

PCM Burst Mode for BelaSigna® 250



ON Semiconductor®

<http://onsemi.com>

Introduction

BelaSigna 250 includes a highly configurable I2S compatible pulse code modulation (PCM) interface that can be used to stream control, configuration or signal data into and out of the device. One of the advantages of this interface is the high communication speed that allows the interface to operate up to the system clock frequency. Additionally, you can configure the interface to operate in either full-duplex mode or half-duplex mode (either transmitting or receiving) because of the separate serial data lines.

Data transfers in BelaSigna 250 using the PCM interface can be divided into two groups, continuous transfer and Burst transfer. In this application note we will discuss the Burst transfer with the help of an application for the PCM interface. Burst transfers provide small blocks of data where the location and contents of every data word is important (ex. control code, data archiving). These transfers are limited to a maximum of 8 frames or 16 sub-frames, each frame being 32 bits long and sub-frames being 16 long. Continuous transfers provide a continuous data stream that is useful for audio processing applications where there is less concern over whether the first/last frame is correct. This type of transfer may also be used in longer control/data transfer applications by defining a synchronization frame or sub-frame, which indicates the start of a transfer.

Concept and Procedure

Existing documentation applies primarily to continuous mode transfers, but may be of assistance in explaining the behavior of and how to configure for Burst Mode transfers. The suggestions listed here provide additional recommendations that either add to or differ from those for continuous transfers in order to assist in the development of systems using burst transfers. All of these suggestions have been applied to a pair of sample applications that implement a PCM burst transfer transmitter and a PCM burst transfer receiver respectively.

For burst transfers the PCM configuration fields `PCM_FRAME_WIDTH` and `PCM_SUBFRAME` should be set to one of the following below during initialization:

`PCM_FRAME_WIDTH_SHORT`
and `PCM_SUBFRAME_DISABLE`
`PCM_FRAME_WIDTH_LONG`
and `PCM_SUBFRAME_ENABLE`

APPLICATION NOTE

Using one of these two settings will assist in synchronizing the transmitter and receiver sides of the PCM communications interface at the start of the first frame of each transmission (as opposed synchronizing the interface at a delay of up to one frame).

The length of a burst transmission in terms of the number of frames transmitted and received should be synchronized between the transmitter and receiver. In order to meet this requirement a user should select a burst transfer length and configure interrupts on the receive side to occur after every transmitted block. This helps to prevent the rising of the frame signal (indicating the completion of the last frame) after a transfer from resulting in additional visible data. Using burst transfers nominally precludes using the interface for full-duplex communications due to conflicting configuration requirements between a burst transfer transmitter and receiver (as the PCM interface should be disabled for a transmitter and should be enabled for a receiver). As an alternative to full-duplex transfers, a user could alternately use sequential unidirectional burst transfers.

The PCM burst transfer application is available from ON Semiconductor; please visit the extranet, <https://extranet.my.amis.com/> to download.

AND8342/D

Transmitting Device Only

PCM burst transmitter application, configures the device to operate as a PCM master sending data to another device. After startup, the application writes an incrementing sequence of data to the PCM buffer transmitting the generated data frames in bursts over the PCM interface. Each burst transfer is followed by a delay effectively resulting in a periodic transmission.

On the transmitting device, **the PCM interface must not be enabled**. Instead a user should use the behavior of the PCM interface to complete transmitting previously queued

data. By not enabling the interface the system prevents garbage data from being transmitted both before and after the actual data block is sent. The data to be transmitted over PCM should be written to the PCM buffer directly and then should manually update the PCM_TX_PTR_STATE to indicate how many frames need to be transmitted. This eliminates issues related to frame data shifting in the underlying buffer which may result in the corruption of the second frame of a transmission. Sample code has been provided below to further explain this mode.

// PCM Transmit Routine: Transmits FRAME_COUNT frames of incrementing data over the PCM interface. A second // copy of this same data is stored to the PCM temporary data buffer in X-Memory for testing purposes.

