

Implementing AC-link With ESAI

by

Ilan Naslavsky and
Leonid Smolyansky

Revision 1.1


Motorola, Incorporated
Semiconductor Products Sector
6501 William Cannon Drive West
Austin, TX 78735-8598

Mfax and OnCE are trademarks of Motorola, Inc.

AC-link is a digital serial interface protocol that is documented in the *Audio Codec '97 Component Specification*. Revision 1.03 (September 1996) and 2.0 (September 1997). Primary Developers: Analog Devices. Creative Labs, Inc. Intel Corporation. National Semiconductor Yamaha Corporation. Copyright 1997, Intel Corporation.

© MOTOROLA INC., 1998

Order by this number: APR37/D

Motorola reserves the right to make changes without further notice to any products herein to improve reliability, function, or design. Motorola does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola 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 Motorola was negligent regarding the design or manufacture of the part. Motorola and  are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

CONTENTS

1	Introduction	
1.1	Scope	1-1
1.2	AC '97 Digital Controller	1-1
1.3	The AC '97 Digital Serial Interface (AC-link)	1-2
2	Physical Connection	
2.1	AC-link Lines	2-1
2.2	ESAI Pin Connections	2-2
3	Data Flow	
3.1	Output Data Streams	3-1
3.2	Input Data Streams	3-1
3.3	Data Flow Model	3-1
3.4	FIFOS	3-2
3.5	Workflow	3-4
4	System Implementation	
4.1	Resources	4-1
4.2	AC-link Application	4-1
4.3	ESAI Usage	4-3
4.4	DMA Usage	4-3
4.4.1	Transmission	4-3
4.4.2	Reception	4-4
4.5	Timer Pulse Modulation	4-4
4.5.1	Cold Reset	4-4
4.5.2	Warm Reset	4-5
4.5.3	ATE Test Mode	4-5
4.5.4	Vendor-Specific Test Mode	4-6
4.6	Interrupts	4-6
4.6.1	Receiver Last Slot Interrupt	4-6
4.6.2	Timer Compare Interrupt	4-7
4.6.3	DMA Channel 0 Interrupt	4-7
4.7	Assumptions And Recommendations	4-8

5	ESAI, DMA, and Timer Configuration	
5.1	ESAI Configuration	5-1
5.1.1	Transmit Clock Control Register (TCCR)	5-1
5.1.2	Transmit Control Register (TCR)	5-2
5.1.3	Receive Clock Control Register (RCCR)	5-3
5.1.4	Receive Control Register (RCR)	5-4
5.1.5	Common Control Register (SAICR)	5-5
5.1.6	Port Control, Direction and Data Registers	5-5
5.2	DMA Configuration - Channel 0	5-7
5.2.1	DMA Control Register for Channel 0 (DCR0)	5-7
5.2.2	DMA Counter Register for Channel 0 (DCO0)	5-8
5.2.3	DMA Offset Registers (DOR0 and DOR1)	5-8
5.2.4	DMA Source Address Register for Channel 0 (DSR0)	5-8
5.2.5	DMA Destination Address Register for Channel 0 (DDR0)	5-8
5.3	DMA Configuration - Channel 1	5-8
5.3.1	DMA Control Register for Channel 1 (DCR1)	5-8
5.3.2	DMA Counter Register for Channel 1(DCO1)	5-9
5.3.3	DMA Offset Registers (DOR2 and DOR3)	5-9
5.3.4	DMA Source Address Register for Channel 1(DSR1)	5-9
5.3.5	DMA Destination Address Register for Channel 1 (DDR1)	5-9
5.4	Timer Configuration	5-9
5.4.1	Timer Control/Status Register (TCSR)	5-10
5.4.2	Timer Load Register (TLR)	5-10
5.4.3	Timer Compare Register (TCPR)	5-10

APPENDIXES:

Appendix A Equates and Code Routines

Appendix B References

FIGURES

Figure 2-1	ESAI - AC '97 Connection: The AC_Link	2-2
Figure 3-1	Audio Application-CODEC Interaction Model	3-3
Figure 3-1	AC-link Application Data Flow	3-4
Figure 4-1	AC-Link Implementation.	4-2
Figure 4-1	AC '97 Cold Reset.	4-5
Figure 4-1	AC '97 Warm Reset	4-5
Figure 4-1	AC '97 ATE Test Mode.	4-6
Figure 4-1	Buffer Swapping.	4-7
Figure 5-1	Transmit Clock Control Register (TCCR).	5-1
Figure 5-2	Transmit Control Register (TCR)	5-2
Figure 5-3	Receive Clock Control Register (RCCR)	5-3
Figure 5-4	Receive Control Register (RCR)	5-4
Figure 5-5	Common Control Register (SAICR)	5-5

TABLES

Table 2-1	AC-link Signals	2-1
Table 3-1	AC-link Application Workflow	3-5
Table 5-1	PCR, PRR and PDR Values	5-6
Table A-1	Data X-Memory Usage	A-1
Table A-2	Data Y-Memory Usage.....	A-2
Table A-3	FIFO Table, Initial Status	A-2

1 Introduction

The Enhanced Serial Audio Interface (ESAI) of the DSP56300 Family provides full capabilities for interfacing with a general AC '97 CODEC through an AC '97 Digital Serial Interface, the AC-link.

The DSP56362 is the first DSP56300 Family derivative that presents the ESAI and is readily enabled to run this application. **Appendix B** lists reference materials pertaining to the DSP56300 Family, the DSP56362, and the *Audio CODEC '97 Component Specification*.

1.1 Scope

This application report describes how to implement an AUDIO CODEC '97 Digital Serial Interface (AC-link) using the ESAI. It is recommended for developers who have previous knowledge of Motorola's DSP56300 Family of Digital Signal Processors and of the AC '97 Component Specification.

Section 2 describes the physical connection between ESAI and an AC '97 CODEC. **Section 3** describes the Data-Flow model. **Section 4** delineates the system concept implemented in the AC-link application. **Section 5** details the configuration of DSP56300 Family resources used in this application. **Appendix A** shows the assembly code and equates of the application. **Appendix B** lists relevant reference information.

1.2 AC '97 Digital Controller

The AC '97 Digital Controller runs one or more audio applications and exchanges data with its analog counterpart, the AC '97 CODEC, through the AC-link. The DSP56300 Family may implement the AC '97 Digital Controller functionality.

1.3 The AC '97 Digital Serial Interface (AC-link)

AC '97 Digital Serial Interface, the AC-link, provides point-to-point, two-way communication between an AC '97 Digital Controller and an AC '97 CODEC. Data exchange is performed over two Time Division Multiplexed (TDM) one-wire channels, one in each direction, supported by two additional timing signals and a hardware reset line.

This report presents an AC-link application for a DSP56300 Family derivative. The application performs all specified AC-link tasks, providing audio applications with a straightforward and modular interface.

2 Physical Connection

This section describes the physical connection between the Enhanced Serial Audio Interface (ESAI) and a generic AC '97-compatible CODEC. The AC '97 DSI or AC-link implementation suggested in this application report uses seven ESAI pins, providing the whole AC-link functionality. This section describes this glueless connection.

2.1 AC-link Lines

Five lines make up the AC-link. On the DSP side, all of the lines are connected to ESAI pins. In some cases, these lines are programmed as GPIO pins in order to accomplish a particular signal's characteristic. **Table 2-1** briefly describes their function and direction related to the ESAI.

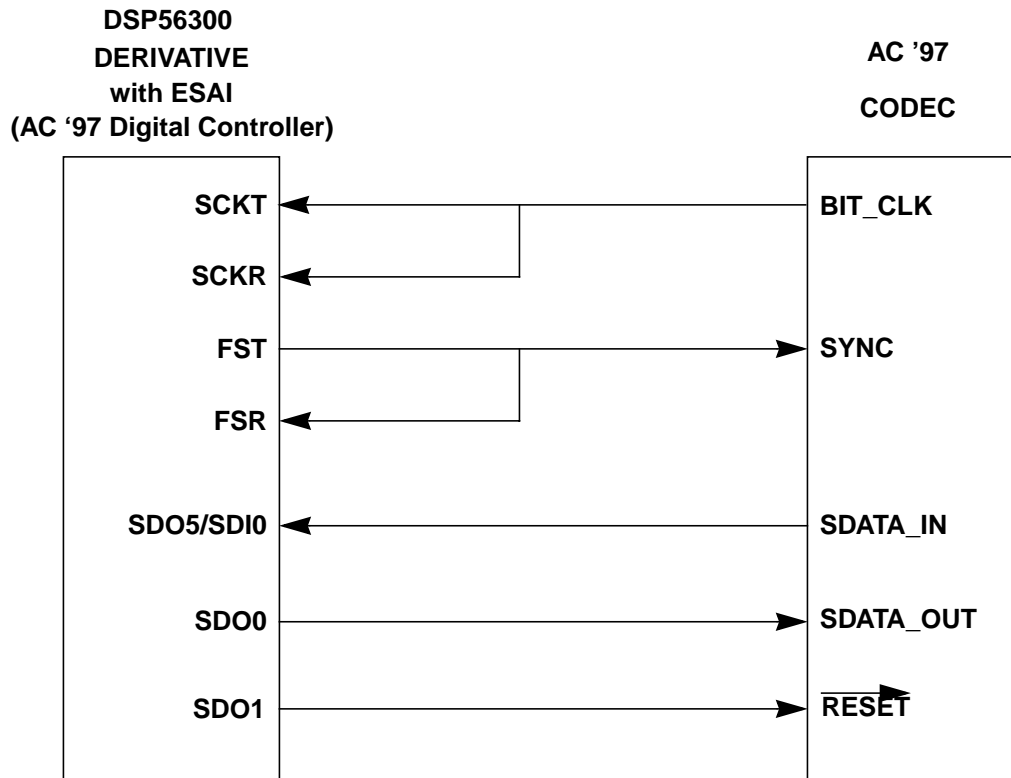
Table 2-1 AC-link Signals

Signal	Description	Direction
BIT_CLK	Serial bit-clock @12.288MHz	input
SYNC	Frame Sync signal @48kHz; high for 16 BIT_CLK periods, low for 240	output
SDATA_IN	Serial Data Stream	input
SDATA_OUT	Serial Data Stream	output
$\overline{\text{RESET}}$	Asynchronous Hardware Reset	output

The AC-link lines should be connected to the CODEC according to specification.

