TMS320C30-IEEE Floating-Point Format Converter

APPLICATION REPORT: SPRA400

Randy Restle
Regional Technology Center
Waltham, Massachusetts
Adam Cron
Digital Signal Processor Products
Semiconductor Group
Texas Instruments

Digital Signal Processing Solutions

IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Certain application using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage ("Critical Applications").

TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.

TRADEMARKS

TI is a trademark of Texas Instruments Incorporated.

Other brands and names are the property of their respective owners.

CONTACT INFORMATION

US TMS320 HOTLINE	(281) 274-2320
US TMS320 FAX	(281) 274-2324
US TMS320 BBS	(281) 274-2323
US TMS320 email	dsph@ti.com

TMS320C30-IEEE Floating-Point Format Converter

Abstract

Certain applications require the TMS320C30's high arithmetic throughput in the IEEE floating-point format. These applications benefit from a custom chip that performs conversions between the TMS320C30 native format and the single-precision IEEE Standard 754-1985. This chapter describes this custom chip.

The description includes the following specific topics:

- External interfaces
- Architectural overview
- Converter operating modes
- Interrupts
- Software application examples
- Hardware application examples
- JTAG/IEEE-1149.1 scan interface

Product Support

World Wide Web

Our World Wide Web site at www.ti.com contains the most up to date product information, revisions, and additions. New users must register with TI&ME before they can access the data sheet archive. TI&ME allows users to build custom information pages and receive new product updates automatically via email.

Email

For technical issues or clarification on switching products, please send a detailed email to *dsph@ti.com.* Questions receive prompt attention and are usually answered within one business day.

Introduction

Certain applications require the exceptionally high arithmetic throughput inherent in the TMS320C30 Digital Signal Processor but must use the IEEE floating-point number format, which differs from the TMS320C30's number format. The TMS320C30 uses a 2's complement format for the mantissa and exponent. Besides making the device more compatible with analog to digital converters, it is computationally more efficient in both speed and die size than the IEEE format. Applications requiring the IEEE format can benefit from the use of a custom chip for this conversion. For this reason, a chip has been designed, built, and tested. This report describes that chip.

The TMS320C30-IEEE Floating-Point Number Format Converter is a peripheral that performs floating-point number conversions between the native format of the TMS320C30 and the Single-Precision IEEE Standard 754-1985. This conversion is performed in hardware and can convert an incoming (IEEE-formatted) or outgoing (TMS320C30-formatted) floating-point number in less than one TMS320C30 instruction cycle. Normally, the part is placed between memory and the TMS320C30.

This peripheral has two operating modes.

- Mode 1 does not pipeline any data through the chip. Instead, one wait state is automatically generated to compensate for the converter's propagation delays. This mode is equivalent in performance to equipping the TMS320C30 with a single-cycle convert instruction. In those applications where speed is of utmost importance, the pipeline mode is provided.
- Mode 2 enables the converter's built-in pipeline.

Because propagation delays through the chip reduce the access time required for TMS320C30 external memory, the pipeline mode allows conversions to take place on one data value while a previously converted value is being read, or written, by the TMS320C30. Depending on the TMS320C30 instruction cycle time and the access time of memories being used, the pipeline mode can eliminate degradation in TMS320C30 throughput entirely. However, it should be noted that values fed through the pipeline appear at the output in the next cycle. Therefore, an extra read or write (i.e., the same operation that was being performed) must be performed to flush the pipeline. Consequently, when pipeline mode is used, data values and their addresses are skewed from one another. This mode is intended for high-speed block transfer/conversion, and the address skew should be acceptable.

All control signals to and from the converter are compatible with TMS320C30 signals so that no extra circuitry is required to use this chip. In fact, it has been designed to appear as much as possible like a simple bus transceiver (e.g., SN74LS245). Consequently, it has two data buses. Data bus A (pins DA31 through DA0) should be connected directly to one of the TMS320C30's data buses and the other to memory. Its direction pin (DIR) should be tied to the read/write pin (R/$\overline{W}$), and its output enable pin ($\overline{OE}$) can be tied to either $\overline{STRB}$ or $\overline{MSTRB}$ of the TMS320C30, depending on where in the TMS320C30 memory map IEEE numbers are stored.

Key Features

This device is designed to fit into systems equipped with TMS320C30 external memory into which IEEE formatted numbers are stored. Below is a list of some specific features of the TMS320C30-IEEE Floating-Point Converter:

- Automatic wait-state generation during conversions
- Automatic interrupt generation when IEEE NaNs are encountered
- Automatic pipeline mode for single-cycle conversions
- Built-in SCOPE (i.e., JTAG) testability logic

Report Overview

- External Interfaces – Describes the external interfaces of this chip, the pinout, and pins.
- Architectural Overview – Describes the functions of the converter. Gives an overview of the TMS320C30 and IEEE Standard 754-1985 number formats and the scope of numbers that can be converted.
- Converter Operating Modes – Describes the converter's operating modes.
- Interrupts – Describes the Not a Number interrupt generated by the converter.
- Software Application Examples – Contains software application examples.
- Hardware Application Examples – Contains hardware application examples.
- JTAG/IEEE-1149.1 Scan Interface – Contains the JTAG/IEEE scan interface description.

Typographical Conventions

In this report, buses are signified with the bus name in capital letters, followed by the range of signals (bits) enclosed in parentheses and separated by a colon. For example, TI(31:0) is bus "TI", bits 31 through 0 (31 is the most significant bit, 0, the least). Table 1 shows the symbols and their corresponding meaning that are used in sections of the report concerning control logic, algorithm overview, and bit-specific conversion algorithms.

Table 1. Symbols and Meanings

Symbol	Name	Meaning
+	plus	arithmetic summation
|	pipe	logical OR
&	ampersand	logical AND
!	exclamation point	one's complement
–	minus	two's complement
^	caret	EXCLUSIVE OR

External Interfaces

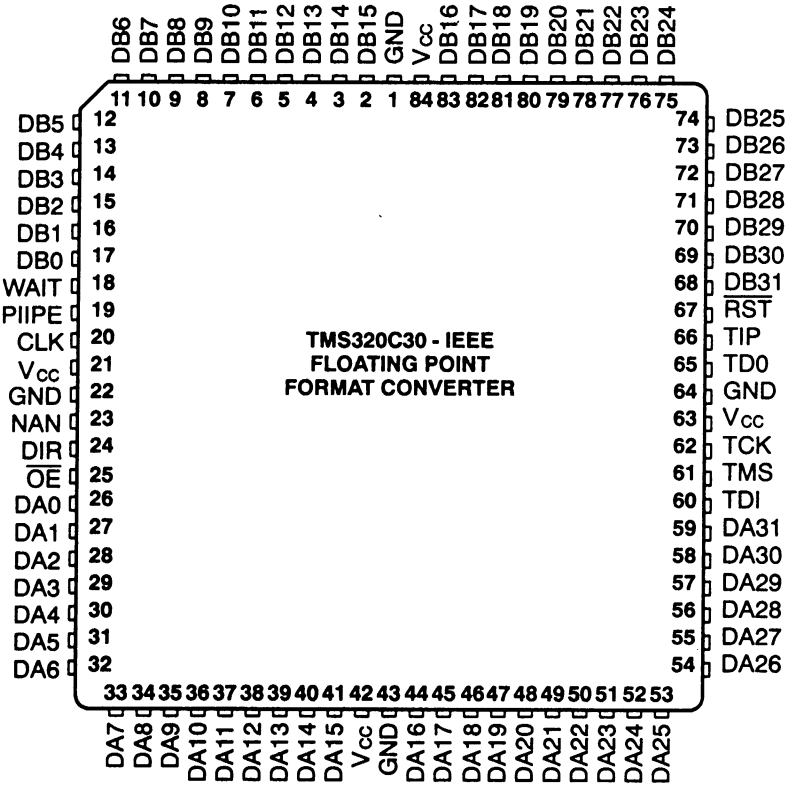
Packaging

The TMS320C30 device is housed in an 84-pin package. This pinout was chosen for efficient flow through connection to the buses. The TMS320C30-IEEE Converter's pin assignments are shown in Table 2, and the pin locations are shown in Figure 1.