PCM_tx:

```
PUSH_6(R1, R4, R5, LC0, AE, AH)
```

```
// Setup by storing registers that will be  
// used in this function.
```

```
LDI R4, D_PCM_DATA0_INDEX0
```

```
// Setup R4 to point to the lower PCM data register  
// (D_PCM_DATA0).
```

```
LDI R1, XL_PCM_TEMP_DATA
```

```
// Setup R1 to point to the base of the temporary  
// PCM transmit data buffer
```

```
LD A, XL_PCM_DATA_COUNTER, X
```

```
// Clear A as the data to be transmitted will be dummy data  
// based on A.
```

```
LDSI LC0, (2*FRAME_COUNT - 1)
```

```
// Setup loop counter 0 for FRAME_COUNT cycles through the  
// following loop.
```

PCM_tx_LOOP:

```
INC A
```

```
// Increment A and write to the buffer starting at
```

```
LD (R4+), AH
```

```
// D_PCM_DATA0_INDEX0 (incrementing R4 to point at the  
// location for the next transfer).
```

```
LD (R1+), AH
```

```
DBNZ0 PCM_tx_LOOP
```

```
// Decrement the counter and loop back to PCM_tx_LOOP if  
// this counter is not zero.
```

```
LDI R5, D_PCM_CTRL
```

```
LDI (R5), 0x181E
```

```
LD XL_PCM_DATA_COUNTER, A, X
```

```
POP_6(R1, R4, R5, LC0, AE, AH)
```

```
// Cleanup and return.
```

```
RET
```

Receiving Device Only

PCM burst receiver application, configures the device to operate as a PCM slave receiving data (along with the frame and clock signal) from another device. After startup, this application waits for PCM interrupts at which point it reads the data received out into a temporary memory buffer. After each block transfer the device disables and re-enables the interface. In the case of an overflow on the PCM interface the device also resets the interface clearing out any pending interrupt events that may not have been cleared due to this error case.

The PCM interface needs be reset between transfers to prevent data received in future transfers being offset by an apparent “extra frame”. This extra frame, which typically contains all zeros (0x00000000) or all ones (0xFFFFFFFF), is caused by the rising edge of the frame signal which terminates the last normal frame. Interface may be reset by disabling and then re-enabling the PCM interface following

each burst transfer. The interface must be disabled for a minimum of one rising edge of the PCM clock.

Enabling the PCM interface during a transfer may result in the interface becoming offset on subsequent transfers. In order to reset the underlying state machine allowing recovery from this state the interface will need to be disabled during at least one PCM frame pulse. Check for an overflow by reading and clearing the PCM_RX_OVERFLOW bit, and resetting the field PCM_RX_PTR_STATE to 0xF after receiving the data from each burst transfer will also assist in maintaining the PCM interface. This will help recovery if the interface is enabled after at least one frame of a burst transfer has already been received. Further, if the system is disabled for a longer period of time then a single rising edge of the PCM clock the device may be given time to realign the complete transmission so that the receiving interface is aligned with the transmitting interface correctly. A sample code has been provided below to further explain this mode.

```
// PCM receive interrupt subroutine: Receive FRAME_COUNT frames of data over the PCM interface. This ISR
// expects that the PCM interface has been configured to with the appropriate PCM_RX_INT_EVERY_X define to // receive
// interrupts after each burst transfer.
```

```
PCM_rx_ISR:
```

```
    ISR_SAVE                // Setup the ISR and save registers that will be used in this ISR.
    PUSH_4(R1, R4, R5, LC0)
    LDI R4, D_PCM_CFG       // Setup pointers to the PCM registers.
    LDI R5, D_PCM_CTRL
    LD AH, (R5)             // Load PCM control and check for overflow on receive.
    TST AH, 4
    BRA PCM_rx_ISR_CONTINUE, Z
// Wait for any possible current transactions to complete, clear any pending interrupts and re-enable the
// interface (clearing any outstanding transfers).
    RES (R4), PCM_ENABLE_POS
    // Delay to ensure that the PCM interface is disabled for an appropriate amount of time. Nominally all that
    // is needed is for a single PCM clock rising edge to occur. This is option #1. In using option #1 when the
    // PCM interface is eventually re-enabled there still exists the possibility that the interface is still
    // receiving from the transaction that caused the overflow in the first place. Option #2: eliminates this
    // possibility by delaying the interface enable for the time to complete an entire transmission (for a PCM
    // clock speed of 128 kHz or higher a complete transmission will take less then 2 mili-seconds). This should
    // only be used if there is a longer time delay between transfers then the time needed to perform the
    // complete transmission.
    Sys_Delay(2)
    // Manually acknowledge the PCM interrupt so it doesn't get triggered again for this corrupt transaction.
    SET D_INT_STATUS, INT_PCM_POS
    SET (R4), PCM_ENABLE_POS // Re-enable the PCM interface and clear out any existing interface state.
    LDI (R5), 0x1F1F
    BRA PCM_rx_ISR_END
PCM_rx_ISR_CONTINUE:      // If an overflow wasn't encountered there should be FRAME_COUNT frames to be read.
    LDI R4, D_PCM_DATA0    // Setup R4 to point to the lower PCM data register (D_PCM_DATA0).
    LDI R1, XL_PCM_TEMP_DATA // Setup R1 to point to the base of the temporary PCM transmit data
    // buffer.
PCM_rx_ISR_LOOP:
    LD AH, (R4+)           // Store the data from D_PCM_DATA0 to the XL_PCM_TEMP_DATA buffer (incrementing
    LD (R1+), AH           // R4 to point at D_PCM_DATA1 to be ready for the next transfer).
```