2.2 ESAI Pin Connections

Figure 2-1 outlines the ESAI-CODEC pin connections.



AA1559

Figure 2-1 ESAI - AC '97 Connection: The AC_Link

As **Figure 2-1** shows, the AC '97 controller drives an asynchronous hardware reset signal through the $\overline{\text{RESET}}$ line. The ESAI P10 pin (SDO1) is configured as a GPIO pin and driven low with each $\overline{\text{RESET}}$ signal. **Section 4** of this application report gives a more detailed description of the AC '97 reset signaling implementation.

A Serial Data Output line, SDATA_OUT, is provided at the ESAI P11 pin and is configured as the ESAI #0 transmitter output (SDO0). Eventually, this pin may be temporarily reconfigured as GPIO, permitting the DSP to signal the AC '97 CODEC to enter ATE Test Mode.

The Frame Sync signal (SYNC) is generated by the transmitter side of ESAI and is output through its P4 pin, configured as Frame Sync for Transmitter (FST). In order to comply with the timing constraints of the AC '97 and DSP56300 Family, the ESAI transmitter and receiver are programmed to work asynchronously. Therefore, Frame Sync must be fed for the receiver side as well. Thus, a feedback connection between FST, output configured, and the ESAI P1 pin (Frame Sync for Receiver - FSR), input configured, is needed. This pin can be reconfigured temporarily as GPIO, permitting the DSP to signal the AC '97 CODEC to enter the Vendor-Specific Test mode.

Similarly, the serial data clock provided by the AC '97 CODEC, (BIT_CLK), must be fed for both the transmitter and receiver portions of ESAI. The required connection links the BIT_CLK line through the ESAI P0 and P3 pins—Receiver Serial Clock (SCKR) and Transmitter Serial Clock (SCKT), respectively—are both configured as input.

The Serial Data Input line, SDATA_IN, is supplied by the AC '97 CODEC, driving the ESAI P6 Serial Transmit/Receive Data pin, SDO5/SDI0, and configured as the ESAI #0 receiver input.

3 Data Flow

This section describes the AC-link application data-flow model, which is an implementation with a five-slot-output and a six-slot-input. This modular concept enables a smooth upgrade for a larger number of data streams.

Two Time-Division-Multiplexed (TDM) one-wire data channels constitute the AC-link data path, one channel for each direction. Each channel can transfer up to 12 data streams, plus a leading validation stream.

3.1 Output Data Streams

The AC-link implementation defines three output data streams that demand five slots of the TDM output channel:

- **PCM Playback:** two-channel composite PCM stream, 2 slots
- **Control:** control register write port, 2 slots
- **Modem Line CODEC Output:** modem line CODEC DAC input stream, 1 slot

3.2 Input Data Streams

In the input direction, the AC-link implementation defines four data streams that demand six slots of the TDM input channel:

- **PCM Record:** two-channel composite PCM stream, 2 slots
- **Status:** control register read port, 2 slots
- **Modem Line CODEC Input:** modem line CODEC ADC output stream, 1 slot
- **Dedicated Microphone Input:** dedicated microphone input stream, 1 slot

3.3 Data Flow Model

The AC '97 Digital Controller sees the AC-link data path as two sets of software implemented as First-In-First-Out queues (FIFOs). Each set reflects one direction of the link, being composed by one FIFO for every data stream, summing up three output FIFOs and four input FIFOs in this particular implementation.

When the Digital Controller has data from any of the three output streams to send to the CODEC, it writes it to the corresponding FIFO. Similarly, each time data from any of the four available input data streams is to be processed, the application reads it from the respective FIFO.

The data transfer path between the FIFOs and the CODEC is transparent to the audio application. This path is achieved through the AC-link application described in this section and the resources and routines detailed in **Section 4** and **Section 5**.

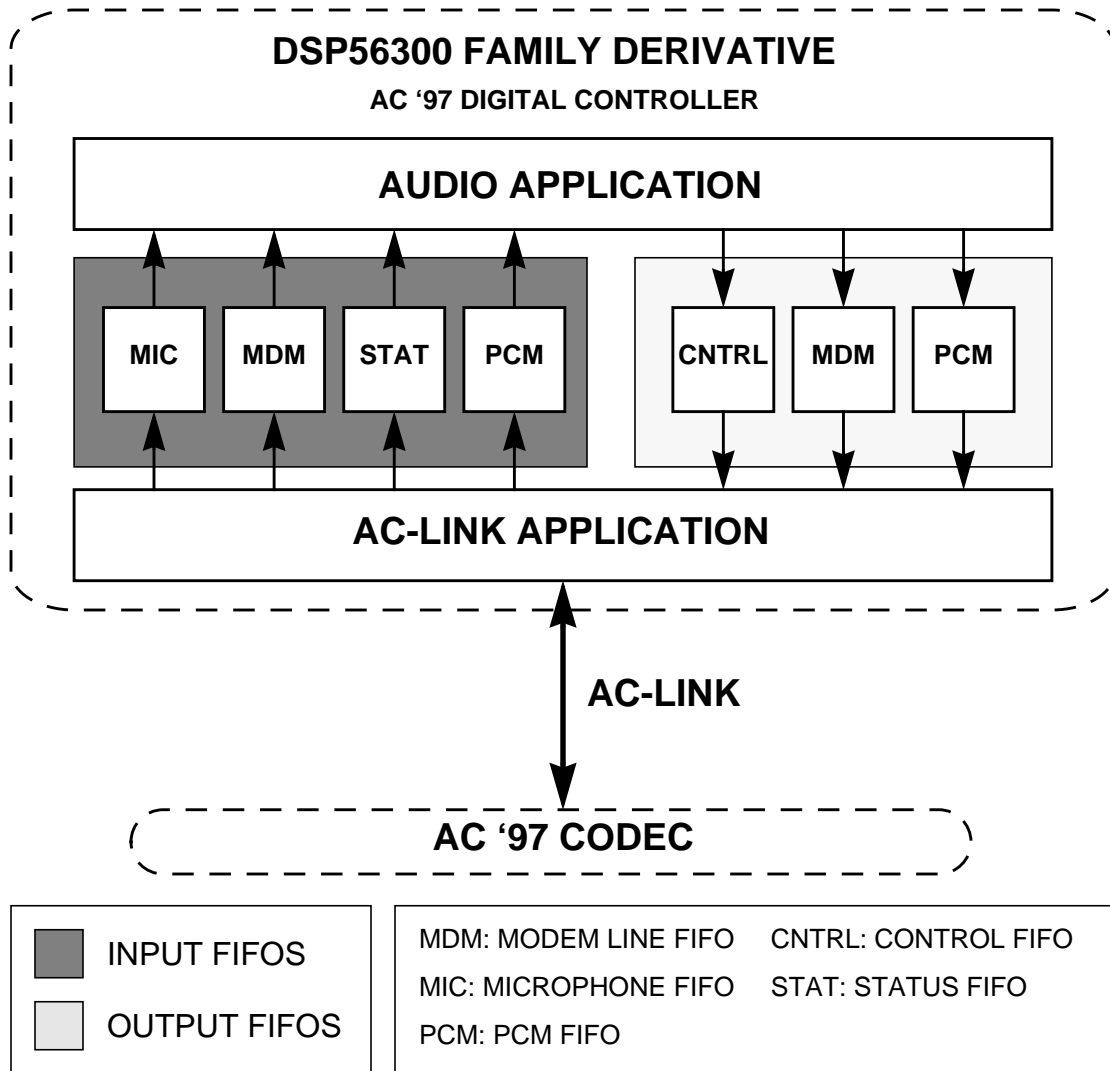
3.4 FIFOS

Every input and output data stream has a corresponding FIFO. The lengths of the FIFOs depend on application throughput and the demand of the corresponding type of data and are independently programmable.

Output FIFOs are read by the AC-link application once per AC '97 Audio Frame, i.e. 48K times per second. Data read from the FIFO is sent through the AC-link. The audio application tailors its data throughput to both the FIFOs capacity and the AC-link transfer frequency, addressing the boundary case of full FIFOs to avoid FIFO overflow. Empty FIFOs are handled by the AC-link application as if their data is not valid, although different approaches can be used with minimal code modification.

In the opposite direction, the AC-link application writes to input FIFOs at the same frequency, 48K times per second. It is the audio application's responsibility to keep input FIFOs under a not-full status, reading data at a compatible frequency. The AC-link Application enters a *crash* routine when any FIFO is full at the moment it tries to write to it. The crash routine procedure should be determined by the Audio Application specification.

Figure 3-1 describes the audio application-CODEC interaction model.