Table 2. Pin Assignments

Pin	Name	Pin	Name	Pin	Name
1	GND	29	DA3	57	DA29
2	DB15	30	DA4	58	DA30
3	DB14	31	DA5	59	DA31
4	DB13	32	DA6	60	TDI
5	DB12	33	DA7	61	TMS
6	DB11	34	DA8	62	TCK
7	DB10	35	DA9	63	VCC
8	DB9	36	DA10	64	GND
9	DB8	37	DA11	65	TDO
10	DB7	38	DA12	66	TIP
11	DB6	39	DA13	67	RST
12	DB5	40	DA14	68	DB31
13	DB4	41	DA15	69	DB30
14	DB3	42	VCC	70	DB29
15	DB2	43	GND	71	DB28
16	DB1	44	DA16	72	DB27
17	DB0	45	DA17	73	DB26
18	WAIT	46	DA18	74	DB25
19	PIPE	47	DA19	75	DB24
20	CLK	48	DA20	76	DB23
21	VCC	49	DA21	77	DB22
22	GND	50	DA22	78	DB21
23	NAN	51	DA23	79	DB20
24	DIR	52	DA24	80	DB19
25	OE	53	DA25	81	DB18
26	DA0	54	DA26	82	DB17
27	DA1	55	DA27	83	DB16
28	DA2	56	DA28	84	VCC

Figure 1. Pin Locations



Pinout Description

Table 3 describes the pin functions.

Table 3. Converter Signals

Signal	Pins	Type	Description
DIR	1	Input	Direction – This pin determines what type of conversion should take place. When it is high, data on bus B is converted from IEEE to TMS320C30 format and output on bus A. When it is low, data on bus A is converted from TMS320C30 to IEEE format and output on bus B. This pin is normally tied directly to the TMS320C30 read/write pin.
$\overline{OE}$	1	Input	Output Enable (active low) – In combination with the DIR pin, this pin disables the currently driven bus (i.e., bus A or B).

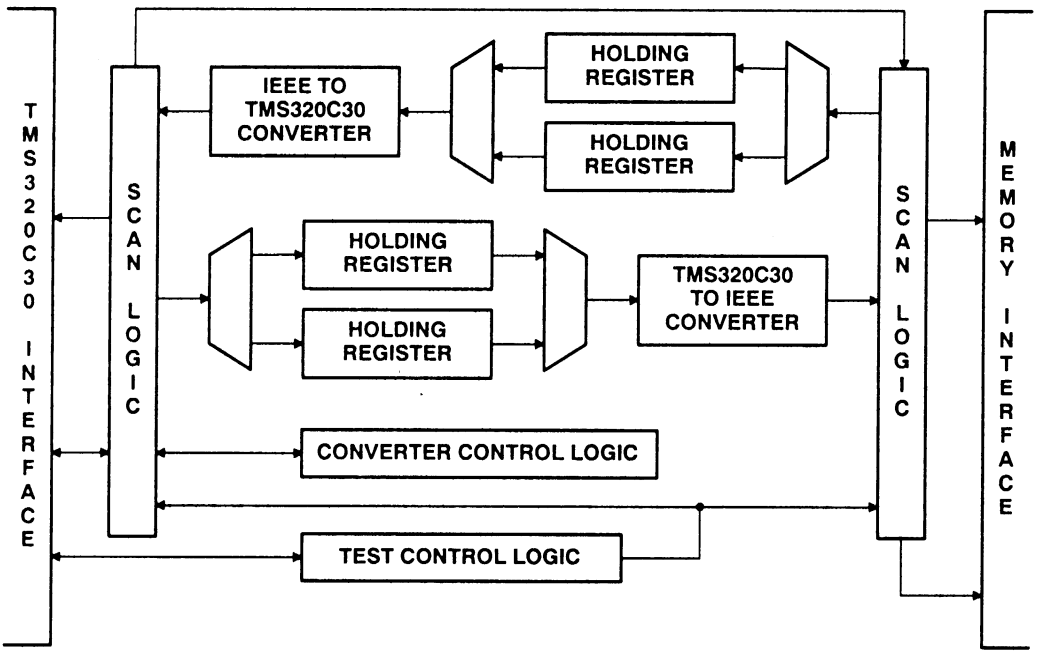
Table 3. Converter Signals (Concluded)

Signal	Pins	Type	Description
WAIT	1	Output	This pin is driven high in nonpipelined operations to signal the TMS320C30 to extend its external memory access to allow the conversion to complete. It can be tied directly to the TMS320C30 ready line. It is appropriately driven for both read and write operations, but is always low in pipelined mode of operation.
PIPE	1	Input	Pipeline Enable – When this is high, the converter is configured in pipeline mode. It must be tied low for nonpipeline mode.
CLK	1	Input	Clock – This clock is the wait-state generator and the pipeline clock. It should be connected directly to the TMS320C30 H1 clock pin.
NAN	1	Output	Not-a-Number Interrupt – This pin is driven low for 1.5 CLK cycles and signals an attempted conversion of the IEEE format: Not-a-Number. This pin can be tied directly to one of the TMS320C30 interrupt pins and can signal command or message passing in multi-processor, shared-memory-type designs.
DA(31:0)	32	Input/Output	Data Bus A – This 32-bit bus should be tied to either one of the two TMS320C30 data buses (i.e., the primary or expansion buses).
DB(31:0)	32	Input/Output	Data Bus B – This 32-bit bus is normally connected to a memory array containing IEEE-formatted data.
TCK	1	Input	Test Clock.
TMS	1	Input	Test Mode Select.
$\overline{\text{RST}}$	1	Input	Reset (active low) – This pin resets all logic on the device.
TDI	1	Input	Test Data In.
TDO	1	Output	Test Data Out.
TIP	1	Output	Test Instruction Register Parity – During instruction register scan, when paused, this output reflects instruction register even parity.

Architectural Overview

Figure 2 shows the block diagram of the converter.

Figure 2. Converter Block Diagram



Introduction

The TMS320C30 attains a peak performance of 33 MFLOPS, largely due to the floating-point format that it uses. In this format, both exponent and mantissa are represented in 2's-complement form.

In the IEEE format, the mantissa is represented in signed-magnitude form, and the exponent includes a bias (i.e., an offset). Additionally, values of numbers are not determined by the same formula. Instead, the exponent is used to flag numbers that are encoded differently. For example, if the exponent is 255, the value is considered not a number (NaN). Another exception is signaled when the exponent is zero. In this case, the mantissa is defined to be denormalized.
The TMS320C30's floating-point format is considerably simpler; most numbers can be converted to it without any loss of precision. However, some denormalized IEEE numbers are smaller than can be represented in TMS320C30 format. When these numbers are converted, they are translated to the closest TMS320C30 values. The error is less than $\pm 2^{-127}$.

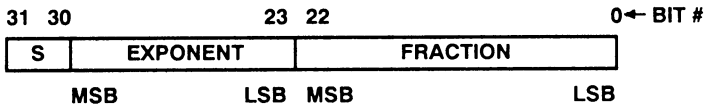
IEEE Floating-Point Format Overview

IEEE Standard 754-1985 defines formats for single-, single-extended-, double- and double-extended-precision floating-point numbers. The single-precision format fits entirely with-

in 32 bits, which is the bus width of the TMS320C30, and is the only format supported by the converter.

The format of the single-precision IEEE Standard 754-1985 is shown below:

Figure 3. Single-Precision IEEE Standard 754-1985 Format



In this format,

S is the sign bit of the mantissa (0 = positive, 1 = negative).

EXPONENT is an unsigned 8-bit field that determines the location of the binary point of the number being encoded.

FRACTION is a 23-bit field containing the fractional part of the mantissa.

LSB is the least significant bit of a field

MSB is the most significant bit of a field

The decimal value (v) of some number X is defined by one of five separate cases shown below:

Case 1: If **EXPONENT** = 255 and **FRACTION** $\neq$ 0, then v is NaN.

