NXP Semiconductors
Application Note

Document Number: AN5375
Rev. 1, September 2017

S32R RADAR Signal Compression

Feature description and guidelines for using signal compression

by: NXP Semiconductors

Contents

1 Introduction

The S32R27x is a 32-bit Power Architecture® based Microcontroller Unit (MCU) targeted for automotive applications. It efficiently supports surround RADAR and mid-range front RADAR applications. The device family members are designed to address advanced RADAR signal processing capabilities combined with automotive microcontroller capabilities for generic software tasks and car bus interfacing. S32R27x meets the high-performance computation demands required by modern beamforming fast chirp modulation RADAR systems by offering unique signal processing acceleration together with a powerful multi-core architecture.

The S32R27x supports automotive safety applications that require a high Safety Integrity Level (ASIL) and adds SHE compliant security features to prevent unauthorized manipulations. The high-integration level of S32R27x enables the customer to build compact, safe and secure, low cost RADAR sensors with leading edge performance.

In addition, the S32R27x MCU supports RADAR signal processing acceleration via the Signal Processing Toolbox (SPT). The SPT is a powerful engine containing high-performance signal processing operations, driven by a user-oriented instruction set. The programmability ensures flexibility for modifications of the signal processing flow.

The CPU cluster is removed from frequent scheduling of hardware operations, but still controls and interacts with the processing flow. The S32R27x MCU incorporates compression technology enabling users to minimize the amount of memory required for RADAR signal processing tasks. A number of different compression types allow tradeoffs between compression ratios, signal quality loss, and the amount of memory saved. This application notes provides a compression overview, lists the compression formats, and documents a basic procedure for enabling compression features using a 4- channel use case.