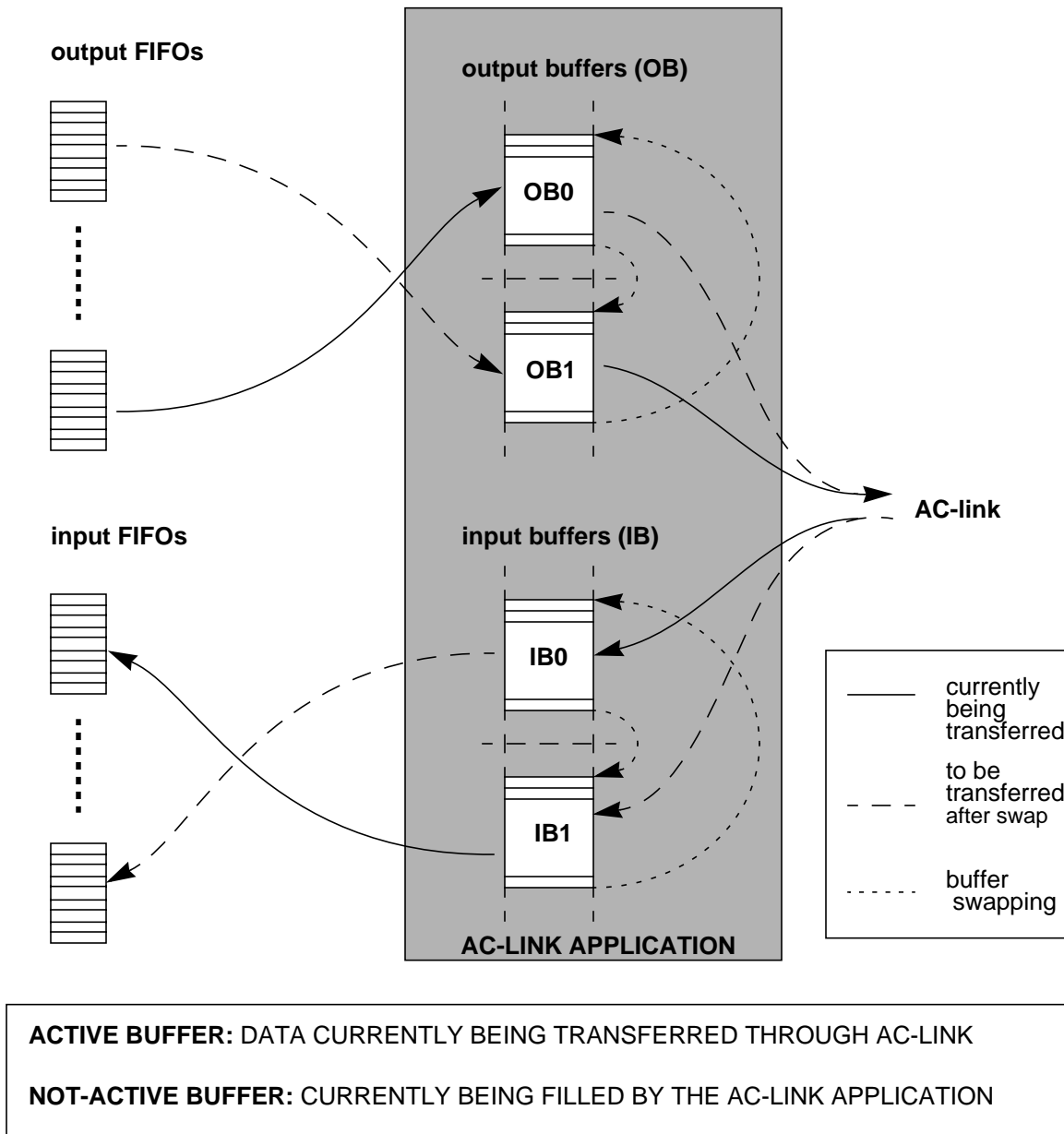


AA1560

Figure 3-1 Audio Application-CODEC Interaction Model

3.5 Workflow

The AC-link application handles data flow between an audio application running on a DSP56300 Derivative and an AC '97 CODEC. It prepares and validates data to be transmitted through the AC-link. A pair of output buffers and a pair of input buffers are allocated for transmission and reception, respectively.



AA1561

Figure 3-1 AC-link Application Data Flow

As **Figure 3-1** shows, during every AC '97 Audio Frame one of the output buffers is *active*, meaning its data is currently being transmitted. At the same time, the second buffer is not-active, and is fed by the AC-link application with the output FIFOs data. Slot validation—that is, slot 0 determination (Audio Frame TAG)—is accomplished concurrently. In the course of the subsequent audio frame, the buffers roles switch.

A symmetrical scheme is adopted in the reception section. The *active* buffer is filled with AC-link incoming data, while the data of the not-active buffer is transferred to input FIFOs. Reception buffers alternate active and not-active states in the same way as the transmitter buffers.

The first and second transmitted frames data after the AC-link reset (F_0 , F_1) are entirely invalidated in order to enable the AC-link Application to enter the steady state. **Table 3-1** outlines the buffer swapping procedure.

Table 3-1 AC-link Application Workflow

Frame	Transmission Section		Reception	Section
	Active Buffer (Buffer -> ESAI)	Not (FIFO -> Buffer)	Active Buffer (ESAI -> Buffer)	Not Active Buffer (Buffer -> FIFO)
F_0	B1(-1) (whole buffer invalidated)	- (no RLS int. occurred)	B1(0)	- (no RLS int. occurred)
F_1	B0 (whole buffer invali- dated)	B1(1)	B0(1)	B1(0)
F_2	B1(1) (first valid buffer transmitted)	B0(2)	B1(2)	B0(1)
...
F_{t-2}	B1(t-3)	B0(t-2)	B1(t-2)	B0(t-3)
F_{t-1}	B0(t-2)	B1(t-1)	B0(t-1)	B1(t-2)
F_t	B1(t-1)	B0(t)	B1(t)	B0(t-1)
F_{t+1}	B0(t)	B1(t+1)	B0(t+1)	B1(t)
F_{t+2}	B1(t+1)	B0(t+2)	B1(t+2)	B0(t+1)
...

4 System Implementation

This section describes the system concept implemented in this application. The AC-link application is implemented with a set of DSP56300 derivative routines that supports ESAI in its AC '97 mode, handling several on-chip resources.

4.1 Resources

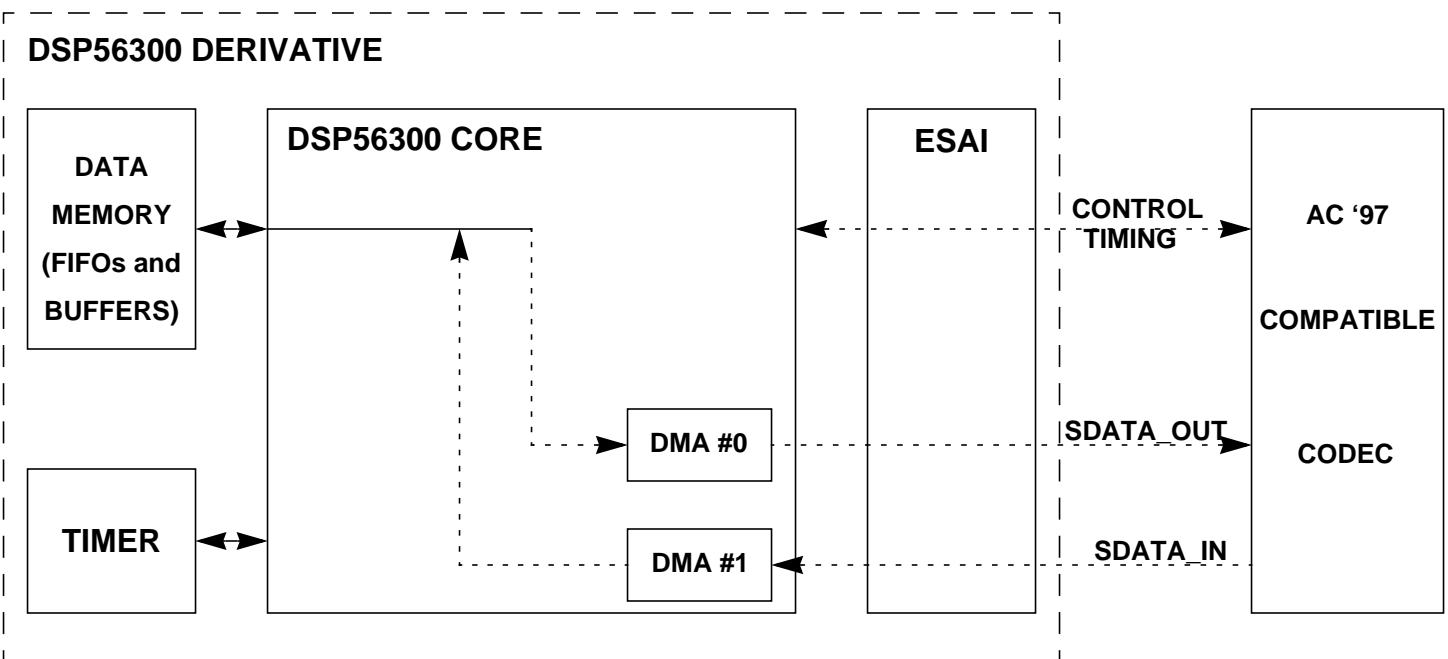
The implemented application uses the following resources of the DSP56300 derivative:

- 1 ESAI transmitter
- 1 ESAI receiver
- 2 Direct Memory Access (DMA) channels
- 1 timer module
- Variable quantity of internal data memory for buffers and FIFOs

4.2 AC-link Application

The AC-link application implements an AC '97 Digital Interface using ESAI on its AC '97 mode and is supported by two DMA channels and a timer module. **Figure 4-1** presents the application block diagram.

The ESAI performs AC-link Audio Frame multiplexing and demultiplexing, as well as timing and control functions. DMA channels transfer data between the ESAI and the audio application running on the DSP56300 derivative. A timer module is necessary in AC-link special signaling.



AA1562

Figure 4-1 AC-link Implementation

4.3 ESAI Usage

The ESAI is programmed to transmit with TX #0 and receive with RX #0 in asynchronous mode:

- with an external clock
- under AC '97 mode
- with 12×20 -bit slots per frame plus a leading 16-bit slot (slot 0, Tag Phase).

The external clock for both the receiver and transmitter is the BIT_CLK signal generated by the AC '97 CODEC. The transmitter section generates the frame synchronization signal (FST/SYNC) supplied to the CODEC and to the receiver section. FST is driven with the falling edge of BIT_CLK, one bit before the beginning of the first slot in the frame.

In AC '97 mode, frame sync is generated high for the first word, a 16-bit word corresponding to the AC '97 tag phase, and low for 12 words, each being one of the 20-bit slots of the AC '97 data phase.

Data is driven out (SDO0/SDATA_OUT) at the falling edge of BIT_CLK, being sampled by the CODEC on the *next* falling edge. Input data (SDO5/SDI0/SDATA_IN) is driven by the CODEC with the rising edge of BIT_CLK and sampled by the receiver on the subsequent falling edge.