Case 2: If **EXPONENT** = 255 and **FRACTION** = 0, then v = $\pm$ infinity.

Case 3: If 0 < **EXPONENT** < 255, then $v = (-1)^{s}\, 2^{\mathbf{exp}-127}\, (1.\mathbf{FRAC})$

where:

S is either 0 or 1

FRAC is the decimal equivalent of **FRACTION**

EXP is the decimal equivalent of **EXPONENT**

Note that an implied 1 exists to the left of the binary point as shown above. This means the mantissa of an IEEE-encoded value has 24 bits of precision.

Case 4: If **EXPONENT** = 0 and **FRACTION** $\neq$ 0, then v is a denormalized number and $v = (-1)^{s}\, 2^{-126}\, (0.\mathbf{FRAC})$

where
S is either 0 or 1

FRAC is the decimal equivalent of **FRACTION**

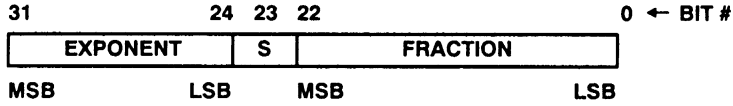
Note that an implied 0 exists to the left of the binary point as shown above. This means the mantissa of an IEEE-encoded value has 24 bits of precision.

Case 5: If **EXPONENT** = 0 and **FRACTION** = 0, then v = ± zero.

TMS320C30 Floating-Point Format Overview

TMS320C30 single-precision floating-point format uses a 2's-complement exponent and mantissa and is shown in Figure 4.

Figure 4. TMS320C30 Single-Precision Floating-Point Format



The decimal value (v) of some number X is determined as follows:

$v = \{(-2)^{S} + (.\mathbf{FRAC})\}\ 2^{\mathbf{exp}}$

where **S** is either 0 or 1

FRAC is the decimal equivalent of **FRACTION**

EXP is the decimal equivalent of **EXPONENT**

An alternate way of describing the TMS320C30 mantissa is as follows:

$\bar{s}$s.fraction

Note that the bit to the left of the binary point is implied and is the complement of the sign bit. This gives the TMS320C30's mantissa 24 bits of precision and not 23 bits as might be expected. For example:

The most positive TMS320C30 mantissa is

$01.1111\ 1111\ 1111\ 1111\ 1111\ 111 = 2 - 2^{-23}$

The least positive TMS320C30 mantissa is

$01.0000\ 0000\ 0000\ 0000\ 0000\ 000 = 1$

The most negative TMS320C30 mantissa is

$10.0000\ 0000\ 0000\ 0000\ 0000\ 000 = -2$

The least negative TMS320C30 mantissa is

$10.1111\ 1111\ 1111\ 1111\ 1111\ 111 = -1 - 2^{-23}$

Note that zero is uniquely identified when the TMS320C30 exponent is –128.

IEEE Number Conversion

This section describes the classifications of IEEE numbers, how they are decoded, and the algorithms necessary to translate them to TMS320C30 format.

IEEE Dynamic Range

Table 4 shows the dynamic range of IEEE numbers. This chart can be used to quickly determine the case classification of an IEEE number.

Table 4. IEEE Range of Numbers

Sign	Exponent	Mantissa	Value	Type	Case
0	FF	0	not applicable	NaN	1
0	FF	0.000...000	+ infinity	+ Infinity	2A
0	FE	1.111...111	$(2-2^{-23})x2^{127}$	+ Normalized Number	3A
0	FE	1.111...110	$(2-2^{-22})x2^{127}$	+ Normalized Number	3A
0	FE	1.111...101	$(2-2^{-21}+2^{-23})x2^{127}$	+ Normalized Number	3A
0	FE	1.111...100	$(2-2^{-21})x2^{127}$	+ Normalized Number	3A
	.	.			
	.	.			
	.	.			
0	FE	1.000...000	2^{127}	+ Normalized Number	3A
0	FD	1.111...111	$(2-2^{-23})x2^{126}$	+ Normalized Number	3A
0	FD	1.111...110	$(2-2^{-22})x2^{126}$	+ Normalized Number	3A
0	FD	1.111...101	$(2-2^{-21}+2^{-23})x2^{126}$	+ Normalized Number	3A
0	FD	1.111...100	$(2-2^{-21})x2^{126}$	+ Normalized Number	3A
	.	.			
	.	.			
	.	.			
0	01	1.000...000	2^{-126}	+ Normalized Number	3A
0	00	0.111...111	$(1-2^{-23})x2^{-126}$	+ Denormalized Number	4A
0	00	0.111...110	$(1-2^{-22})x2^{-126}$	+ Denormalized Number	4A
0	00	0.111...101	$(1-2^{-21}+2^{-23})x2^{-126}$	+ Denormalized Number	4A
0	00	0.111...100	$(1-2^{-21})x2^{-126}$	+ Denormalized Number	4A
	.	.			
	.	.			
	.	.			
0	00	0.100...000	2^{-127}	+ Denormalized Number	4A
0	00	0.011...111	$(1-2^{-22})x2^{-127}$	– Denormalized Number	4B
0	00	0.011...110	$(1-2^{-21})x2^{-127}$	– Denormalized Number	4B
0	00	0.011...101	$(1-2^{-20}+2^{-22})x2^{-127}$	– Denormalized Number	4B
	.	.			
	.	.			
	.	.			
0	00	0.000...011	$(1+2^{-1})x2-148$	– Denormalized Number	4B
0	00	0.000...010	2^{-148}	– Denormalized Number	4B
0	00	0.000...001	2^{-149}	– Denormalized Number	4B
0	00	0.000...000	+ 0.0	+ Zero	5
1	00	0.000...000	– 0.0	– Zero	5
1	00	0.000...001	$-(2^{-149})$	– Denormalized Number	4D
1	00	0.000...010	$-(2^{-148})$	– Denormalized Number	4D
1	00	0.000...011	$-(1+2^{-1})x2^{-148}$	– Denormalized Number	4D
	.	.			
	.	.			
	.	.			

Table 4. IEEE Range of Numbers (Concluded)

Sign	Exponent	Mantissa	Value	Type	Case
1	00	0.011...111	$-(1-2^{-22})x2^{-127}$	– Denormalized Number	4D
1	00	0.100...000	$-(2^{-127})$	– Denormalized Number	4D
1	00	0.100...001	$-(1+2^{-22})x2^{-127}$	– Denormalized Number	4C
1	00	0.100...010	$-(1+2-21)x2-127$	– Denormalized Number	4C
1	00	0.100...011	$-(1+2^{-21}+2^{-22})x2^{-127}$	– Denormalized Number	4C
	.	.			
	.	.			
	.	.			
1	00	0.111...111	$-(1-2^{-23})x2^{-126}$	– Denormalized Number	4C
1	01	1.000...000	$-(2^{-126})$	– Normalized Number	3C
1	01	1.000...001	$-(1+2-^{23})x2^{-126}$	– Normalized Number	3B
1	01	1.000...010	$-(1+2^{-22})x2^{-126}$	– Normalized Number	3B
1	01	1.000...011	$-(1+2^{-22}+2^{-23})x2^{-126}$	– Normalized Number	3B
	.	.			
	.	.			
	.	.			
1	01	1.111...111	$-(2-2^{-23})x2^{-126}$	– Normalized Number	3B
1	02	1.000...000	$-(2^{-125})$	– Normalized Number	3C
1	02	1.000...001	$-(2+2^{-23})x2^{-125}$	– Normalized Number	3B
1	02	1.000...010	$-(2+2^{-22})x2^{-125}$	– Normalized Number	3B
1	02	1.000...011	$-(1+2^{-22}+2^{-23})x2^{-125}$	– Normalized Number	3B
	.	.			
	.	.			
	.	.			
1	FE	1.111...100	$-(2-2^{-21})x2^{127}$	– Normalized Number	3B
1	FE	1.111...101	$-(2-2^{-21}+2^{-23})x2^{127}$	– Normalized Number	3B
1	FE	1.111...110	$-(2-2^{-22})x2^{127}$	– Normalized Number	3B
1	FE	1.111...111	$-(2-2^{-23})x2^{127}$	– Normalized Number	3B
1	FF	= 0	– infinity	– Infinity	2B

IEEE-to-TMS320C30 Control Logic

The control logic that classifies incoming IEEE data in order to perform correct translation to TMS320C30 format is shown below. The form of the expressions was chosen to minimize propagation delay through the device.

