A Novel Approach to Service Serial Peripheral for Data Acquisition using EDMA3 in TMS320C6748 DSP

Harikrishnan Prabhakaran
Tismo Technology Solutions (P) Ltd.
21 June 2021

ABSTRACT

This article summarizes a novel approach in which Enhanced Direct Memory Access (EDMA3) in TMS320C6748 is used to service a serial peripheral that acquires digital data from external Analog to Digital Converters (ADC). The discussed approach is implemented in the application firmware using TI EDMA3 drivers with TI SYSBIOS as the RTOS kernel.

INTRODUCTION

With technology today any sort of quantity or information can be translated into electrical or an electromagnetic signal. The signal of interest is often a continuous time varying voltage or current. In other words, they are inherently analog in nature. For the various advantages, computers in today's world are designed to process data that is discrete in quantity and in time. To bridge the gap between real world and computing world, Analog to Digital Converters (ADC) are used to convert the analog signals to digital data. Today's market provides lots of varieties of ADC as integrated circuits that can be used for different applications. Most of them typically provide a serial interface like SPI (Serial Peripheral Interface) or I2C (Inter-Integrated Circuit) to share the digital data with microcontrollers or digital signal processors.

This article describes a problem statement in which the TMS320C6748[1], a low power DSP from Texas Instruments (TI) communicates with two external ADCs, one primary and other secondary. The primary ADC is capable of conversion of 8 channels of data all at a maximum sampling rate of 32 kHz, and the secondary ADC is capable of conversion of 6 channels of data all at a maximum sampling rate of 1 kHz. Both of the ADCs provide a Serial Peripheral Interface (SPI). The SPI module allows a duplex, synchronous, serial communication between the DSP/MCU with peripheral devices.[2] The protocol provides the flexibility of connecting multiple devices to the same SPI bus. The TMS320C6748 DSP has two SPI peripherals integrated into it. One of them is dedicated for communication with another processor. The second SPI peripheral in DSP is used in Master mode to communicate with the SPI bus to which both primary and secondary ADCs are connected.

The data read by the SPI peripheral in DSP can be collected using multiple approaches. One approach is to poll for the SPI transfer flags for each and every byte received and transmitted by the peripheral. Second approach is to use the interrupt capabilities of SPI peripheral to interrupt the DSP core every time a byte is transmitted or received. Both of these approaches have significant disadvantages in terms of wastage of valuable processing clock cycles which otherwise could be used for other useful

Copyright © 2021 Tismo Technology Solutions (P) Ltd.

activities. Also, in the above mentioned approaches, the DSP is almost incapable of monitoring each and every byte of data at higher data rates.

The article suggests an approach that utilizes the potential of Enhanced Direct Memory Access (EDMA3) in TMS320C6748 to service the SPI peripheral. Direct Memory Access (DMA) is a widely used solution to offload the data transfers between two memory points from the CPU. EDMA3 primarily performs user programmed data transfers between two memory slave points. Other capabilities include servicing of event driven peripherals such as serial ports, software paging transfers between external memory and internal memory etc. In the approach discussed, EDMA3 peripheral is user programmed to do automatic transfer of data from SPI by configuring EDMA3 to listen to SPI DMA events as well as the GPIO bank interrupt events. The EDMA3 is also programmed to auto reload the transfer context after a transfer configuration is exhausted.

The sections below discuss the EDMA approach in detail.

TERMINOLOGY

EDMA3

The architecture of the EDMA3 controller in TMS320C6748 has two principal blocks.

- EDMA3 channel controller (EDMA3CC)
- EDMA3 transfer controller (EDMA3TC)

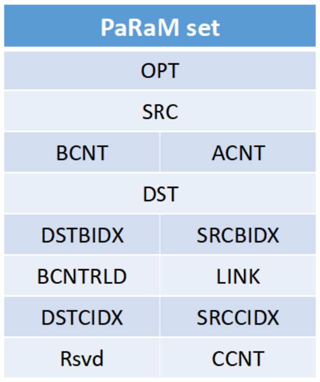
EDMA3 channel controller acts as the interface for the user to configure the EDMA3 controller. It also serves to prioritize the requests from software, or peripheral events and raise transfer requests (TR) to the transfer controller (EDMA3TC). EDMA3 transfer controller performs the actual data movement based on the Transfer Request Packet(TRP) submitted by the channel controller. TRP includes the transfer context information based on which EDMA3TC performs data read/write to source and destination addresses. The EDMA3 controller of TMS320C6748 has two EDMA3 channel controllers (EDMA3CC) and three EDMA3 transfer controllers (EDMA3TC). Each EDMA3CC supports upto 32 programmable channels for servicing peripherals and memory. Out of the two EDMA3CC, the synchronisation events (or channels) of interest in the application are in the first EDMA3 channel controller.