4.4 DMA Usage

The DMA performs data transfer between the Audio application and ESAI using one channel for each direction. DMA channel #0 transfers output data, and channel #1 transfers input data.

4.4.1 Transmission

The ESAI specification requires that the first transmitted word be written to the ESAI transmitter register by a DSP Core move *before* transmitter enabling. Accordingly, the first transmitted audio frame requires a DMA configuration in which only the data phase is transferred from the active output buffer to the ESAI. The tag phase (slot 0) is programmed as first data for the ESAI transmitter.

From the second frame on, DMA is re-programmed to work in a continuous mode, servicing the ESAI transmitter upon request and transferring both tag and data

phases. The current active output buffer is always being accessed by DMA, while the not-active output buffer is updated and validated by the core. In steady state, DMA demands no interrupt servicing, synchronously swapping output buffers without core interference.

4.4.2 Reception

On the reception side, DMA is programmed to work in a continuous mode from the first frame, transferring both tag and data phases from the ESAI receiver register to the active input buffer and servicing the ESAI receiver upon request. The current active input buffer is always being accessed by DMA while the core transfers the not-active input buffer to the FIFOs. DMA demands no interrupt servicing and synchronously swaps input buffers without core interference.

4.5 Timer Pulse Modulation

Cold Reset, Warm Reset, ATE Test Mode and Vendor-Specific Test Mode signaling are implemented through the ESAI pins, re-programmed as GPIO. Specific signal generation is attained with the timer module.

4.5.1 Cold Reset

Cold Reset pulse is achieved by bringing the ESAI P10 pin ($\overline{\text{SDO1/RESET}}$) low for 1.0 μs or more. This pin remains constantly configured as a GPIO output, as discussed in **Section 2**.

The timer module is programmed to count a number of internal clocks sufficient to produce the specified pulse ($> 1.0 \mu\text{s}$). The pin is driven low with timer enabling; upon a Timer Compare Interrupt that corresponds to the end of counting, the pin is brought back high.

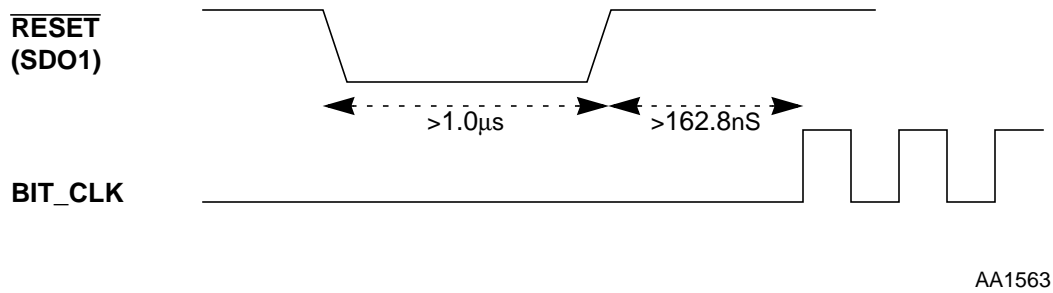


Figure 4-1 AC '97 Cold Reset

4.5.2 Warm Reset

A similar pulse generation mechanism, supported by the timer module, is used for Warm Reset. This time the timer module is programmed to count to a typical $1.3 \mu\text{s}$. Re-programming the ESAI P4 pin (FST/SYNC) as GPIO output allows a high pulse to be produced on this pin. The pin is driven high with timer enabling. Upon a timer Compare Interrupt that corresponds with the end of counting, the pin is brought back low.

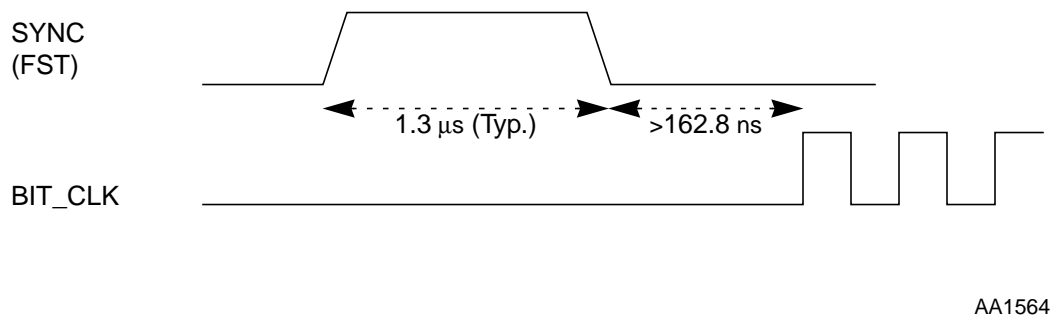
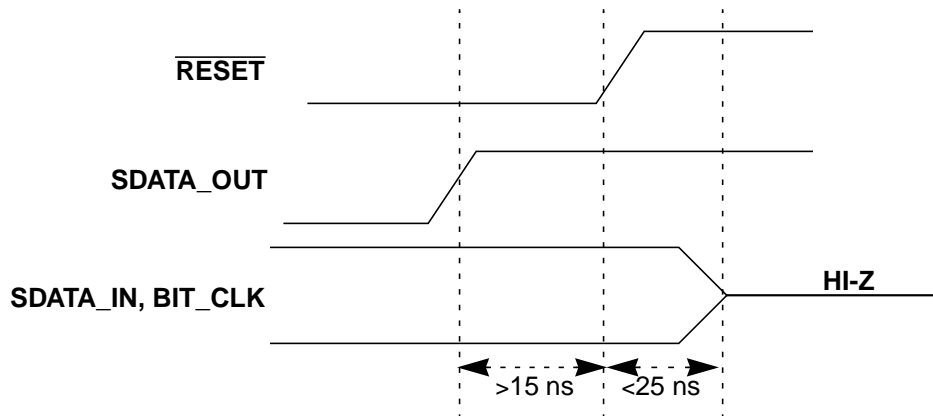


Figure 4-1 AC '97 Warm Reset

4.5.3 ATE Test Mode

ATE Test Mode signaling uses the ESAI P11 pin (SDO0/SDATA_OUT). SDATA_OUT must be sampled high at the trailing edge of the $\overline{\text{RESET}}$ signal. This signaling is achieved by configuring the ESAI P11 pin as GPIO output and bringing it high with timer enabling for reset pulse counting. Upon a Timer Compare Interrupt and subsequent deassertion of $\overline{\text{RESET}}$, the pin is kept high.



AA1565

Figure 4-1 AC '97 ATE Test Mode

4.5.4 Vendor-Specific Test Mode

Vendor-Specific Test Mode is implemented equivalently to ATE Test Mode. The ESAI P4 (FST/SYNC) pin is used instead of P11.

4.6 Interrupts

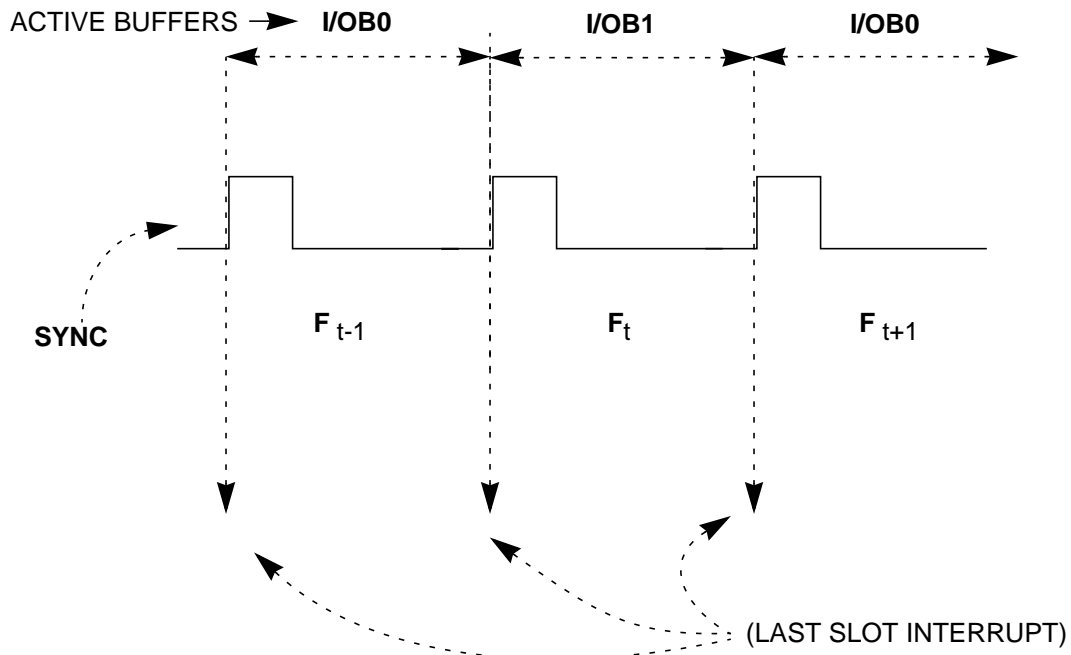
Some of the AC-link application functions use program interrupts. The following paragraphs describe their usage.

4.6.1 Receiver Last Slot Interrupt

Not-Active/Active buffer swapping occurs once every Audio Frame, at the ESAI Receiver Last Slot Interrupt (RLSI). This interrupt occurs at the end of the last slot of frame F_{t-1} when the ESAI finishes transmitting the last word of the output buffer and receiving the last word for input buffer, respectively—for example, OB0 and IB0. During frame F_t , the AC-link Application continues to supply data to ESAI from the OB1 transmission buffer, while transferring received data from ESAI to IB1 through DMA transfers.

During the interrupt at frame F_t , active buffers are switched from I/OB0 to I/OB1 and *vice versa*. Then the AC-link application fills up the not-active transmission buffer OB0 (in case any of the output FIFOs is not empty) validating the

corresponding valid-bits of the OB0 TAG (slot 0). The new OB0 buffer data is transmitted next time OB0 is active, i.e. during F_{t+1} . Similarly, input FIFOs are written with data from valid slots received during F_{t-1} and allocated at IB0.



NOTE: Timing representation out of scale.