The logic is simplified if the following three factors are used (refer to typographical definitions for symbols used):

```
EXPFF =  IEEE(30)   & IEEE(29)   & IEEE(28)   & IEEE(27)   &
         IEEE(26)   & IEEE(25)   & IEEE(24)   & IEEE(23)

EXP00 = !( IEEE(30) | IEEE(29)   | IEEE(28)   | IEEE(27)   |
         IEEE(26)   | IEEE(25)   | IEEE(24)   | IEEE(23)   )

MANT0 = !( IEEE(21) | IEEE(20)   | IEEE(19)   | IEEE(18)   |
         IEEE(17)   | IEEE(16)   | IEEE(15)   | IEEE(14)   |
```

IEEE(13) | IEEE(12) | IEEE(11) | IEEE(10) |
IEEE(9) | IEEE(8) | IEEE(7) | IEEE(6) |
IEEE(5) | IEEE(4) | IEEE(3) | IEEE(2) |
IEEE(1) | IEEE(0))

Then

Case 1: NaN

= EXPFF & (IEEE(22) | !MANT0)

Case 2A: positive infinity

= !IEEE(31) & EXPFF & !(IEEE(22) | !MANT0)

Case 2B: negative infinity

= IEEE(31) & EXPFF & !(IEEE(22) | !MANT0)

Case 3A: positive normalized numbers

= !IEEE(31) & !EXP00 & !EXPFF

Case 3B: negative normalized numbers with fraction $\neq 0$

= IEEE(31) & !EXP00 & !EXPFF & (!MANT0 | IEEE(22))

Case 3C: negative normalized numbers with fraction = 0

= IEEE(31) & !EXP00 & !EXPFF & !(!MANT0 | IEEE(22))

Case 4A: positive denormalized numbers $\geq 2^{-127}$

= !IEEE(31) & EXP00 & IEEE(22)

Case 4B: positive denormalized numbers $< 2^{-127}$

= !IEEE(31) & EXP00 & !IEEE(22) & !MANT0

Case 4C: negative denormalized numbers $\leq (-1 - 2^{-23}) \times 2^{-127}$

= IEEE(31) & EXP00 & IEEE(22) & !MANT0

Case 4D: negative denormalized numbers $> (-1 - 2^{-23}) \times 2^{-127}$

= IEEE(31) & EXP00 & (IEEE(22) ^ !MANT0)

Case 5: positive and negative zero

= EXP00 & !IEEE(22) & MANT0

IEEE-to-TMS320C30 Conversion Algorithm Overview

Table 5 shows the conversion algorithms used on the sign, exponent, and mantissa fields of IEEE numbers to produce the corresponding TMS320C30 fields. These fields are broken down into bit-specific algorithms in the following section.

Table 5. Conversion Algorithms from IEEE to TMS320C30 Format

	TMS320C30		
Case	**Exponent**	**Sign**	**Fraction**
1.	e_{IEEE}	s_{IEEE}	f_{IEEE}
2A.	7Fh	s_{IEEE}	7F FFFFh
2B.	7Fh	s_{IEEE}	00 0000h
3A.	e_{IEEE} + 81h	s_{IEEE}	f_{IEEE}
3B.	e_{IEEE} + 81h	s_{IEEE}	$-f_{IEEE}$
3C.	e_{IEEE} ^ 80h	s_{IEEE}	$-f_{IEEE}$
4A.	81h	s_{IEEE}	2 x f_{IEEE}
4B.	80h	s_{IEEE}	00 0000h
4C.	81h	s_{IEEE}	2 x $-f_{IEEE}$
4D.	80h	0	00 0000h
5.	80h	0	00 0000h

Note: Fraction, above, has only 23-bits

IEEE-to-TMS320C30 Bit-Specific Conversion Algorithms

These circuits were designed by examining Table 5 and finding all possible choices for each bit. The different choices were fed into data selectors, whose addresses were derived from the case-identifying logic described in the preceding section on control logic.

For maximum performance, all data selectors were designed from NAND gates. This also permitted minimization by eliminating all NAND gates that had an input of 0 and by reducing the number of NAND inputs where a bit was always 1. However, for clarity, no minimization is shown here. Instead, that detail can be seen in the following figures.

The following bit algorithms are shown in bit descending order, starting with IEEE bit 31.

Figure 5. IEEE Bit 31 to TMS320C30 Bit 23

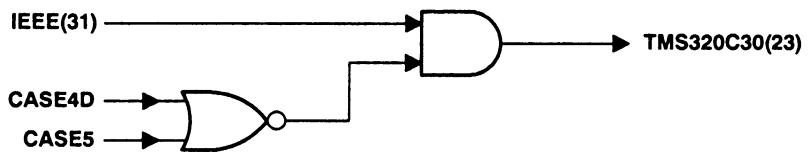
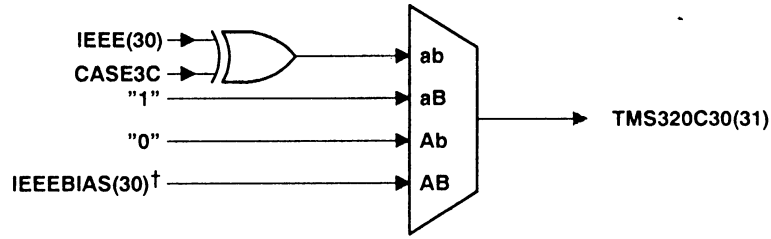


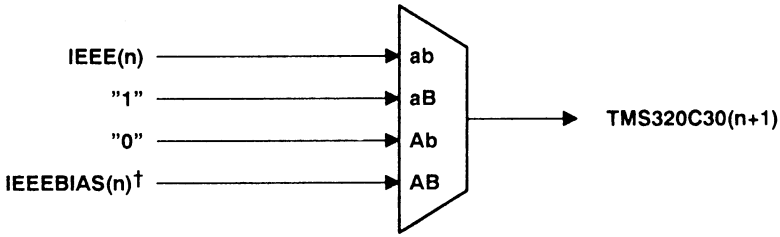
Figure 6. IEEE Bit 30 to TMS320C30 Bit 31



† IEEEBIAS is the result of the case 3A and case 3B arithmetic operations shown in Table 5 under the heading "Exponent". The arithmetic shown is unsigned addition, ignoring the carry.

b = CASE1 | CASE2A | CASE2B | CASE3C
B = !b
A = CASE2A | CASE2B | CASE3A | CASE3B
a = !A

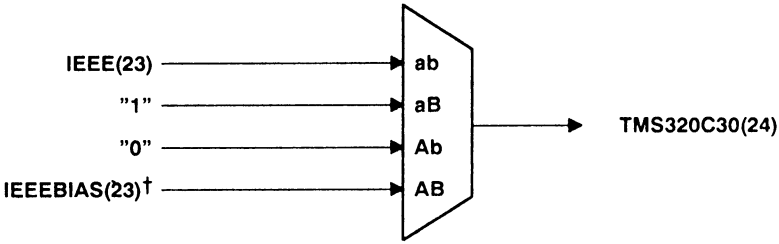
Figure 7. IEEE Bit n to TMS320C30 Bit n+1, Where 29 ≥ n ≥ 24



† IEEEBIAS is the result of the case 3A and case 3B arithmetic operations shown in Table 5 under the heading "Exponent". The arithmetic shown is unsigned addition, ignoring the carry.