EDMA3 of TMS320C6748, the controller provides a lot of potential features.[3] Of these, the features significant to this article are: multidimensional transfer support upto 3 dimensions, flexible transfer definition specifically linking feature and interrupt generation for transfer completion and error conditions.

EDMA3 is programmed for data transfers using the parameter RAM (PaRaM), a programmable RAM space that stores parameter RAM sets used by DMA/QDMA/Linking channels. Each channel controller in TMS320C6748 supports upto 128 PaRaM set entries. Of these, the first 32 in each controller are reserved for DMA channels supported by the respective channel controllers. Rest of the PaRaM sets can be used for QDMA or linking entries. Each PaRaM set is a 32 byte transfer definition that stores the transfer context for the data transfer corresponding to the DMA channel. EDMA3CC services the PaRaM sets when the trigger events occur. Trigger events for DMA channels can be either manual trigger (CPU triggered), external event trigger, or chain triggered. The PaRaM set structure supports linking / autoreloading feature. With this, a PaRaM set can be auto loaded with new transfer characteristics after completion of transfer defined by the current transfer definition. The new transfer characteristics can be stored in the linking PaRaM set entries.

An EDMA3 transfer is always defined in terms of 3 dimensions. These three dimensions are:

- 1st dimension or Array(A): The 1st dimension consists of ACNT contiguous bytes
- 2nd dimension or Frame(B): The 2nd dimension consists of BCNT arrays of ACNT bytes
- 3rd dimension or Block(C): The 3rd dimension consists of CCNT frames of BCNT arrays of ACNT bytes



PaRaM set	
OPT	
SRC	
BCNT	ACNT
DST	
DSTBIDX	SRCBIDX
BCNTRLD	LINK
DSTCIDX	SRCCIDX
Rsvd	CCNT

Figure 1. PaRaM set contents

Each PaRaM set is organised into eight 32 bit words or 32 bytes as shown in *Figure 1*.

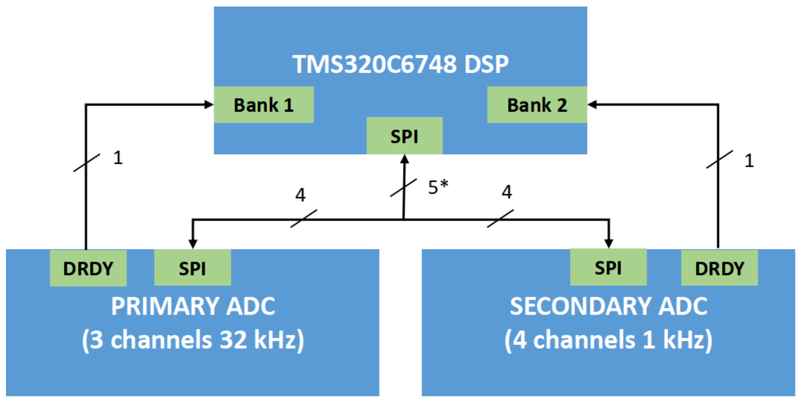
For more information regarding the PaRaM set entries and other features, refer to EDMA3 user guide.[3]

SPI

The Serial Peripheral Interface (SPI) is a high speed synchronous serial duplex port that allows a serial bit stream to be shifted in and out of the device at a programmable bit transfer rate. SPI typically consists of 4 lines: CLK (SPI Clock), MOSI (Master Out Serial In), MISO (Master In Serial Out) and CS (Chip Select). There are two SPI peripherals in TMS320C6748. The SPI peripheral in TMS320C6748 provides features to program clock frequency, clock polarity, clock phase and character length. In addition it provides DMA support by providing DMA synchronization events for receive (REVT) and transmit (XEVT). These events are used by EDMA to trigger DMA transfers. When a character (say 1 byte) is to be transmitted, the SPI peripheral raises a XEVT signal. The EDMA controller which is preprogrammed to do a transfer with PaRaM set for the XEVT of SPI peripheral, transfers data from source buffer to the SPI transmit data register (SPIDAT1). Similarly, when a character is received, SPI peripheral signals DMA with REVT. The EDMA controller with the transfer context information corresponding to REVT, starts reading the data from SPI receive buffer register (SPIBUF) and transfers it to a destination buffer.[4]