AA1566

Figure 4-1 Buffer Swapping

4.6.2 Timer Compare Interrupt

A Timer Compare interrupt occurs when the timer counter matches the compare register and permits the completion of the AC '97 reset and test signaling.

4.6.3 DMA Channel 0 Interrupt

Because of the special transmission at the first frame, the DMA Channel 0 interrupt is needed to enable DMA re-programming to work in a continuous mode. This re-programming includes disabling this interrupt.

4.7 Assumptions And Recommendations

The audio application developer should take into account some assumptions and recommendations made for this implementation:

- Once an RLSI interrupt occurs it must be fully serviced before the next occurrence of the same interrupt. This guarantees the completion of buffer swap, including data update and validation, avoiding DMA switching access to an out-of-date buffer. Therefore, a coherent Interrupt Priority Level setting is required.
- The active buffer and not-active buffer should be located in different RAM-memory modules (256 words module) in order to avoid CORE-DMA contention.
- The FIFOs and Buffers base addresses should comply with modulo addressing conditions in order to allow implemented routines to work properly.
- A case might exist when an external buffer is needed in order to delay the SDATA_IN signal. In the *Audio Codec '97 Component Specification*, data hold-time (both SDATA_IN and SDATA_OUT) from the falling edge of BIT_CLK is stated as 5 nsec, measured from the beginning of the edge. If BIT_CLK fall time is to be considered in its worst case, 6 nsec, there is an effective -1 nsec hold-time, meaning data are not stable in the falling edge. Refer to the data sheet of the particular AC'97 device, or contact the vendor in order to decide on external buffer needs.

5 ESAI, DMA, and Timer Configuration

This section details how the resources to be used, ESAI, DMA, and Timer, are configured.

5.1 ESAI Configuration

The following paragraphs describe ESAI configuration and control register programming.

5.1.1 Transmit Clock Control Register (TCCR)

11	10	9	8	7	6	5	4	3	2	1	0
TDC2	TDC1	TDC0	TPSR	TPM7	TPM6	TPM5	TPM4	TPM3	TPM2	TPM1	TPM0
1	0	0	0	0	0	0	0	0	0	0	0

23	22	21	20	19	18	17	16	15	14	13	12
THCKD	TFSD	TCKD	THCKP	TFSP	TCKP	TFP3	TFP2	TFP1	TFP0	TDC4	TDC3
0	1	0	0	0	1	0	0	0	0	0	1

AA1567

Figure 5-1 Transmit Clock Control Register (TCCR)

The TCCR is initialized with a value of **\$441800**, corresponding to the following configuration:

- TPM7-TPM0, TPSR: no prescale is used, so Prescale Modulus Select and Prescale Range bits are irrelevant.
- TDC4-TDC0: Frame Rate Divider Control bits are initialized with 12 (\$c), the AC '97 number of slots per frame, minus one.
- TFP3-TFP0: High-Frequency Clock Divider Control bits are irrelevant since the bit clock is externally generated and the high speed clock is not used.
- TCKP: data out and frame sync are driven with the falling edge of the transmit clock, so the receiver Clock Polarity bit is set.
- TFSP: receiver Frame Sync Polarity is positive, then TFSP is cleared.

- THCKP: the Transmitter High-Speed Clock Polarity bit is irrelevant for this model, since the high-speed clock is not used.
- TCKD: the clock source is external, thus a cleared Clock Source Direction bit.
- TFSD: the FST/SYNC pin is output, so the Frame Sync Signal Direction bit is set.
- THCKD: High-Frequency Clock Direction is irrelevant in this application.

5.1.2 Transmit Control Register (TCR)

11	10	9	8	7	6	5	4	3	2	1	0
TSWS1	TSWS0	TMOD1	TMOD0	TWA	TSHFD	TE5	TE4	TE3	TE2	TE1	TE0
1	1	1	1	0	0	0	0	0	0	0	0/1

23	22	21	20	19	18	17	16	15	14	13	12
TLIE	TIE	TEDIE	TEIE				TFSR	TFSL	TSWS4	TSWS3	TSWS2
0	0	0	0	0	0	0	1	0	0	0	0

AA1568

Figure 5-2 Transmit Control Register (TCR)

The TCR is initially configured with a value of **\$010f00**. With transmitter #0 enabling, the register value is updated to **\$010f01**. This program corresponds to the following configuration:

- TE5-TE1: Transmitter #5 to #1 Enable bits are always cleared since none of these transmitters are used.
- TE0: the Transmitter #0 Enable bit is zero for initial programming and is set to enable this transmitter.
- TSHFD: the Transmitter Shift Direction bit is cleared so that data is shifted out Most Significant Bit first.
- TWA: Transmitter Word Alignment Control is irrelevant to AC '97 mode since word and slot length are equal.
- TMOD1-TMOD0: AC '97 mode is chosen through Transmitter Network Mode Control bits

- TSW4-TSW0: Transmitter Slot and Word Length Select bits values configure a 20-bits slot and a 20-bits word
- TFSL: Transmitter Frame Sync Length bit is zero, for a word-length frame sync
- TFSR: Transmitter Frame Sync Relative Timing bit is set so that frame sync will occur one serial clock cycle earlier
- TEIE, TEDIE, TIE, TLIE: no transmitter interrupt is enabled

5.1.3 Receive Clock Control Register (RCCR)

11	10	9	8	7	6	5	4	3	2	1	0
RDC2	RDC1	RDC0	RPSR	RPM7	RPM6	RPM5	RPM4	RPM3	RPM2	RPM1	RPM0
1	0	0	0	0	0	0	0	0	0	0	0
23	22	21	20	19	18	17	16	15	14	13	12
RHCKD	RFSD	RCKD	RHCKP	RFSP	RCKP	RFP3	RFP2	RFP1	RFP0	RDC4	RDC3
0	0	0	0	0	0	0	0	0	0	0	1
											AA1569

Figure 5-3 Receive Clock Control Register (RCCR)

RCCR is initialized with a value of **\$001800**, corresponding to the following configuration:

- RPM7-RPM0, RPSR: no prescale is used, so the Prescale Modulus Select and Prescale Range bits are irrelevant.
- RDC4-RDC0: the Frame Rate Divider Control bits are initialized with 12 (SC), the AC '97 number of slots per frame, minus one.
- RFP3-RFP0: the High-Frequency Clock Divider Control bits are irrelevant since the bit clock is externally generated and high speed clock is not used.
- RCKP: data out and frame sync are driven with the rising edge of the receive clock, so the Receiver Clock Polarity bit is cleared.
- RFSP: When Receiver Frame Sync Polarity is positive, RFSP is cleared.

- RHCKP: the Receiver High-Speed Clock Polarity bit is irrelevant for this model since the high-speed clock is not used.
- RCKD: The clock source is external, thus a cleared Clock Source Direction bit.
- RFSD: The RST pin is input, so the Frame Sync Signal Direction bit is cleared.
- RHCKD: High-Frequency Clock Direction is irrelevant in this application.

5.1.4 Receive Control Register (RCR)

11	10	9	8	7	6	5	4	3	2	1	0
RSWS1	RSWS0	RMOD1	RMOD0	RWA	RSHFD			RE3	RE2	RE1	RE0
1	1	1	1	0	0	0	0	0	0	0	0/1

23	22	21	20	19	18	17	16	15	14	13	12
RLIE	RIE	REDIE	REIE				RFSR	RFSL	RSWS4	RSWS3	RSWS2
1	0	0	0	0	0	0	1	0	0	0	0

AA1570

Figure 5-4 Receive Control Register (RCR)

RCR is initially configured with a value of **\$810f00**. With receiver #0 enabling, the register value is updated to **\$810f01**. This program corresponds to the following configuration:

- RE3-RE1: the Receiver #3 to #1 Enable bits are always cleared since none of these receivers are used.
- RE0: the Receiver #0 Enable bit is zero for initial programming, and is set to enable this receiver.
- RSHFD: the Receiver Shift Direction bit is cleared so that data is shifted in Most Significant Bit first.
- RWA: Receiver Word Alignment Control is irrelevant to AC '97 mode since word and slot length are equal.
- RMOD1-RMOD0: AC '97 mode is chosen through the Receiver Network Mode Control bits.

- RSW4-RSW0: the Receiver Slot and Word Length Select bit values configure a 20-bit slot and a 20-bit word.
- RFSL: the Receiver Frame Sync Length bit is zero for a word-length frame sync.
- RFSR: the Receiver Frame Sync Relative Timing bit is set since frame sync occurs one serial clock cycle earlier.
- REIE, REDIE, RIE, RLIE: only the Receiver Last Slot Interrupt is enabled.

5.1.5 Common Control Register (SAICR)

11	10	9	8	7	6	5	4	3	2	1	0
			ALC	TEBE	SYN				OF2	OF1	OF0
0	0	0	0	0	0	0	0	0	0	0	0

23	22	21	20	19	18	17	16	15	14	13	12
0	0	0	0	0	0	0	0	0	0	0	0

AA1571

Figure 5-5 Common Control Register (SAICR)

SAICR is initialized with a value of **\$000000**, corresponding to the following configuration:

- OF2-OF0: these Serial Output Flag bits are irrelevant in this application because ESAI is in the asynchronous mode.
- SYN: Asynchronous mode is chosen with a zero Synchronous/Asynchronous bit.
- TEBE, ALC: these bits are irrelevant for this configuration.

5.1.6 Port Control, Direction and Data Registers

PCR, PRR and PDR mirror the pin functionality, direction, and state required in every task performed throughout the application. **Table 5-1** condenses all the combinations used. In this table *VST* stands for Vendor-Specific Test.

