Product Specification

AHA4525

IEEE 802.16a Compliant Turbo Product Code Encoder/Decoder

This product is covered under multiple patents held or licensed by Comtech AHA Corporation.

This product is covered by a Turbo Code Patent License from France Telecom - TDF - Groupe des ecoles des telecommunications.

PS4525_0204

AHA

A subsidiary of Comtech Telecommunications Corporation

■ 2345 NE Hopkins Court ■ Pullman WA 99163 ■ tel: 509.334.1000 ■ fax: 509.334.9000 ■ www.aha.com ■

Table of Contents

List of Figures

List of Tables

1.0 INTRODUCTION

The AHA4525 device is a single-chip Turbo Product Code (TPC) Forward Error Correction (FEC) Encoder/Decoder. This device integrates independent TPC encoder and decoder functions, and can be configured for full or half duplex operation.

The encoder and decoder accept data and configuration through a synchronous 3-wire DSP type interface or asynchronous data bus.

Encoder and decoder configuration registers are written and read through the same interface as the data. Configuration registers are accessible after a reset and before the chip is enabled for data transfers.

The encode datapath, after configuration for any of the IEEE 802.16a BTC coding options, inputs data on UDATA, calculates and inserts error correction code (ECC) bits, and outputs the data on the EDATA interface. The decoder datapath is the reverse of the encoder datapath. The received data is input either serially or one soft metric per transfer at the CDATA interface, decoded with ECC bits removed, then output in a serial bit stream on the DDATA bus.

The Input/Output data interfaces each have a separate clock input. This coupled with an on-chip PLL for the 4x internal system clock allow flexibility in the system and lower on-board clock frequencies. This device implements the same code shortening schemes defined in the IEEE 802.16a standard. Removing X axis rows, Y axis columns, and bits from the beginning of the original message are easily programmed configuration register parameters.

1.1 FEATURES

PERFORMANCE:

- 60 Mbit/sec channel rate and 50 Mbit/sec payload data rate for (64,57)x(64,57) code with 3 iterations
- Supports all IEEE 802.16a BTC code configurations

FLEXIBILITY:

- Code rates from 0.25 to 0.97
- Encoded block sizes from 64 bits to 4K bits
- Programmable code shortening supports exact block sizes
- Programmable decoder input quantization for up to 4 bit wide soft metrics
- Programmable iterations up to 255 per block
- On chip PLL allows low frequency system clock

CHANNEL INTERFACE:

- Synchronous 3-wire input and output ports designed to be compatible with DSP serial ports
- Bus mode input and output ports designed to be compatible with a DSP bus
- Chip selects on encoder and decoder ports for full or half-duplex operation
- Pin selectable interface control signal polarity
- Decoder supports up to 4 bit parallel soft metric input data for fast decode operation

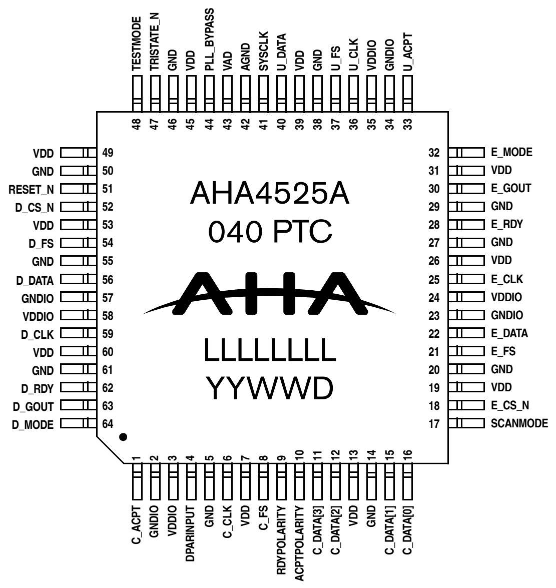
SYSTEM INTERFACE:

- Configuration registers are accessed through the data ports

ELECTRICAL:

- 3.3V I/O, 1.8V core operation
- 5V tolerant inputs
- TTL signal compatible
- 64 pin TQFP Package
- Commercial or Industrial temperature rating

Figure 1: Pinout



Note: YYWWD = Date Code; LLLLLLLL = Lot Number

1.2 GLOSSARY OF TERMS

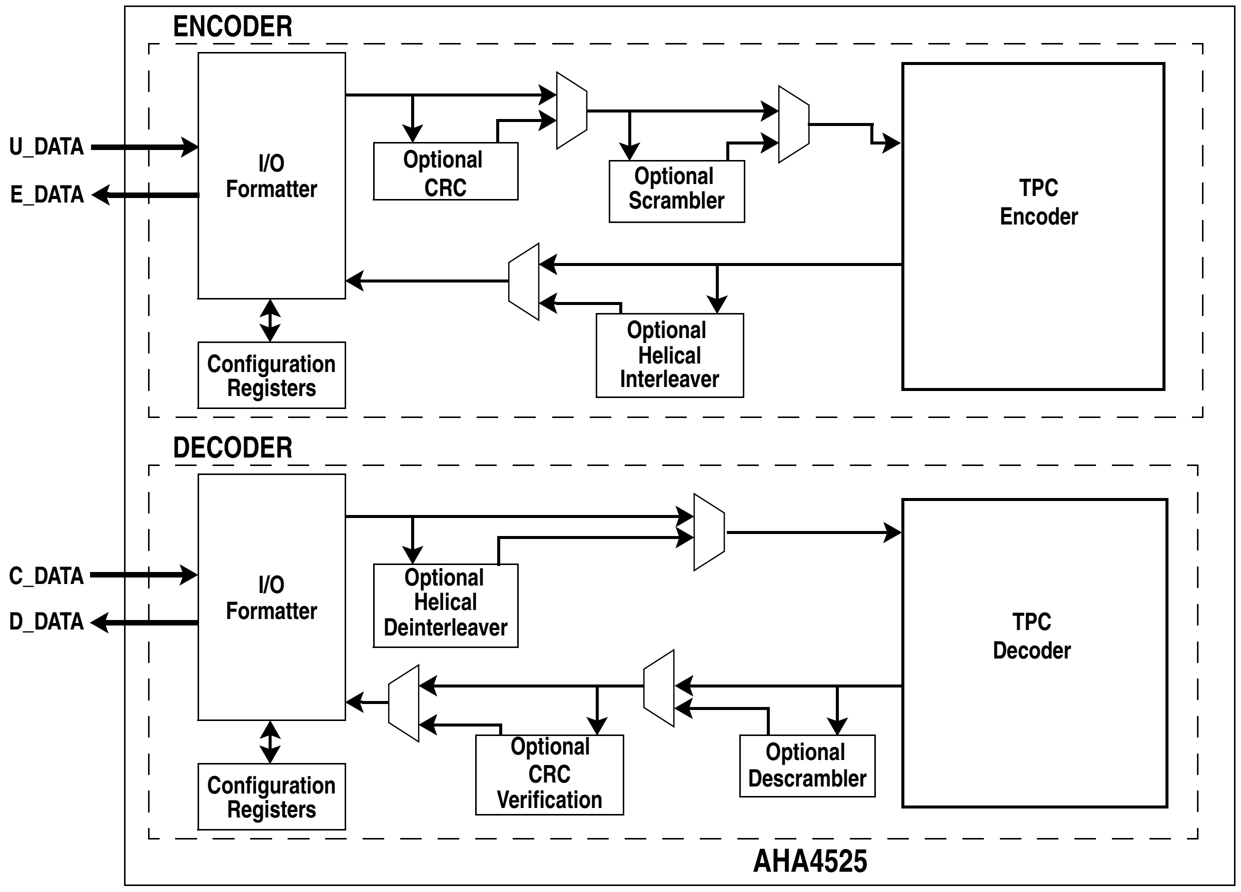
Block (TPC) –	A TPC error correction block, including user data bits, CRC bits, and TPC error correction bits.
Channel Rate -	The bit rate output from the encoder, or input to the decoder, including all user data bits, CRC bits, and TPC error correction bits.
Data Rate -	The bit rate input to the encoder, or output from the decoder, including only user data bits. This is sometimes referred to as the payload data rate.

1.3 DOCUMENT CONVENTIONS

The following are formatting examples for specific document elements.

– **SIGNAL OR PINS -** Electrical connections available to the system.
– **Register Bit -** Bit(s) within a register. When the same register exists in both the encoder and decoder, an '**x**' is used to designate both register bits, as in **ECRCEnable** and **DCRCEnable**.
– Hex values are represented with a prefix of "0x," such as Register "0x00." Binary values are represented with a prefix of "0b".
– Active low signals have "**_N**" appended to the signal name, as in **E_RD_N**.
– Signals labelled "= 1" are tied to Vddio. Signals labelled "= 0" are tied to ground.
– "configuration header or footer bit" - Configuration header control bits are listed in quotes, as in "last" or "rwn".

Figure 2: AHA4525 Functional Block Diagram



2.0 DATA AND CONFIGURATION INPUT/OUTPUT

The input/output ports of the AHA4525 chip may be configured to operate in synchronous mode or bus mode. Synchronous mode signals consist of a continuous port clock, a frame sync signal, and data. Bus mode signals consist of a read or write strobe and data.

Bus mode is selected for the encoder when pin **E_MODE** = 1 on the rising edge of **RESET_N**, and is selected for the decoder when pin **D_MODE** = 1 on the rising edge of **RESET_N**. Synchronous mode is selected for the encoder when pin **E_MODE** = 0 and for the decoder when pin **D_MODE** = 0.

The interface modes are independent. The encoder is not required to use the same interface mode as the decoder. The input port cannot be configured to use a different mode than the output port within the encoder or decoder.

The interfaces into and out of the encoder and out of the decoder are always serial in either mode. The soft decision channel data into the decoder may be input symbol per transfer, up to 4 bits parallel, in either mode. The parallel channel input is selected when pin **DPARINPUT** = 1 on the rising edge of **RESET_N**. When pin **DPARINPUT** = 0, the channel data is input serially.

A summary of the differences between synchronous and bus interface modes is:

1)Synchronous mode requires a continuous - not gated - port clock. Bus mode uses a data strobe.

2)Synchronous mode requires a frame sync signal to assert 1 clock before data on each block. Bus mode does not require a frame sync.

3)Data input and output can be faster in the synchronous mode than in bus mode. The bus mode data strobe is limited to the **SYSCLK** frequency and the synchronous port clock allows an input clock rate of 1.9***SYSCLK** (refer to Section 11.2 *U_CLK, E_CLK, C_CLK, D_CLK Clock Timing* and Section 11.3 *U_WR_N, E_RD_N, C_WR_N, D_RD_N Strobe Timing*).

With the exception of the 4 bit parallel channel input to the decoder, the encoder interface is identical to the decoder interface. The following sections describe the interface in terms of the decoder, but all control descriptions and signal timing, except where explicitly stated, apply to the encoder signals. The encoder and decoder are shown on all connection diagrams.

2.1 CONFIGURATION WRITES

The AHA4525 configuration registers are split into 2 sets of register banks: one for the encoder datapath functions and one for the decoder datapath functions.

All encoder configuration registers are written through the encoder unencoded data input port. All decoder configuration registers are written through the decoder channel data input port. The configuration registers can be written before the start of a block transfer - not during a block transfer.

Configuration data is written in a 16 bit configuration cycle format (refer to Section 2.3 *Configuration Cycle Format*). When writing configuration data to the decoder and the decoder is configured for a parallel input by setting pin **DPARINPUT** = 1, the configuration data must be input 4 bits per transfer. When writing the configuration in parallel, the first of 4 data transfers expects bit 15 to be input on **CDATA[3]** down to bit 12 input on **CDATA[0]**. All 4 **CDATA** inputs are always used when writing configuration data in parallel.

The configuration cycle starts with a “rwn” bit to indicate whether the current configuration cycle is a read or write cycle and is followed by the “last” bit to indicate that the current configuration cycle is the last configuration write cycle. Configuration cycles are expected to continue until a cycle is received with the “last” bit set.

If the “last” bit is set, the AHA4525 expects data to immediately follow the current configuration cycle unless the last configuration cycle writes a one to the **NoBlock** bit (see configuration Section 7.7.1 *Control*). An example of the configuration cycle is shown in Figure 3.

Figure 3: Configuration Cycle Followed by Block 1

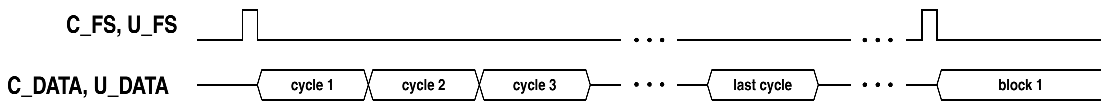


Figure 4: Fixed Configuration Mode

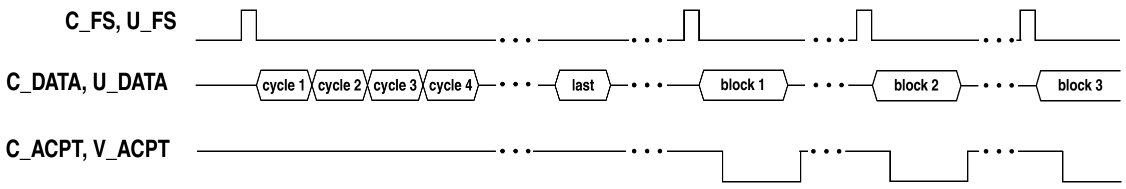
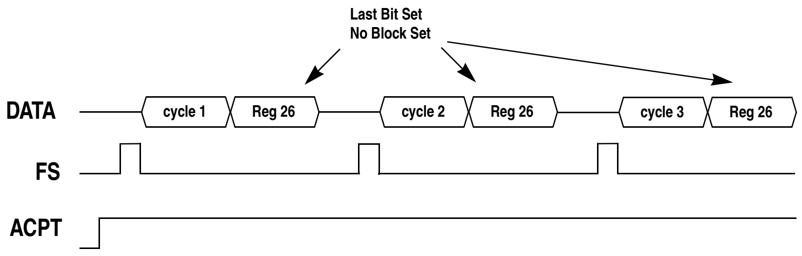


Figure 5: Configuration Example for Gaps Between Configuration Writes



The "last" bit is followed by a 6 bit address and then 8 bits of data to be written to the specified address. Refer to Section 7.0 *Register Descriptions* for configuration register functions and addresses.

The AHA4525 is configured for a fixed configuration mode with only 1 block code. An example of this configuration mode is shown in Figure 4.

If system constraints make it necessary to allow for dead time in between configuration writes you may use the configuration timing shown in Figure 5. Cycle 1, 2, 3 are normal configuration writes followed by a register 0x26 write. Register 0x26 must have the "**last**" bit set and the **xNoBlock** bit set.

The fixed configuration mode is selected during the initial configuration following **RESET_N**. When a 1 is written to the **xNoConfig** bit in the control register (Refer to Section 7.7.1 *Control*), all data blocks are received as config 0. After the chip is configured, there is no access to the configuration registers until **RESET_N** is asserted.

2.2 CONFIGURATION READS

All encoder configuration read requests are written through the encoder unencoded data input port. All decoder configuration read requests are written through the decoder channel data input port.

The encoder configuration registers can only be read when the encoder datapath is empty. The decoder configuration registers can only be read when the decoder datapath is empty. The AHA4525 has a general output pin that may be used to monitor the datapath empty status (refer to Section 7.7.1 *Control*).

All encoder configuration registers are read from the encoder encoded data output port. All decoder configuration registers are read from the decoder decoded data output port.

A configuration data read request is written in a 16 bit configuration cycle format (refer to Section 2.3 *Configuration Cycle Format*). The configuration cycle starts with a "rwn" bit to indicate whether the current configuration cycle is a read or write cycle. When the "rwn" bit is set, the configuration cycle is a read request and it is by default the last configuration cycle. The "last" bit is irrelevant in a configuration read because each read must wait for the read data to be output before a new read can be requested. Data cannot directly follow a read configuration request.

The "last" bit is followed by a 6 bit address to specify which register to read. The 8 bits of data that follow the address field of the configuration read cycle are don't cares, but they must be input to finish the configuration cycle (refer to Section 7.0 *Register Descriptions* for configuration register functions and addresses).

2.3 CONFIGURATION CYCLE FORMAT

Bit 15 is input first, bit 0 is input last.

bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
rwn	**last**	**a[5]**	**a[4]**	**a[3]**	**a[2]**	**a[1]**	**a[0]**	**d[7]**	**d[6]**	**d[5]**	**d[4]**	**d[3]**	**d[2]**	**d[1]**	**d[0]**

"rwn" -	Read/Write_not. When "rwn" = 1, the configuration cycle is a read cycle. When "rwn" = 0, the configuration cycle is a write cycle.
"last" -	Last configuration cycle. When set, the current configuration cycle is the last configuration cycle before data begins. When cleared, the current configuration cycle will be followed by another configuration cycle.
"a[5:0]" -	Configuration address. Address of configuration read or write.
"d[7:0]" -	Configuration data. When "rwn" = 0, this is the data to be written to the addressed configuration register. When "rwn" = 1, this data is a don't care.

2.4 DECODER STATUS OUTPUT

The AHA4525 has the option to output status information with each decoded block. There is no option to output any status information with encoded blocks.

The status of a decoded block is output as 16 bits at the end of every decoded block when the **DStatus** configuration bit is set to 1 (see Section 7.7.1 *Control*). The structure of the status output is shown in Section 2.5 *Decoder Status Output Format*. The status is output in serial, msb first.

The status includes a parity error flag "perr" which is asserted when the CRC is enabled and the output block failed the CRC verification.

The last status information is the number of corrections "c[11:0]". This is a count of the number of bit errors corrected in the current block, including user data, inserted CRC bits, and ECC bits.

2.5 DECODER STATUS OUTPUT FORMAT

Bit 15 is output first, bit 0 is output last.

bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
res			**perr**	**c[11]**	**c[10]**	**c[9]**	**c[8]**	**c[7]**	**c[6]**	**c[5]**	**c[4]**	**c[3]**	**c[2]**	**c[1]**	**c[0]**

"perr" -	Parity error. Asserted when CRC is enabled and the output block failed the CRC verification.
"c[11:0]" -	Correction count. The number of bit errors corrected in the current block including user data, inserted CRC bits, and ECC bits.