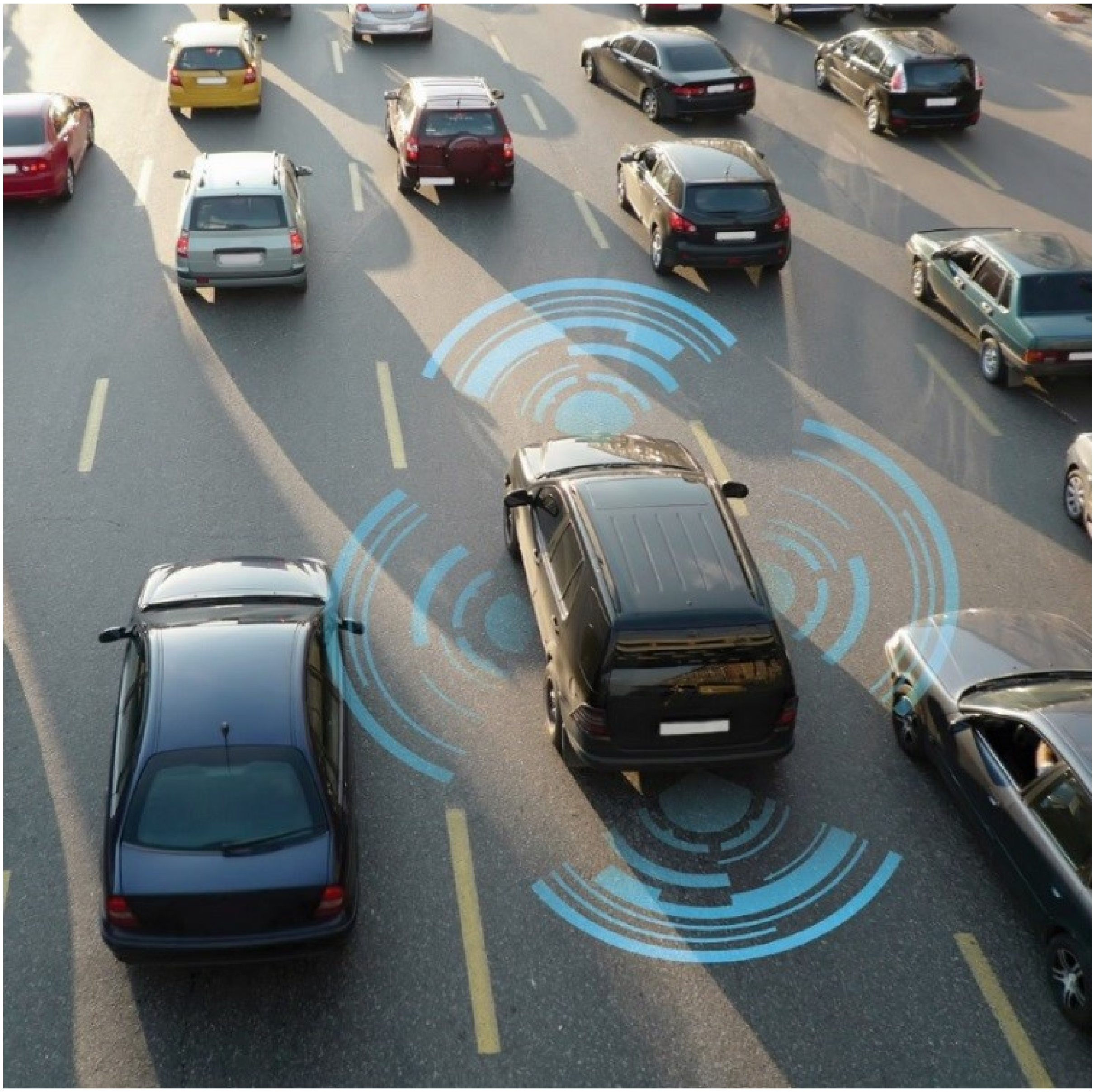

Figure 1. Typical Automotive RADAR use

The following block diagram illustrates a high-level system view of a RADAR application:

