams. The subsequent details and breakup of each of the channels is shown only for the 1st approach in section 7.1.

7.1 Steady State Multi-channel Sequencing

At a high level, Figure 1 shows the steady state multi-channel sequencing for any one measurement. In this case, the pulse excitation and capture of each channel is followed immediately by the algorithmic processing of that channel before the next channel is excited and processed. The figure shows the scenario of 4 channels.

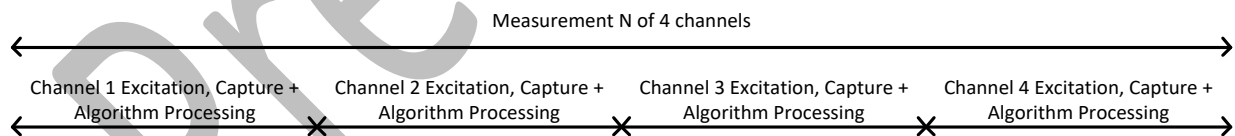


Figure 1 Steady State Sequencing of Multi-channel Processing

7.2 Alternate Steady State Multi-channel Sequencing

An alternate approach that can be considered by some customers is shown in Figure 2. In this scenario, the pulse excitation and capture of all the channels are completed before the captured data of any of the channels is processed by the algorithms. In a system that has 4 channels, this requires allocating 4x capture buffers and possibly copy of the data to and from LEARAM to FRAM if the LEARAM is not sufficiently large to hold captures of all 4 channels in addition to the algorithm objects for each of the

individual channels. Of course, if the LEARAM is large enough to hold 4 UPS & DNS captures from the 4 channels, it is not necessary to do the copies to and from FRAM.

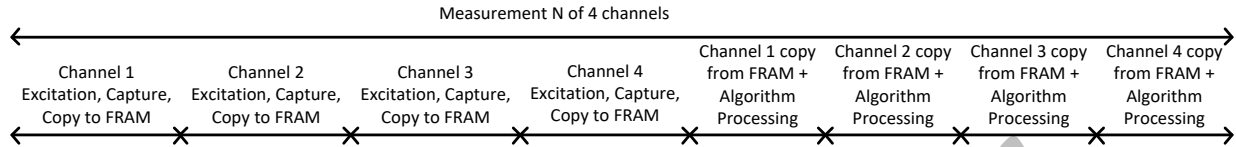


Figure 2 Steady State Sequencing of Multi-channel Processing of an Alternate Approach

7.3 Sequencing of Measurement K of Channel N

The sequence of operations for each channel N for any measurement is shown in more detail below in Figure 3.

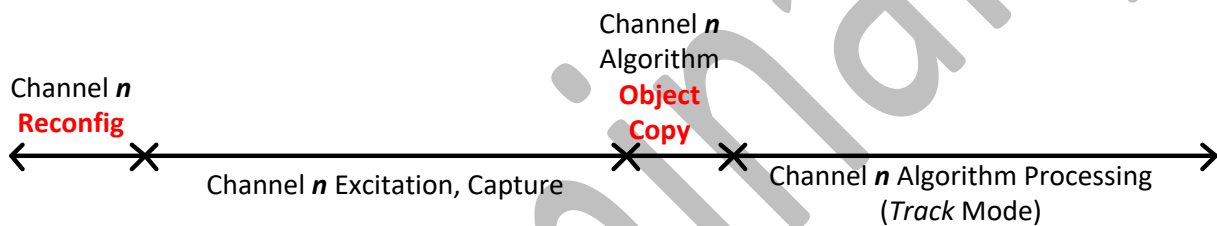


Figure 3 Sequence of Operations for Channel N in each measurement

The “Channel *n* Reconfig” corresponds to the *USS_reconfigureUltrasonicMeasurement()* and has a relatively very small processing overhead. The “Channel *n* Algorithm Object Copy” is *USS_copyAlgorithmObject()* and basically copies the algorithm object of the *n*th channel from FRAM for processing by *USS_runAlgorithmsFixedPoint()*. This is also a relatively low overhead function. Hence the time taken for total excitation, capture and algorithm processing is nearly same as the time taken for a single transducer pair solution.

In comparison, the sequence of operations for single transducer pair solution is shown below in Figure 4. It is not to scale but for practical systems, the time taken for the sequences in Figure 3 and Figure 4 are very close to each other.

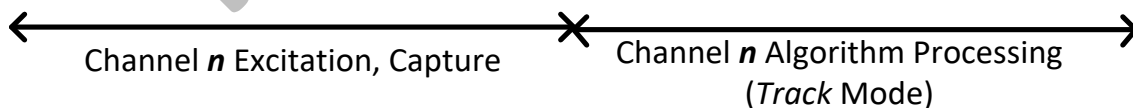


Figure 4 Sequence of Operations for Single Transducer Pair in each measurement

7.4 Sequencing of Transducer Excitation and Capture of each Channel N

The sequence of operations to complete the excitation and capture for each channel N for any measurement is shown in additional detail below. To be clear, this is the “Channel *n* Excitation, Capture” in section 7.3.

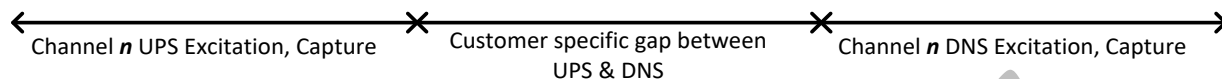


Figure 5 Sequence of Operations for Transducer Excitation and Capture for each channel in each measurement

The “Customer specific gap between UPS & DNS” is dependent on the customer pipe design.