b = CASE2A | CASE2B | CASE3A | CASE3B
B = !b
a = CASE2A | CASE2B | CASE1 | CASE3C
A = !a

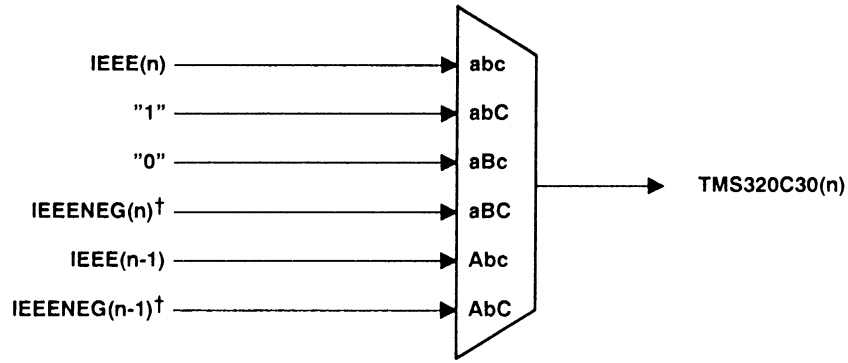
Figure 8. IEEE Bit 23 to TMS320C30 Bit 24



† IEEEBIAS is the result of the case 3A and case 3B arithmetic operations shown in Table 5 under the heading "Exponent". The arithmetic shown is unsigned addition, ignoring the carry.

b = CASE1 | CASE3C | CASE4B | CASE4D | CASE5
B = !b
A = CASE4B | CASE4D | CASE5 | CASE3A | CASE3B
a = !A

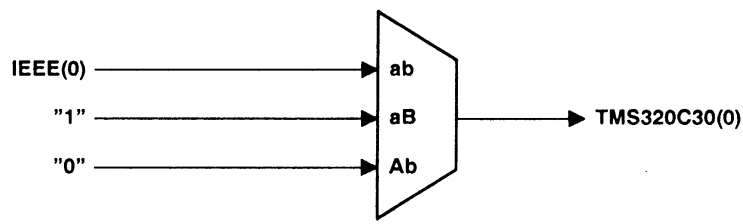
Figure 9. IEEE Bit n to TMS320C30 Bit n, Where 22 ≥ n ≥ 1



† IEEENEG is the result of taking the 2s complement of the fractional part of the IEEE number for case 3A and case 3B as shown in Table 5 under the heading "Fraction".

C = CASE2A | CASE3B | CASE3C | CASE4C
c = !C
b = CASE1 | CASE2A | CASE3A | CASE4A | CASE4C
B = !b
A = CASE4A | CASE4C
a = !A

Figure 10. IEEE Bit 0 to TMS320C30 Bit 0



B = CASE2A
b = !B
A = CASE1 | CASE2A | CASE3A | CASE3B | CASE3C
a = !A

TMS320C30 Number Conversion

This section describes the classifications of TMS320C30 numbers, how they are decoded, and the algorithms necessary to translate them to IEEE format.

TMS320C30 Dynamic Range

Shown in Table 6 is the dynamic range of TMS320C30 numbers. As with Table 4, this table can be used to quickly determine case classification of a TMS320C30 number.

Table 6. TMS320C30 Range of Numbers

Exponent	Sign	Mantissa	Value	Type	Case
7F	0	1.111...111	$(2-2^{-23})x2^{127}$	Positive Number	6
7F	0	1.111...110	$(2-2^{-22})x2^{127}$	Positive Number	6
7F	0	1.111...101	$(2-2^{-21}+2^{-23})x2^{127}$	Positive Number	6
7F	0	1.111...100	$(2-2^{-21})x2^{127}$	Positive Number	6
.		.			
.		.			
.		.			
7F	0	1.000...000	2^{127}	Positive Number	6
7E	0	1.111...111	$(2-2^{-23})x2^{126}$	Positive Number	6
7E	0	1.111...110	$(2-2^{-22})x2^{126}$	Positive Number	6
7E	0	1.111...101	$(2-2^{-21}+2^{-23})x2^{126}$	Positive Number	6
.		.			
.		.			
.		.			
00	0	1.000...000	1	Positive Number	6
FF	0	1.111...111	$1-2^{-24}$	Positive Number	6
FF	0	1.111...110	$1-2^{-23}$	Positive Number	6
FF	0	1.111...101	$1-2^{-22}+2^{-24}$	Positive Number	6
.		.			
.		.			
.		.			
FF	0	1.000...000	2^{-1}	Positive Number	6
FE	0	1.111...111	$(2-2^{-23})x2^{-2}$	Positive Number	6
FE	0	1.111...110	$(2-2^{-22})x2^{-2}$	Positive Number	6
FE	0	1.111...101	$(2-2^{-21}+2^{-23})x2^{-2}$	Positive Number	6
.		.			
.		.			
.		.			
82	0	1.000...000	2^{-126}	Positive Number	6
81	0	1.111...111	$(2-2^{-23})x2^{-127}$	Positive Number	7 (note 1)
81	0	1.111...110	$(2-2^{-22})x2^{-127}$	Positivr Number	7 (note 1)
81	0	1.111...101	$(2-2^{-21}+2^{-23})x2^{-127}$	Positive Number	7 (note 1)
81	0	1.111...100	$(2-2^{-21})x2^{-127}$	Positive Number	7 (note 1)
.		.			
.		.			
.		.			
81	0	1.000...010	$(1+2^{-22})x2^{-127}$	Positive Number	7 (note 1)
81	0	1.000...001	$(1+2^{-23})x2^{-127}$	Positive Number	7 (note 1)
81	0	1.000...000	2^{-127}	Positive Number	7 (note 1)
80	0	0.111...111	(note 2)	Implied Zero	8
80	0	0.111...110	(note 2)	Implied Zero	8
80	0	0.111...101	(note 2)	Implied Zero	8
.		.			
.		.			
.		.			
80	0	0.000...001	(note 2)	Implied Zero	8

Table 6. TMS320C30 Range of Numbers (Concluded)

Exponent	Sign	Mantissa	Value	Type	Case
80	0	0.000...000	0.0	Zero	8
80	1	10.111...111	(note 2)	Implied Zero	(note 3)
80	1	10.111...110	(note 2)	Implied Zero	(note 3)
80	1	10.111...101	(note 2)	Implied Zero	(note 3)
.		.			
.		.			
.		.			
80	1	10.000...011	(note 2)	Implied Zero	(note 3)
80	1	10.000...010	(note 2)	Implied Zero	(note 3)
80	1	10.000...001	(note 2)	Implied Zero	(note 3)
80	1	10.000...000	(note 2)	Implied Zero	8
81	1	10.111...111	$(-1-2^{-23})x2^{-127}$	Negative Number	9 (note 1)
81	1	10.111...110	$(-1-2^{-22})x2^{-127}$	Negative Number	9 (note 1)
81	1	10.111...101	$(-1-2^{-21}+2^{-23})x2^{-127}$	Negative Number	9 (note 1)
.		.			
.		.			
.		.			
81	1	10.000...010	$(-2+2^{-22})x2^{-127}$	Negative Number	9 (note 1)
81	1	10.000...001	$(-2+2^{-23})x2^{-127}$	Negative Number	9 (note 1)
81	1	10.000...000	$-(2^{-126})$	Negative Number	10
82	1	10.111...111	$(-1-2^{-23})x2^{-126}$	Negative Number	11
82	1	10.111...110	$(-1-2^{-22})x2^{-126}$	Negative Number	11
82	1	10.111...101	$(-1-2^{-21}+2^{-23})x2^{-126}$	Negative Number	11
.		.			
.		.			
.		.			
FF	1	10.000...001	$-1+2^{-24}$	Negative Number	11
FF	1	10.000...000	-1	Negative Number	10
00	1	10.111...111	$(-1-2^{-23})x2^{-1}$	Negative Number	11
00	1	10.111...110	$(-1-2^{-22})x2^{-1}$	Negative Number	11
00	1	10.111...101	$(-1-2^{-21}+2^{-23})x2^{-1}$	Negative Number	11
.		.			
.		.			
.		.			
00	1	10.000...001	$-2+2^{-23}$	Negative Number	11
00	1	10.000...000	-2	Negative Number	10
01	1	10.111...111	$-2-2^{-22}$	Negative Number	11
01	1	10.111...110	$-2-2^{-21}$	Negative Number	11
01	1	10.111...101	$-2-2^{-20}+2^{-22}$	Negative Number	11
.		.			
.		.			
.		.			
7F	1	10.000...001	$(-2+2^{-23})x2^{127}$	Negative Number	11
7F	1	10.000...000	$-(2^{128})$	Negative Number	12

Notes: 1) Numbers converted to IEEE denormalized values lose one least significant bit of accuracy.