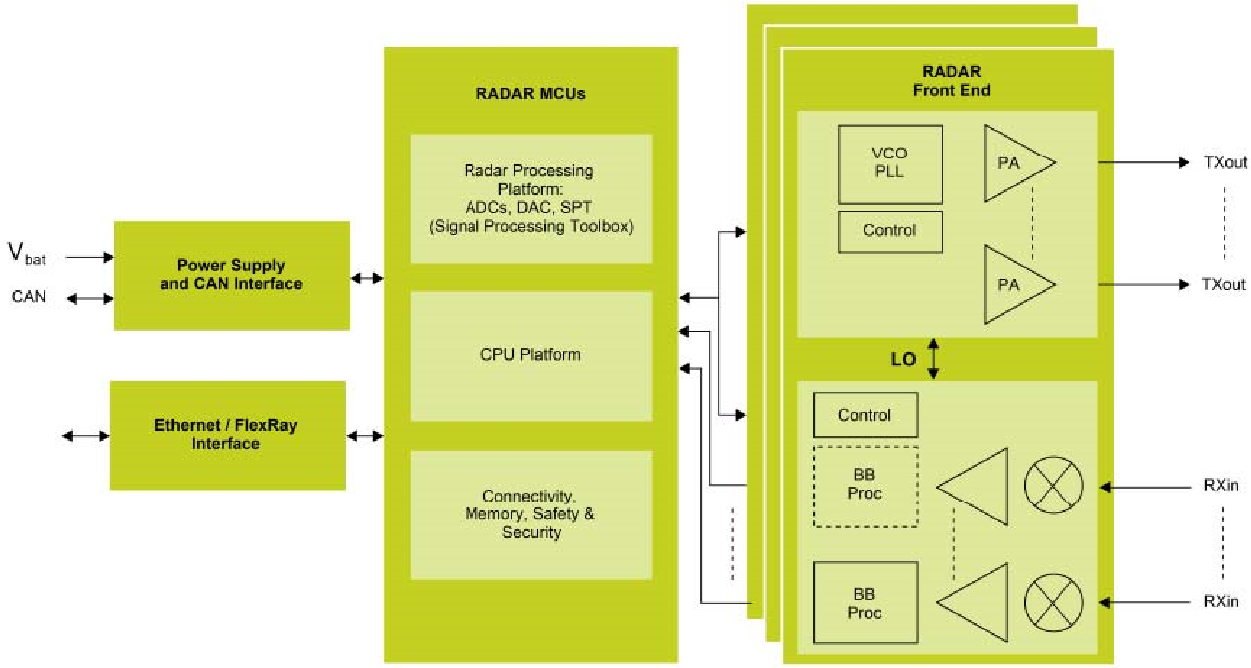

Figure 2. High-level system view for RADAR solution

The products supporting advanced RADAR processing are shown below:

Device	Application	Details
MR2001	77 GHz multi-channel transceiver	Chipset includes 2 channel TX, 3 channel RX, VCO – scalable to 4 TX and 12 RX
MR3003	77 GHz multi-channel transceiver, single chip	Single device integration of the MR2001 chipset 3 Transmit, 4 Receive
TEF810x	77 GHz RF CMOS single chip transceiver	3 Transmit, 4 Receive
S32R27x	RADAR MCU with embedded security and MIPI-CSI input	• embedded SPT • ASIL safety • SHE compliant security • 4-channel analog front-end • MIPI-CSI input
MPC577xK	RADAR MCU for long-range and mid-range applications	• embedded SPT • ASIL safety • 8-channel analog front-end • Parallel data input

NOTE

The S32R27x RADAR MCU inter-operates with both NXP and non-NXP RF front-ends via MIPI-CSI2. The compression features describe in this application note are supported in the S32R27x with either NXP or non-NXP RF front-ends.

2 Compression overview

The MPC5775xK and S32R27x RADAR Micro-Controller Units (MCUs) contain an efficient and powerful embedded Digital Signal Processor (DSP) called the Signal Processing Toolkit (SPT). The SPT supports a lightweight command syntax enabling users to manage multiple incoming RADAR streams, performing Fast Fourier Transforms (FFTs) in real-time. The SPT Programmable Direct Memory Access (DMA) is one of the key elements supporting transfer of RADAR streams into and out of the SPT core. The Programmable Direct Memory Access (PDMA) command supports a number of different compression types yielding compression ratios from 2.8:1 to 6:1, thereby saving space in valuable system RAM. Some compression types can result in slight signal quality loss manifested as a measurable impact on Signal-to-Noise Ratio (SNR). This application note shows how to enable the various compression types.

The compression technology:

- Reduces the number of bits and increases transfer throughput
- Ensures optimal memory filling without redundant bits
- Has near lossless operation for most cases, but guarantees certain SNR for detection [1]
- Supports a variety of options to match application needs

To understand how compression is relevant in the S32R27x, it is very important to have a basic understanding of the RADAR signal processing flow. This flow is described in the following diagram and supporting text:

[1] For compression ratios 2.8 : 1 and 3 : 1, loss is near zero (typically < 1/4 dB). For higher compression ratios (e.g., 6 : 1), signal loss is < 1 dB.