2.6 ENCODER AND DECODER - SYNCHRONOUS I/O - FULL DUPLEX

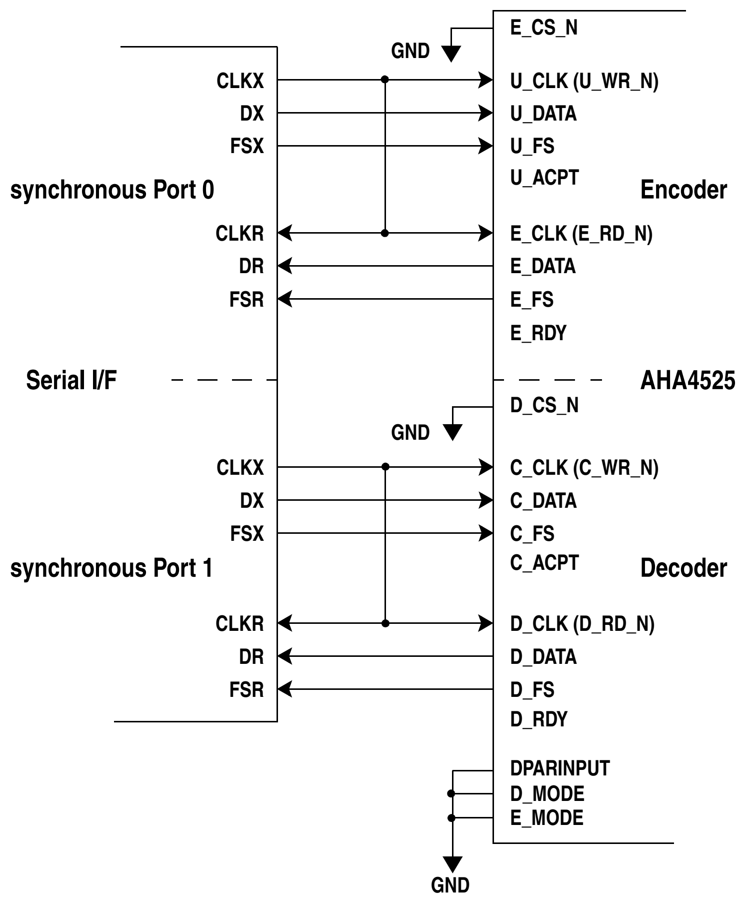
An AHA4525 synchronous interface full duplex connection diagram is shown in Figure 6. Data blocks and configuration information are input through the **U_DATA** port when encoding or the **C_DATA** port when decoding.

The soft decision channel data into the decoder is serially input when **DPARINPUT** = 0 on the rising edge of **RESET_N**, which reduces the input rate by up to a factor of 4 when a 4 bit soft metric is used. The channel input port may be configured to allow 4 bit parallel input on **C_DATA** when **DPARINPUT** = 1 on the rising edge of **RESET_N**. When **DPARINPUT** = 1, the **C_DATA** port is configured in parallel mode and the configuration data is expected to be received in 4 bit parallel. When **DPARINPUT** = 0, the **C_DATA** port resets into serial mode and expects to receive configuration data serially.

The synchronous interface timing for the decoder signals is shown in Figures 7 and 8. The signal timing for the encoder is similar.

A data or configuration write is started when **C_FS** is asserted for one **C_CLK** cycle. The **C_FS** signal is a sync signal that is asserted once for every block when the **DWordEqBlk** configuration bit is set to 1 or asserted once for every word when the **DWordEqBlk** configuration bit is set to 0 (see Section 7.6.1 *Transfer Word Size*). The word size is configured using the **DWordSize[4:0]** register also shown in Section 7.6.1 *Transfer Word Size*. The **DWordSize[4:0]** register resets to 16 bits per word. Figure 27 shows the input timing requirements for **C_FS**.

Figure 6: Full Duplex Connection Diagram



All AHA4525 configuration is done immediately following the first **C_FS** of a block and the interface remains in the configuration mode until a configuration cycle is input with the “last” bit set. The structure of a configuration cycle is shown in Section 2.3 *Configuration Cycle Format* and in the timing diagram Figure 7 in the **C_DATA** input data stream.

In Figure 7, A[3:0] and B[3:0] following the configuration cycle indicate the start of the data block where A is a 4 bit soft metric and B is a 4 bit soft metric.

If the configuration cycle is a read request, the read data is available on **D_DATA** on the **D_CLK** cycle following when the **D_FS** output signal is asserted for one **D_CLK** cycle. Data cannot follow a configuration read and decoder configuration reads are only allowed when the decoder datapath is empty. The read data must be read from the **D_DATA** port before another configuration cycle is started.

The **D_FS** signal timing is shown in Figures 7 and 8 and the electrical characteristics are shown in Figure 30. **D_FS** asserts to indicate that the block is finished processing and the output data will start on the next **D_CLK** cycle. The **D_FS** signal is a sync signal that is asserted once for every block when the **DWordEqBlk** configuration bit is set to 1 or asserted once for every word when the **DWordEqBlk** configuration bit is set to 0 (see Section 7.6.1 *Transfer Word Size*). The word size is configured using the **DWordSize[4:0]** register also shown in Section 7.6.1 *Transfer Word Size*. The **DWordSize[4:0]** register resets to 16 bits per word. The output data is read serially through the **D_DATA** port.

D_RDY asserts with the **D_FS** at the start of a block and stays asserted until the last data, including block status, is clocked out of the decoder (see Figures 7 and 8). Note that the encoder does not output a block status footer. **D_RDY** is not required for systems where the receive port is set up to expect a certain output block size, but it may be used in systems that do not count the bits in the received block.

C_ACPT asserts when the input buffer is ready to accept a block and is deasserted after the data portion of the block transfer is started or a configuration read is started (see Figures 7 and 8). This signal may be used to allow the decoder to process more than one block at a time. **C_ACPT** is not required for systems that process one block at a time (i.e. the first block is input, processed, and output before a second block is input).

The synchronous interface requires continuous port clocks **C_CLK** and **D_CLK**. Gated port clocks are not allowed. The continuous **D_CLK** automatically fills any padding required by a serial receive port.

The following timing diagrams Figure 7 and Figure 8 show a block input and output through the decoder in serial mode (**DPARINPUT** = 0). The **C_FS** and **D_FS** signals shown in the following diagrams are shown assuming **DWordEqBlk** is set.

Figure 7: Synchronous Data/Configuration Communication Timing - Start of Block

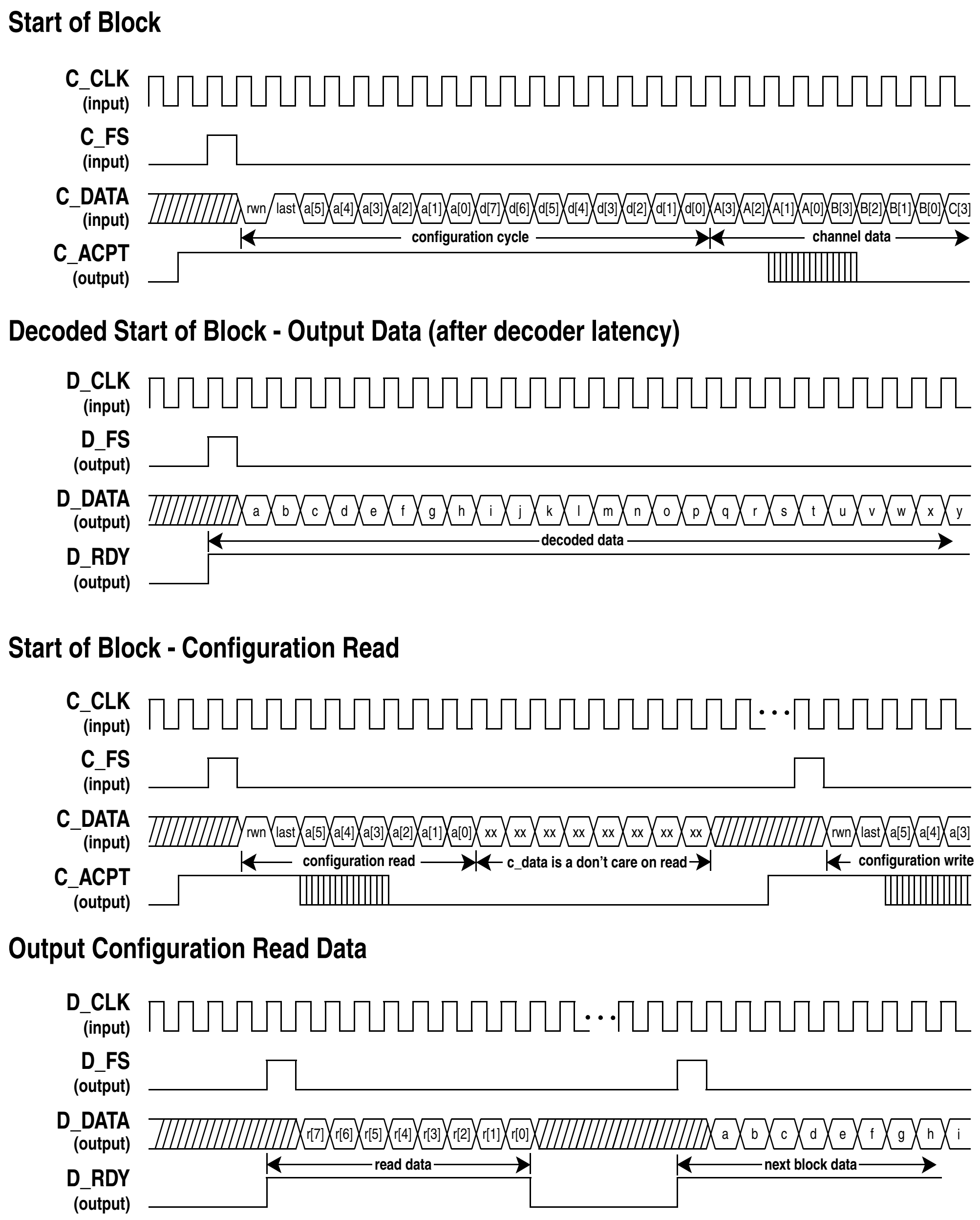
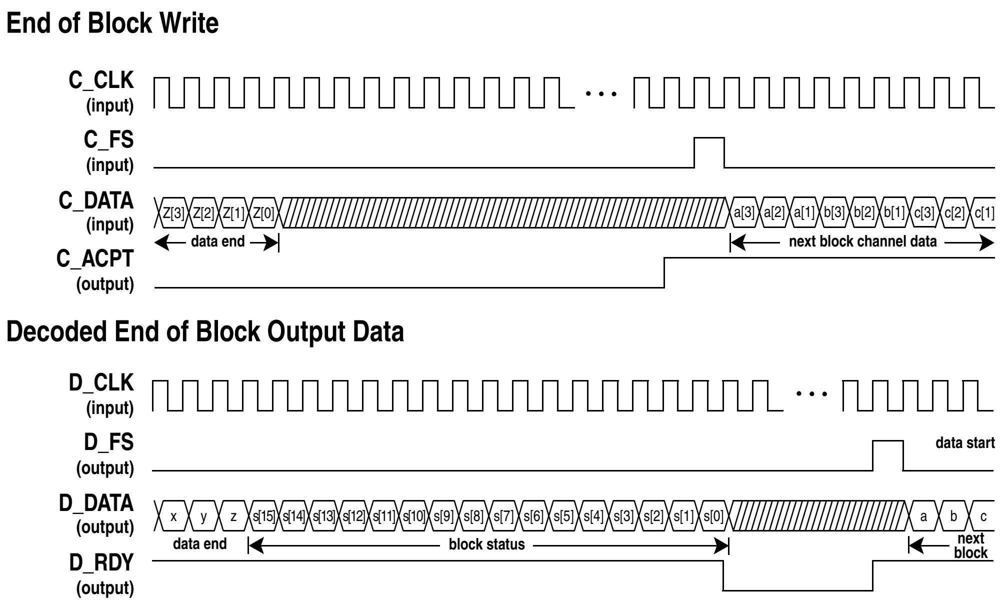

Start of Block
C_CLK (input)
C_FS (input)
C_DATA (input)
rwn last a[5] a[4] a[3] a[2] a[1] a[0] d[7] d[6] d[5] d[4] d[3] d[2] d[1] d[0] A[3] A[2] A[1] A[0] B[3] B[2] B[1] B[0] C[3]
configuration cycle
channel data
C_ACPT (output)
Decoded Start of Block - Output Data (after decoder latency)
D_CLK (input)
D_FS (output)
D_DATA (output)
a b c d e f g h i j k l m n o p q r s t u v w x y
decoded data
D_RDY (output)
Start of Block - Configuration Read
C_CLK (input)
C_FS (input)
C_DATA (input)
rwn last a[5] a[4] a[3] a[2] a[1] a[0] xx xx xx xx xx xx xx xx rwn last a[5] a[4] a[3]
configuration read
c_data is a don't care on read
configuration write
C_ACPT (output)
Output Configuration Read Data
D_CLK (input)
D_FS (output)
D_DATA (output)
r[7] r[6] r[5] r[4] r[3] r[2] r[1] r[0] a b c d e f g h i
read data
next block data
D_RDY (output)

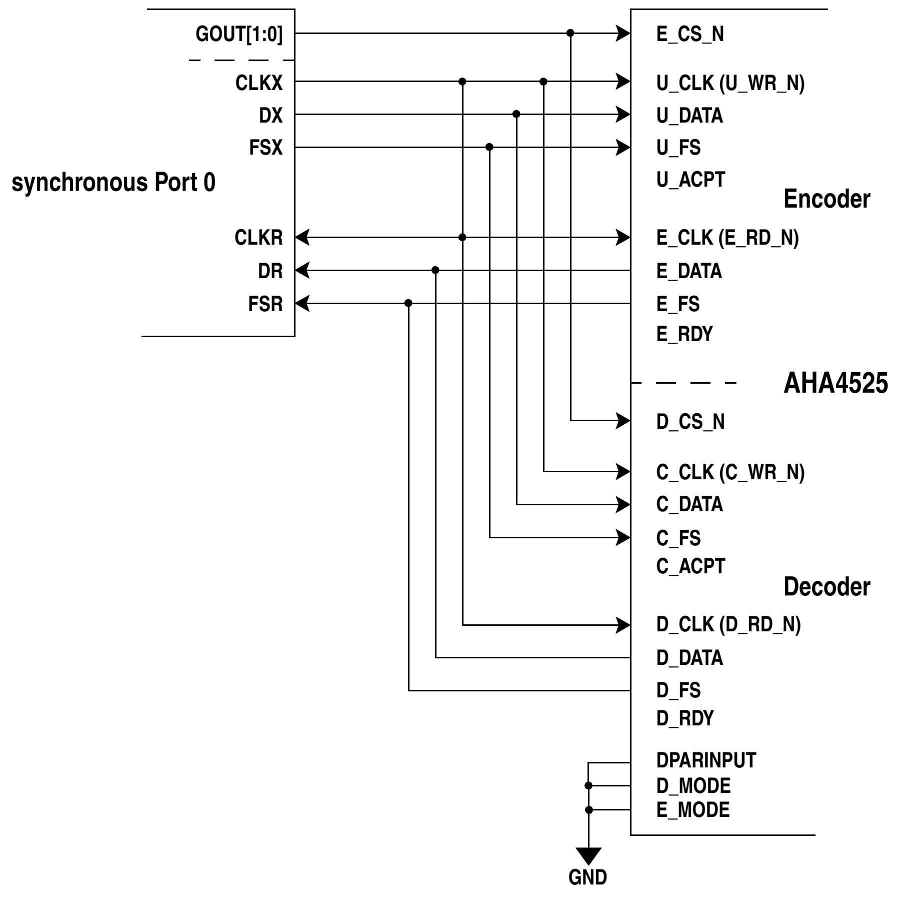
Figure 8: Synchronous Data/Configuration Communication Timing - End of Block



2.7 ENCODER AND DECODER SYNCHRONOUS - HALF DUPLEX

This interface is the same as the full duplex synchronous mode except the **E_CS_N** and **D_CS_N** inputs are used to switch between the encoder and decoder.

Figure 9: Half Duplex Connection Diagram



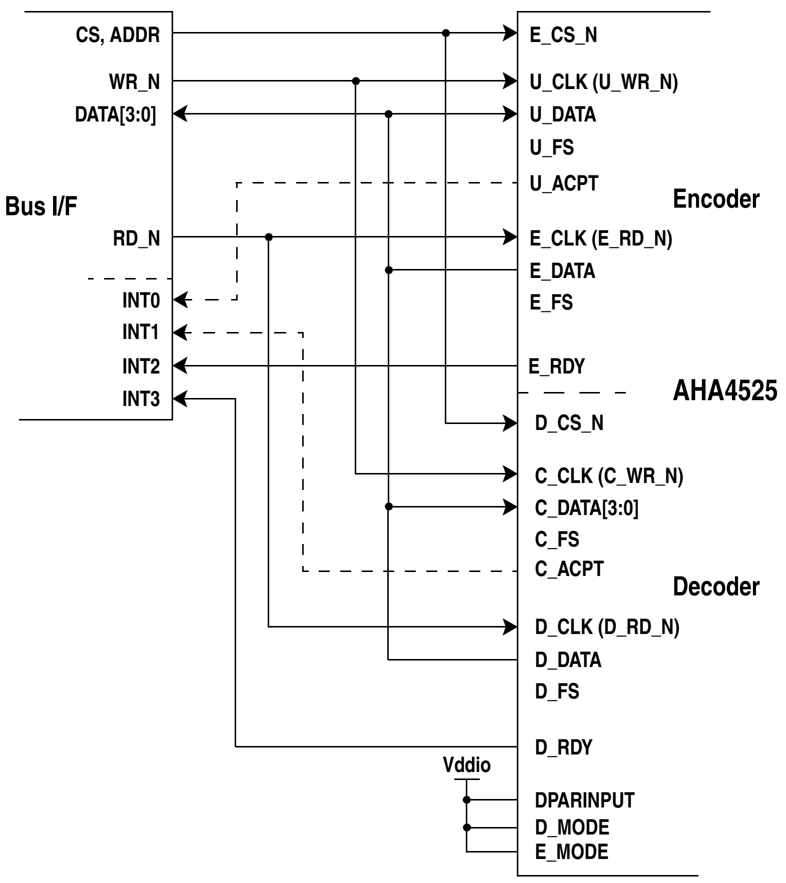
The differences in the communication timing between the full duplex and half duplex synchronous configurations are the tristated **D_FS** and **D_DATA** outputs when the decoder **D_CS_N** is not asserted, and the tristated **E_FS** and **E_DATA** outputs when the encoder **E_CS_N** is not asserted. Refer to the full duplex synchronous configuration Section 2.6 *Encoder and Decoder - Synchronous I/O - Full Duplex* for communication timing diagrams.

2.8 BUS MODE - HALF DUPLEX

The AHA4525 bus interface half duplex connection diagram is shown in Figure 10. The bus interface is selected for the encoder when **E_MODE** = 1 and is selected for the decoder when **D_MODE** = 1. Data blocks and configuration information are input through the **U_DATA** port when encoding or the **C_DATA** port when decoding.

The bus mode interface allows a 4 bit parallel channel input through the **C_DATA** port to allow soft metrics to be written to the decoder at a rate of one symbol per transfer. The interfaces to **D_DATA**, **U_DATA**, and **E_DATA** are serial. When bus mode is selected at reset, the status of the **DPARINPUT** pin determines if the **C_DATA** port is configured in serial or 4 bit parallel mode. When **DPARINPUT** = 1 at reset, the **C_DATA** port is configured in 4 bit parallel mode and the first configuration cycle is expected to be received in 4 bit parallel.