PROBLEM STATEMENT

Refer to *Figure 2*, the system consists of two external ADCs, primary and secondary.



* SPI bus includes SPI MOSI, SPI MISO, SPI CLK, CS0 and CS1

Figure 2. DSP along with primary ADC and secondary ADC

The primary ADC is capable of performing data conversion of 8 channels, all of which can be done with a maximum sampling rate of 32 kHz. In addition to the SPI bus lines, DRDY line (Data ready) of primary ADC is connected to GPIO (General Purpose Input Output) pin DSP. This connection is used by primary ADC to inform the DSP that the conversion of one set of samples is complete. In other words, this line is pulled low by primary ADC when the data corresponding to a sample conversion is ready. At 32 kHz, the line is pulled low every 31.25 microseconds.

The secondary ADC connected to the same SPI bus, is capable of performing data conversion of 6 channels all at a maximum sampling rate of 1 kHz. Similar to primary ADC, the DRDY line of secondary ADC is connected to a GPIO pin in DSP.

For the application, DSP requires 3 channels of data at 32 kHz from primary ADC and 4 channels of data at 1 kHz from secondary ADC. In response to the DRDY from the primary ADC, DSP needs to activate the primary ADC using the respective chip select and fetch ADC data from the primary ADC. The data size in primary ADC is 12 bytes (3 status bytes + 3 bytes * 3 channels). The transfer has to happen within 31.25 microseconds after which the next sample is available in the data registers of primary ADC. Similarly, the secondary ADC which is operating asynchronously with respect to primary ADC interrupts the DSP every 1 milliseconds. DSP needs to use the same SPI bus to communicate with the secondary ADC after activating the corresponding chip select. The read data opcode size for secondary ADC is 16 bytes.

The primary ADC and secondary ADC operate with different SPI clock frequency and clock phase. SPI clock for primary ADC can be a maximum of 20 MHz and that for secondary ADC can be a maximum of 16 MHz. The challenge is to fetch data from both of these ADCs ensuring that no samples are missed by the DSP from any of these ADCs. In addition, there should be enough time in DSP to perform various DSP routines on blocks of data collected from both primary and secondary ADC.

Copyright © 2021 Tismo Technology Solutions (P) Ltd.

SOLUTION

The primary objective of the solution is to ensure that the data transfer with the ADCs are done reliably with minimal intervention of the DSP core so that the valuable system clock cycles are available for DSP routine. This is done by utilizing the EDMA3 of DSP in serial peripheral servicing mode. The configuration of EDMA3 for this usage mode is described in sections below. The solution is implemented using the TI processor SDK RTOS for OMAPL138 (v 06.03.00.106). The processor SDK has following versions of SYSBIOS and EDMA3 Low Level Driver (LLD).

SYSBIOS	6_76_03_01
EDMA3 LLD	2_12_05_30E

SPI Configuration

SPI peripheral for communication with the ADCs are initialized with the below mentioned configurations.

- Two SPI Data format registers, say SPIFMT0 and SPIFMT1 are initialized with the clock, phase and polarity configurations for primary and secondary ADC respectively. One of these data formats can be used later during data transmission by setting DFSEL bits in the SPIDAT1 register.
- SPI is configured in 4 pin mode and also the chip select lines corresponding to primary and secondary ADC is enabled. The chip select can be later selected by using CSNR bits in the SPIDAT1 register.
- Character length in SPI peripheral is configured to be 8 bit (1 byte)
- Interrupts from the SPI peripheral are disabled by default.
- After the initial configuration communication with the primary ADC and secondary ADC, the SPI peripheral is disabled.