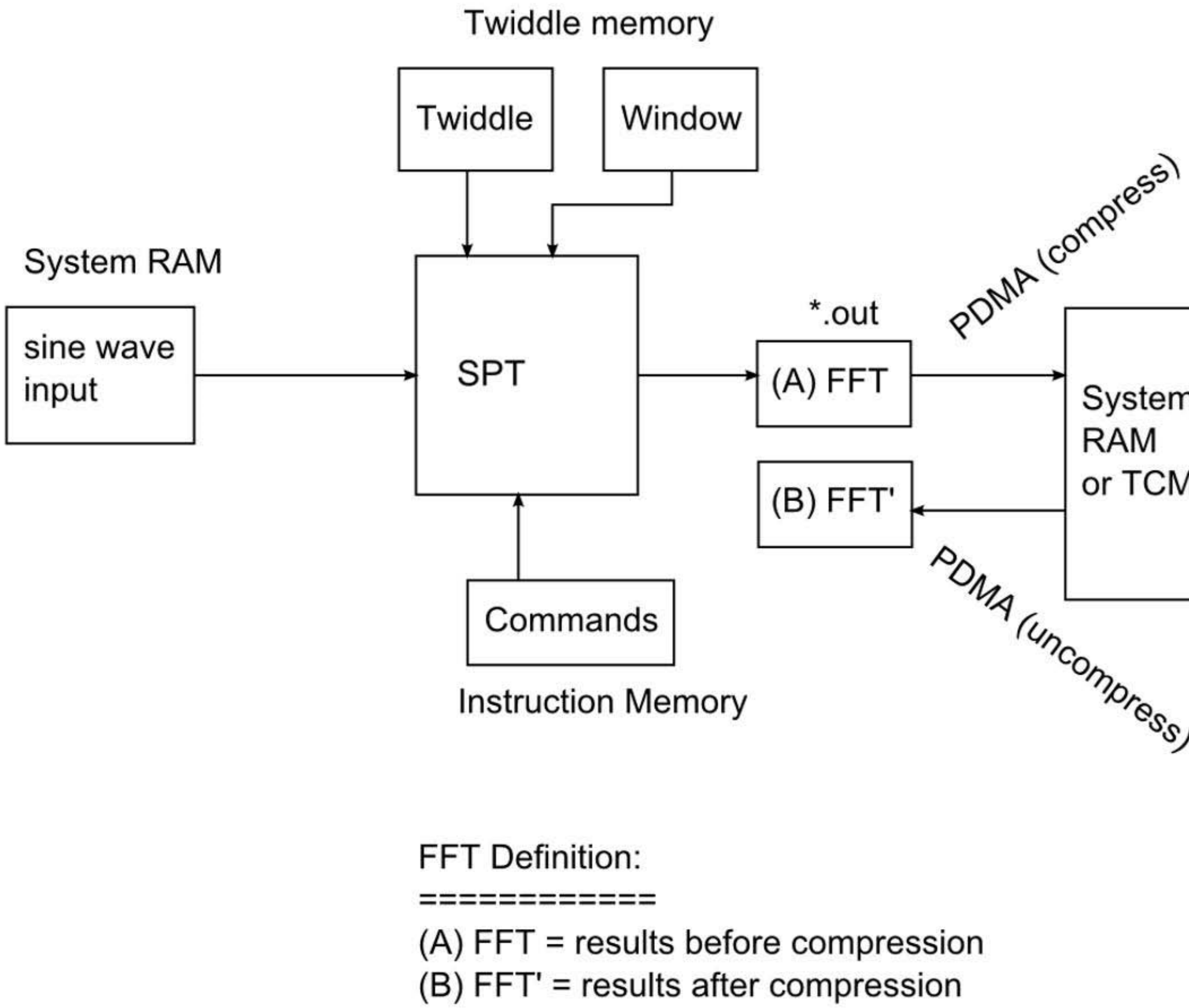

Figure 3. Simplified block diagram of the SPT

RADAR signal processing in the MCU occurs in the following steps:

Step 1: Analog to Digital Convert (ADC) or MIPI-CSI2 data samples are transferred to System RAM (SRAM) / Tightly Coupled Memory (TCM).

Step 2: Once an ADC sample frame has been acquired, data is transferred to the SPT for Range Fast Fourier Transform (FFT) processing. For example, the SPT uses four Radix 4 (RDX4) commands to calculate a 256-point range FFT.

Step 3: SPT stores results of range FFT in internal SPT Operand RAM memory (space constrained). PDMA transfers results to SRAM/TCM. Compression type can be selected in this step.

Step 4: A second PDMA command transfers data from SRAM / TCM into SPT Operand RAM for Doppler FFT.

3 Compression formats and ratios

The SPT supports the following compression types with associated compression ratios. See the S32R27x Reference Manual for more details.