7.5 Channel Configuration vs. Reconfiguration

In general, the Channel measurement configuration, `USS_configureUltrasonicMeasurement()`, that is invoked during `USSLibGUIApp_Init()`, consumes a significant amount of time. Hence, it is important to **re-configure** only the parameters that differ between the channels. This is accomplished by `USS_reconfigureUltrasonicMeasurement()`. The below comparison in Figure 6 is not exactly to scale but the relative timing is depicted to convey the concept pictorially.

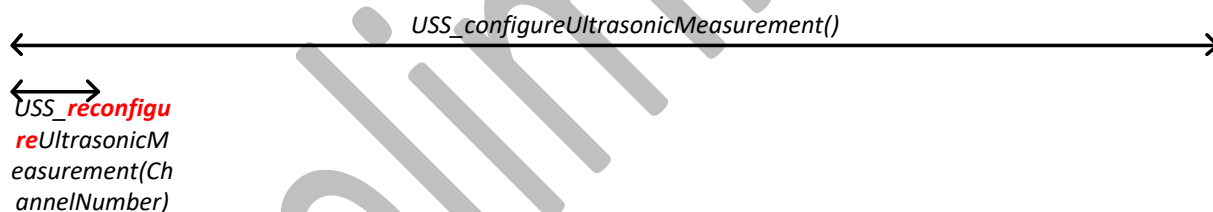


Figure 6 Timing Comparison of Complete Measurement Configuration and Re-Configuration

7.6 Algorithm *AbsToF* Search vs. Track

In steady state, the Absolute Time of Flight (*AbsToF*) estimation algorithm usually operates in a *Track* mode. See section 2.2.1 of the application note “[Waveform capture based ultrasonic sensing water flow metering technology](#)” for explanation of the *Search* and *Track* modes. The algorithm usually goes into *Search* mode when the flow rate or temperature changes significantly. As explained in the application note, the *Search* mode consumes significantly higher amount of time compared to *Track* mode. The below comparison in Figure 7 is not exactly to scale but the relative timing is depicted to convey the concept pictorially.

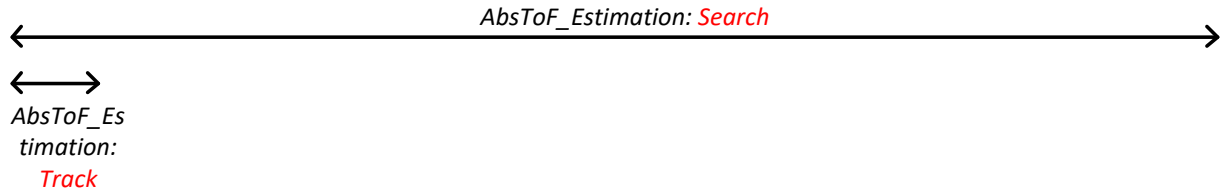


Figure 7 Timing Comparison of Absolute Time of Flight (AbsToF) Search & Track Modes

8 Using the Design Center GUI

User Param 10 in advanced parameters is used to change the channel that the mux is switched to and send configuration information. The following command codes can be issued in param 10.

0 – When 0 is sent in param 10, the mux will be switched to channel 0 and parameters in the GUI will be updated for this channel.

1 - When 1 is sent in param 10, the mux will be switched to channel 1 and parameters in the GUI will be updated for this channel.

2 - When 2 is sent in param 10, the mux will be switched to channel 2 and parameters in the GUI will be updated for this channel.

3 - When 3 is sent in param 10, the mux will be switched to channel 3 and parameters in the GUI will be updated for this channel.

*please note that the number of transducer pairs supported must be set in USS_userconfig.h. The code snippet below shows this configured for 2 pairs of transducers.

```
////////////////////////////////////
```

```
#define USS_MT_NUM_OF_TRANS_PAIRS          2
```

```
////////////////////////////////////
```

20 – When 20 is sent in param 10, the mux will switch to channel 0, but no parameters, except UPS0 to UPS1 Gap, will be updated for this channel.

21 – When 21 is sent in param 10, the mux will switch to channel 1, but no parameters, except UPS0 to UPS1 Gap, will be updated for this channel.

22 - When 22 is sent in param 10, the mux will switch to channel 2, but no parameters, except UPS0 to UPS1 Gap, will be updated for this channel.

23 - When 23 is sent in param 10, the mux will switch to channel 3, but no parameters, except UPS0 to UPS1 Gap, will be updated for this channel.

24 – When 24 is sent in param 10, interleave mode is enabled. The software will switch between all available channels that are enabled. The results can be viewed in the waveforms tab.

9 References

- [1] Application Note [Waveform capture based ultrasonic sensing water flow metering technology](#),
March 2019

Revision History

Changes from June 24, 2018 to July 3, 2019:

1. Added Table of Contents and Section 7 Conceptual Sequence Diagrams

Preliminary

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI's products are provided subject to TI's Terms of Sale (www.ti.com/legal/termsofsale.html) or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI's provision of these resources does not expand or otherwise alter TI's applicable warranties or warranty disclaimers for TI products.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265

Copyright © 2019, Texas Instruments Incorporated