Figure 10: Half Duplex Connection Diagram



The bus interface timing for the decoder signals is shown in Figure 11. The signal timing for the encoder is typical.

A data or configuration write cycle is started when **C_WR_N** is asserted low. There is no block start signal in bus mode. The **C_WR_N** signal is a write strobe signal, with data being registered on the rising edge (see Figure 33).

All configuration is done immediately following reset. The AHA4525 counts the number of input data bits and compares the input count to the **DBlockSize** configuration setting (see Section 7.4.2 *Block Size*) to find the end of an input block.

The interface remains in the configuration mode until a configuration cycle is input with the "last" bit set. The structure of a configuration cycle is shown in Section 2.3 *Configuration Cycle Format*. The configuration cycle is shown as c[3:0] in Figure 11 in the **C_DATA** input stream. Note that the encoder does not have the parallel input option and all configuration bits must be input serially. The first 16 bits (4 writes) in the **C_DATA** stream are configuration write bits with bits (15:11) written first and bits (3:0) written last. The configuration starts with a "rwn" (read/write_not) bit and is followed by a "last" bit which indicates that the current configuration cycle is the last configuration cycle. The "last" bit is followed by a 6 bit address a[5:0] and then the data d[7:0] to be written to the address.

The AHA4525 expects data to immediately follow the last configuration write cycle unless the last configuration cycle writes a one to the **NoBlock** bit (see configuration Section 7.7.1 *Control*). A and B following the configuration cycle indicate the start of the data block where A is a 4 bit soft metric and B is a 4 bit soft metric.

If the configuration cycle is a read request, the read data is available on **D_DATA** when **D_RDY** is asserted. The configuration data is read when **D_RD_N** is asserted low and the data is valid until the **D_RD_N** rising edge (see Figure 35). Data cannot follow a configuration read and decoder configuration reads are only allowed when the decoder datapath is empty. Each configuration read is considered a last configuration cycle similar to a last configuration write cycle with the **NoBlock** bit set. The read data must be read from the **D_DATA** port before another configuration cycle is started.

D_RDY asserts when valid data is available to be read at the output and stays asserted until the last data, including block status, is strobed out of the decoder (see Figure 11). **D_RDY** can be monitored to know when to start an output read, or the output can be read after decoder block latency number of **SYSCLK**s. The decoder latency calculations are shown in Section 4.2.7 *Decoder Latency*. Any **D_RD_N** strobes beyond the end of a block are ignored until the **D_RDY** signal is asserted.

C_ACPT asserts when the input buffer is ready to accept a block and deasserts after the data portion of the block transfer is started or a configuration read is started (see Figure 11). This signal may be used to allow the decoder to process more than one block at a time. **C_ACPT** is not required for systems that process one block at a time (i.e. the first block is input, processed, and output before a second block is input).

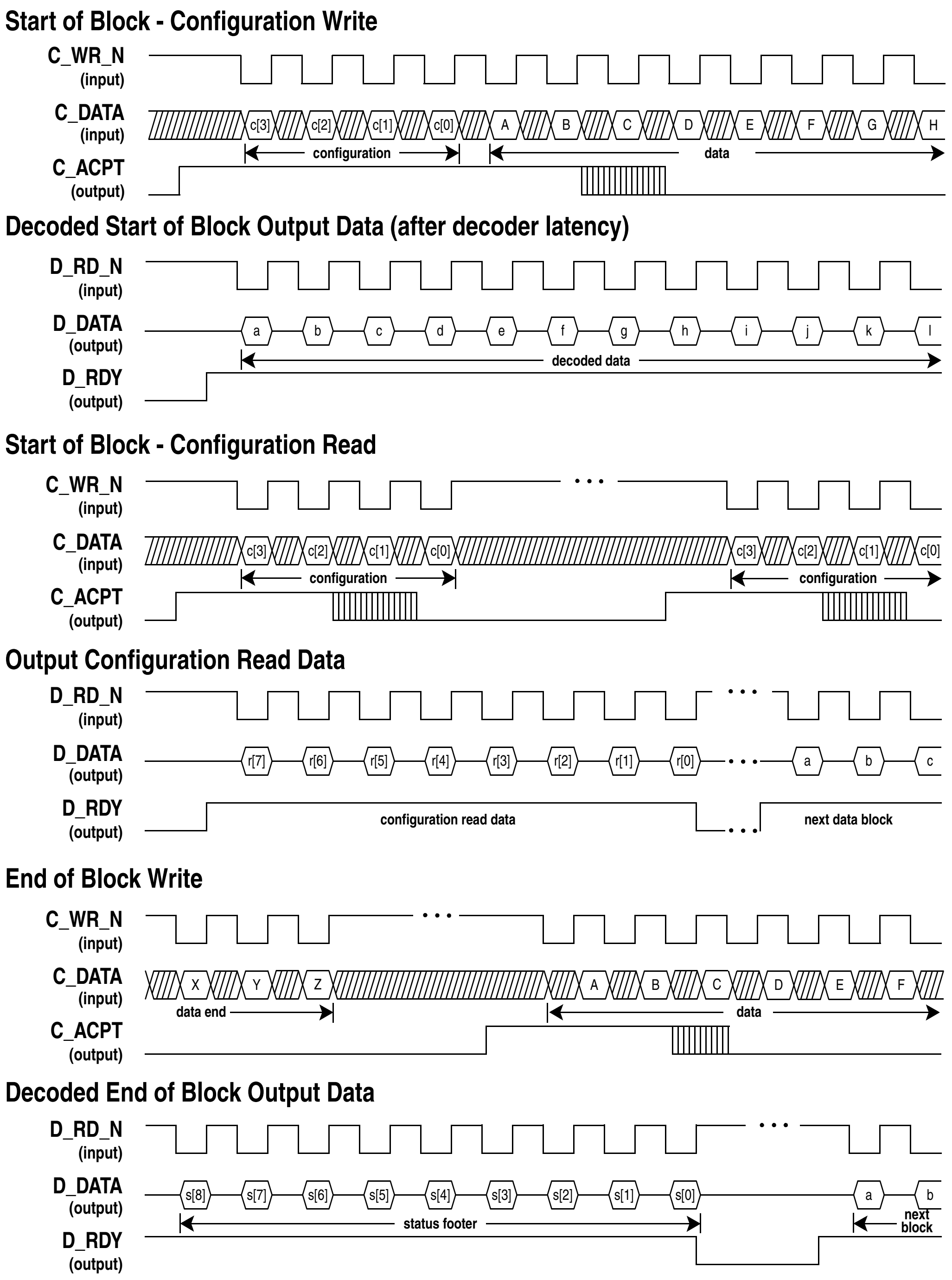
The status of the decoded block is output in 16 bits at the end of every decoded block when the **DStatus** configuration bit is set to 1 (see control Section 7.7.1 *Control*). The structure of the status output is shown in Section 2.5 *Decoder Status Output Format* and partially in the timing diagram Figure 11 in the **D_DATA** output data stream as s[8:0] (s[15:9] are not shown). There is not an option to output status information at the end of encoded blocks.

In bus mode, the AHA4525 is a slave to the system. The **C_WR_N** and **D_RD_N** signals are both inputs to AHA4525. Blocks do not need to be read or written in a continuous stream in bus mode.

The signal differences between synchronous mode and bus mode are:

a) the **C_CLK** and **U_CLK** inputs become **C_WR_N** and **U_WR_N** in bus mode,
b) the **D_CLK** and **E_CLK** inputs become **D_RD_N** and **E_RD_N** inputs in bus mode,
c) the **D_DATA** bus is tristated when **D_RD_N** is not asserted, and
d) the **E_DATA** bus is tristated when **E_RD_N** is not asserted.

Figure 11: Bus Interface Data/Configuration Communication Timing - Start of Block

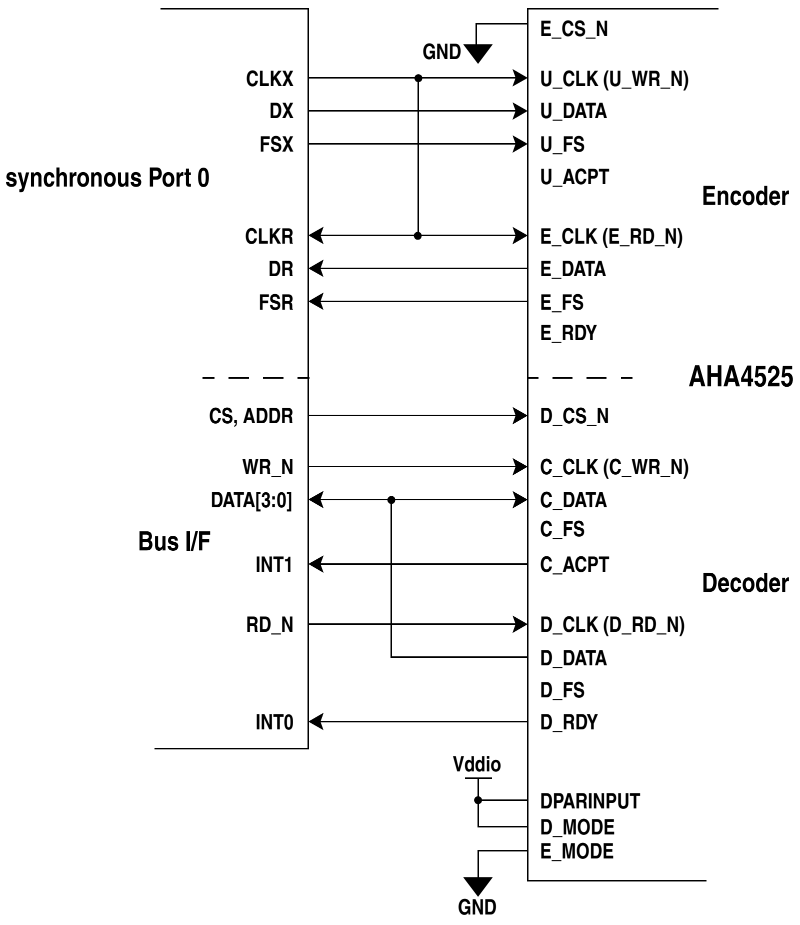


2.9 DECODER BUS INTERFACE, ENCODER SYNCHRONOUS INTERFACE - FULL DUPLEX

The full duplex connection diagram shown in Figure 12 is used to show the flexibility of the AHA4525 interface. The decoder uses the bus connection with a 4 bit parallel channel input while the encoder uses the synchronous port.

In this configuration, the **C_ACPT** output is used to signal that the **C_DATA** input is ready to accept data. The AHA4525 decoder resets into 4 bit parallel mode when **DPARINPUT** = 1, so the first decoder configuration write must be done in 4 bit parallel.

Figure 12: Full Duplex Connection Diagram



3.0 ENCODER

Figure 13: Encoder Block Diagram

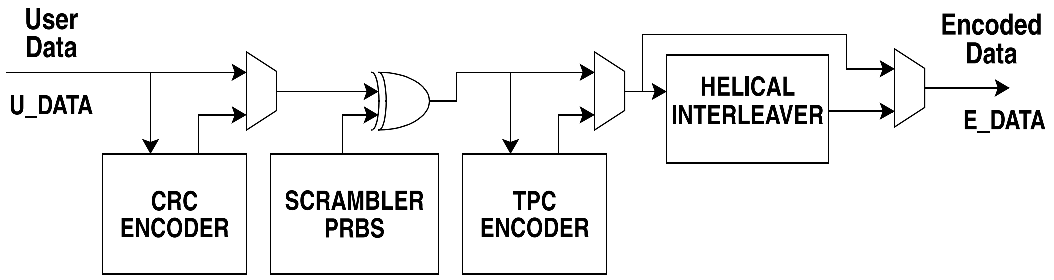


Figure 13 shows a block diagram of the encode path. Note that all of the blocks in the encode path except the helical interleaver operate by only inserting or modifying data in the stream. Therefore, the entire encode path has low latency.

Data is input serially through **U_DATA**. The CRC engine computes a CRC over each block of data which is inserted at the end of the uncoded data block. This data is scrambled by exclusive-ORing with the output of a Pseudo Random Binary Sequence (PRBS) generator. The scrambler ensures adequate bit transitions in the data stream, which are often required to allow improved DC balance and to accelerate clock recovery in the demodulator. The scrambled data is input to a TPC encoder, which computes ECC bits and inserts them at the appropriate locations in the data stream.

The helical interleaver improves burst error performance of the decoder. However, it adds a one block latency to the datapath. The encoded block is serially output through **E_DATA**.

3.1 CRC ENCODER

The cyclic redundancy check (CRC) encoder is a 32 bit linear feedback shift register with a programmable polynomial. Figure 14 shows a diagram of the shift register. Each TPC block has a separate CRC encoded with the resultant CRC bits appended to the block.

The polynomial for the CRC encoder is written into the **ECRCPoly** register. The highest order of the polynomial (defined by **ECRCSize**) is assumed to be a 1, and bit 0 of the polynomial register corresponds to the 0th order of the polynomial, as shown in Figure 14. The desired size of the CRC, in bits, is written into the **ECRCSize** register.

Table 1 gives a suggested list of polynomials for various length CRCs. The POLY column of table 1 gives the value to program into the **ECRCPoly** register. The detection capability column gives the probability that an incorrect block will be detected and flagged as incorrect by the CRC decoder.

Figure 14: CRC Encoder

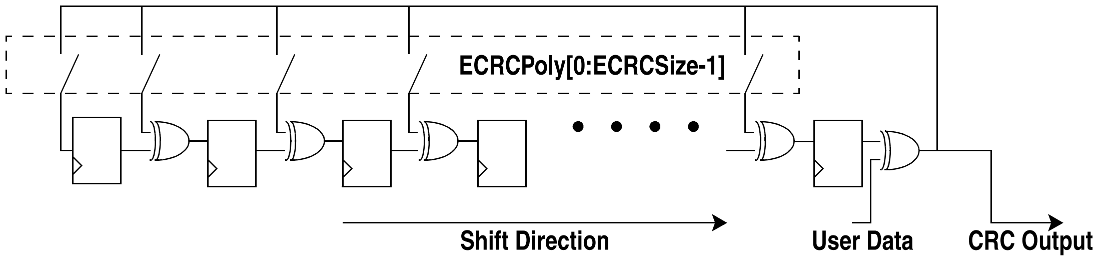


Table 1: Recommended CRC Polynomials

CRC	*SIZE (bits)*	*POLY Program Value (hex)**	*DETECTION CAPABILITY***
4	4	1f	0.9375
8	8	1d5	0.99609
12	12	180f	0.999756
ANSI	16	18005	0.999985
CCITT	16	11021	0.999985
SDLC	16	1a097	0.999985
24	24	1805101	0.9999999404
32A	32	1404098e2	0.99999999953
32B	32	104c11db7	0.99999999977

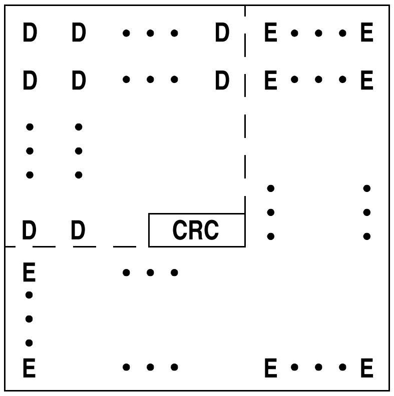
Notes:

*	*The leading '1' in these values is assumed by the device and should not be written to the register.*
**	*This detection capability is the probability that an incorrect block is marked in error. The probability of an undetected block is computed by multiplying the block error rate by (1 - Detection Capability).*

The shift register is reset to all 0s at the beginning of each TPC block. Data from the block is shifted into the circuit until the number of bits programmed into **EBlkSize** is reached. Note that the value written into **EBlkSize** does not include the CRC bits. After the entire TPC block is shifted into the CRC encoder, the data input and feedback of the CRC shifter are disabled, and the contents of the CRC registers are shifted out and inserted into the data stream.

Figure 15 shows the location of the CRC word inserted in a 2D TPC encoded block. D denotes data bits and E denotes computed error correction bits.

Figure 15: 2D TPC Encoded Block with CRC



3.2 SCRAMBLER

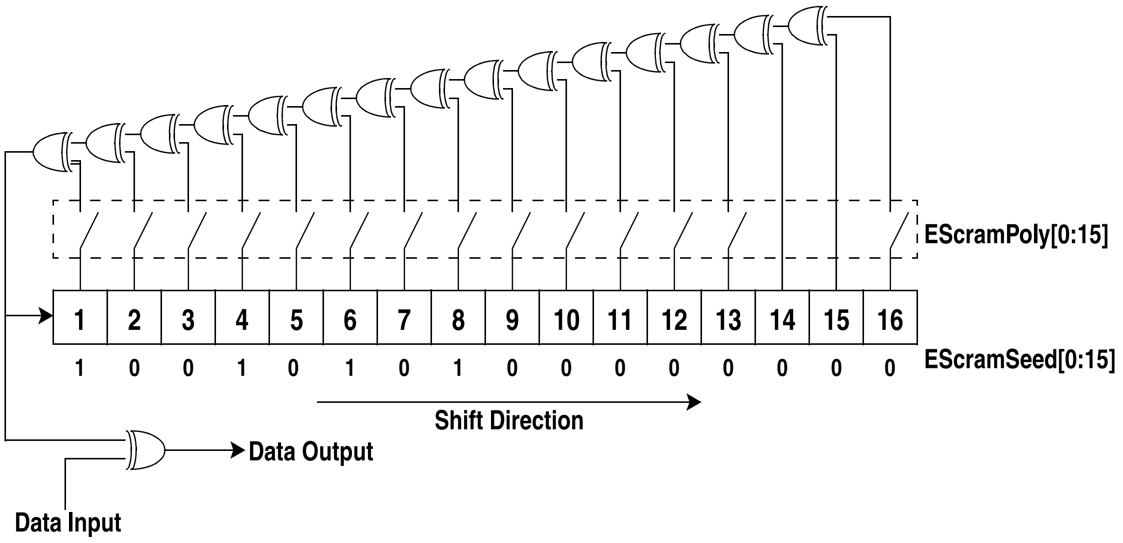
The scrambler is built with a 16 bit pseudo-random binary sequence generator with programmable polynomial, length, and initialization seed. Figure 16 shows the configuration of the scrambler. The shift register is clocked once for each bit. As shown in Figure 16, the output of the shift register is exclusive-ORed with the data to be scrambled.

Figure 16 shows an example configuration for the generator polynomial sequence:

$$1 + X^{14} + X^{15}$$

This sequence is programmed into the **EScramPoly** register as 0b0110000000000000. The seed for the shift register that is shown in the diagram is programmed into **EScramSeed** as 0b0000000010101001. Every time the scrambler is reset it is initialized with this seed value.