AND8342/D

```
LD AH, (R4-) // Store the data from D_PCM_DATA1 to the XL_PCM_TEMP_DATA buffer
LD (R1+), AH // (decrementing R4 to point at D_PCM_DATA0 to be ready for the
// next transfer).

INC (R2) // Increment the received PCM frame counter.
LD AH, (R5) // Loop back to PCM_rx_ISR_LOOP if there is more data available
ANSI A, 0x08 // (if the buffer isn't empty).
CMSI A, 0x08
BRA PCM_rx_ISR_LOOP, NZ
LDI R4, D_PCM_CFG // Reset the PCM RX pointer by cycling the PCM interface
// configuration.


RES (R4), PCM_ENABLE_POS
// Similar to above, all that nominally is needed to reset the PCM RX pointer is for a single PCM clock
// rising edge to occur. However, since there should be a delay between burst transmissions take some time
// here and disable the interface to reduce the risk that if this transmission was offset (ie. by starting
// the interface in the middle of a burst transaction) that the interface will overflow the RX data
// register.
Sys_Delay(2)
LDI (R5), 0x1F0F // Clear out the current PCM state.
SET (R4), PCM_ENABLE_POS // Re-enable the PCM interface.
PCM_rx_END:
POP_4(R1, R4, R5, LC0) // Cleanup (including enabling interrupts) and return.
ISR_RESTORE
Enable_Int
RET
```

Company or Product Inquiries

For more information about ON Semiconductor, our technology and our products, visit our website at: <http://www.onsemi.com>.

For technical support, email: dsp.support@onsemi.com.

BelaSigna is a registered trademark of Semiconductor Components Industries, LLC (SCILLC).

ON Semiconductor and  are registered trademarks of Semiconductor Components Industries, LLC (SCILLC). SCILLC reserves the right to make changes without further notice to any products herein. SCILLC makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does SCILLC assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation special, consequential or incidental damages. "Typical" parameters which may be provided in SCILLC data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. SCILLC does not convey any license under its patent rights nor the rights of others. SCILLC products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the SCILLC product could create a situation where personal injury or death may occur. Should Buyer purchase or use SCILLC products for any such unintended or unauthorized application, Buyer shall indemnify and hold SCILLC and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that SCILLC was negligent regarding the design or manufacture of the part. SCILLC is an Equal Opportunity/Affirmative Action Employer. This literature is subject to all applicable copyright laws and is not for resale in any manner.

PUBLICATION ORDERING INFORMATION

LITERATURE FULFILLMENT:
Literature Distribution Center for ON Semiconductor
P.O. Box 5163, Denver, Colorado 80217 USA
Phone: 303-675-2175 or 800-344-3860 Toll Free USA/Canada
Fax: 303-675-2176 or 800-344-3867 Toll Free USA/Canada
Email: orderlit@onsemi.com

N. American Technical Support: 800-282-9855 Toll Free
USA/Canada
Europe, Middle East and Africa Technical Support:
Phone: 421 33 790 2910
Japan Customer Focus Center
Phone: 81-3-5773-3850

ON Semiconductor Website: www.onsemi.com
Order Literature: <http://www.onsemi.com/orderlit>
For additional information, please contact your local
Sales Representative