Compression Mode	Number of Channels / Number of Complex operands (24, 16bit)	Compression Ratio [bits_opram/ bits_ram]	Compression Ratio [bits_ram / bits_ram]
CP4, CP4D	4	3	4
CP4FMTA, CP4DFMTA	4	6	8
CP4FMTB, CP4DFMTB	4	2.82	3.8
CP6	6	4.5	6
CP8FMTA	8	6	8
CP8FMTB	8	2.91	3.9
CP16FMTB	16	2.95	3.9

4 SPT programming procedure

To prepare for SPT execution, the user must perform the following steps:

1. Generate sine waves with noise, place in Operand RAM (OPRAM) or TCM or SRAM.
2. Calculate twiddle factors and Window coefficients and place in Twiddle RAM (TRAM).
3. Determine SPT commands including:
   - Window command
   - Radix 4 commands
   - COPY transpose command
   - PDMA write with compression
   - PDMA read with decompression

This section details each of the steps described above.

Step 1: Build a sample sine wave:

Using Matlab, the following steps define the waveforms required:

```
Generate sine waveform: z = (sin(2*pi*48*t+phi))/2
Add white Gaussian noise: z = awgn(z,Noise,'measured',3)
Create Window coefficients: z = z.*chebywin(nPts, 60)
Scale waveform to 15 bits: z = floor(z*26214);
```

For static signal testing, the sine wave stimulus is stored in system RAM. For real world signal testing, an ADC sample stream is assigned to the system RAM or TCM buffer.

Steps 2: Setup Twiddles and Window coefficients in TRAM

The SPT contains dedicated memory for Twiddles and Window coefficients.

The test configuration uses Chebychev windowing with 256-sample points and 60 dB attenuation. The application uses Matlab to generate Chebychev coefficients as follows:

```
numPts = 256; % number of sample points
attenuation = 60; % attenuation level for the Chebychev windowing function
wc = chebwin(numPts, attenuation); % call the Chebychev windowing function
wvtool(wc); % display the results
```

Step 3: Determine SPT commands

For this application example, the SPT pulls data in from system RAM to Operand RAM (PDMA command), performs a Window operation (WIN command), performs four Radix 4 operations (RDX4 command), a PDMA with compression operation, and a PDMA with decompression operation (PDMA command). The user must specify each command to execute the necessary functions in the SPT. See the Reference Manual for the command details.

The command list encompassing steps 4 through 8 for a four-channel use case is shown below.

Step 4: Execute SPT PDMA Sine Wave from system RAM to OPRAM (16-bit real pack)

```
0xA10D0100; 0x00000000; 0x80000000; 0x01000001 // Channel A
// Channel B
// Channel C
// Channel D
```

Step 5: Execute SPT WIN command

```
0x85800100; 0x8000A000; 0x43000101; 0x00010240 // WINdow, scale by 4 bits left shift
```

Step 6: Execute SPT RDX4 commands – 4 rounds for FFT256

```
0x8A010100; 0xA000A100; 0x40080101; 0x00000000 // radix 4 round 0
0x8A110100; 0xA100A200; 0x40080101; 0x00000000 // radix 4 round 1
0x8A210100; 0xA200A300; 0x40080101; 0x00000000 // radix 4 round 2
0x8A310100; 0xA3008600; 0x40080104; 0x00000000 // radix 4 dest increment address 4
# COPY transpose forward 8x4 / vector length is 1024. Copy from source 0x8600 to
destination 0x8A00
0x950D0400; 0x86008A00; 0x08080101; 0x00000000
```

Step 7: Execute PDMA from OPRAM to system RAM with CP4D compression selected

```
0xA12F0400; 0x00000800; 0x8A000000; 0x01000001
```