Figure 16: Scrambler



Note that the length of the generator is directly determined by the polynomial configuration. The sequence shown here is generated by a 15 stage shift register. Since the highest order of the polynomial is 15, the 16th bit XOR feedback will be disabled and only 15 bits of the shift register are used. Also note that the 0th order of the polynomial is assumed to be 1.

The scrambler is reset at the beginning of every TPC block.

3.3 TPC ENCODER

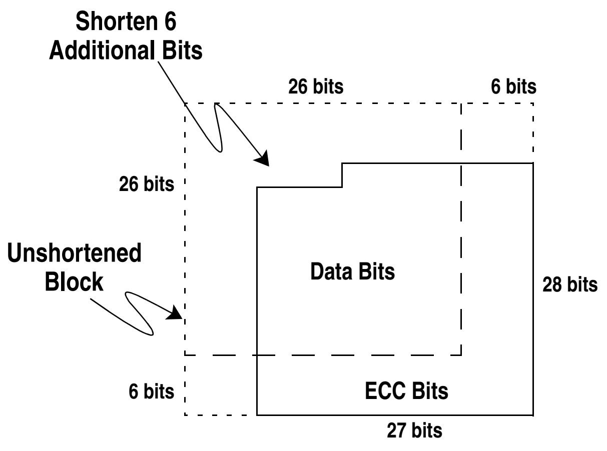
The TPC encoder supports 2D codes with constituent code lengths of up to 64 bits and overall block size up to 4096 bits. The encoder supports both extended Hamming and parity only constituent codes. See Section 7.4.1 *TPC Constituent Code* for a description of supported codes and shortening configurations.

3.3.1 ENCODER CODE SHORTENING

There are two methods of shortening product codes. The first method is to remove an entire row or column from a 2D code. This is equivalent to shortening the constituent codes that make up the product code and is accomplished by writing the amount to shorten into the **xShortX** and **xShortY** registers (see Section 7.4.4 *Shortening Configuration*). This method enables a coarse granularity on shortening and at the same time maintaining the highest code rate possible by removing both data and parity symbols. Further shortening is obtained by removing individual bits from the first row of a 2D code (using **xShortB**).

The following examples discuss shortening in a 2D code. Assume a 456 bit block size is required with code rate of approximately 0.6. The base code chosen before shortening is the (32,26)x(32,26) code which has a data size of 676 bits. Shortening all rows by 5 and all columns by 4 results in a (27,21)x(28,22) code with a data size of 462 bits. To get the exact block size, the first row of the product is shortened by an addition 6 bits. The final code is a (750,456) code with a code rate of 0.608. Figure 17 shows the structure of the resultant block. This shortening is programed into the device by writing **xShortX** to 5, **xShortY** to 4, and **xShortB** to 6.

Figure 17: Structure of Shortened Code



3.3.2 HELICAL INTERLEAVER

Helical interleaving transmits data in a helical fashion. When the channel introduces a burst of errors, the helical deinterleaver in the decoder will spread these errors across all axes of the code. The use of helical interleaving increases the burst error correcting capability of the code and increases the block latency (refer to Section 3.3.4 *Encoder Latency*).

The helical interleaver is enabled by setting **EHelical** to one (see Section 7.4.1 *TPC Constituent Code*). When helical interleaving is enabled, the **EShortX** and **EShortB** values must be set to 0 (see Section 7.4.4 *Shortening Configuration*). This constrains the shortening resolution to one row for 2D codes. All of the blocks in the encode datapath must be either interleaved or not interleaved. A mixture of interleaving blocks and non-interleaving blocks in the encoder datapath at the same time is not allowed.

Helical interleaving is applied along a diagonal path through the encoded block. Data is output along diagonal lines from the upper left to lower right corner. The first diagonal output starts with the bit row 1, column 1 followed by the diagonal starting at row 1, column 2.

The example below shows how interleaving is applied for a 2D (64,57)x(64,57) code.

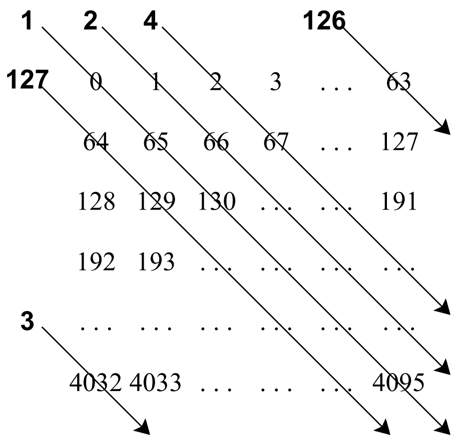
Figure 18: Input Block

0	1	2	3	. . .	63
64	65	66	67	. . .	127
128	129	130	. . .	. . .	191
192	193	. . .	. . .	. . .	. . .
. . .	. . .	. . .	. . .	. . .	. . .
4032	4033	. . .	. . .	. . .	4095

Note: The number reflects the bit order including generated ECC bits.

The encoded, interleaved data output is taken along diagonal lines starting with bit zero as shown below. The order of the interleaving is noted for each diagonal line.

Figure 19: 2D Interleaving



For the (64,57)x(64,57) block, the data output is: 0, 65, 130,..., 4095, 1, 66,..., 4031, 4032, 2, 67,...,..., 63, 64,..., 4094 for a total of 4096 bits output. The AHA4525 operating as a decoder deinterleaves the block to restore it to its original order.

Figure 20: Encoded/Interleaved Data Output

0	65	130	. . .	4030	4095
1	66	131	. . .	4031	4032
2	67	132	. . .	3968	4033
3	68	. . .	. . .	. . .	. . .
. . .	. . .	. . .	. . .	. . .	. . .
63	64	129	. . .	4029	4094

Data bits are output from the encoder in row order from left to right.

3.3.3 INTERNAL BUFFERING

An internal encoder buffer is used to allow data to stream into and out of AHA4525. The encoder buffer is always enabled because the data flow into the encoder is not throttled by AHA4525 in either bus mode or synchronous mode. The logical size of the encoder input buffer is set via **EBufferSize** (refer to Section 7.4.3 *Buffer Configuration (Encoder Only)* for an equation to calculate **EBufferSize**).

3.3.4 ENCODER LATENCY

The encoder latency is defined as the time from when the first un-encoded bit is input until the **E_RDY** signal is asserted to indicate that a block is ready to be output. When helical interleaving is disabled and **E_MODE** = 0, the approximate encoder latency in **SYSCLK**s when **PLLBYPASS** = 0 is:

$$\frac{(\mathbf{EBufferSize} \times 16) + 31}{2} + \left(2 \times \frac{\mathbf{SYSCLK}\ (\mathrm{MHz})}{\mathbf{U\_CLK}\ (\mathrm{MHz})}\right) + \left(2 \times \frac{\mathbf{SYSCLK}\ (\mathrm{MHz})}{\mathbf{E\_CLK}\ (\mathrm{MHz})}\right)$$

When helical interleaving is disabled and **E_MODE** = 1, the approximate encoder latency in **SYSCLK**s is:

$$\frac{(\mathbf{EBufferSize} \times 16) + 36}{2}$$

The **EBufferSize** input buffer is used to guarantee a streaming input and output.

Helical interleaving adds one block of latency. When helical interleaving is enabled, the encoder input buffer is not used. The worst case encoder latency in **SYSCLK**s when helical interleaving is enabled and **E_MODE** = 0 is:

$$\left(1 \text{ encoded block size} \times \frac{\textbf{SYSCLK}\text{ (MHz)}}{\textbf{U\_CLK}\text{ (MHz)}}\right) + \left(2 \times \frac{\textbf{SYSCLK}\text{ (MHz)}}{\textbf{U\_CLK}\text{ (MHz)}}\right) + 24 + \left(2 \times \frac{\textbf{SYSCLK}\text{ (MHz)}}{\textbf{E\_CLK}\text{ (MHz)}}\right)$$

When helical interleaving is enabled and **E_MODE** = 1, the worst case encoder latency in **SYSCLK**s is:

$$\left(1 \text{ encoded block size} \times \frac{\textbf{SYSCLK}\text{ (MHz)}}{\textbf{U\_CLK}\text{ (MHz)}}\right) + 26$$

If the PLL is bypassed (**PLLBYPASS** = 1), then the latency result in **SYSCLK**s must be multiplied by 4.

4.0 DECODER

The decode path of AHA4525 includes a counterpart for each encoder module. The encoder and decoder are isolated paths. This allows full duplex operation where the encoder and decoder may be operating with different code types and data rates.

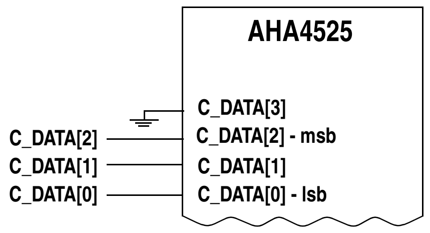
4.1 CHANNEL INTERFACE

The channel interface formats the channel data for decoding by the Turbo Product Code decoder. For best decoder performance, soft metric information from the channel is necessary.

4.1.1 CHANNEL INPUT FORMATTING

When decoding, data may be input either serially or one quantization value per handshake on **C_DATA[q-1:0]** where q is the number of input quantization bits. The number of quantization bits is configurable based on the setting of **QBIT[1:0]** within the Quantization register in Section 7.5.1. The **QMODE[1:0]** bits within the Quantization register determine the type of input data. The input data may be 2's complement, sign/magnitude, or unsigned. All unused **C_DATA** inputs are ignored except in configuration cycles. Note that all **C_DATA[3:0]** inputs are used for configuration cycles in parallel mode. In parallel configuration cycles, configuration bits (15:11) are written first with configuration bit 15 input on **C_DATA[3]**. In parallel data cycles, the lsb of the soft metric value is input on **C_DATA[0]** and the msb is determined by **QBits[1:0]**. Figure 21 shows example connections when **QBits[1:0]** = "11" (3 input bits).

Figure 21: C_DATA Interface



For QBits = "11" connect C_DATA as shown

4.2 TPC DECODER

The Turbo Product Code decoder supports block sizes up to 4096 encoded bits. The decoder supports iterative decoding of 2D codes built from extended Hamming or parity only constituent codes of length up to 64 bits.

4.2.1 HELICAL DEINTERLEAVER

The helical deinterleaver is enabled by setting **DHelical** to 1 in the configuration register shown in Section 7.4.1 *TPC Constituent Code*. See Section 3.3.2 for a description of helical interleaving.

4.2.2 CODE CONFIGURATION

Turbo Product Codes are specified by the constituent codes of each axis in the code. The code configuration registers in Section 7.4.1 *TPC Constituent Code* specify the code type (extended Hamming or Parity) and length of each axis.

To generate a specific block size, the product code is shortened. Shortening is discussed in detail in Section 4.2.3.

The decoder can be configured to run a specific number of iterations by programming the number of **Iterations** in the configuration register shown in Section 7.4.6 *Iterations (Decoder Only)*. The decoder can be set to detect when a block converges and stop iterating. This allows the decoder to spend more time, up to the number of iterations specified in **Iterations**, on difficult blocks and less time on less difficult blocks. This feature is enabled by setting **StopIter** in the configuration register shown in Section 7.4.5 *Feedback (Decoder Only)*.

4.2.3 DECODER CODE SHORTENING

Refer to Section 3.3.1 *Encoder Code Shortening*.

4.2.4 CORRECTIONS COUNT

In order to assist the system with estimation of channel bit error rates, the AHA4525 device reports the number of bits corrected in each block decoded.

The **Corrections** register (see Section 7.7.2 *Status and Correction Count (Decoder Only)*) contains the number of corrections between incoming (channel) data and outgoing (decoded) data. This corrections value is the corrections done on all bits in the TPC block including user data, CRC, and TPC ECC bits. The correction count is updated in the **Corrections** register after each block is decoded and is also optionally output at the end of every block as a status footer (see Section 2.5 *Decoder Status Output Format*).

4.2.5 DESCRAMBLER

The descrambler is built with a 16 bit pseudo-random binary sequence generator with programmable polynomial, length, and initialization seed. The descrambler logic matches that of the scrambler shown in Figure 16 (refer to Section 3.2).

The generator polynomial sequence is programmed into the **DScramPoly** register. The seed for the shift register shown in Figure 16 is programmed into **DScramSeed**. Every time the descrambler is reset, it initializes to the **DScramSeed** value. The descrambler is reset at the start of every block.

4.2.6 CRC CHECKING

The CRC parity bits are checked after decoding. Each packet has a separate CRC with variable length up to 32 bits. The CRC comparator circuit matches that of the encoder shown in Figure 14.

The shift register is reset at the beginning of each TPC block. Data from the TPC block is shifted into the circuit until the number of bits programmed into **DBlkSize** is reached. At this point, the contents of the CRC shift register are examined. If all bits are 0, then the CRC is correct. Otherwise, a CRC error is detected and the **CRCErr** status flag is set in the decoder status footer (see Section 2.5 *Decoder Status Output Format*). The **CRCErr** flag can also be read from the status register shown in Section 7.7.2 *Status and Correction Count (Decoder Only)*.

4.2.7 DECODER LATENCY

The decoder latency is defined as the time from when the last channel bit is input until the **D_RDY** signal is asserted to indicate that a block is ready to be output. In the following equations, **DCodeX** is **DCodeX[2:0]**, **DCodeY** is **DCodeY[2:0]**. The approximate decoder latency in **SYSCLK**s when **PLLBYPASS** = 0 is:

If **DCodeX** is not 0,

$$\text{x\_latency} = \frac{2^{\mathbf{DCodeX}} \times (2^{\mathbf{DCodeY}} - \mathbf{DShortY})}{4} + 2^{\mathbf{DCodeX}} + 16$$

If **DCodeX** = 0, x_latency = 0.

If **DCodeY** is not 0,

$$y\text{\_latency} = \frac{2^{\mathbf{DCodeY}} \times (2^{\mathbf{DCodeX}} - \mathbf{DShortX})}{4} + 2^{\mathbf{DCodeY}} + 16$$

If **DCodeY** = 0, y_latency = 0.

$$\text{iter\_clocks} = (\text{x\_latency} + \text{y\_latency}) \times \mathbf{Iterations}$$

When **D_MODE** = 1,

$$\text{latency} = 29 + \frac{\text{iter\_clocks}}{2} + \frac{\mathbf{DShortB}}{4}$$

When **D_MODE** = 0,

$$\text{latency} = \frac{2 \times \mathbf{SYSCLK}\ (\text{MHz})}{\mathbf{C\_CLK}\ (\text{MHz})} + 27 + \frac{2 \times \mathbf{SYSCLK}\ (\text{MHz})}{\mathbf{D\_CLK}\ (\text{MHz})} + \frac{\text{iter\_clocks}}{2} + \frac{\mathbf{DShortB}}{4}$$

If the PLL is bypassed (**PLLBYPASS** = 1), then the latency result in **SYSCLK**s must be multiplied by 4.

Table 2 gives an abridged list of possible codes supported by AHA4525, along with the latency in **SYSCLK**s for 4, 6, and 12 iterations with the CRC disabled. This table applies to a synchronous (**D_MODE** = 0) port configuration with the internal PLL enabled (**PLLBYPASS** = 0) and assumes that the frequency of **C_CLK** = **SYSCLK**. This is a very small subset of supported codes. Comtech AHA Corporation (CAC) can provide software to assist the code selection process.

Table 2: Partial Code List and Decoder Datapath Latency

CODE (X)x(Y)x(Z)	BLOCK SIZE (bits)	DATA SIZE (bits)	4 ITERATIONS (SYSCLKs)*	6 ITERATIONS (SYSCLKs)*	12 ITERATIONS (SYSCLKs)*
(64,63)x(64,63)	4096	3969	4445	6665	13277
(64,57)x(64,57)	4096	3249	4445	6653	13277
(32,31)x(32,31)	1024	961	1245	1853	3677
(32,26)(x(32,26)	1024	676	1245	1853	3677
(64,57)x(32,26)	2048	1482	2333	3485	6941
(64,57)x(64,63)	4096	3591	4445	6665	13277
(64,57)x(32,31)	2048	1767	2333	3485	6941
(32,26)x(16,11)	512	286	701	1037	2045
(32,26)x(16,15)	512	390	701	1037	2045

* *The number of* SYSCLK*s is assuming the PLL is enabled (PLLBYPASS=0) and is accurate to within 2%. If the PLL is not enabled (PLLBYPASS=1), then the latency in SYSCLKs must be multiplied by 4.*

Table 3: Partial Code List and Performance - Synchronous Interface

CODE (X)x(Y)x(Z)	BLOCK SIZE (bits)	DATA SIZE (bits)	CODE RATE	6 ITER CODING GAIN (dB)	4 ITER Channel/ Data* (Avg Mbit/sec)	6 ITER Channel/ Data* (Avg Mbit/sec)	12 ITER Channel/ Data* (Avg Mbit/sec)
(64,63)x(64,63)	4096	3969	0.969	3.2	46.4/45.0	31.0/30.0	15.5/15.0
(64,57)x(64,57)	4096	3249	0.793	7.3	46.4/36.9	31.0/24.5	15.5/12.3
(32,31)x(32,31)	1024	961	0.938	3.2	42.3/39.6	28.1/26.4	14.1/13.3
(32,26)x(32,26)	1024	676	0.660	7.2	42.3/27.9	28.1/18.6	14.1/9.3
(64,57)(x(32,26)	2048	1482	0.724	7.2	44.5/32.3	29.6/21.5	14.9/10.8
(64,57)x(64,63)	4096	3591	0.877	5.4	46.4/40.8	31.0/27.1	15.5/13.6
(64,57)x(32,31)	2048	1767	0.863	5.4	44.5/38.4	29.6/25.6	14.9/12.8
(32,26)x(16,11)	512	286	0.559	6.8	38.4/21.4	25.5/14.3	12.8/7.1
(32,26)x(16,15)	512	390	0.762	5.4	38.4/29.3	25.5/19.5	12.8/9.8

* *The input and output rates assume a 50 MHz SYSCLK frequency when the PLL is enabled or a 200 MHz SYSCLK frequency when the PLL is bypassed. The port clocks C_CLK and D_CLK are assumed to be running at 95 MHz.*

Table 4: Partial Code List and Performance - Bus Interface

CODE (X)x(Y)x(Z)	BLOCK SIZE (bits)	DATA SIZE (bits)	CODE RATE	6 ITER CODING GAIN (dB)	4 ITER Channel/ Data* (Avg Mbit/sec)	6 ITER Channel/ Data* (Avg Mbit/sec)	12 ITER Channel/ Data* (Avg Mbit/sec)
(64,63)x(64,63)	4096	3969	0.969	3.2	44.3/43.0	29.6/28.6	14.9/14.4
(64,57)x(64,57)	4096	3249	0.793	7.3	44.3/35.3	29.6/23.4	14.9/11.8
(32,31)x(32,31)	1024	961	0.938	3.2	40.4/37.9	26.9/25.3	13.5/12.6
(32,26)x(32,26)	1024	676	0.660	7.2	40.0/26.3	26.9/17.8	13.5/8.9
(64,57)x(32,26)	2048	1482	0.724	7.2	42.5/30.8	28.3/20.5	14.3/10.3
(64,57)x(64,63)	4096	3591	0.877	5.4	44.3/38.9	29.6/25.9	14.9/13.0
(64,57)x(32,31)	2048	1767	0.863	5.4	42.5/36.6	28.3/24.5	14.3/12.1
(32,26)x(16,11)	512	286	0.559	6.8	36.6/20.4	24.4/13.6	12.1/6.8
(32,26)x(16,15)	512	390	0.762	5.4	36.6/28.0	24.4/18.6	12.1/9.3

* *The input and output rates assume a 50 MHz SYSCLK frequency when the PLL is enabled or a 200 MHz SYSCLK frequency when the PLL is bypassed. The read and write strobes C_WR_N and D_RD_N are assumed to be running at 50 MHz.*

4.2.8 CODE PERFORMANCE

Tables 3 and 4 give code rate, coding gain, and maximum channel and user (decoded) data rates for the same abridged list of possible codes that was shown in Table 2. The data rate calculations assume that the frequency of **SYSCLK** is 50 MHz. The coding gain is measured on a Binary Input Additive White Gaussian Noise (AWGN) channel at 10^{-6} Bit Error Rate (BER) and 6 iterations. CAC can provide software to assist in code BER performance analysis.