Table 5-1 PCR, PRR and PDR Values

PIN NUMBER	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
ESAI FUNCTION	SDO0	SDO1	SDO2 /SDI3	SDO3 /SDI2	SDO4 /SDI1	SDO5 /SDI0	HCKT	FST	SCKT	HCKR	FSR	SCKR
INITIAL PCR	0	0	0	0	0	0	0	0	0	0	0	0
INITIAL PRR	1	1	1	1	1	1	1	1	1	1	1	1
INITIAL PDR	0	1	0	0	0	0	0	0	0	0	0	0
ACTIVE PCR	1	0	0	0	0	1	0	1	1	0	1	1
ACTIVE PDR	0	1	0	0	0	0	0	0	0	0	0	0
PCR AFTER RESET	0	0	0	0	0	0	0	0	0	0	0	0
COLD RST PCR	0	0	0	0	0	0	0	0	0	0	0	0
COLD RST PDR	0	0	0	0	0	0	0	0	0	0	0	0
PDR AFTER COLD RST	0	1	0	0	0	0	0	0	0	0	0	0
WARM RST PCR	0	0	0	0	0	0	0	0	0	0	0	0
WARM RST PDR	0	1	0	0	0	0	0	1	0	0	1	0
PDR AFTER WARM RST	0	1	0	0	0	0	0	0	0	0	0	0
ATE PCR	0	0	0	0	0	0	0	0	0	0	0	0
ATE PDR	1	0	0	0	0	0	0	0	0	0	0	0
PDR AFTER ATE	1	1	0	0	0	0	0	0	0	0	0	0
VST PCR	0	0	0	0	0	0	0	0	0	0	0	0
VST PDR	0	0	0	0	0	0	0	1	0	0	1	0

Table 5-1 PCR, PRR and PDR Values (Continued)

PIN NUMBER	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
ESAI FUNCTION	SDO0	SDO1	SDO2 /SDI3	SDO3 /SDI2	SDO4 /SDI1	SDO5 /SDI0	HCKT	FST	SCKT	HCKR	FSR	SCKR
PDR AFTER VST	0	1	0	0	0	0	0	1	0	0	1	0

5.2 DMA Configuration - Channel 0

The properties of the first transmitted audio frame are discussed in **Section 3** of this application report. Channel 0 of DMA is specially programmed for 12 data transfers to ESAI during this frame and subsequently re-programmed for continuous data transfer from transmission data buffers to the ESAI transmitter. The DMA Control Register values for channel 0 are listed in **Section 5.2.1**.

5.2.1 DMA Control Register for Channel 0 (DCR0)

The first frame's DCR0 value is **See6250**:

- DMA channel enabled
- DMA interrupt enabled
- DMA Transfer Mode: word, triggered by ESAI request, DE not cleared
- Highest Channel Priority
- Continuous Mode disabled
- Source Address Generation Mode: linear
- Destination Address Generation Mode: no update
- Source and Destination in X memory

Continuous operation DCR0 value is **\$ae6620**:

- DMA channel enabled
- DMA interrupt disabled
- DMA Transfer Mode: word, triggered by ESAI request, DE not cleared
- Highest Channel Priority

- Continuous Mode disabled
- Three-Dimensional Address Generation Mode
- Source Address Generation Mode: three-dimensional with offset registers DOR0 and DOR1
- Destination Address Generation Mode: no update
- Mode E counter
- Source and Destination in X memory

5.2.2 DMA Counter Register for Channel 0 (DCO0)

For the first frame, the counter is initialized with \$B, corresponding to 12 transfers minus one. Subsequently, the counter is re-programmed for 2 times 13 transfers, repeated circularly.

5.2.3 DMA Offset Registers (DOR0 and DOR1)

DOR0 and DOR1 are initialized with adequate values in order to swap from buffer OB0 to buffer OB1 and back to buffer OB0 in a circular way defined by DCR0 and DCO0.

5.2.4 DMA Source Address Register for Channel 0 (DSR0)

DSR0 points at any time to the current word being transferred by DMA. It is initialized with the base address of the active buffer.

5.2.5 DMA Destination Address Register for Channel 0 (DDR0)

DDR0 is programmed as the ESAI Transmit Data Register for transmitter #0 (TX0) and remains unchanged.

5.3 DMA Configuration - Channel 1

Channel 1 is programmed from the beginning for continuous data transfer from the ESAI receiver to reception data buffers. The DMA Control Register values for channel 1 are listed in **Section 5.3.1**.

5.3.1 DMA Control Register for Channel 1 (DCR1)

Continuous operation DCR1 value is **\$ae5e64**:

- DMA channel enabled
- DMA interrupt disabled
- DMA Transfer Mode: word, triggered by ESAI request, DE not cleared
- Highest Channel Priority
- Continuous Mode disabled
- Three-Dimensional Address Generation Mode
- Source Address Generation Mode: no update
- Destination Address Generation Mode: three-dimensional with offset registers DOR2 and DOR3
- Mode E counter
- Source in X memory and Destination in Y memory

5.3.2 DMA Counter Register for Channel 1(DCO1)

This register is programmed in the same way as DCO0.

5.3.3 DMA Offset Registers (DOR2 and DOR3)

DOR2 and DOR3 are initialized with adequate values in order to swap from buffer IB0 to buffer IB1 and back to buffer IB0 in a circular way defined by DCR1 and DCO1.

5.3.4 DMA Source Address Register for Channel 1(DSR1)

DSR1 is programmed as the ESAI Receive Data Register for receiver #0 (RX0) and remains unchanged.

5.3.5 DMA Destination Address Register for Channel 1 (DDR1)

DDR1 points at any time to the current word being transferred by DMA. It is initialized with the base address of the first active buffer.

5.4 Timer Configuration

The timer module is used for pulse generation on AC-link reset and test signalling. It is programmed only to count the number of internal clock cycles corresponding to the desired pulse duration.

5.4.1 Timer Control/Status Register (TCSR)

For all pulse generation routines, TCSR is programmed with the same value: **\$4**. This corresponds to:

- Timer Compare Interrupt enabled
- Timer Mode with internal Clock

5.4.2 Timer Load Register (TLR)

The timer counter is cleared before timer enabling, by writing **\$0** to TLR.

5.4.3 Timer Compare Register (TCPR)

TCPR is programmed with a clock cycles count corresponding to the desired pulse.

Appendix A Equates and Code Routines

This appendix presents the defined equates and assembly code routines for this application.

A.1 EQUATES

As part of the respective DSP56300 derivative's I/O and interrupt equates, the AC-link application assembly code uses the equates defined in the following paragraphs.

A.2.1 General Equates

```

;-----
;          AC-LINK APPLICATION  EQUATES
;-----
;          General
;-----
START          equ          $100          ; Main Program Starting Address
TOGGLE_MODULE  equ          $1ff          ; Module Addressing Factor for Active
                                           ;Buffer Toggling

```

A.2.2 ESAI Configuration Equates

```

;-----
;          ESAI
;-----
TCR_INIT      equ          $010f00      ; Initial Value for TCR
RCR_INIT      equ          $810f00      ; Initial Value for RCR
TCCR_INIT     equ          $441800      ; Initial Value for TCCR
RCCR_INIT     equ          $001800      ; Initial Value for RCCR
SAICR_INIT    equ          $000000      ; Initial Value for SAICR
PRR_INIT      equ          $000fff      ; Initial Value for PRR (reset)
PCR_INIT      equ          $000000      ; Initial Value for PCR (reset)
PDR_INIT      equ          $000400      ; Initial Value for PDR (reset)
;-----
TCR_ACT       equ          $010f01      ; ESAI  Enabling Value for TCR
RCR_ACT       equ          $810f01      ; ESAI  Enabling Value for RCR
PDR_ACT       equ          $000400      ; ESAI  Enabling Value for PDR
PCR_ACT       equ          $00085b      ; ESAI  Enabling Value for PCR

```

Equates

A.2.3 DMA Programming Equates

```
-----  
;          DMA  
-----  
DMA0_COUNT_F0      equ      $b          ; First Audio Frame DMA Channel #0 ;Counter  
DMA_COUNT           equ      $00100c    ; DMA counter prog: 2 * 13 words  
DMA_OFFSET_0       equ      $f4         ; DOR0 Value  
DMA_OFFSET_1       equ      $fffef4    ; DOR1 Value  
DMA_OFFSET_2       equ      $fffef4    ; DOR2 Value  
DMA_OFFSET_3       equ      $f4         ; DOR3 Value  
DMA_0_CTRL_F0      equ      $ce6250     ; DMA configuration for channel 0, First  
;Frame  
DMA_0_CTRL          equ      $ae6620     ; DMA configuration for channel 0  
DMA_1_CTRL          equ      $ae5e64     ; DMA configuration for channel 1
```

A.2.4 Timer Programming Equates

```
-----  
;          TIMER  
-----  
TCSR1_RST           equ      $000004  
TCSR1_RST_ENABLE   equ      $000005
```

A.2.5 Data Structure Equates

```
-----  
;          AC-LINK SLOTS  
-----  
SLOT_1_MASK        equ      $c00000  
SLOT_2_MASK        equ      $a00000  
SLOT_3_MASK        equ      $900000  
SLOT_4_MASK        equ      $880000  
SLOT_5_MASK        equ      $840000  
-----  
;          BUFFERS  
-----  
DATA_B0            equ      $0          ; First Data Buffer Base Address  
DATA_B1            equ      $100       ; Second Data Buffer Base Address  
-----  
;          FIFOS  
-----  
AC_LINK_TABLE      equ      $f0  
-----  
;          FIFOs' SCHEDULE  
-----  
OFIFOS_ORDER       equ      $70  
IFIFOS_ORDER       equ      $80
```



```

;-----
;           FIFOs' TABLE
;-----
FIFOS_TABLE      equ      $90
;columns
FIFO_BEG         equ      $0
FIFO_END         equ      $1
FIFO_SIZE        equ      $2
FIFO_RD          equ      $3
FIFO_WR          equ      $4
FIFO_FGS         equ      $5
;flags
FULL_FG          equ      $0
EMPTY_FG         equ      $1

;-----
;           Output FIFOs
;-----
OPCM_FIFO_BEG    equ      $010
OPCM_FIFO_END    equ      $017
OPCM_FIFO_SIZE   equ      (OPCM_FIFO_END-OPCM_FIFO_BEG) ; size -1
;-----
OCMD_FIFO_BEG    equ      $018
OCMD_FIFO_END    equ      $01f
OCMD_FIFO_SIZE   equ      (OCMD_FIFO_END-OCMD_FIFO_BEG) ; size -1
;-----
OMDM_FIFO_BEG    equ      $020
OMDM_FIFO_END    equ      $027
OMDM_FIFO_SIZE   equ      (OMDM_FIFO_END-OMDM_FIFO_BEG) ; size -1
;-----
;           Input FIFOs
;-----
IPCM_FIFO_BEG    equ      $028
IPCM_FIFO_END    equ      $02f
IPCM_FIFO_SIZE   equ      (IPCM_FIFO_END-IPCM_FIFO_BEG) ; size -1
;-----
ISTAT_FIFO_BEG   equ      $030
ISTAT_FIFO_END   equ      $037
ISTAT_FIFO_SIZE  equ      (ISTAT_FIFO_END-ISTAT_FIFO_BEG) ; size -1
;-----
IMDM_FIFO_BEG    equ      $038
IMDM_FIFO_END    equ      $03f
IMDM_FIFO_SIZE   equ      (IMDM_FIFO_END-IMDM_FIFO_BEG) ; size -1
;-----
IMIC_FIFO_BEG    equ      $040
IMIC_FIFO_END    equ      $047
IMIC_FIFO_SIZE   equ      (IMIC_FIFO_END-IMIC_FIFO_BEG) ; size -1