EDMA3 configuration

All the synchronisation events of interest, namely the GPIO bank interrupt event, SPI1 Receive event and SPI Transmit event are associated with the EDMA3 Channel controller 0. Therefore, the EDMA3 instance 0 is initialized using the EDMA3 initialization functions provided by the EDMA3 LLD.

GPIO configuration

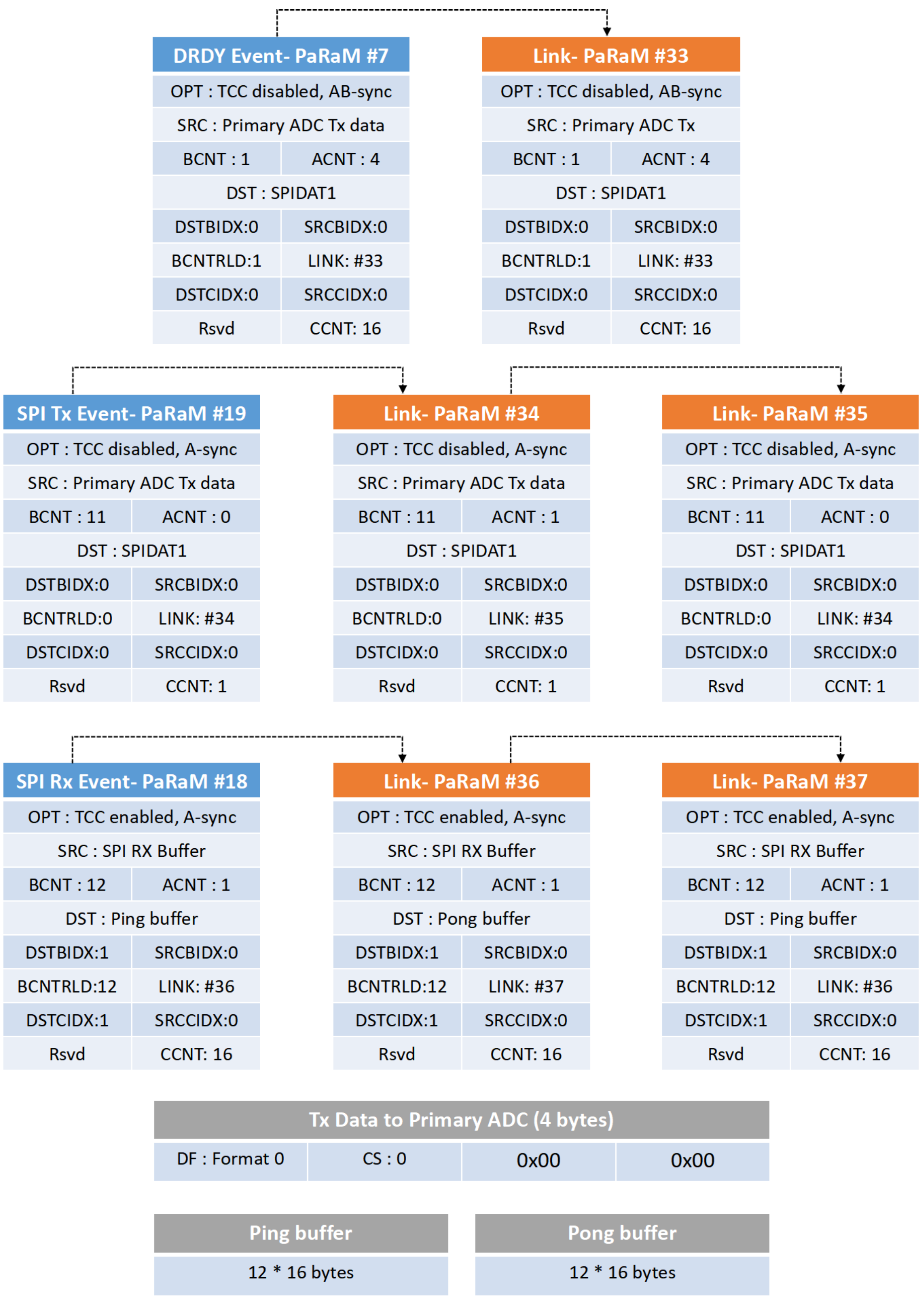
Bank level and pin level interrupt configurations are configured appropriately for GPIO pins connected to the DRDY of primary ADC and secondary ADC. The GPIO pin connected to primary ADC is in Bank 1 and that of secondary ADC is in Bank 2. A SYSBIOS Hardware Interrupt (HWI) instance is configured for the DRDY of secondary ADC.