5.0 PHASE LOCK LOOP

The AHA4525 contains an internal PLL which is used to multiply the input **SYSCLK** frequency by 4 for internal clocking. The internal PLL has a minimum input **SYSCLK** frequency of 20 MHz and a maximum input **SYSCLK** frequency of 50 MHz (see Section 11.1 *SYSCLK Clock Timing*). The PLL can not lock to frequencies below 20 MHz.

The internal PLL is enabled when pin **PLLBYPASS** = 0. If **PLLBYPASS** = 1, the internal PLL is bypassed and the input **SYSCLK** directly drives the internal logic.

When the PLL is bypassed, the latency and throughput are reduced by a factor of 4 given the same input **SYSCLK** frequency. The throughput and latency are identical when the PLL is bypassed and a 4x **SYSCLK** frequency is used when compared to the throughput and latency when the PLL is enabled and a 1x **SYSCLK** frequency is used (refer to Section 3.3.4 *Encoder Latency* and Section 4.2.7 *Decoder Latency*).

6.0 GENERAL OUTPUT SIGNALS

The AHA4525 contains a general output signal for the encoder and a general output signal for the decoder that can be used to control external logic or for system debugging. The signals can be driven to a 1 or 0 or can be driven according to events within the device including load complete, decode complete, empty, etc. (see Section 7.7.1 for a complete list of available status outputs).

7.0 REGISTER DESCRIPTIONS

7.1 CONFIGURATION SEQUENCE

The following sequence should be followed when configuring the device. The encoder and decoder configuration sequences are identical. The following configuration sequence is described in terms of the decoder, but all descriptions, except where explicitly stated, apply to the encoder.

The decoder is reset when **RESET_N** = 0. After the rising edge of **RESET_N**, the decoder is idle. The first write to the decoder is expected to start a configuration cycle. The order of configuration is not important. The last configuration cycle is specified by either the "rwb" bit = 1 or "last" bit = 1 (see Section 2.3 *Configuration Cycle Format*). Data is not expected to follow the configuration cycles if the last configuration cycle is a read or if the DNoBlock bit was set (see Section 7.7.1 *Control*).

7.2 REGISTER LIST

There are 2 complete register sets in the AHA4525 device: one for the encoder and one for the decoder. Registers used in both the encoder and decoder have the same address but are accessed through either the encoder or decoder data interface. In Table 5, the register bit name does not include the E for encoder register or D for decoder register when the register is common to both the encoder and decoder. Registers that are specific to the encoder or decoder are labelled with the E for encoder or D for decoder. All register bits labelled as "*res*" are reserved and must be written to 0. Reads from reserved registers return unpredictable values.

Table 5: Register Bits - Alphabetical

REGISTER BIT NAME	CONFIG	ADDRESS	BIT	REGISTER SECTION
ActualIterations[7:0]	read only	0x3d	7:0	Section 7.7.3 *Actual Iterations (Decoder Only)*
BlkSize[11:8]	0	0x11	3:0	Section 7.4.2 *Block Size*
	1	0x11	7:4	
	2	0x12	3:0	
	3	0x12	7:4	
BlkSize[7:0]	0	0x13	7:0	
	1	0x14		
	2	0x15		
	3	0x16		
CodeX[3:0]	0	0x09	7:4	Section 7.4.1 *TPC Constituent Code*
	1	0x0b		
	2	0x0d		
	3	0x0f		
CodeY[3:0]	0	0x09	3:0	
	1	0x0b		
	2	0x0d		
	3	0x0f		
Corrections[11:8]	read only	0x3b	3:0	Section 7.7.2 *Status and Correction Count (Decoder Only)*
Corrections[7:0]	read only	0x3c	7:0	
CRCEnable	all	0x00	5	Section 7.3.1 *CRC and Scrambler Configuration*
CRCErr	read only	0x3b	4	Section 7.7.2 *Status and Correction Count (Decoder Only)*
CRCPoly[31:24]	all	0x01	7:0	Section 7.3.1 *CRC and Scrambler Configuration*
CRCPoly[23:16]		0x02		
CRCPoly[15:8]		0x03		
CRCPoly[7:0]		0x04		
CRCSize[4:0]	all	0x00	4:0	
DOutputECC	all	0x0a	5	Section 7.4.1 *TPC Constituent Code*
DStatus	all	0x26	6	Section 7.7.1 *Control*

REGISTER BIT NAME	CONFIG	ADDRESS	BIT	REGISTER SECTION
EBufferSize[8]	0	0x28	7:0	Section 7.4.3 *Buffer Configuration (Encoder Only)*
	1	0x29		
	2	0x2a		
	3	0x2b		
EPassThrough	all	0x0a	6	Section 7.4.1 *TPC Constituent Code*
FeedbackX[4:0]	0	0x28	4:0	Section 7.4.5 *Feedback (Decoder Only)*
	1	0x2a		
	2	0x2c		
	3	0x2e		
FeedbackY[4:2]	0	0x28	7:5	
	1	0x2a		
	2	0x2c		
	3	0x2e		
FeedbackY[1:0]	0	0x29	1:0	
	1	0x2b		
	2	0x2d		
	3	0x2f		
GOutConfig[2:0]	all	0x26	5:3	Section 7.7.1 *Control*
Helical	0	0x0a	7	Section 7.4.1 *TPC Constituent Code*
	1	0x0c		
	2	0x0e		
	3	0x10		
Iterations[7:0]	0	0x36	7:0	Section 7.4.6 *Iterations (Decoder Only)*
	1	0x37		
	2	0x38		
	3	0x39		
NoBlock	all	0x26	2	Section 7.7.1 *Control*
NoConfig	all		7	
QBits[1:0]	all	0x3a	3:2	Section 7.5.1 *Quantization (Decoder Only)*
QMode[1:0]	all		1:0	
ScramEnable	all	0x00	6	Section 7.3.1 *CRC and Scrambler Configuration*
ScramPoly[15:8]	all	0x05	7:0	
ScramPoly[7:0]	all	0x06		
ScramSeed[15:8]	all	0x07		
ScramSeed[7:0]	all	0x08		

REGISTER BIT NAME	CONFIG	ADDRESS	BIT	REGISTER SECTION
ShortB[5:2]	0	0x18	7:4	Section 7.4.4 *Shortening Configuration*
	1	0x1b		
	2	0x1e		
	3	0x21		
ShortB[1:0]	0	0x19	1:0	
	1	0x1c		
	2	0x1f		
	3	0x22		
ShortX[5:0]	0	0x17	5:0	
	1	0x1a		
	2	0x1d		
	3	0x20		
ShortY[5:4]	0	0x17	7:6	
	1	0x1a		
	2	0x1d		
	3	0x20		
ShortY[3:0]	0	0x18	3:0	
	1	0x1b		
	2	0x1e		
	3	0x21		
StopIter	0	0x29	7	Section 7.4.5 *Feedback (Decoder Only)*
	1	0x2b		
	2	0x2d		
	3	0x2f		
Version[7:0]	read only	0x3f	7:0	Section 7.8.1 *Version*
WordEqBlk	all	0x27	5	Section 7.6.1 *Transfer Word Size*
WordSize[4:0]	all	0x27	4:0	Section 7.6.1 *Transfer Word Size*

7.3 USER DATA FORMATTING REGISTERS

7.3.1 CRC AND SCRAMBLER CONFIGURATION

Read/Write
Reset Value (hex) = 00 00 00 00 00 00 00 00 00

address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x00	*res*	**xScramEnable**	**xCRCEnable**	**xCRCSize[4:0]**				
0x01	**xCRCPoly[31:24]**							
0x02	**xCRCPoly[23:16]**							
0x03	**xCRCPoly[15:8]**							
0x04	**xCRCPoly[7:0]**							
0x05	**xScramPoly[15:8]**							
0x06	**xScramPoly[7:0]**							
0x07	**xScramSeed[15:8]**							
0x08	**xScramSeed[7:0]**							

xCRCEnable - CRC Enable. When set, the encoder will compute and insert CRC bits and the decoder will compute, check and remove CRC.

xCRCSize[4:0] - CRC Size. A value of 0 represents 32 bits. This value must always be less than or equal to **xBlockSize[11:0]**.

xCRCPoly[31:0] - CRC Polynomial. Refer to Table 1 on page 15 for configuration of the polynomial.

xScramEnable - Scrambler/Descrambler Enable. When set, the scrambler or descrambler will scramble the data according to the polynomial. When cleared, the scrambler or descrambler is disabled.

xScramPoly[15:0] - Scrambler/Descrambler polynomial. For each exponential term in the polynomial, X^N, a binary one should be loaded at bit location N-1. X^0 is assumed to be a term in the polynomial. For example, the polynomial $1+X^3+X^5+X^8$ is programmed as 0b0000000010010100. Note that the length of the PRBS is determined by the most significant bit location containing a one in **xScramPoly[15:0]**. Refer to Figure 16 on Page 15 for configuration of the polynomial.

xScramSeed[15:0] - Seed for scrambler and descrambler. The PRBS is reset to this seed at the beginning of every TPC block. Refer to Figure 16 on Page 15 for configuration of the seed.

7.4 CODE CONFIGURATION REGISTERS

7.4.1 TPC CONSTITUENT CODE

Read/Write
Reset Value (hex) = 00 00 00 00   00 00 00 00

address	*bit7*	*bit6*	*bit5*	*bit4*	*bit3*	*bit2*	*bit1*	*bit0*
0x09	**xCodeX[3:0] - config 0**				**xCodeY[3:0] - config 0**			
0x0a	**xHelical - config 0**	**EPass Thru**	**DOutput ECC**	**Reserved**	**Reserved**			
0x0b	**Reserved**							
0x0c	**Reserved**							
0x0d	**Reserved**							
0x0e	**Reserved**							
0x0f	**Reserved**							
0x10	**Reserved**							

xCodeX[3:0] - X axis code. **xCodeX[3]** determines whether the code is an extended Hamming or parity only code. When set, the code is a parity only code, when cleared, it is an extended Hamming code. **xCodeX[2:0]** determines the size of the code, as follows:

Code[2:0]	*EXTENDED HAMMING*	*PARITY ONLY*
0x2	N/A	(4,3)
0x3	(8,4)	(8,7)
0x4	(16,11)	(16,15)
0x5	(32,26)	(32,31)
0x6	(64,57)	(64,63)

xCodeY[3:0] - Y axis code. Defined the same as **xCodeX[3:0]**. The maximum length for y-code is 64 bits.

EPassThru - Encoder Pass Through. When set, the encoder does not add ECC bits. The block size is still configured by the code configuration, but the encoder simply passes bits through from the input to the output. This is a reserved bit in the decoder configuration register stack.

DOutputECC - Decoder Output ECC. When set, the decoder will output both the user data and the TPC ECC bits. Descrambling and CRC checking must be disabled when this bit is set. This is a reserved bit in the encoder configuration register stack.

xHelical - Helical Interleaver/Deinterleaver Enable. See Section 3.3.2 for a description of helical interleaving. The following rules must be followed when helical interleaving.

The length of the x dimension vector (**xCodeX - xShortX**) must be a multiple of 2.

xShortB must be 0.

7.4.2 BLOCK SIZE

Read/Write
Reset Value (hex) = 00 00 00 00   00 00

address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x11	**Reserved**				**xBlockSize[11:8] - config 0**			
0x12					**Reserved**			
0x13	**xBlockSize[7:0] - config 0**							
0x14	**Reserved**							
0x15	**Reserved**							
0x16	**Reserved**							

xBlockSize - Block Size. This is the number of input bits per block when encoding or decoding. When decoding, **DBlockSize** includes the CRC bits if they were inserted by the encoder. When encoding, **EBlockSize** does not include CRC bits. **EBlockSize** + **ECRCSize** must always be less than 4096 bits. When set to 0, **xBlockSize** is 4096 bits. The minimum number of data bits in a block is 16 and the number of data bits must be greater than or equal to **xCRCSize[4:0]**.

An example of (16,11) × (16,11) with 16 bit CRC gives

D BlockSize = 16 × 16 = 256 = 0 × 100
E BlockSize = (11 × 11) - 16 = 105 = 0 × 069

When decoding, **DBlockSize** is equal to the channel block for a maximum of 4096 bits.When encoding **EBlockSize** is equal to the data block size minus CRC bits.

7.4.3 BUFFER CONFIGURATION (ENCODER ONLY)

Read/Write
Reset Value (hex) = 00 00 00 00

address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x28	**EBufferSize[7:0] - config 0**							
0x29	**Reserved**							
0x2a	**Reserved**							
0x2b	**Reserved**							

EBufferSize[7:0] - Encoder Logical Buffer Size * 16. The encoder input buffer is used to allow data blocks to stream into and out of the TPC encoder. When set to 0, **EBufferSize[7:0]** is 256. **EBufferSize[7:0]** should be set according to the following equations:

If the frequency of **U_CLK** is greater than or equal to the frequency of **E_CLK**, or if **EHelical** = 1, then

$$\textbf{EBufferSize} = 1.$$

If the frequency of **E_CLK** is greater than the frequency of **U_CLK**, and **EHelical** = 0, then in order to stream out a block continuously,

$$\textbf{EBufferSize} = \left\lceil [n_x \times n_y + n_x \times (n_y - k_y) + n_x - k_x]\frac{cr}{16} \right\rceil \times ((eclk)/(uclk))$$

Where the codes (n_x,k_x), (n_y,k_y) refer to the shortened constituent codes and *cr* is the code rate of the input block. The code rate is the number of input bits divided by the number of output bits including bits inserted by the CRC encoder.

If **EBufferSize***16 is calculated to be greater than **EBlockSize**, then

$$\textbf{EBufferSize = EBlockSize/16}$$

7.4.4 SHORTENING CONFIGURATION

Read/Write
Reset Value (hex) = 00 00 00 00 00 00 00 00 00 00 00 00 00 00

address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x17	**xShortY[5:4] - config 0**		**xShortX[5:0] - config 0**					
0x18	**Reserved**				**xShortY[3:0] - config 0**			
0x19	**Reserved**						**xShortB[1:0] - config 0**	
0x1a	**Reserved**							
0x1b	**Reserved**							
0x1c	**Reserved**							
0x1d	**Reserved**							
0x1e	**Reserved**							
0x1f	**Reserved**							
0x20	**Reserved**							
0x21	**Reserved**							
0x22	**Reserved**							
0x23	**Reserved**							
0x24	**Reserved**							

xShortX[5:0] - Number of bits to shorten from the X axis code. This value must be set to an even value when helical interleaving is enabled. The X axis must always contain at least 2 data bits.

xShortY[5:0] - Number of bits to shorten from the Y axis code.

xShortB[5:0] - Number of bits to shorten from the first row. This value must be set to 0 when helical interleaving is enabled.

Table 6: Summary of Code Shortening Rules

Code Type	ShortX	ShortY	ShortZ	ShortB	ShortR
2D			not allowed		not allowed
2D Helical	even		not allowed	not allowed	not allowed

7.4.5 FEEDBACK (DECODER ONLY)

Read/Write
Reset Value (hex) = 00 00 00 00 00 00 00 00 00 00 00 00 00 00

address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x28	**FeedbackY[4:2] - config 0**			**FeedbackX[4:0] - config 0**				
0x29	**StopIter - config 0**	**Reserved**					**FeedbackY[1:0] - config 0**	
0x2a	**Reserved**							
0x2b	**Reserved**							
0x2c	**Reserved**							
0x2d	**Reserved**							
0x2e	**Reserved**							
0x2f	**Reserved**							
0x30	**Reserved**							
0x31	**Reserved**							
0x32	**Reserved**							
0x33	**Reserved**							
0x34	**Reserved**							
0x35	**Reserved**							

FeedbackX[4:0] - Feedback for the X axis. The output of the SISO is multiplied by **FeedbackX[4:0]**, then shifted by 5 (divided by 32) before being input to following iterations.

FeedbackY[4:0] - Feedback for the Y axis.

StopIter - When set, decoder will stop iterating before reaching the maximum number of iterations specified in the **Iterations** register if convergence is detected (see Section 7.4.6 *Iterations (Decoder Only)*). When clear, the decoder will iterate the full number of **Iterations**.

7.4.6 ITERATIONS (DECODER ONLY)

Read/Write
Reset Value (hex) = 00 00 00 00

address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x36	**Iterations[7:0] - config 0**							
0x37	**Reserved**							
0x38	**Reserved**							
0x39	**Reserved**							

Iterations[7:0] - Maximum number of iterations to perform. When set to 0, the decoder outputs the hard decision values for each bit with no corrections.

7.5 CHANNEL INTERFACE REGISTERS

7.5.1 QUANTIZATION (DECODER ONLY)

Read/Write
Reset Value (hex) = 00

address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x3a	*res*				**QBits[1:0]**		**QMode[1:0]**	

QBits[1:0] - Quantization bits. A value of 0 represents 4 bits. **QBits** = 01 is not valid unless **QMode** = 00.

QMode[1:0] - Quantization Mode for soft input data. When set to "11," input data is assumed to be in signed 2's complement notation (mid-tread). When set to "01," data is assumed to be mid-riser sign/magnitude notation. When set to "00," data is assumed to be mid-riser unsigned. When set to "10," data is assumed to be in "half 2's complement" mid-riser notation. The confidence mapping for each mode is shown below with four bit quantization.