2) The TMS320C30 does not produce these numbers under normal arithmetic operations. Because the exponent of these numbers is –128, the TMS320C30 considers them zero. TMS320C30 Boolean operations are capable of producing numbers of these forms. Because of this, proper conversion to IEEE format is unclear and should be avoided. See note 3.

3) Case 8 & Case 9 are activated simultaneously. This is the only instance where the cases are not mutually exclusive. The TMS320C30 does not produce these numbers under normal arithmetic operations. Because the exponent of these numbers is –128, the TMS320C30 considers them zero. TMS320C30 Boolean operations are capable of producing numbers of these forms. Because of this, proper conversion to IEEE format is unclear. This dilemma can be resolved with minor modification to the case qualifier logic. See note 2.

TMS320C30-to-IEEE Control Logic

Conversion from TMS320C30 format to IEEE format is qualified with a different set of Boolean equations. To eliminate confusion between IEEE and TMS320C30 cases, different case numbers are used.

The logic is simplified if the following three factors are used:

```
EXP80_81 =  !C30(31) | C30(30) | C30(29) | C30(28) |
            C30(27) | C30(26) | C30(25)

EXP7F =     !C30(31) & C30(30) & C30(29) & C30(28) &
            C30(27) & C30(26) & C30(25) & C30(24)

MANT0 =     C30(22) | C30(21) | C30(20) | C30(19) |
            C30(18) | C30(17) | C30(16) | C30(15) |
            C30(14) | C30(13) | C30(12) | C30(11) |
            C30(10) | C30(9)  | C30(8)  | C30(7)  |
            C30(6)  | C30(5)  | C30(4)  | C30(3)  |
            C30(2)  | C30(1)  | C30(0)
```

Then,

Case 6: positive numbers $\geq 2^{-126}$

= !EXP80_81 & !C30(23)

Case 7: positive numbers N such that

$(2-2^{-23}) \times 2^{-127} \geq N \geq 2-127$

= EXP80_81 & C30(24) & !C30(23)

Case 8: zero

= EXP80_81 & C30(24)

Case 9: negative numbers N such that

$(-1-2^{-23})x2^{-127} \geq N \geq (-2+2^{-23})x2^{-127}$

= EXP80_81 & C30(23) & !MANT0

Case 10: negative numbers N such that

$-(2^{-126}) \geq N \geq -(2^{127})$ and whose fraction is 0

= !(EXP80_81 & !C30(24)) & !EXP7F & C30(23) & MANT0

Case 11: negative numbers N such that

$-(2^{-126}) > N > -(2^{128})$ and whose fraction $\neq 0$

= !EXP80_81 & C30(23) & !MANT0

Case 12: negative 2^{128}

= EXP7F & C30(23) & MANT0

TMS320C30-to-IEEE Conversion Algorithm Overview

Table 7 shows the conversion algorithms used on the sign, exponent, and mantissa fields of TMS320C30 numbers to produce the corresponding IEEE fields. These fields are broken down into bit-specific algorithms in the next section.

Table 7. Conversion Algorithms from TMS320C30 to IEEE Format

	IEEE		
Case	**Sign**	**Exponent**	**Fraction**
6	s_{C30}	e_{C30}+7Fh	f_{C30}
7	s_{C30}	00	(f_{C30}/2)+400000h
8	0	00	00 0000h
9	s_{C30}	00	($\bar{f}_{C30}$+1)/2+400000h
10	s_{C30}	e_{C30}+80h	00 0000h
11	s_{C30}	e_{C30}+7Fh	$\bar{f}_{C30}$+1
12	s_{C30}	FFh	00 0000h

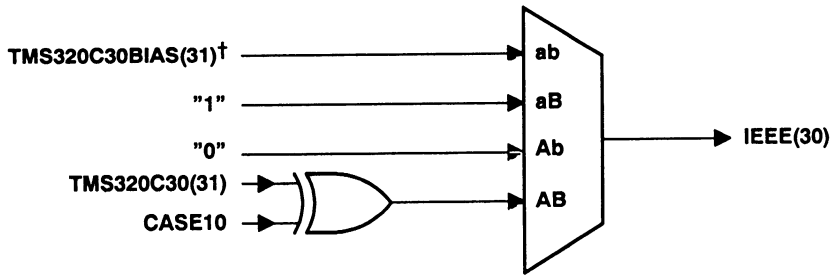
TMS320C30-to-IEEE Bit-Specific Conversion Algorithms

These circuits were designed by examining Table 7 and finding all possible choices for each bit. The different choices were fed into data selectors whose addresses were derived from the case-identifying logic described in the preceding section on TMS320C30 to IEEE control logic.

Just as in the IEEE case-identifying logic, all data selectors were designed from NAND gates for maximum performance. This also permitted minimization by eliminating all NAND gates having an input of 0 and by reducing the number of NAND inputs where a bit was always 1. However, for clarity, no minimization is shown here. Instead, that detail can be seen in the following figures.

The following bit algorithms are shown in bit-descending order, starting with TMS320C30 bit 31.

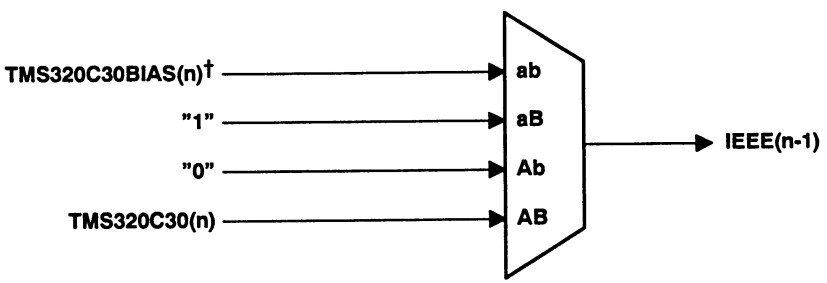
Figure 11. TMS320C30 Bit 31 to IEEE Bit 30



† TMS320C30BIAS is the result of the arithmetic operations for cases 6 through 9 and 11 shown in Table 7 under the heading "Exponent". The arithmetic shown is unsigned addition, ignoring the carry.

```
B = CASE10 | CASE12
b = !B
a = CASE6 | CASE11 | CASE12
A = !a
```

Figure 12. TMS320C30 Bit n to IEEE Bit n–1, Where 31 ≥ n ≥ 24



† TMS320C30BIAS is the result of the arithmetic operations for cases 6 through 9 and 11 shown in Table 7 under the heading "Exponent". The arithmetic shown is unsigned addition, ignoring the carry.

```
B = CASE10 | CASE12
b = !B
a = CASE6 | CASE11 | CASE12
A = !a
```

Figure 13. TMS320C30 Bit 23 to IEEE Bit 31

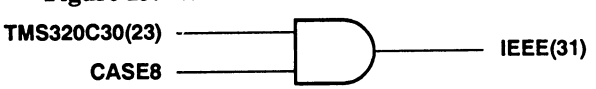
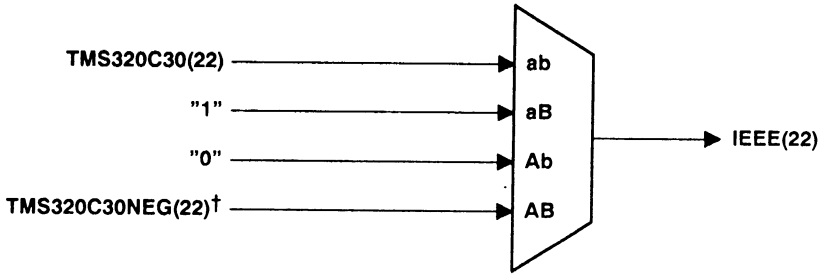