Communication with Primary ADC

Before starting the conversion process in primary ADC, the below mentioned PaRaM sets in EDMA3 are programmed. The PaRaM set entries for each of the PaRaM sets are depicted in *Figure 3*. The EDMA3 PaRaM set is configured to do 12 byte SPI transfer for each of DRDY interrupt from primary ADC for 16 times, after which the DSP is interrupted to process the block of data collected for last 500 microseconds (assuming 32 kHz sampling rate). Using the linking feature, the PaRaM sets are also configured to do an auto load with the new transfer definitions for the next sixteen 12 byte SPI transfers.

Note: EDMA3 LLD mandates that the DMA/Link channels for the below channels are requested before the PaRaM sets are configured.

A. GPIO Bank 1 Interrupt (PaRaM #7) : DRDY line of primary ADC is connected to Bank 1. Therefore, the PaRaM set corresponding to Bank 1 needs to be configured in such a way that a SPI transfer is triggered by DRDY signal from primary ADC. The transfer defined in PaRaM set is to perform a 4 byte write to SPIDAT1 register of the SPI peripheral. Data to be written to the SPIDAT1 register is shown in *Figure 3*. A write to SPIDAT1 register causes the SPI peripheral to clock out 1 byte. It is to be noted that ACNT is set as 4, BCNT is set as 1 and CCNT is set as 16. Also, in the OPT parameter entry, the PaRaM set is configured to perform AB synchronised transfer so that BCNT*ACNT bytes are transferred for CCNT events.
B. Link set for GPIO event (PaRaM #33): This set is a copy of PaRaM #7. In PaRaM #7, link address points to the PaRaM #33. This is to auto load the PaRaM #7 after completion of 16 transfers corresponding to 16 DRDY interrupts.
C. SPI1 Transmit Event (PaRaM #19) : The initial configuration in the PaRaM set is a dummy transfer (since ACNT is set as 0). Therefore for the first XEVT from SPI, there won't be any meaningful data transfer. The reason for this configuration is explained in the descriptions following. The link address is configured to link to PaRaM #34, the first link channel for SPI Transmit event.
D. Link set 1 for SPI Transmit event (PaRaM #34): After the first dummy transfer by PaRaM#19, the entries of PaRaM #34 is loaded to the PaRaM #19. The PaRaM #34 have transfer definition to perform 11 bytes of transfer as the first byte is written by the GPIO PaRaM #7. It is to be noted that ACNT is set as 1, BCNT as 11 and CCNT as 1. The transfer type is A-Synchronised transfer. Therefore the PaRaM set is exhausted after 11 XEVT from the SPI peripheral. The link address points to PaRaM #35, second link set for SPI Transmit event.
E. Link set 2 for SPI Transmit event (PaRaM #35) : This set is a exact copy of PaRaM #19. The significance of this set is to auto load the SPI PaRaM #19 with the initial transfer definition for dummy transfer.
F. SPI1 Receive event (PaRaM #18): The transfer definition in this PaRaM



DRDY Event- PaRaM #7	
OPT : TCC disabled, AB-sync	
SRC : Primary ADC Tx data	
BCNT : 1	ACNT : 4
DST : SPIDAT1	
DSTBIDX:0	SRCBIDX:0
BCNTRLD:1	LINK: #33
DSTCIDX:0	SRCCIDX:0
Rsvd	CCNT: 16

Link- PaRaM #33	
OPT : TCC disabled, AB-sync	
SRC : Primary ADC Tx	
BCNT : 1	ACNT : 4
DST : SPIDAT1	
DSTBIDX:0	SRCBIDX:0
BCNTRLD:1	LINK: #33
DSTCIDX:0	SRCCIDX:0
Rsvd	CCNT: 16

SPI Tx Event- PaRaM #19	
OPT : TCC disabled, A-sync	
SRC : Primary ADC Tx data	
BCNT : 11	ACNT : 0
DST : SPIDAT1	
DSTBIDX:0	SRCBIDX:0
BCNTRLD:0	LINK: #34
DSTCIDX:0	SRCCIDX:0
Rsvd	CCNT: 1