Step 8: Execute PDMA from system RAM to OPRAM with CP4D decompression selected

```
0xA12D0200; 0x00000800; 0x90000000; 0x01000001
# For other types of the compression, the following code can be used:
# CP4DFMTA compression
0xA1470400; 0x00000800; 0x8A000000; 0x01000001 # read from OPRAM into system RAM
0xA1450200; 0x00000800; 0x90000000; 0x01000001 # write back from system RAM to OPRAM
# CP4DFMTB compression
0xA14F0400; 0x00000800; 0x8A000000; 0x01000001 # read from OPRAM into system RAM
0xA14D0100; 0x00000800; 0x90000000; 0x01000001 # write back from system RAM to OPRAM
# CP8 compression
0xA1270400; 0x00000800; 0xA0000000; 0x01000001 # read from OPRAM into system RAM
0xA1250100; 0x00000800; 0xB0000000; 0x01000001 # write back from system RAM to OPRAM
# CP8FMTB compression
0xA1530400; 0x00000800; 0xA0000000; 0x01000001 # read from OPRAM into system RAM
0xA1510100; 0x00000800; 0xB0000000; 0x01000001 # write back from system RAM to OPRAM
```

The following block diagram illustrates memory contents and SPT setup:

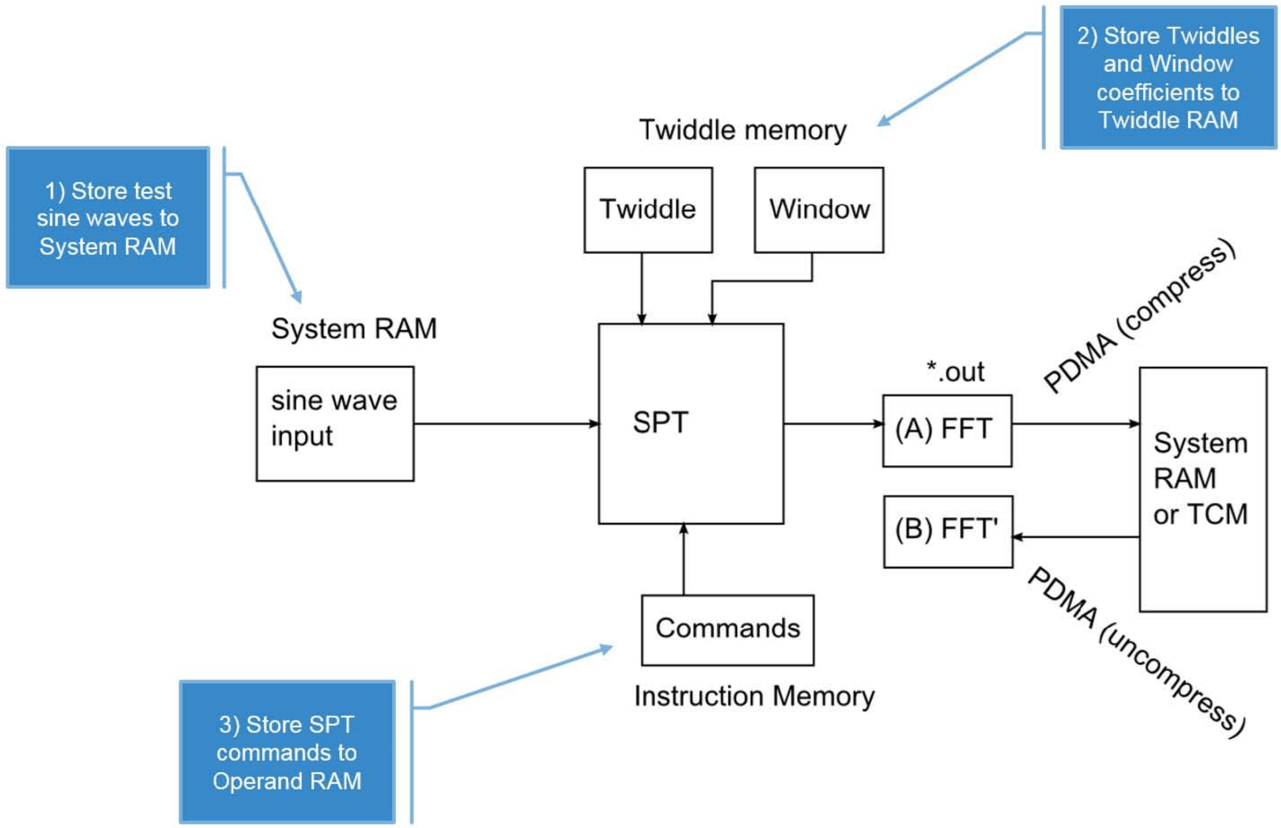

Figure 4. Setting up SPT memory contents for processing

SPT commands and memory contents developed in the previous steps govern SPT execution. Once SPT execution completes, FFT results before compression and after compression will be present in Operand RAM. The following diagram describes SPT execution:

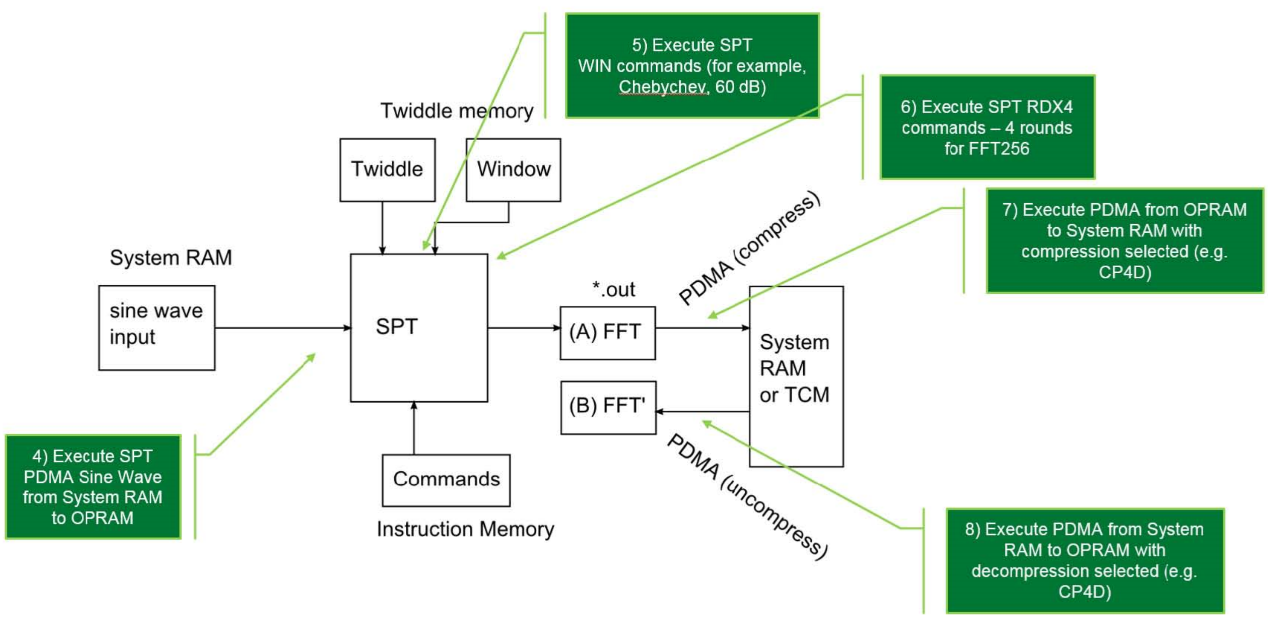

Figure 5. SPT execution with PDMA compression / decompression diagram

5 Conclusion

This application note presented compression user information for the S32R family of RADAR MCUs. The note showed the user how to setup a simple signal processing workflow enabling various 4-channel and 8-channel use cases. In conclusion:

- Compression is a powerful tool, saving memory space.
- Compression algorithms have minimal impact on SNR.
- Best performance requires signal scaling to maximize Operand bit use (optimizes dynamic range).

6 Revision history

Rev. No.	Date	Substantive Change(s)
0	January 2017	Initial version.
1	September 2017	1. Updated the section SPT programming procedure on page 6. 2. Updated the document from the editorial perspective.

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

Document Number: AN5375
Rev. 1, September 2017