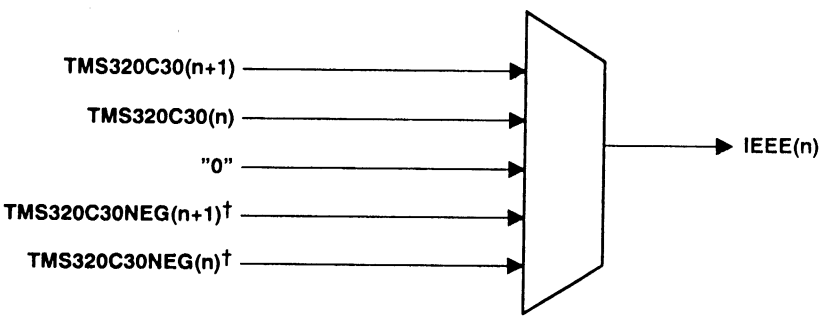
Figure 14. TMS320C30 Bit 22 to IEEE Bit 22



† TMS320C30NEG is the result of the Case 9 and Case11 arithmetic operations shown in Table 7 under the heading "Fraction". The logic that generates TMS320C30NEG must encompass the 400000h value, which is easily derived by using the CASE9 signal itself (CASE9 = 0 when CASE11 is active).

```
B = CASE7 | CASE9 | CASE11
b = !B
a = CASE6 | CASE7 | CASE9
A = !a
```

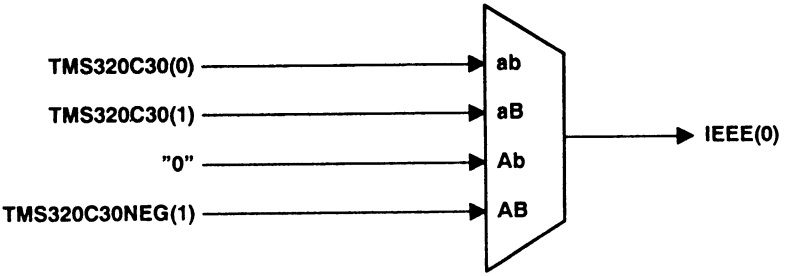
Figure 15. TMS320C30 Bit n to IEEE Bit n, Where 21 ≥ n ≥ 1



† TMS320C30NEG is the result of the Case 9 and Case11 arithmetic operations shown in Table 7 under the heading "Fraction". The logic that generates TMS320C30NEG must encompass the 400000h value, which is easily derived by using the CASE9 signal itself (CASE9 = 0 when CASE11 is active).

```
C = CASE6 | CASE9
c = !C
b = CASE6 | CASE7 | CASE11
B = !b
A = CASE11
a = !A
```

Figure 16. TMS320C30 Bit 0 to IEEE Bit 0:



B = CASE7 | CASE9
b = !B
a = CASE6 | CASE7 | CASE11
A = !a

Scope of Conversion

This section describes the actions taken by the converter when it converts to and from the IEEE format. When there is not a match between formats, the converter forces the translated number to the closest approximation.

IEEE-to-TMS320C30 Exceptions

The match is not exact in translating from four sets of IEEE numbers to TMS320C30 numbers. They are: NaN, ± infinity, ± zero and denormalized numbers too small to represent.

NaN (Not a Number)

The NaN format is especially useful in passing commands to another process. So that commands can be passed through the converter, NaNs are not converted. However, the bit positions of the sign and exponent bits are altered. That is, the sign bit of the IEEE number is transferred to the sign bit of the TMS320C30 format. Likewise, the exponent field is transferred. In this way, the sign of the NaN is preserved which may aid in quick detection of the code. In other words, the TMS320C30 Branch on Positive instruction (BP) or Branch on Negative instruction (BN) are effective. So that the command can be acted on quickly, a NaN interrupt is generated.

± *Infinity*

When positive or negative infinity is passed through the converter, the most positive or negative TMS320C30 number is produced.

Denormalized numbers whose magnitude $< 2^{-126}$

Half of the denormalized IEEE numbers are out of range of TMS320C30 numbers. These denormalized numbers have very small magnitudes and are therefore forced to zero when converted.

± *Zero*

The IEEE format includes representations for positive and negative zero, but the TMS320C30 format does not. The converter forces each of these numbers to the singular TMS320C30 zero format.

TMS320C30-to-IEEE Exceptions

There are two sets of TMS320C30 numbers that do not perfectly match IEEE numbers. One set consists of a single value (-2^{127}). The other consists of numbers converted to IEEE denormalized numbers.

-2^{127}

The single value, -2^{127}, is a very large negative number. When this number is translated, negative infinity is produced.

Numbers Translated to Denormalized Values

When the exponent is –127, denormalized IEEE numbers are produced, and one least significant bit of accuracy is lost. This occurs because the TMS320C30 mantissa must be right-shifted one bit in order that the exponent be increased to –126, which is the most negative exponent the IEEE format can use.

Converter Operating Modes

The converter is controlled by the TMS320C30. Conversions occur when the converter's output enable pin ($\overline{OE}$) is active (i.e., low) and the TMS320C30 performs a read or write over its primary ($\overline{STRB}$ active) or expansion ($\overline{MSTRB}$ active) buses. This requires the converter to be placed directly between the TMS320C30 and external memory. That memory is where IEEE data will be stored. If direct (i.e., no conversion wanted) access to that memory is desired, transceivers like the SN74LS245 should be added in parallel with the converter. However, doing so requires that only one data path be enabled at a time. If unused, one of the XF pins of the TMS320C30 can be dedicated to perform this selection.

During a read, data is converted from IEEE format to TMS320C30 format. During a write, data is converted from TMS320C30 format to IEEE format. This will happen if the TMS320C30 R/$\overline{W}$ or XR/$\overline{W}$ pin is tied to the converter's direction (DIR) pin. Table 8 shows how to put the converter into its two operating modes and briefly describes each mode.

Table 8. Converter Operating Modes

Mode	Pin	Description
Memory	PIPE=0	Flow-Through Conversion Enabled – In this mode, the converter essentially behaves like a simple bus transceiver, such as an SN74LS245, except with an integrated floating-point format converter. When this mode is used, conversions take two cycles. Because of this, the converter automatically generates a wait state, which will halt the TMS320C30 for one cycle until the conversion is complete.
Pipeline	PIPE=1	Converter's Pipeline Registers Enabled Internally – This mode permits single-cycle conversion. As one data value is being converted, a previously converted value is output.

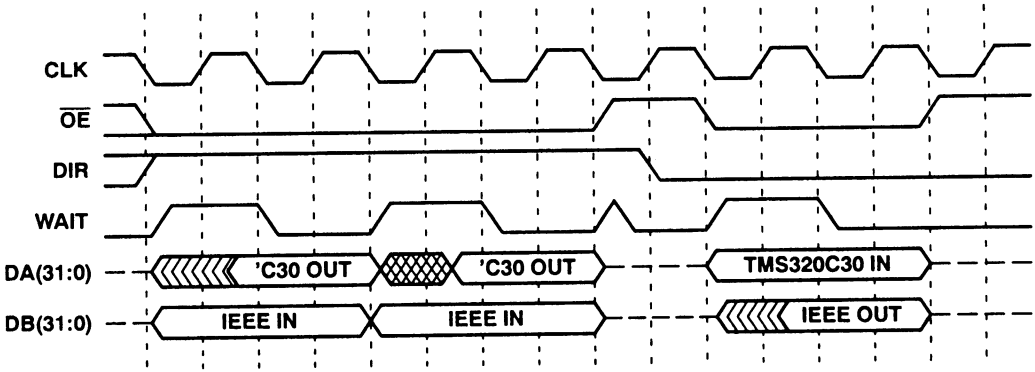
Memory Mode Operation

In this mode, one wait cycle is automatically generated during conversions from

- IEEE format to TMS320C30 format (reads)
- TMS320C30 format to IEEE format (writes)

The converter will not generate wait cycles of any other length and requires that the TMS320C30 H1 clock pin be tied to the converter's CLK pin. Figure 17 shows the timing diagram for this mode of operation.