Link- PaRaM #34	
OPT : TCC disabled, A-sync	
SRC : Primary ADC Tx data	
BCNT : 11	ACNT : 1
DST : SPIDAT1	
DSTBIDX:0	SRCBIDX:0
BCNTRLD:0	LINK: #35
DSTCIDX:0	SRCCIDX:0
Rsvd	CCNT: 1

Link- PaRaM #35	
OPT : TCC disabled, A-sync	
SRC : Primary ADC Tx data	
BCNT : 11	ACNT : 0
DST : SPIDAT1	
DSTBIDX:0	SRCBIDX:0
BCNTRLD:0	LINK: #34
DSTCIDX:0	SRCCIDX:0
Rsvd	CCNT: 1

SPI Rx Event- PaRaM #18	
OPT : TCC enabled, A-sync	
SRC : SPI RX Buffer	
BCNT : 12	ACNT : 1
DST : Ping buffer	
DSTBIDX:1	SRCBIDX:0
BCNTRLD:12	LINK: #36
DSTCIDX:1	SRCCIDX:0
Rsvd	CCNT: 16

Link- PaRaM #36	
OPT : TCC enabled, A-sync	
SRC : SPI RX Buffer	
BCNT : 12	ACNT : 1
DST : Pong buffer	
DSTBIDX:1	SRCBIDX:0
BCNTRLD:12	LINK: #37
DSTCIDX:1	SRCCIDX:0
Rsvd	CCNT: 16

Link- PaRaM #37	
OPT : TCC enabled, A-sync	
SRC : SPI RX Buffer	
BCNT : 12	ACNT : 1
DST : Ping buffer	
DSTBIDX:1	SRCBIDX:0
BCNTRLD:12	LINK: #36
DSTCIDX:1	SRCCIDX:0
Rsvd	CCNT: 16

Tx Data to Primary ADC (4 bytes)			
DF : Format 0	CS : 0	0x00	0x00

Ping buffer	Pong buffer
12 * 16 bytes	12 * 16 bytes

Figure 3 : EDMA PaRaM set configuration for Primary ADC. Buffers used are also shown in the image.

Copyright © 2021 Tismo Technology Solutions (P) Ltd.

set performs one byte transfer from SPI Receive buffer to the destination address. It is to be noted that ACNT is set as 1, BCNT is set as 12 and CCNT is set as 16 and transfer type is set as A-Synchronised transfer. Therefore, the PaRaM set is exhausted only after 12*16 REVT events from SPI peripheral. The destination address in the PaRaM set is to buffer location, say Ping buffer. Also, the CIDX and BIDX for destination is set as 1 so that the address is incremented for each REVT. The PaRaM set is linked to PaRaM #36, the first link set for SPI Receive event

G. Link set 1 for SPI Receive event (PaRaM #36): Compared to PaRaM #18, the only difference is destination address and link address. The destination address points to the Pong buffer. The link address points to PaRaM #37, second link set for SPI receive event
H. Link set 2 for SPI receive event (PaRaM #37): This set is a exact copy of initial configuration of SPI receive event PaRaM set, PaRaM #18. This is to auto load the PaRaM #18 with the initial configuration after the transfer of 12*16 bytes each to Ping and Pong buffers.

After the above PaRaM sets are configured, the events corresponding to GPIO Bank 1 interrupt (Event #7), SPI Transmit event (Event #19) and SPI Receive event (Event #18) are enabled. Thereafter the SPI peripheral and SPI DMA event support is enabled.

Immediately after the SPI peripheral and SPI DMA event is enabled, a SPI Transmit event (XEVT) is signalled by the SPI peripheral because the transmit buffer is empty. For this event, a dummy transfer defined initially in PaRaM #19 is performed by EDMA3 and PaRaM #19 is loaded with PaRaM #34. For this dummy transfer to SPIDAT1 register, no byte is clocked out by SPI to primary ADC. After the acquisition is started in primary ADC, the first DRDY interrupt to DSP is received after 31.25 microseconds. EDMA3 on receiving the DMA event corresponding to GPIO Bank 1 interrupt, submits a transfer request based on PaRaM #7, to do a 4 byte write to SPIDAT1 register.