QMode[1:0]	INPUT DATA TYPE	HARD DECISION 0 CONFIDENCE RANGE			NO CONFIDENCE	HARD DECISION 1 CONFIDENCE RANGE		
		Max	. . .	Min		Min	. . .	Max
00	Unsigned	0000	. . .	0111	N/A	1000	. . .	1111
01	Sign/Magnitude	0111	. . .	0000	N/A	1000	. . .	1111
10	Half 2's Compl	1000	. . .	1111	N/A	0000	. . .	0111
11	2's Complement	1000	. . .	1111	0000	0001	. . .	0111

7.6 DATA INPUT/OUTPUT CONFIGURATION

7.6.1 TRANSFER WORD SIZE

Read/Write
Encoder Reset Value (hex) =e0
Decoder Reset Value (hex) =e0

address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x27	*res*		**xWordEqBlk**	**xWordSize[4:0]**				

xWordEqBlk - Word size equals block size. Only valid when port is configured as a synchronous port (**x_MODE** = 0). When set, the input port expects 1 **x_FS** per block and the output will generate 1 **x_FS** per block. When cleared, the input port expects 1 **x_FS** pulse for every **xWordSize[4:0]** input bits and the output port generates 1 **x_FS** pulse for every **xWordSize[4:0]** output bits.

xWordSize[4:0] - Word size. Only valid when port is configured as a synchronous port (**x_MODE** = 0) and **xWordEqBlk** is cleared. The input port expects 1 **x_FS** pulse for every **xWordSize[4:0]** input bits and the output port generates 1 **x_FS** pulse for every **xWordSize[4:0]** output bits. Valid values for **xWordSize[4:0]** range from 2 to 32 bits. When set to 0, **xWordSize[4:0]** is 32 bits.

7.7 CONTROL AND STATUS

7.7.1 CONTROL

Read/Write
Reset Value (hex) = 00

address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x26	**xNoConfig**	**DStatus**	**xGOutConfig[2:0]**			**xNoBlock**	**Reserved**	

xGOutConfig[2:0] - General output control. These registers control the state of the general output signals (**E_GOUT** and **D_GOUT**). The general outputs can be driven to a 1 or 0 or any one of six signals can be muxed to either of the general output signals. This is used for system debugging or control of an external device. The following table shows the options for each **GOUT** signal. Note that **xGOutConfig[2]** is only used in the decoder.

DGOut Config[2:0]	*SELECTED OUTPUT SIGNAL*
0	Drive **D_GOUT** to 0 (low).
1	Drive **D_GOUT** to 1 (high).
2	Drive decoder load complete event. **D_OUT** will change state each time the decoder completes loading one block of data.
3	Drive decode complete event. **D_GOUT** will change state each time the decoder completes decoding one block of data.
4	Drive unload complete event. **D_GOUT** will change state each time the decoder completes unloading one block of data.
5*	Drive **D_GOUT** to 1 when the decoder datapath is empty, 0 when not empty.

EGOut Config[1:0]	*SELECTED OUTPUT SIGNAL*
0	Drive **E_GOUT** to 0 (low).
1	Drive **E_GOUT** to 1 (high).
2	Drive encoder load complete event. **E_GOUT** will change state each time the encoder completes loading one block of data. This is only valid when helical interleaving is enabled.
3*	Drive **E_OUT** to 1 when the encoder datapath is empty and not processing a block, 0 when not empty or processing a block.

* *The datapath empty flags may assert between a configuration read request and the read data output.*

xNoBlock - No block input following configuration. This bit is written to indicate that the configuration write cycle will not be followed by data. The **xNoBlock** bit must be written to a 1 for each set of configuration cycles that is not finished with a configuration read or followed by a data block. The **xNoBlock** bit is automatically cleared between blocks. Reads of **xNoBlock** will always return a 0.

xNoConfig - No configuration cycles. This bit is written after configuration 0 is programmed and no further configuration is needed. The configuration cycle with **xNoConfig** set must be the last configuration cycle. The AHA4525 expects data to follow this configuration cycle unless **xNoBlock** is set.

DStatus - Decoder Status. When this bit is asserted, a status footer is attached at the end of every decoded block. The status footer contains the **Corrections** count and the **CRCErr** flag.

7.7.2 STATUS AND CORRECTION COUNT (DECODER ONLY)

Read Only
Reset Value (hex) = 00 00

address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x3b	*res*			**CRCErr**	**Corrections[11:8]**			
0x3c	**Corrections[7:0]**							

Corrections[11:0] - Corrections per block. The total number of bits corrected by the TPC decoder in the previous block. This value is output as a footer at the end of the TPC block when **DStatus** is set. The **Corrections** value applies to the last decoded block. Writes to these bits have no effect.

CRCErr - CRC verification error. This decoder signal is asserted when the data block did not pass the CRC verification. This value is output as a footer at the end of the TPC block when **DStatus** is set. The **CRCErr** status is updated with the last read of a block and is valid until the last read of the next block. A write to this bit has no effect.

7.7.3 ACTUAL ITERATIONS (DECODER ONLY)

Read Only
Reset Value (hex) = 00

address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x3d	**ActualIterations[7:0]**							

ActualIterations[7:0] - Iterations per block. This is the actual number of iterations performed on the previous block. This number will always equal **Iterations** when **StopIter** is not asserted. The **ActualIterations** count value can only be read when the decoder datapath is empty. The **ActualIterations** value applies to the last decoded block. **ActualIterations** may be used as a tool to tune the **Iterations** setting for specific noise levels. Writes to these bits have no effect.

7.8 MISCELLANEOUS

7.8.1 VERSION

Read Only
Reset Value (hex) = 2b

address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x3f	**xVersion[7:0]**							

xVersion[7:0] - Version number of the device.

8.0 SIGNAL DESCRIPTIONS

This section contains descriptions for all the pins. Each signal has a type code associated with it. The type codes are described in the following table.

TYPE CODE	*DESCRIPTION*
I	Input only pin
O	Output only pin

8.1 SYSTEM CONTROL AND MISCELLANEOUS

SIGNAL	*TYPE*	*DESCRIPTION*
RESET_N	I	Refer to Section 11.6 *Reset_n Timing* for **RESET_N** timing rules.
TRISTATE_N	I	When low, all outputs are tristated. Tie high for normal operation.
TESTMODE	I	Test mode input signal. Tie low for normal operation.
SCANMODE	I	Scan mode input signal. Tie low for normal operation.
DPARINPUT	I	Decoder parallel input. When **DPARINPUT** = 1 on the **RESET_N** rising edge, **C_DATA[3:0]** is configured as a 4 bit input bus. When **DPARINPUT** = 0 on the **RESET_N** rising edge, **C_DATA[0]** is configured as a serial input.
PLL_BYPASS	I	PLL bypass input signal. Tie low for normal operation.
E_MODE	I	Encoder interface mode. When **E_MODE** = 0 on the **RESET_N** rising edge, the encoder interface is configured as a synchronous port. When **E_MODE** = 1 on the **RESET_N** rising edge, the encoder interface is configured as a bus interface.
D_MODE	I	Decoder interface mode. When **D_MODE** = 0 on the **RESET_N** rising edge, the decoder interface is configured as a synchronous port. When **D_MODE** = 1 on the **RESET_N** rising edge, the decoder interface is configured as a bus interface.
E_CS_N	I	Encoder chip select, active low. When deasserted, **E_DATA** and **E_FS** are tristated.
D_CS_N	I	Decoder chip select, active low. When deasserted, **D_DATA** and **D_FS** are tristated.
SYSCLK	I	System Clock.
E_GOUT	O	Encoder general output signal. The functionality of this signal is configured in Section 7.7.1 *Control*.This signal is driven from an internal clock domain. There is no timing relationship between the external clocks and the GOUT signals.
D_GOUT	O	Decoder general output signal. The functionality of this signal is configured in Section 7.7.1 *Control*. This signal is driven from an internal clock domain. There is no timing relationship between the external clocks and the GOUT signals.
RDYPOLARITY	I	Ready polarity signal. When **RDYPOLARITY** = 1, **E_RDY** and **D_RDY** are active high. When **RDYPOLARITY** = 0, **E_RDY** and **D_RDY** are active low. This can only change when **RESET_N** is asserted.
ACPTPOLARITY	I	Accept polarity signal. When **ACPTPOLARITY** = 1, **U_ACPT** and **C_ACPT** are active high. When **ACPTPOLARITY** = 0, **U_ACPT** and **C_ACPT** are active low. This input can only change when **RESET_N** is asserted.

8.2 UNENCODED DATA INTERFACE

SIGNAL	*TYPE*	*DESCRIPTION*
U_CLK or **U_WR_N**	I	Unencoded Data Clock. Refer to Section 11.2 *U_CLK, E_CLK, C_CLK, D_CLK Clock Timing* for timing rules.
U_DATA	I	Unencoded Input Data. When **E_MODE** = 0, **U_DATA** is synchronous to **U_CLK**. Data is transferred into the device on the rising edge of **U_CLK**. When **E_MODE** = 1, **U_DATA** is latched on the rising edge of **U_WR_N**.
U_FS	I	Unencoded Frame Sync. Only used when **E_MODE** = 0, **U_FS** is synchronous to **U_CLK**. Indicates to the device that data will be transferred on the next rising edge of **U_CLK**.
U_ACPT	O	Unencoded Data Accept. Indicates that the device can accept **U_DATA**. The polarity of this signal is selected with the **ACPTPOLARITY** pin.

8.3 ENCODED DATA INTERFACE

SIGNAL	*TYPE*	*DESCRIPTION*
E_CLK or **E_RD_N**	I	Encoded Data Clock. Refer to Section 11.2 *U_CLK, E_CLK, C_CLK, D_CLK Clock Timing* for timing rules.
E_DATA	O	Encoded Output Data. When **E_MODE** = 0, **E_DATA** is synchronous to **E_CLK**. Data is transferred into the device on the rising edge of **E_CLK**. When **E_MODE** = 1, **E_DATA** is transferred on the rising edge of **E_RD_N**.
E_FS	O	Encoded Frame Sync. Only used when **E_MODE** = 0, **E_FS** is synchronous to **E_CLK**. Indicates to the device that data will be transferred on the next rising edge of **E_CLK**.
E_RDY	O	Encoded Data Ready. Indicates that the device has data ready to be transferred across **E_DATA**. The polarity of this signal is selected with the **RDYPOLARITY** pin.

8.4 CHANNEL DATA INTERFACE

SIGNAL	TYPE	DESCRIPTION
C_CLK or **C_WR_N**	I	Channel Data Clock. Refer to Section 11.2 *U_CLK, E_CLK, C_CLK, D_CLK Clock Timing* for timing rules.
C_DATA[3:0]	I	Channel Input Data. When **D_MODE** = 0, **C_DATA[3:0]** is synchronous to **C_CLK**. Data is transferred into the device on the rising edge of **C_CLK**. When **D_MODE** = 1, **C_DATA[3:0]** is latched on the rising edge of **C_WR_N**. If **DPARINPUT** = 1 on the rising edge of **RESET_N**, data is received in parallel on **C_DATA[3:0]**. If **DPARINPUT** = 0 on the rising edge of **RESET_N**, data is received serially on **C_DATA[0]**.
C_FS	I	Channel Frame Sync. Only used when **D_MODE** = 0, **C_FS** is synchronous to **C_CLK**. Indicates to the device that data will be transferred on the next rising edge of **C_CLK**.
C_ACPT	O	Channel Data Accept. Indicates that the device can accept **C_DATA**. The polarity of this signal is selected with the **ACPTPOLARITY** pin.

8.5 DECODED DATA INTERFACE

SIGNAL	TYPE	DESCRIPTION
D_CLK or **D_RD_N**	I	Decoded Data Clock. Refer to Section 11.2 *U_CLK, E_CLK, C_CLK, D_CLK Clock Timing* for timing rules.
D_DATA	O	Decoded Output Data. When **D_MODE** = 0, **D_DATA** is synchronous to **D_CLK**. Data is transferred into the device on the rising edge of **D_CLK**. When **D_MODE** = 1, **D_DATA** is transferred on the rising edge of **D_RD_N**.
D_FS	O	Decoded Frame Sync. Only used when **D_MODE** = 0, **D_FS** is synchronous to **D_CLK**. Indicates to the device that data will be transferred on the next rising edge of **D_CLK**.
D_RDY	O	Decoded Data Ready. Indicates that the device has data ready to be transferred across **D_DATA**. The polarity of this signal is selected with the **RDYPOLARITY** pin.

9.0 PINOUT

Table 7: Pinout - Pin Number Order

PIN	SIGNAL	PIN	SIGNAL
1	C_ACPT	33	U_ACPT
2	GNDIO	34	GNDIO
3	VDDIO	35	VDDIO
4	DPARINPUT	36	U_CLK
5	GND	37	U_FS
6	C_CLK	38	GND
7	VDD	39	VDD
8	C_FS	40	U_DATA
9	RDYPOLARITY	41	SYSCLK
10	ACPTPOLARITY	42	AGND
11	C_DATA[3]	43	VAD
12	C_DATA[2]	44	PLL_BYPASS
13	VDD	45	VDD
14	GND	46	GND
15	C_DATA[1]	47	TRISTATE_N
16	C_DATA[0]	48	TESTMODE
17	SCANMODE	49	VDD
18	E_CS_N	50	GND
19	VDD	51	RESET_N
20	GND	52	D_CS_N
21	E_FS	53	VDD
22	E_DATA	54	D_FS
23	GNDIO	55	GND
24	VDDIO	56	D_DATA
25	E_CLK	57	GNDIO
26	VDD	58	VDDIO
27	GND	59	D_CLK
28	E_RDY	60	VDD
29	GND	61	GND
30	E_GOUT	62	D_RDY
31	VDD	63	D_GOUT
32	E_MODE	64	D_MODE

10.0 DC ELECTRICAL SPECIFICATIONS

Information in this section represents design goals. While every effort will be made to meet these goals, values presented should be considered targets until characterization is complete. Please consult with Comtech AHA Corporation for the most up-to-date values.

10.1 OPERATING CONDITIONS

SYMBOL	PARAMETER	MIN	MAX	UNITS	NOTES
Vdd	Core Supply voltage (1.8V nominal)	1.7	1.9	V	7
Vddio	Input/Output Supply Voltage (3.3V nominal)	3.0	3.6	V	7
Vad	Analog PLL Supply Voltage (1.8V nominal)	1.7	1.9	V	3
Idd	Supply current (static)		1	mA	
Iddio	Input/Output Supply current (static)		0.6	mA	
Idd	Supply current (active)		110	mA	5
Iddio	Input/output supply current (active)		10.3		5
Idd	Supply current (active)		90	mA	6
Iddio	Input/Output Supply current (active)		8	mA	6
Ta	Ambient temperature	0	70	°C	2
Vil	Input low voltage	-0.3	0.8	V	
Vih	Input high voltage	2.0	5.5	V	
Iin	Input leakage current	-5	5	uA	4
Vol	Output low voltage (Iol=-2mA)		0.4	V	
Voh	Output high voltage (Ioh=2mA)	2.4		V	
Iol	Output low current		8	mA	
Ioh	Output high current		8	mA	
Ioz	Output leakage current during tristate	-5	5	uA	
Cin	Input capacitance		10	pF	
Cio	Input/Output capacitance		10	pF	
Cout	Output load capacitance		30	pF	1

Notes:
1) *Timings referenced to this load.*
2) *May require external heat sink.*
3) *See Appendix A for Vad recommendations.*
4) *Measured at 0V and Vddio nominal.*
5) *At maximum SYSCLK frequency, nominal Vdd and Vddio, and Full Duplex operation.*
6) *At maximum SYSCLK frequency, nominal Vdd and Vddio, and Encode operation.*
7) *During power on sequencing, Vdd must reach 1.7V concurrent or before Vddio reaches 1.7V.*

10.2 ABSOLUTE MAXIMUM STRESS RATINGS

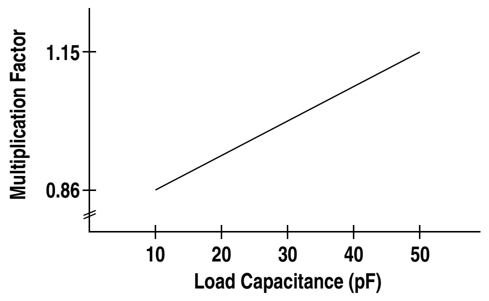
SYMBOL	PARAMETER	MIN	MAX	UNITS	NOTES
Tstg	Storage temperature	-50	150	°C	
Vsupply	Supply voltage	-0.5	4.6	V	
Vpin	Voltage applied to any signal pin		6.0	V	

10.3 TEST CONDITIONS

PARAMETER	VALUE
AC timing reference	1.4 V

* *The timing diagrams for these signals assume a capacitive load of 30pF. The specified signal timings must be derated by the factor shown in Figure 22 when operating at loads other than 30pF.*

Figure 22: Signal Timing vs. Output Load



LOAD CAPACITANCE	MULTIPLICATION FACTOR
10 pF	0.86
20 pF	0.92
30 pF	1.00
40 pF	1.07
50 pF*	1.15

*Production test conditions

11.0 AC ELECTRICAL SPECIFICATIONS

11.1 SYSCLK CLOCK TIMING

Figure 23: SYSCLK Clock Timing

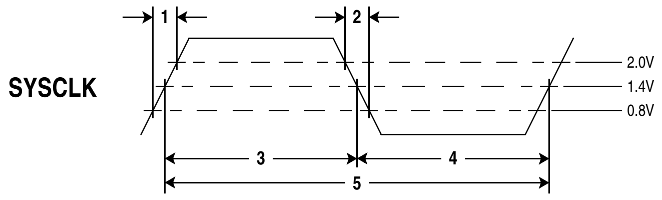


Table 8: SYSCLK Clock Timing with PLLBYPASS = 0

NUMBER	PARAMETER	MIN	MAX	UNITS
1	**SYSCLK** rise time		1	ns
2	**SYSCLK** fall time		1	ns
3	**SYSCLK** high time	8		ns
4	**SYSCLK** low time	8		ns
5	**SYSCLK** period (T_{cp})	20		ns
	SYSCLK frequency	20	50	MHz
	Cycle to cycle jitter	-125	125	ps
	Long term jitter	-200	200	ps

Table 9: SYSCLK Clock Timing with PLLBYPASS = 1

NUMBER	PARAMETER	MIN	MAX	UNITS
1	**SYSCLK** rise time		1	ns
2	**SYSCLK** fall time		1	ns
3	**SYSCLK** high time	2.0		ns
4	**SYSCLK** low time	2.0		ns
5	**SYSCLK** period (T_{cp})	5.0		ns
	SYSCLK frequency		200	MHz
	Cycle to cycle jitter	-125	125	ps
	Long term jitter	-200	200	ps