```

Equates

A.2.6 AC-Link Reset Routines Equates

```
-----  
;          RESET  
-----  
PDR_AFTER_RST      equ      (AC_LINK_TABLE)  
CR_COUNT           equ      $0001B8 ; Timer Counter Value for Cold Reset  
WR_COUNT           equ      $48    ; Timer Counter Value for Warm Reset  
PCR_COLD_RST       equ      $000000  
PDR_COLD_RST       equ      $000000  
PDR_AFTER_COLD     equ      $000400 ; esai/gpio pins data (reset exit)  
PCR_WARM_RST       equ      $000000  
PDR_WARM_RST       equ      $000412  
PDR_AFTER_WARM     equ      $000400 ; esai/gpio pins data (reset exit)  
PCR_AFTER_RST      equ      $000000 ; esai pin configuration (reset exit)
```

A.2.7 AC-Link Test Routines Equates

```
-----  
;          TEST  
-----  
PCR_ATE            equ      $000000  
PDR_ATE            equ      $000800 ; esai/gpio pins data for ATE transition  
; (SDATA_OUT high)  
PDR_AFTER_ATE      equ      $000c00 ; esai/gpio pins data (reset exit)  
PCR_VST            equ      $000000  
PDR_VST            equ      $000012 ; esai/gpio pins data for VST transition  
; (SYNC high)  
PDR_AFTER_VST      equ      $000412 ; esai/gpio pins data (reset exit)
```

A.6 MAIN CODE ROUTINES

A.2.1 Initial Procedure

```
initial  
  move    #DATA_B0,r7      ; not-active active  buffer core pointer  
  clr     a  
  move    #TOGGLE_MODULE,m7  
  move    #(DATA_B1-DATA_B0),n7; buffer size for module add. updating  
  movep   #$000303,x:M_IPRP ; set ESAI0 and TIMER priority to 2  
  movep   #$00F000,x:M_IPRC ; set DMA priority to 2  
  move    #$0,sr          ; DSP interrupt priority = 0 --> enable ;int.  
  move    a,x:(r7)        ; Audio Frame Not-valid  
  move    a,x:(r7+n)      ; Audio Frame Not-valid
```

A.2.2 ESAI Configuration

```

esai_prog
    movep    #TCR_INIT,x:M_TCR
    movep    #RCR_INIT,x:M_RCR
    movep    #TCCR_INIT,x:M_TCCR
    movep    #RCCR_INIT,x:M_RCCR
    movep    #SAICR_INIT,x:M_SAICR
    movep    #PRR_INIT,x:M_PRR
    movep    #PCR_INIT,x:M_PCR
    movep    #PDR_INIT,x:M_PDR

```

A.2.3 DMA Programming

```

dma_prog
    movep    #(DATA_B1+1),x:M_DSR0
    movep    #DMA_OFFSET_0,x:M_DOR0
    movep    #DMA_OFFSET_1,x:M_DOR1
    movep    #M_TX0,x:M_DDR0
    movep    #DMA0_COUNT_F0,x:M_DCO0

    movep    #M_RX0,x:M_DSR1
    movep    #(DATA_B1),x:M_DDR1
    movep    #DMA_COUNT,x:M_DCO1
    movep    #DMA_OFFSET_2,x:M_DOR2
    movep    #DMA_OFFSET_3,x:M_DOR3

```

A.2.4 DMA Activation

```

activate_dma
    movep    #DMA_0_CTRL_F0,x:M_DCR0
    movep    #DMA_1_CTRL,x:M_DCR1

```

A.2.5 ESAI Activation

```

activate_esai
    movep    #PCR_ACT,x:M_PCR
    movep    #$000000,x:M_TX0    ; First data to Transmitter, ZERO, since
                                ; Frame is not valid

    movep    #TCR_ACT,x:M_TCR
    movep    #RCR_ACT,x:M_RCR

```

A.6 AC_LINK TEST ROUTINES

A.2.1 ATE

```
ate_test
    move    #PDR_AFTER_ATE,a
    move    a,x:PDR_AFTER_RST
    movep   #$0,x:M_TLR1           ; zero timer load register
    movep   #CR_COUNT,x:M_TCPRI    ; count #COUNT timer clocks
    movep   #TCSR1_RST,x:M_TCSR1  ; program timer
    movep   #PDR_ATE,x:M_PDR
    movep   #PCR_ATE,x:M_PCR
    movep   #TCSR1_RST_ENABLE,x:M_TCSR1; enable timer
    rts
```

A.2.2 Vendor Specific Test

```
vendor_test
    move    #PDR_AFTER_VST,a
    move    a,x:PDR_AFTER_RST
    movep   #$0,x:M_TLR1           ; zero timer load register
    movep   #CR_COUNT,x:M_TCPRI
    movep   #TCSR1_RST,x:M_TCSR1  ; program timer
    movep   #PDR_VST,x:M_PDR
    movep   #PCR_VST,x:M_PCR
    movep   #TCSR1_RST_ENABLE,x:M_TCSR1 ; enable timer
    rts
```

A.6 AC_LINK RESET ROUTINES

A.2.1 Cold Reset

```
cold_reset
    move    #PDR_AFTER_COLD,a
    move    a,x:PDR_AFTER_RST
    movep   #$0,x:M_TLR1           ; zero timer load register
    movep   #CR_COUNT,x:M_TCPRI
    movep   #TCSR1_RST,x:M_TCSR1  ; program timer
    movep   #PDR_COLD_RST,x:M_PDR
    movep   #PCR_COLD_RST,x:M_PCR
    movep   #TCSR1_RST_ENABLE,x:M_TCSR1 ; enable timer
    rts
```

A.2.2 Warm Reset

```
warm_reset
    move    #PDR_AFTER_WARM,a
    move    a,x:PDR_AFTER_RST
    movep   #$0,x:M_TLR1           ; zero timer load register
    movep   #WR_COUNT,x:M_TCPR1
    movep   #TCSR1_RST,x:M_TCSR1   ; program timer
    movep   #PDR_WARM_RST,x:M_PDR
    movep   #PCR_WARM_RST,x:M_PCR
    movep   #TCSR1_RST_ENABLE,x:M_TCSR1 ; enable timer
    rts
```

A.6 INTERRUPT ROUTINES

A.2.1 Timer Interrupt

```
timer_1
    bclr    #$2,x:M_TCSR1         ; disable timer interrupt
    bclr    #$0,x:M_TCSR1         ; disable timer
    movep   x:PDR_AFTER_RST,x:M_PDR ; pulse conclusion
    movep   #PCR_AFTER_RST,x:M_PCR ; enable ESAI with new
                                         ;configuration
    rti
```

A.2.2 DMA Channel #0 Interrupt

```
dma_0
    movep   #DMA_COUNT,x:M_DCO0    ; number of words
    movep   #DATA_B0,X:M_DSR0
    movep   #DMA_0_CTRL,x:M_DCR0   ; enable DMA, this time with
                                         ;interrupt disabled
    rti
```

A.2.3 ESAI Receive Last Slot Interrupt

```
rx_last_slot_int
    move    (r7)+n                ; toggle B0<->B1, r7
    move    #OFIFOS_ORDER,r2      ; point to FIFOs' schedule
    move    #FIFO_FGS,n3          ; point to FLAGS column
    move    r7,x0                  ; save r7
    clr    a        (r7)+         ; clear TAG , point R7 to next
                                         ;data slot
    do     #$5,end_rx_last_slot_int_read
```

Equates

```
move    x:(r2)+,r3                ; point to current FIFOs' row
move    x:(r2)+,y1                ; get fifos' mask
jset    #EMPTY_FG,x:(r3+n),eend_rx_last_slot_int_read; read from fifo
                                         unless it's EMPTY
move    x:(r3+FIFO_RD),r1         ; get read pointer
move    x:(r3+FIFO_SIZE),m1       ; get FIFO's size for module
                                         addressing
move    x:(r1)+,x1                ; read from FIFO
bclr    #FULL_FG,x:(r3+n)         ; clear FULL flag
move    x1,x:(r7)                 ; write to buffer
move    r1,x:(r3+FIFO_RD)         ; update RD pointer
or      y1,a                       ; update tag
move    x:(r3+FIFO_WR),b         ; get write pointer
move    r1,x1
cmp     x1,b
jne     eend_rx_last_slot_int_read ; set EMPTY flag in case RD=WR
bset    #EMPTY_FG,x:(r3+n)
eend_rx_last_slot_int_read
move    (r7)+                     ; point R7 to next data slot
nop
nop
nop
nop
end_rx_last_slot_int_read
; now handle input fifos
move    x0,r7                     ; recover r7
move    a1,x:(r7)                 ; update TAG
move    #IFIFOS_ORDER,r2         ; point to FIFOs' schedule
move    y:(r7)+,a                 ; get input buffer's TAG
lsl     a
jcc     end_rx_last_slot_int_write ; end in case this frame is NOT
                                         ;VALID
do      #$6,end_rx_last_slot_int_write
move    x:(r2)+,r3                ; point to current FIFOs' table
jset    #FULL_FG,x:(r3+n),crash   ; write to FIFO unless it's FULL
lsl     a
jcc     eend_rx_last_slot_int_write ; end in case this slot is NOT
                                         ;VALID
move    x:(r3+FIFO_WR),r1         ; get write pointer
move    x:(r3+FIFO_SIZE),m1       ; get FIFO's size for module
                                         addressing
move    y:(r7),x1                ; read buffer
bclr    #EMPTY_FG,x:(r3+n)         ; clear EMPTY flag
move    x1,x:(r1)+                ; write to FIFO
move    r1,x:(r3+FIFO_WR)         ; update WR pointer
move    x:(r3+FIFO_RD),b         ; get read pointer
move    r1,x1
cmp     x1,b
jne     eend_rx_last_slot_int_write ; set FULL flag in case RD=WR
bset    #FULL_FG,x:(r3+n)
eend_rx_last_slot_int_write
move    (r7)+                     ; point R7 to next data slot
nop
```

```

        nop
        nop
        nop
        nop
        nop
end_rx_last_slot_int_write
        move    x0,r7                ; recover r7
        nop
        nop
        nop
        nop
        rti