SPI peripheral after clocking out 1 byte to primary ADC emits a SPI Transmit event (XEVT) for the transmit buffer is now empty. Almost immediately after this, the receive event (REVT) is also emitted because it has received a byte back from primary ADC. The PaRaM #19 performs a single byte for each received XEVT. After each byte is clocked out, a XEVT is emitted and PaRaM #19 services 11 such XEVT events. After the last 11th byte, PaRaM#19 is loaded with PaRaM #36. This corresponds to a dummy transfer. Therefore, the SPI does not clock out data until it is triggered again by the PaRaM#7.

SPI ParaM #18 processes the received data transfers for the SPI receive events received for each of the transmit writes made above.
The initial transfer definition is to write to a Ping buffer. The destination address is switched to Pong buffer after 12 bytes * 16 transfers (after 500 microseconds). This is required so that DSP can use the Ping buffer data for further data processing while

EDMA has already started transferring the next set of samples to the Pong buffer.

Therefore, the DSP is interrupted after completion of data transfer from primary ADC every 500 microseconds. Meanwhile, the EDMA controller in the background continues to do the transfer for DRDY interrupt received 31.25 microseconds after the transfer complete interrupt to DSP.

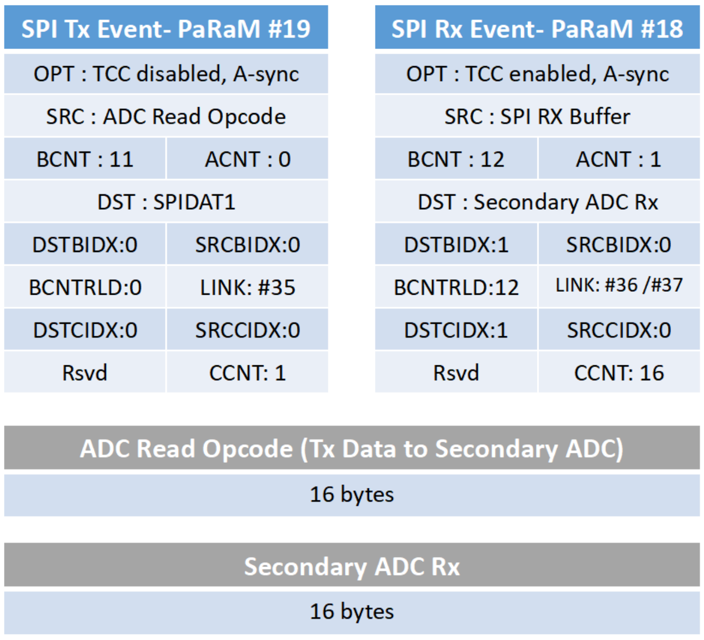
Communication with secondary ADC

Since the secondary ADC is connected to the same SPI peripheral, it is required to make use of EDMA PaRaM #19 and EDMA PaRaM #18 to transfer data with secondary ADC as well. However, these PaRaM sets cannot be modified, if there is an ongoing transfer with primary ADC.

The DSP firmware is configured to be interrupted for every DRDY interrupt from secondary ADC. This happens every 1 millisecond, if the secondary ADC is configured to have a sampling rate of 1 kHz. In the interrupt callback for the DRDY signal from secondary ADC, DSP sets a service request flag (SRF) for fetching ADC data from secondary ADC.

If the aforementioned communication with primary ADC is active, the DSP is interrupted every 500 microseconds by the transfer completion interrupts for data from primary ADC. At 32 kHz sampling rate of primary ADC, there is a time frame of 31.25 microseconds before which SPI and its associated EDMA SPI PaRaM sets can be used for communication with secondary ADC. Therefore in the transfer completion callbacks of primary ADC, DSP checks if SRF is set. If this is set, the DSP first disables the DRDY event from primary ADC to prevent any chances of PaRaM #7 disrupting the transaction with the secondary ADC. Thereafter it modifies the EDMA PaRaM set for SPI Transmit and Receive events as mentioned below (Refer to *Figure 4*).