11.2 U_CLK, E_CLK, C_CLK, D_CLK CLOCK TIMING

Figure 24: Clock Timing

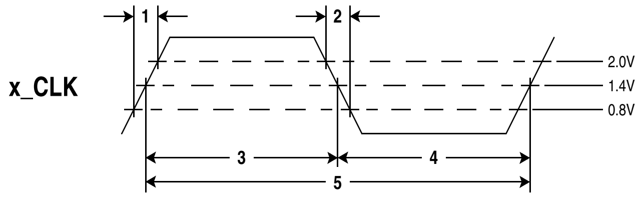


Table 10: Clock Timing

NUMBER	PARAMETER	MIN	MAX	UNITS
1	**U_CLK, E_CLK, C_CLK, D_CLK** rise time		1	ns
2	**U_CLK, E_CLK, C_CLK, D_CLK** fall time		1	ns
3	**U_CLK, E_CLK, C_CLK, D_CLK** high time	4		ns
4	**U_CLK, E_CLK, C_CLK, D_CLK** low time	4		ns
5	**U_CLK, E_CLK, C_CLK, D_CLK** frequency with **PLLBYPASS** = 0.		1.9***SYSCLK**	MHz
5	**U_CLK, E_CLK, C_CLK, D_CLK** frequency with **PLLBYPASS** = 1.		**SYSCLK**/2.1	MHz

11.3 U_WR_N, E_RD_N, C_WR_N, D_RD_N STROBE TIMING

Figure 25: Strobe Timing

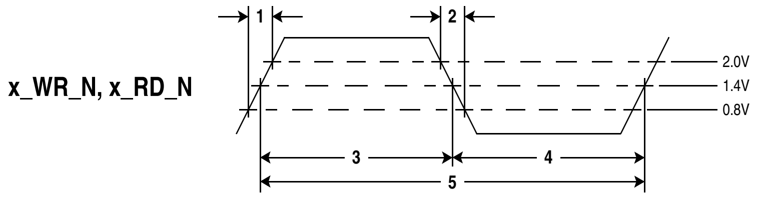


Table 11: Strobe Timing with PLLBYPASS = 0

NUMBER	PARAMETER	MIN	MAX	UNITS
1	**U_WR_, E_RD_N, C_WR_N, D_RD_N** rise time.		1	ns
2	**U_WR_, E_RD_N, C_WR_N, D_RD_N** fall time.		1	ns
3	**U_WR_, E_RD_N, C_WR_N, D_RD_N** high time.	8		ns
4	**U_WR_, E_RD_N, C_WR_N, D_RD_N** low time.	8		ns
5	**U_WR_, E_RD_N, C_WR_N, D_RD_N** frequency with **PLLBYPASS** = 0.		**SYSCLK**	MHz
5	**U_WR_, E_RD_N, C_WR_N, D_RD_N** frequency with **PLLBYPASS** = 1.		**SYSCLK**/4	MHz

11.4 SYNCHRONOUS DATA INTERFACE

Figure 26: Encoder Interface Data Input Timing

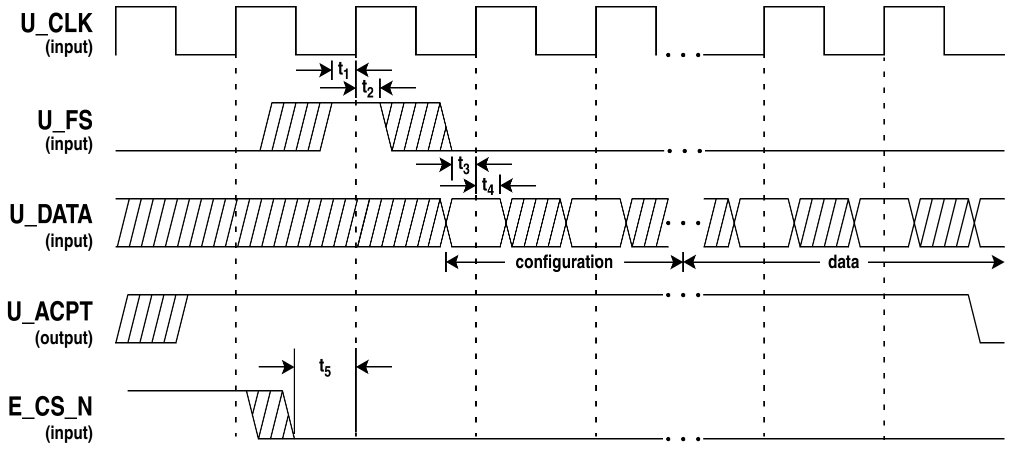


Table 12: Encoder Interface Data Input Timing

NUMBER	PARAMETER	MIN	MAX	UNITS	NOTES
t_1	**U_FS** setup to **U_CLK**.	5		ns	
t_2	**U_FS** hold from **U_CLK**.	3		ns	
t_3	**U_DATA** setup to **U_CLK**.	5		ns	
t_4	**U_DATA** hold from **U_CLK**.	3		ns	
t_5	**E_CS_N** setup to **U_CLK** rising edge that **U_FS** is asserted.	9		ns	

Notes:

1) *U_ACPT is driven from an internal clock domain. There is no timing relationship between U_CLK and U_ACPT. The function of U_ACPT is to indicate that the encoder can accept a block of data. U_ACPT deasserts following the first data transfer or after a configuration read is started.*
2) *There is no timing relationship between U_CLK and SYSCLK.*

Figure 27: Decoder Interface Data Input Timing

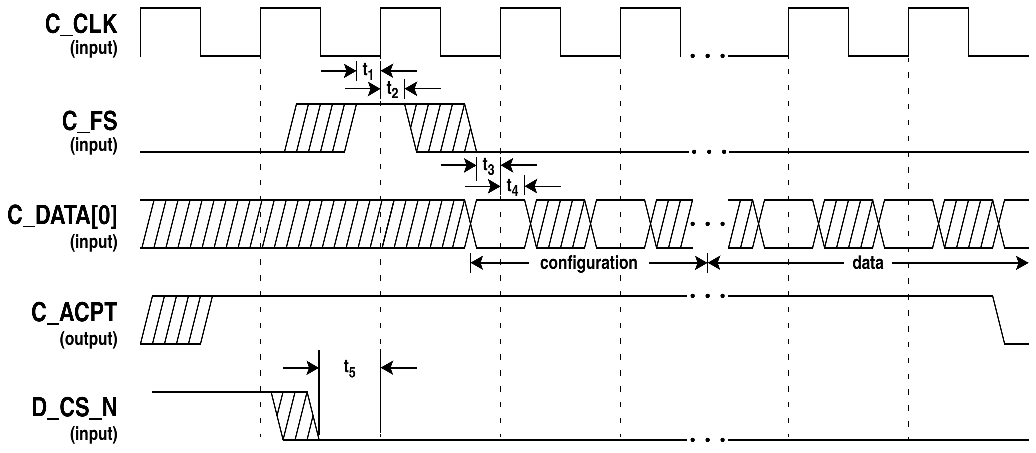


Table 13: Decoder Interface Data Input Timing

NUMBER	*PARAMETER*	*MIN*	*MAX*	*UNITS*	*NOTES*
t_1	**C_FS** setup to **C_CLK**.	5		ns	
t_2	**C_FS** hold from **C_CLK**.	3		ns	
t_3	**C_DATA[0]** setup to **C_CLK**.	5		ns	
t_4	**C_DATA[0]** hold from **C_CLK**.	3		ns	
t_5	**D_CS_N** setup to **C_CLK** rising edge that **C_FS** is asserted.	9		ns	

Notes:

1) *C_ACPT is driven from an internal clock domain. There is no timing relationship between C_CLK and C_ACPT. The function of C_ACPT is to indicate that the encoder can accept a block of data. C_ACPT deasserts following the first data transfer or after a configuration read is started.*
2) *There is no timing relationship between C_CLK and SYSCLK.*

Figure 28: Encoder Interface Data Output Timing Using Chip Select

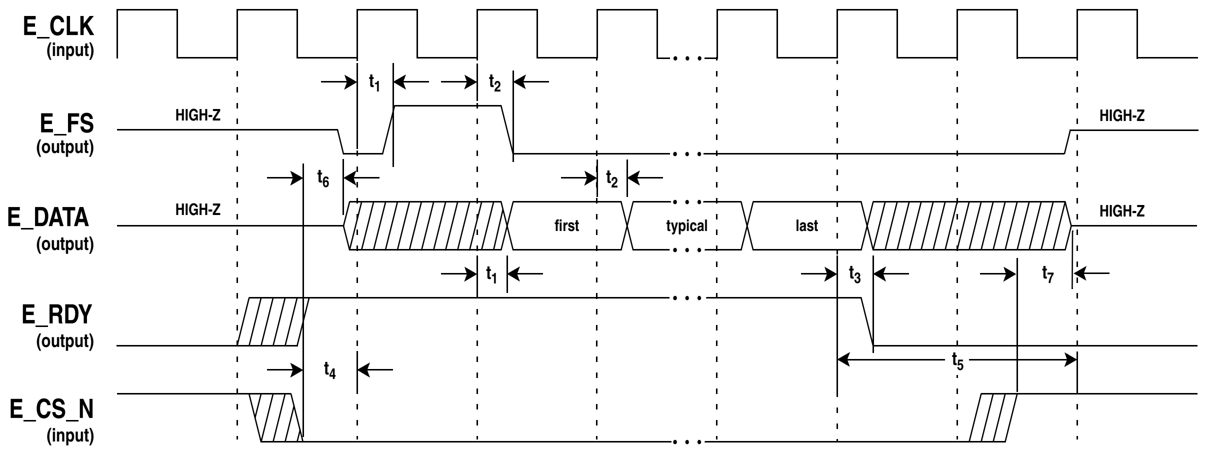


Table 14: Encoder Interface Data Output Timing Using Chip Select

NUMBER	PARAMETER	MIN	MAX	UNITS	NOTES
t_1	**E_FS, E_DATA** delay from **E_CLK**.		8	ns	
t_1	**E_FS, E_DATA** valid from **E_CLK** following **E_CS_N** asserted.		8	ns	1
t_2	**E_FS, E_DATA** hold from **E_CLK**.	1		ns	
t_3	**E_RDY** delay from **E_CLK**.		8	ns	
t_4	**E_CS_N** setup to **E_CLK.**	9		ns	
t_5	last **E_DATA** read to **E_CS_N** deasserted.		5	**SYSCLK**	1
t_6	**E_CS_N** asserted to **E_FS, E_DATA** valid.		8	ns	
t_7	**E_CS_N** deasserted to **E_FS, E_DATA** tristate.		8	ns	

Notes:

1) *It is valid for E_CS_N to always be asserted.*
2) *There is no timing relationship between E_CLK and SYSCLK.*

Figure 29: Encoder Interface Data Output Timing

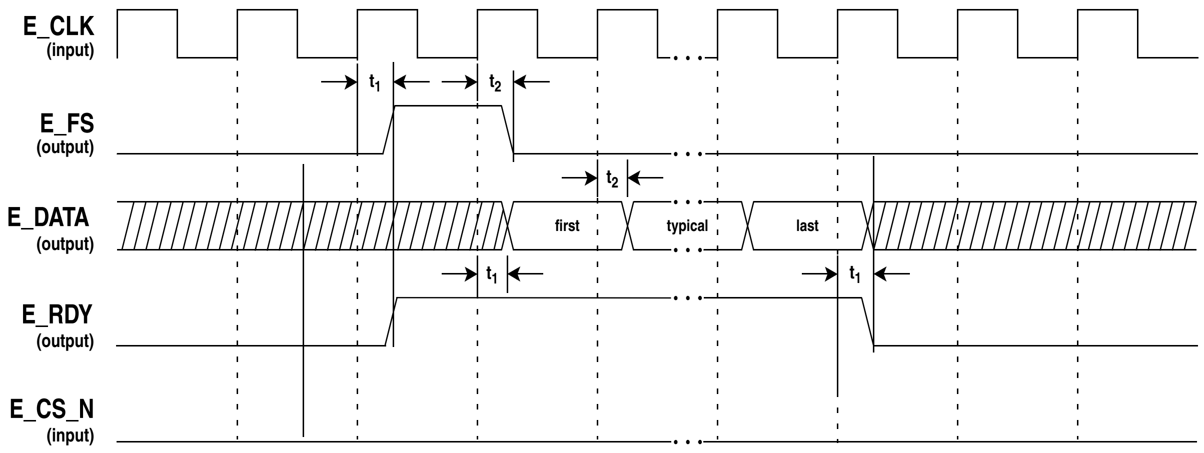


Table 15: Encoder Interface Data Output Timing

NUMBER	*PARAMETER*	*MIN*	*MAX*	*UNITS*	*NOTES*
t_1	**E_FS, E_RDY, E_DATA** delay from **E_CLK**.		8	ns	
t_1	**E_FS, E_RDY, E_DATA** valid from **E_CLK** following **E_CS_N** asserted.		8	ns	1
t_2	**E_FS, E_DATA** hold from **E_CLK**.	1		ns	
t_3	**E_RDY** delay from **E_CLK**.		8	ns	

Notes:

1) *It is valid for E_CS_N to always be asserted.*
2) *There is no timing relationship between E_CLK and SYSCLK.*

Figure 30: Decoder Interface Data Output Timing Using Chip Select

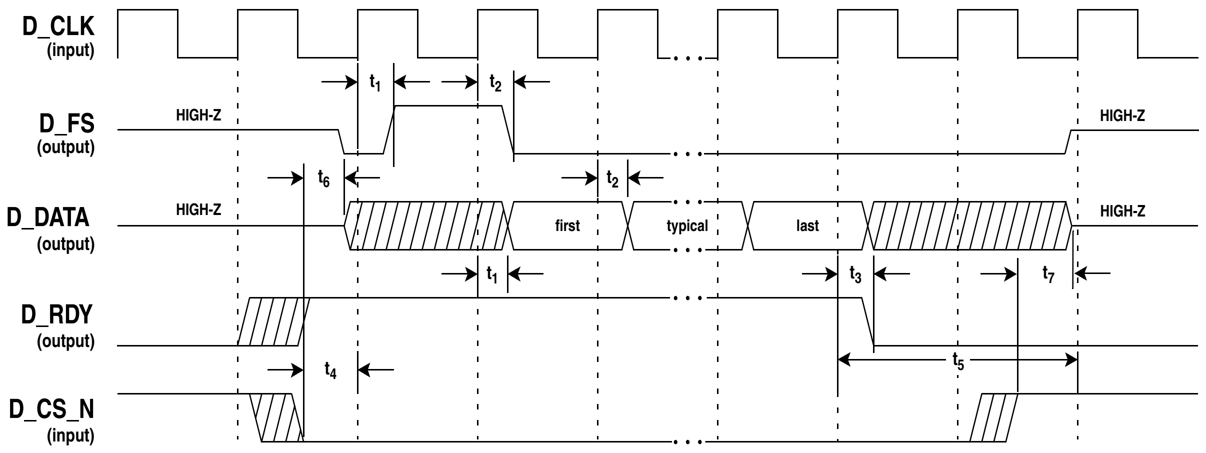


Table 16: Decoder Interface Data Output Timing Using Chip Select

NUMBER	PARAMETER	MIN	MAX	UNITS	NOTES
t_1	**D_FS, D_DATA** delay from **D_CLK**.		8	ns	
t_1	**D_FS, D_DATA** valid from **D_CLK** following **D_CS_N** asserted.		8	ns	1
t_2	**D_FS, D_DATA** hold from **D_CLK**.	1		ns	
t_3	**D_RDY** delay from **D_CLK**.		8	ns	
t_4	**D_CS_N** setup to **D_CLK**.	9		ns	
t_5	last **D_DATA** read to **D_CS_N** deasserted.		5	**SYSCLK**	1
t_6	**D_CS_N** asserted to **D_FS, D_DATA** valid.		8	ns	
t_7	**D_CS_N** deasserted to **D_FS, D_DATA** tristate.		8	ns	

Notes:
1) *It is valid for D_CS_N to always be asserted.*
2) *There is no timing relationship between D_CLK and SYSCLK.*

Figure 31: Decoder Interface Data Output Timing

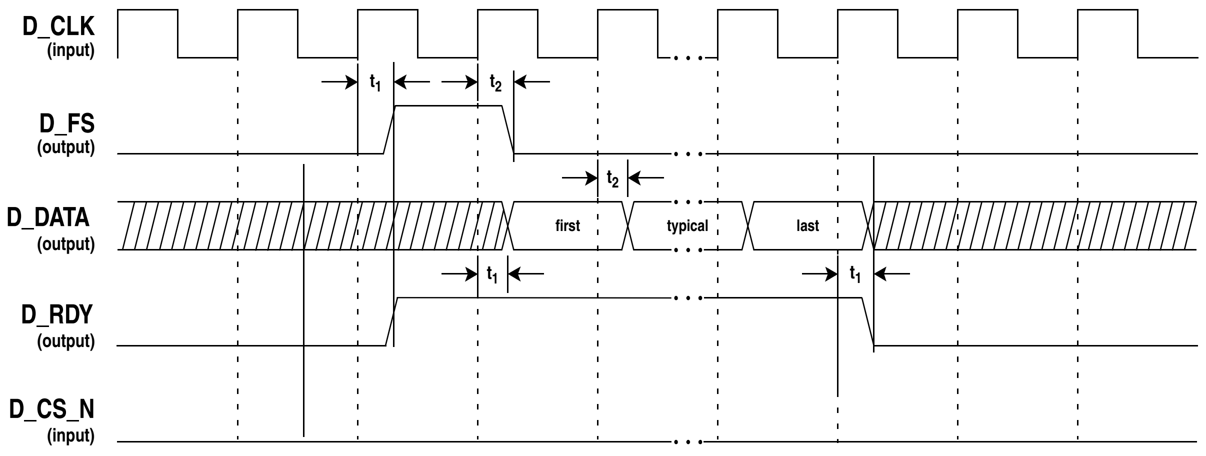


Table 17: Decoder Interface Data Output Timing

NUMBER	PARAMETER	MIN	MAX	UNITS	NOTES
t_1	**D_FS, D_RDY, D_DATA** delay from **D_CLK**.		8	ns	
t_1	**D_FS, D_RDY, D_DATA** valid from **D_CLK** following **D_CS_N** asserted.		8	ns	1
t_2	**D_FS, D_DATA** hold from **D_CLK**.	1		ns	
t_3	**D_RDY** delay from **D_CLK**.		8	ns	

Notes:

1) *It is valid for D_CS_N to always be asserted.*
2) *There is no timing relationship between D_CLK and SYSCLK.*