```

A.6 CRASH ROUTINE

The AC-link application enters a *crash* routine when it tries to write to any input FIFO that is full. An audio application should handle this situation and define an applicable procedure (for example, immediately process the FIFO's contents or enlarge the FIFO's capacity).

In the sample provided here, we resume the *receive last slot ISR* without updating the corresponding full FIFO. The data in the input buffer is therefore lost.

```

crash
    nop
    nop
    nop
    jmp eend_rx_last_slot_int_write
    nop
    nop
    nop

```

A.6 A RUNNING EXAMPLE

The preceding routines are put together in the example covered in this section. The code for the routines is not copied here and should be included before assembling, according to the instructions in the comment lines.

```

;-----
; Interrupt Space
;-----
    org    p:I_RESET        ;Hardware RESET
    jmp    >START

    dup    (I_INTEND-**+1)  ;fill vector space
    jmp *                    ;

```

Equates

```
    endm

    org P:I_DMA0
    jsr <dma_0
    nop

    org P:I_TIMER1
    jsr <timer_1
    nop

    org P:I_SIRLS
    jsr <rx_last_slot_int
    nop

    org      p:(I_INTEND+1)
    dup      (START-I_INTEND-1)      ;fill with nops
    nop
    endm

;-----
; Application code begins here
;-----

    org      P:START

    ; PLL initialization
    movep    #$75001C,x:M_PCTL      ;(DSP_clk = 67.2M, when input clk=16.8M)
    rep     #$fff
    nop
; include below <Initial Procedure> code

; include below <ESAI Configuration> code

; include below <DMA Programming> code

; include below <DMA Activation> code

; write to command FIFO
    move     #$18,r5
    move     #$800000,x1
    move     x1,x:(r5)+             ; reading codec add 00(reset reg) .
    move     #$0,x1
    move     x1,x:(r5)+
    move     #$fe0000,x1
    move     x1,x:(r5)+             ; reading codec add 7e(vendor reg) .
    move     #$0,x1
    move     x1,x:(r5)+
    move     #$fc0000,x1
    move     x1,x:(r5)+             ; reading codec add 7c(vendor reg) .
    move     #$0,x1
    move     x1,x:(r5)+
; reset CODEC
    jsr     <cold_reset
; wait for the end of the reset pulse
```



```

        jset    #$0,x:M_TCSR1,*

; include below <ESAI Activation> code

; wait until CODEC transmit CODEC Ready bit
wait4codecready:
        jset #23,y:(r6)+n,setfifo
        jset #23,y:(r6)+n,setfifo
        jmp <wait4codecready

; update command FIFO's entry in table to enable transmission
setfifo:
        bclr   #EMPTY_FG,x:(FIFOS_TABLE+FIFO_FGS+$20)    ;update FIFO_FGS
        move   r5,x:(FIFOS_TABLE+FIFO_WR+$20)           ;update FIFO_WR pointer

; infinite loop, for observation
        jmp    *

; include below the code for RESET, TEST, INTERRUPTS and CRASH routines

;-----
; DATA memory
;-----
;-----
;          AC-LINK TABLE
;-----
        org    X:AC_LINK_TABLE

        dc     $0                                     ; PDR_AFTER_RST
;-----
;          INPUT FIFOS ORDER
;-----
        org    X:IFIFOS_ORDER

        dc     FIFOS_TABLE
        dc     FIFOS_TABLE
        dc     FIFOS_TABLE+$8
        dc     FIFOS_TABLE+$8
        dc     FIFOS_TABLE+$10
        dc     FIFOS_TABLE+$18
;-----
;          OUTPUT FIFOS ORDER
;-----
        org    X:OFIFOS_ORDER

        dc     FIFOS_TABLE+$20
        dc     SLOT_1_MASK
        dc     FIFOS_TABLE+$20
        dc     SLOT_2_MASK
        dc     FIFOS_TABLE+$28
        dc     SLOT_3_MASK
        dc     FIFOS_TABLE+$28
        dc     SLOT_4_MASK

```

Equates

```
dc      FIFOS_TABLE+$30
dc      SLOT_5_MASK
;-----
;      FIFOS TABLE
;-----
org     X:FIFOS_TABLE

dc      ISTAT_FIFO_BEG
dc      ISTAT_FIFO_END
dc      ISTAT_FIFO_SIZE
dc      ISTAT_FIFO_BEG;      ISTAT_RD
dc      ISTAT_FIFO_BEG;      ISTAT_WR
dc      $2                    ;      ISTAT_FGS
;-----
org     X:FIFOS_TABLE+$8

dc      IPCM_FIFO_BEG
dc      IPCM_FIFO_END
dc      IPCM_FIFO_SIZE
dc      IPCM_FIFO_BEG;      IPCM_RD
dc      IPCM_FIFO_BEG;      IPCM_WR
dc      $2                    ;      IPCM_FGS
;-----
org     X:FIFOS_TABLE+$10

dc      IMDM_FIFO_BEG
dc      IMDM_FIFO_END
dc      IMDM_FIFO_SIZE
dc      IMDM_FIFO_BEG;      IMDM_RD
dc      IMDM_FIFO_BEG;      IMDM_WR
dc      $2                    ;      IMDM_FGS
;-----
org     X:FIFOS_TABLE+$18

dc      IMIC_FIFO_BEG
dc      IMIC_FIFO_END
dc      IMIC_FIFO_SIZE
dc      IMIC_FIFO_BEG;      IMIC_RD
dc      IMIC_FIFO_BEG;      IMIC_WR
dc      $2                    ;      IMIC_FGS
;-----
org     X:FIFOS_TABLE+$20

dc      OCMD_FIFO_BEG
dc      OCMD_FIFO_END
dc      OCMD_FIFO_SIZE
dc      OCMD_FIFO_BEG;      OCMD_RD
dc      OCMD_FIFO_BEG;      OCMD_WR
dc      $2                    ;      OCMD_FGS
;-----
org     X:FIFOS_TABLE+$28

dc      OPCM_FIFO_BEG
```

```
dc      OPCM_FIFO_END
dc      OPCM_FIFO_SIZE
dc      OPCM_FIFO_BEG      ;      OPCM_RD
dc      OPCM_FIFO_BEG;    OPCM_WR
dc      $2                  ;      OPCM_FGS
;-----
org     X:FIFOS_TABLE+$30

dc      OMDM_FIFO_BEG
dc      OMDM_FIFO_END
dc      OMDM_FIFO_SIZE
dc      OMDM_FIFO_BEG;    OMDM_RD
dc      OMDM_FIFO_BEG;    OMDM_WR
dc      $2                  ;      OMDM_FGS
;-----
```


Appendix B References

The following manuals, which may contain data pertinent to this application, may be viewed or downloaded at the indicated web sites.

- <http://www.mot.com/SPS/DSP/documentation/DSP56300.html>
DSP56300 Digital Signal Processor Family Manual
DSP56362 Digital Signal Processor User's Manual
DSP56362 Digital Signal Processor Data Sheet
DSP56300 Digital Signal Processor Family: DMA Application Note
Application Optimization Note for the DSP56300/DSP56600 Family
- <http://developer.intel.com/pc-supp/platform/ac97>
Audio Codec `97 Component Specification. Revision 1.03 (September 1996) and 2.0 (September 1997). Primary Developers: Analog Devices. Creative Labs, Inc. Intel Corporation. National Semiconductor Yamaha Corporation. Copyright 1997, Intel Corporation.

Order by this number: APR37/D

OnCE and Mfax are registered trademarks of Motorola, Inc.

Motorola reserves the right to make changes without further notice to any products herein. Motorola makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Motorola 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 consequential or incidental damages. "Typical" parameters which may be provided in Motorola 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. Motorola does not convey any license under its patent rights nor the rights of others. Motorola 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 life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola 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 Motorola was negligent regarding the design or manufacture of the part. Motorola and  are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

How to reach us:

USA/Europe/Locations Not Listed:

Motorola Literature Distribution
P.O. Box 5405
Denver, Colorado 80217
1 (800) 441-2447
1 (303) 675-2140

Motorola Fax Back System (Mfax™):

TOUCHTONE (602) 244-6609
1 (800) 774-1848
RMFAX0@email.sps.mot.com

Asia/Pacific:

Motorola Semiconductors H.K. Ltd.
8B Tai Ping Industrial Park
51 Ting Kok Road
Tai Po, N.T., Hong Kong
852-26629298

Technical Resource Center:

1 (800) 521-6274

DSP Helpline

dsphelp@dsp.sps.mot.com

Japan:

Nippon Motorola Ltd
SPD, Strategic Planning Office
4-32-1, Nishi-Gotanda
Shinagawa-ku, Tokyo 141, Japan
81-3-5487-8488

Internet:

<http://www.motorola-dsp.com/>



MOTOROLA