A. SPI1 Transmit event (PaRaM #19) : This PaRaM set contains the transfer definition to transfer the data byte by byte from the buffer that contains the ADC read opcode (16 bytes) to the SPIDAT1 register. It is to be noted that the source address points from the second byte of the buffer for the first byte is written by DSP to trigger the SPI transfer. ACNT is set as 1, BCNT is set as 15 and CCNT is set as 1 and transfer type is set as A-Synchronised transfer. The PaRaM set also links to PaRaM #35 used for primary ADC communication
B. SPI1 Receive event (PaRaM #18): This PaRaM set contains the transfer definition to transfer the data from SPI receive buffer (SPIBUF) to the secondary ADC data buffers in DSP for every received byte. It is to be noted that ACNT is set as 1, BCNT as 16, CCNT as 1 and transfer type as A-Synchronised transfer. The link address is set as PaRaM #36 if the last buffer used for primary ADC received data is the Ping buffer. If the last buffer is Pong buffer, the link address is set PaRaM #37.



SPI Tx Event- PaRaM #19	
OPT : TCC disabled, A-sync	
SRC : ADC Read Opcode	
BCNT : 11	ACNT : 0
DST : SPIDAT1	
DSTBIDX:0	SRCBIDX:0
BCNTRLD:0	LINK: #35
DSTCIDX:0	SRCCIDX:0
Rsvd	CCNT: 1

SPI Rx Event- PaRaM #18	
OPT : TCC enabled, A-sync	
SRC : SPI RX Buffer	
BCNT : 12	ACNT : 1
DST : Secondary ADC Rx	
DSTBIDX:1	SRCBIDX:0
BCNTRLD:12	LINK: #36 /#37
DSTCIDX:1	SRCCIDX:0
Rsvd	CCNT: 16

ADC Read Opcode (Tx Data to Secondary ADC)
16 bytes

Secondary ADC Rx
16 bytes

Figure 4: PaRaM set configuration for secondary ADC

Note that the PaRaM sets for the SPI events are configured to load the PaRaM sets of primary ADC so that transfer context is set back to the transfer context of primary ADC automatically after the transfer with secondary ADC is complete.

After the PaRaM sets are configured for secondary ADC transfer, SPI transaction is triggered by DSP. This is done by a 4 byte write to the SPIDAT1 register. The 4 bytes written to have information regarding the data format and chip select to be used for secondary ADC and the first byte to be transmitted to the secondary ADC. This write further triggers the SPI transmit and receive events. This is handled by the PaRaM sets #19 and #18 for secondary ADC. After the transfer of 16 bytes is complete, the EDMA PaRaM #19 and PaRaM #18 is auto loaded with the transfer context of primary ADC. A transfer complete callback for the 16 bytes received from secondary ADC is also raised to DSP. In the callback the DRDY interrupt event for primary ADC is enabled so that EDMA controller resumes its transfer for primary ADC.

To summarise, the DSP core is interrupted every 500 microseconds by EDMA for the transfer completion with primary ADC for the last 16 sets of samples. In the transfer completion callbacks for primary ADC, DSP may switch the EDMA transfer context if the data is ready in secondary ADC.

CONCLUSION

The article discussed a promising approach to utilize the EDMA for servicing SPI peripheral based on the data ready signals from primary ADC and secondary ADCs. The data integrity of the data received from the ADCs with the discussed approach was verified. Clearly, the rich capabilities of EDMA is flexible enough to suit any data transfer even more complex than the data transfer discussed in this article.

REFERENCES

1. *TMS320C6748 Fixed and Floating point DSP (Rev G), Texas Instruments (June 2009-Revised January 2017). TI Literature number : SPRS590G*

2. *SPI Block Guide V03.06, Motorola, Inc (Revised: 04 Feb 2003)*

3. *TMS320C6748 Technical Reference Manual, Texas Instruments (April 2013-Revised September 2016). TI Literature number : SPRUH79C*

4. *TMS320C6748/46/42 and OMAP-L138 Processor Enhanced DMA Controller User's Guide, Texas Instruments (April 2010). TI Literature number : SPRUGP9B*

5. *EDMA3 User guide, Texas instruments (November 2014 - Document version 02.12.XX.XX)*

6. *PaRaM set configuration for primary ADC discussed in this article is inspired from the TI E2E forum discussion : https://bit.ly/3zO3U5i*

TISMO Embedding Excellence Copyright © 2021 Tismo Technology Solutions (P) Ltd.