11.5 BUS INTERFACE

Figure 32: Encoder Bus Interface Data Input Timing

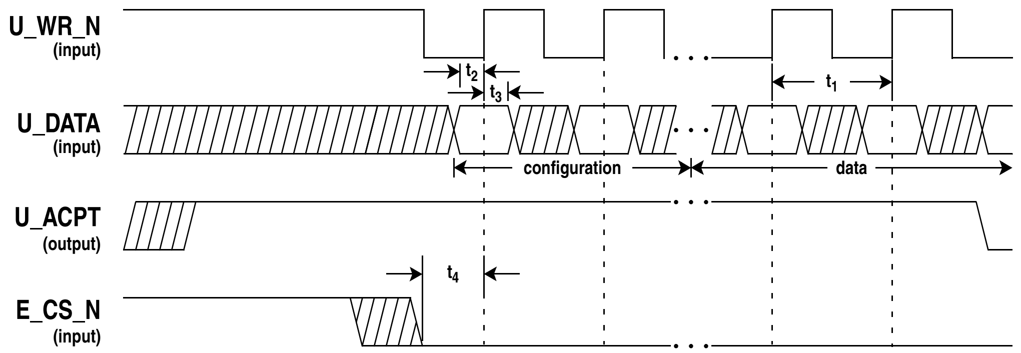


Table 18: Encoder Bus Interface Data Input Timing

NUMBER	PARAMETER	MIN	MAX	UNITS	NOTES
t_1	**U_WR_N** frequency with **PLLBYPASS** = 0.		**SYSCLK**	MHz	2, 3
t_1	**U_WR_N** frequency with **PLLBYPASS** = 1.		**SYSCLK**/4	MHz	2, 3
t_2	**U_DATA** setup to **U_WR_N** rising edge.	5		ns	
t_3	**U_DATA** hold from **U_WR_N** rising edge.	3		ns	
t_4	**E_CS_N** valid before **U_WR_N** strobe rising edge.	9		ns	

Notes:

1) *U_ACPT is driven from an internal clock domain. There is no timing relationship between U_WR_N and U_ACPT. The function of U_ACPT is to indicate that the encoder can accept a block of data. U_ACPT deasserts following the first data transfer or after a configuration read is started.*
2) *There is no timing relationship between U_WR_N and SYSCLK.*
3) *U_WR_N is used internally as a clock. U_WR_N must not glitch.*

Figure 33: Decoder Bus Interface Data Input Timing

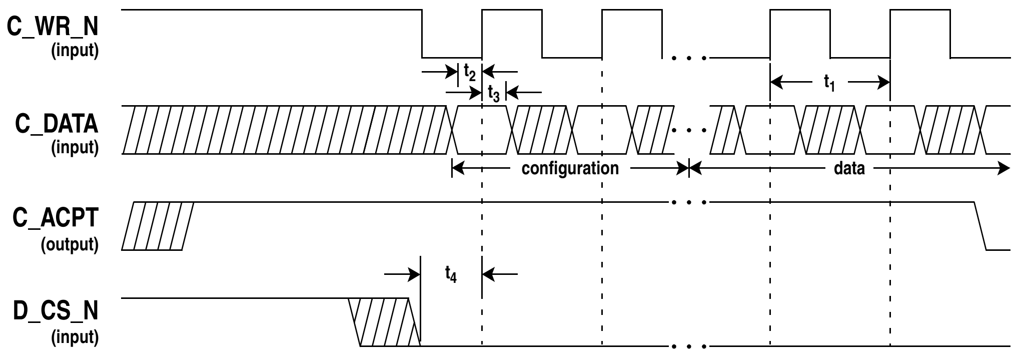


Table 19: Decoder Bus Interface Data Input Timing

NUMBER	PARAMETER	MIN	MAX	UNITS	NOTES
t_1	**C_WR_N** frequency with **PLLBYPASS** = 0.		**SYSCLK**	MHz	2, 3
t_1	**C_WR_N** frequency with **PLLBYPASS** = 1.		**SYSCLK**/4	MHz	2, 3
t_2	**C_DATA** setup to **C_WR_N** rising edge.	5		ns	
t_3	**C_DATA** hold from **C_WR_N** rising edge.	3		ns	
t_4	**D_CS_N** valid before **C_WR_N** strobe rising edge.	9		ns	

Notes:

1) *C_ACPT is driven from an internal clock domain. There is no timing relationship between C_CLK and C_ACPT. The function of C_ACPT is to indicate that the encoder can accept a block of data. C_ACPT deasserts following the first data transfer or after a configuration read is started.*
2) *There is no timing relationship between C_WR_N and SYSCLK.*
3) *C_WR_N is used internally as a clock. C_WR_N must not glitch.*

Figure 34: Encoder Bus Interface Data Output Timing

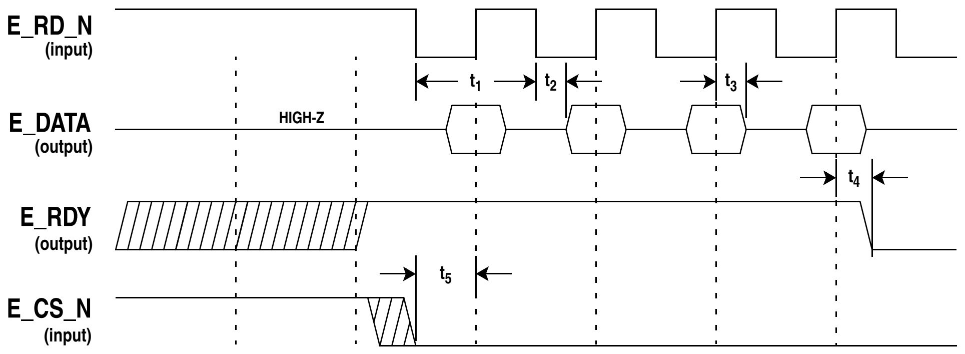


Table 20: Encoder Bus Interface Data Output Timing

NUMBER	PARAMETER	MIN	MAX	UNITS	NOTES
t_1	**E_RD_N** frequency with **PLLBYPASS** = 0.		**SYSCLK**	MHz	1, 2
t_1	**E_RD_N** frequency with **PLLBYPASS** = 1.		**SYSCLK**/4	MHz	1, 2
t_2	**E_DATA** valid from **E_RD_N** falling edge.		8	ns	
t_3	**E_DATA** hold from **E_RD_N** rising edge.	1	5	ns	
t_4	**E_RDY** delay from **E_RD_N** rising edge.		8	ns	
t_5	**E_CS_N** setup to **E_RD_N** rising edge.	9		ns	

Notes:

1) There is no timing relationship between E_RD_N and SYSCLK.
2) E_RD_N is used internally as a clock. E_RD_N must not glitch.

Figure 35: Decoder Bus Interface Data Output Timing

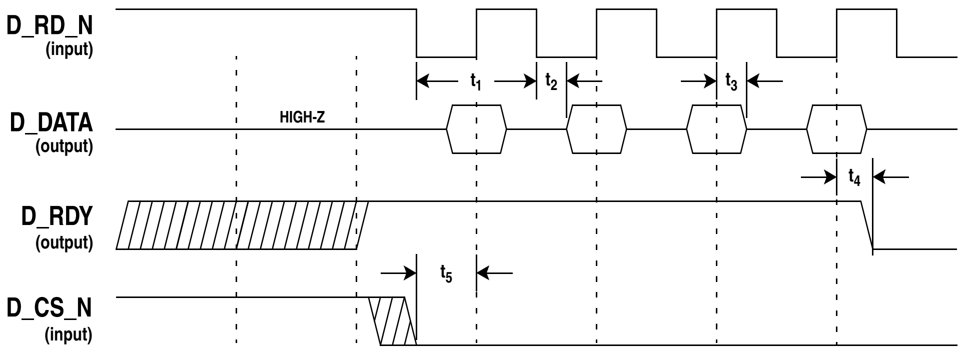


Table 21: Decoder Bus Interface Data Output Timing

NUMBER	PARAMETER	MIN	MAX	UNITS	NOTES
t_1	**D_RD_N** frequency with **PLLBYPASS** = 0.		**SYSCLK**	MHz	1, 2
t_1	**D_RD_N** frequency with **PLLBYPASS** = 1.		**SYSCLK**/4	MHz	1, 2
t_2	**D_DATA** valid from **D_RD_N** falling edge.		8	ns	
t_3	**D_DATA** hold from **D_RD_N** rising edge.	1	5	ns	
t_4	**D_RDY** delay from **D_RD_N** rising edge.		8	ns	
t_5	**D_CS_N** setup to **D_RD_N** rising edge.	9		ns	

Notes:

1) There is no timing relationship between D_RD_N and SYSCLK.
2) D_RD_N is used internally as a clock. D_RD_N must not glitch.

11.6 RESET_N TIMING

Figure 36: RESET_N Timing

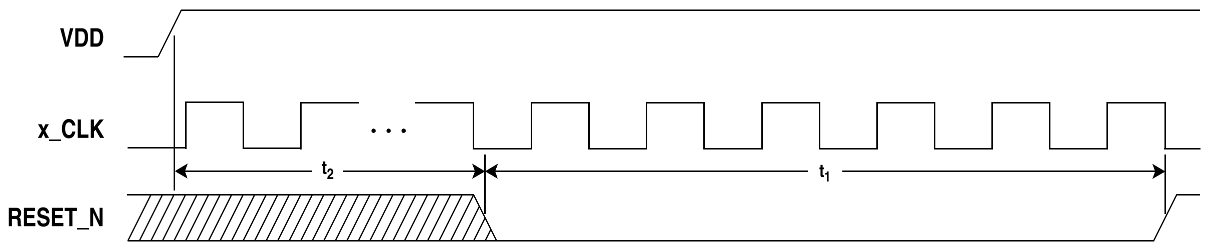


Table 22: RESET_N Timing

NUMBER	*PARAMETER*	*MIN*	*MAX*	*UNITS*	*NOTES*
t_1	**RESET_N** pulsewidth with **E_MODE** = 1, **D_MODE** = 1, and **PLLBYPASS** = 0.	10		**SYSCLK**	
t_1	**RESET_N** pulsewidth with **E_MODE** = 1, **D_MODE** = 1, and **PLLBYPASS** = 1.	40		**SYSCLK**	
t_1	**RESET_N** pulsewidth with **E_MODE** = 0 or **D_MODE** = 0.	10		T_{xCLK}	1,2,3,4
t_2	PLL lock time		10	us	5

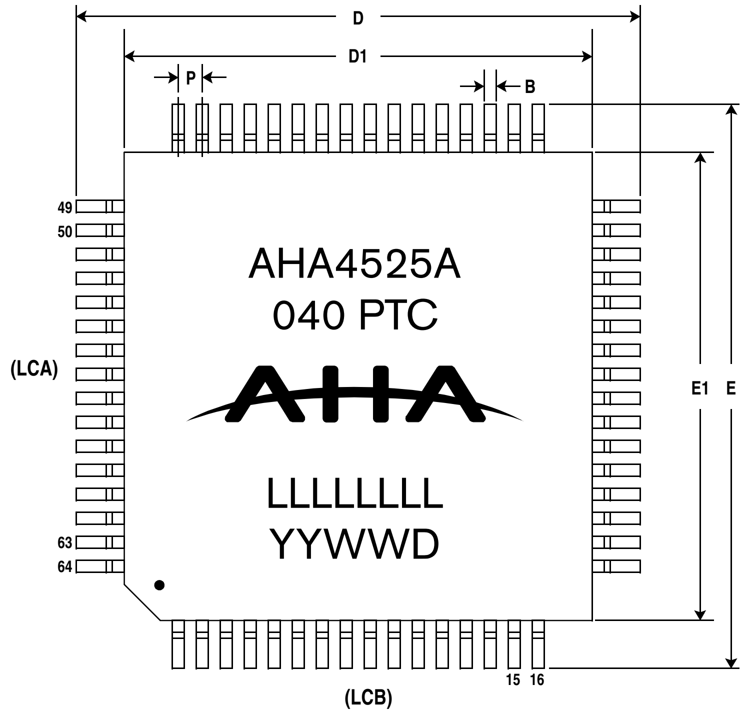
Notes:

1) *If E_MODE = 0 and PLLBYPASS = 0, T_{xCLK} is the period of the lowest frequency of U_CLK, E_CLK and SYSCLK.*
2) *If E_MODE = 0 and PLLBYPASS = 1, T_{xCLK} is the period of the lowest frequency of U_CLK, E_CLK and SYSCLK/ 4.*
3) *If D_MODE = 0 and PLLBYPASS=0, T_{xCLK} is the period of the lowest frequency of C_CLK, D_CLK and SYSCLK.*
4) *If D_MODE = 0 and PLLBYPASS=1, T_{xCLK} is the period of the lowest frequency of C_CLK, D_CLK and SYSCLK/4.*
5) *SYSCLK must be running for 10us to lock the internal PLL before RESET_N is asserted.*

12.0 PACKAGE SPECIFICATIONS

12.1 PACKAGE DIMENSIONS

Figure 37: Package Dimensions - Top View



NOTE: YYWWD = Date Code; LLLLLLLL = Lot Number

Figure 38: Package Dimensions - Cross Section View

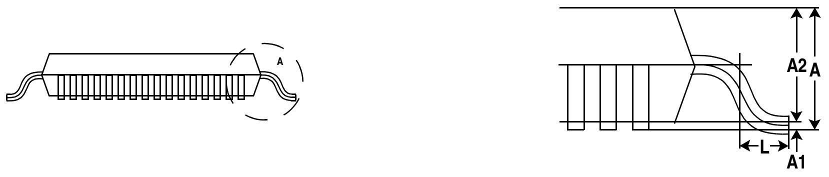


Table 23: TQFP (Thin Quad Flat Pack) 7 x 7 mm Package Dimensions

(All dimensions are in mm)

SYMBOL	***NUMBER OF PIN AND SPECIFICATION DIMENSION***		
	64		
	MIN	NOM	MAX
(LCA)	16		
(LCB)	16		
A		1.1	1.2
A1	0.05	0.1	0.15
A2	0.95	1.0	1.05
D		9.0	
D1		7.0	
E		9.0	
E1		7.0	
L	0.45	0.60	0.75
P		0.40	
B	0.13	0.18	0.23

Notes: *All dimensions are in millimeters.*
More detailed mechanical information is available upon request.

13.0 ORDERING INFORMATION

13.1 AVAILABLE PARTS

PART NUMBER	***DESCRIPTION***
AHA4525A-040 PTC	IEEE 802.16a Compliant Turbo Product Code Encoder/Decoder
AHA4525A-040 PTI	IEEE 802.16a Compliant Turbo Product Code Encoder/Decoder - Industrial Temp

13.2 PART NUMBERING

AHA	***4525***	***A-***	***040***	***P***	***T***	***C, I***
Manufacturer	Device Number	Revision Level	Internal Reference	Package Material	Package Type	Temperature Specification

Device Number:

AHA4525

Revision Letter:

A

Package Material:

P	Plastic

Package Type:

T	Thin Quad Flat Pack

Temperature Specifications:

C	Commercial	0°C to +70°C
I	Industrial	-40°C to +85°C

14.0 RELATED TECHNICAL PUBLICATIONS

DOCUMENT #	DESCRIPTION
PB4524EVB	Product Brief - AHA4524 TPC Evaluation Board
PB4525	Product Brief - AHA4525 IEEE 802.16a Compliant Turbo Product Code Encoder/Decoder
PBGalaxy-EVSW	Product Brief - AHA Galaxy TPC Windows Evaluation Software
ANTPC01	Application Note – Primer: Turbo Product Codes
ANTPC02	Application Note – Use and Performance of Shortened Codes with the AHA4501 TPC Encoder/Decoder)
ANTPC03	Application Note – Turbo Product Code Encoder/Decoder with Quadrature Amplitude Modulation (QAM)
ANTPC04	Application Note – Use and Performance of the AHA4501 TPC Encoder/ Decoder with Differential Phase Shift Keying (DPSK)
ANTPC06	Application Note – AHA Turbo Product Codes Frequently Asked Questions (FAQ)
ANTPC07	Application Note – Turbo Product Codes for LMDS
ANTPC09	Application Note – TPC Code selection and Parameter Determination
TPCEVAL	Evaluation Software – Turbo Product Codes - Windows Evaluation Software

APPENDIX A: VAD RECOMMENDATIONS

Analog Power Supply Filter for PLL usage

The analog power supply for the internal PLL (Vad) ideally comes from an analog power supply having low pass filter characteristics of -3dB at < 1KHz and < -70dB in the absorption band. Actual component selection will limit the -3dB point and good board layout is vital to achieving good high frequency absorption. An R-C or R-L-C filter can be used with the "C" being composed of multiple devices to achieve a wide spectrum of noise absorption.

For DC reasons the series resistance of this filter should be limited. There should be considerably less than 5% voltage drop across the resistor under worst-case conditions. High quality series inductors should be used with a series resistor to prevent a high gain series resonator from being created.

For the low-frequency cutoff an electrolytic capacitor should be used in the filter design. The filter also needs to sustain its attenuation into high frequencies (i.e. > 100 MHz) so additional non-electrolytic capacitance should be added in parallel. All leads of the high frequency capacitor(s) should be kept short. This includes the board wires, vias, and component leads, and within the component package (so select the component carefully). Board layout around the high-frequency capacitance and the path from there to the pads is critical. It is vital that the quiet ground and power are treated like analog signals.

The power (Vad) path must be a single trace from the IC package pin to the high frequency capacitance, then to the low frequency capacitance, then through the series element (e.g. resistor), then to board power (Vdd). The distance from the IC pin to the high frequency cap should be kept as short as possible.

The ground (AGND) path should be from the IC pin to the high frequency capacitance to the low frequency capacitance with the distances being very short. The PLL has the DC ground connection made on chip, so the external Gnd pin must not be connected to PCB ground, but only to the power supply filter. The power and ground traces should be run close and parallel as far as possible with large spacing between adjacent traces. This will help minimize noise, especially non common-mode noise.

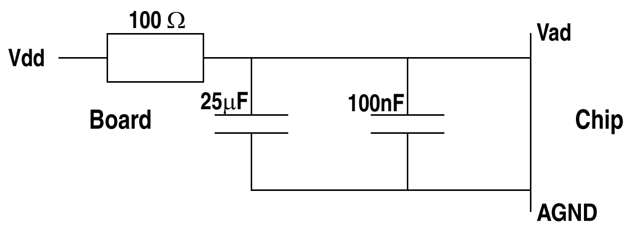
The power loop consisting of the high frequency capacitor, Vad and Gnd traces to the IC, and the IC itself must be designed to keep area and impedance to a minimum. Layout the board to enable the total analog power circuit to be small with short and adjacent wire traces. No extra connections should be made to board power planes; only connect as described above.

Care should be exercised in component selection to insure that there are no resonant absorptions or resonant non-absorptions throughout the attenuating frequency range,. This means that the series element will either be a resistor or a very poor (i.e. resistive) inductor. For a series element with high impedance (i.e. 100 ohms), the electrolytic is often chosen as the biggest capacitance tantalum available which fits reasonably on the board (i.e. 25uF). Similarly, the other capacitor is typically the largest value high frequency capacitor available in a small package (i.e. 100nF).

Example Schematic

This schematic represents an example of both the circuit and the device placement. Actual component values may be different.

Figure A1: Example External Circuit Component Configuration


100 Ω
Vdd
Vad
25µF
100nF
Board
Chip
AGND