Figure 17. Memory Mode Timing Diagram

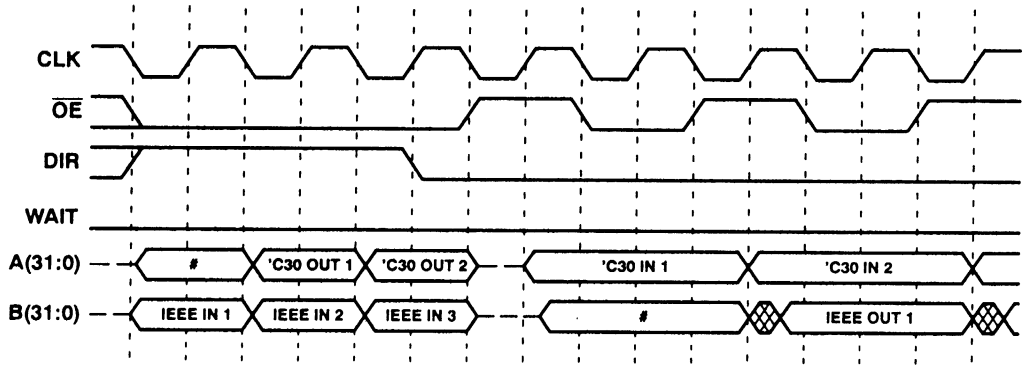


Pipelined Operation

Pipeline mode permits consecutive conversions every instruction cycle without wait cycles. However, because the pipeline has two internal stages, it takes two consecutive occurrences of the same operation (i.e., two reads or two writes) before it is filled. Therefore, the first read after a transition from a write will not provide properly converted data, and vice versa.

There is an address skew of one address when consecutive data values are converted. This should not be a major problem when blocks of memory are converted. The only added task will be to perform one extra transfer (read or write) to convert the last value remaining in the pipeline. With this exception, operation is identical to the Memory mode. Figure 18 shows a timing diagram for this mode of operation.

Figure 18. Pipeline Mode Timing Diagram



Interrupts

The converter automatically generates an interrupt whenever the conversion of an IEEE number classified as Not a Number (NaN) is attempted. The interrupt pulse is 1.5 H1 cycles wide. This is compatible with the TMS320C30 edge-triggered interrupt types. Table 9 shows this interrupt and its trigger. Note that the converter does not change the value of the NaN, but it does alter its bit positions. This assures that the sign bit of the IEEE number remains a sign bit in the TMS320C30 format. The same is true of the exponent field. The fractional field is left unchanged. If NaN is used to pass a code or command to the TMS320C30, interpretation of the code requires only the alteration of the comparison mask in software. For more information, refer to the previous subsection NaN (Not a Number).

Table 9. NaN Interrupt

Name	Function	Sources
NAN	Not a Number	IEEE CASE1: NaN

Software Application Examples

Simple Nonpipelined Conversion

If an external device (i.e., RAM, ROM, dual bus RAM, latch, etc.) contains a single-precision IEEE floating-point number and the corresponding TMS320C30 number is needed, the following TMS320C30 code will perform the required conversion:

```
EXTD    .word   0800000h        ; put address of external device here
*
        LDI     @EXTD,AR0       ; load AR0 w/address of external device
        LDF     *AR0,R0         ; R0=C30 formatted number
```

The following example performs TMS320C30-to-IEEE format conversion:

```
EXTD    .word   0800000h        ; put address of external device here
*
        LDI     @EXTD,AR0       ; load AR0 w/address of external device
        STF     R0,*AR0         ; location pointed to by AR0=IEEE formatted
*                               ;   number
```

Simple Pipelined Conversion

This example illustrates the overhead when the converter's pipeline mode is used. Since a single value will be converted, it is necessary to read the converter one extra time to flush the pipeline. Once again, assume that an external device (i.e., RAM, ROM, dual bus RAM, latch, etc.) contains a single-precision IEEE floating-point number, and the corresponding TMS320C30 number is needed.

```
EXTD    .word   0800000h        ; put address of external device here
*
        LDI     @EXTD,AR0       ; load AR0 w/address of external device
        LDF     *AR0,R0         ; ignore loaded value, 1st load queues
*                               ;   pipeline
        LDF     *AR0,R0         ; R0=C30 formatted number, address is
*                               ;   immaterial
```

The following example performs TMS320C30 to IEEE format conversion:

```
EXTD    .word   0800000h        ; put address of external device here
*
        LDI     @EXTD,AR0       ; load AR0 w/address of external device
        STF     R0,*AR0         ; value stored not correct until 2nd store
        STF     R0,*AR0         ; location pointed to by AR0=IEEE formatted
*                               ;   number
```

Pipelined Block Conversions

In the previous subsection, the pipeline was used, but not efficiently. This example shows a more typical application of pipeline mode. Again, external memory contains IEEE formatted data.

```
N       .set    03FFh           ; N = # of values to convert - 1
EXTD    .word   0800000h        ; put external address here
DADR    .word   0809800h        ; put destination address here
*
```

```
        LDI     @EXTD,AR0       ; load AR0 w/address of external device
        LDI     @DADR,AR1       ; load AR1 w/destination address
        LDF     *AR0++,R0       ; prime (preload) the converter's pipeline
        LDI     N,RC            ; block will be repeated N (0400h) times
        RPTB    RCR             ; specify end address of block repeat
        LDF     *AR0++,R0       ; read converted values into R0
RCR:    STF     R0,*AR1++       ; store converted values into on-chip
*                               ;   memory
```

This is more efficient:

```
N       .set    03FEh           ; N = # of values to convert - 2
EXTD    .word   0800000h        ; put external address here
DADR    .word   0809800h        ; put destination address here
*
        LDI     @EXTD,AR0       ; load AR0 w/address of external device
        LDI     @DADR,AR1       ; load AR1 w/destination address
        LDF     *AR0++,R0       ; prime (preload) the converter's pipeline
        LDF     *AR0++,R0       ; read 1st converted value for 1st STF
        RPTS    N               ; repeat next instruction N-1 (03FFh)
*                               ;   times, extra loop is to store last
*                               ;   value converted
        LDF     *AR0++,R0       ; read converted values into R0
||      STF     R0,*AR1++       ; store converted values into on-chip
*                               ;   memory, 1st store will save junk
```

The following example performs TMS320C30 to IEEE format conversion:

```
N       .set    0400h           ; N equals number of values to convert
EXTD    .word   0800000h        ; put external address here
SADR    .word   0809800h        ; put source data address here
*
        LDI     @EXTD,AR0       ; load AR0 w/address of external device
        LDI     @SADR,AR1       ; load AR1 w/source data address
        LDI     N,RC            ; block will be repeated N+1 (0401h) times,
*                               ;   extra loop is to store last value
*                               ;   converted
        RPTB    AC              ; specify end address of block repeat
        LDF     *AR1++,R0       ; read TMS320C30 format numbers into R0
AC:     STF     R0,*AR0++       ; store converted values into external
*                               ;   device
```

This is more efficient:

```
N       .set    03FFh           ; N equals number of values to convert - 1
EXTD    .word   0800000h        ; put external address here
SADR    .word   0809800h        ; put source data address here
*
        LDI     @EXTD,AR0       ; load AR0 w/address of external device
        LDI     @SADR,AR1       ; load AR1 w/source data address
        LDF     *AR0++,R0       ; read 1st converted value for 1st STF
        RPTS    N               ; repeat next instruction N (0400h) times,
*                               ;   extra loop is to store last value
*                               ;   converted
        LDF     *AR1++,R0       ; read converted values into R0
||      STF     R0,*AR0++       ; store converted values into external
*                               ;   device
        STF     R0,*AR0++       ; store last value
```

Using TMS320C30 External Flag 0 (XF0)

As mentioned in the section on converter operating modes, one of the TMS320C30's XF pins can be tied to the converter's output enable (OE) pin to enable the data path through the converter

or to bypass it, as the case may be. The following TMS320C30 code uses the TMS320C30 XF0 pin to do this (see Hardware Applications Examples section later in this report for the hardware configuration). Nonpipelined mode is assumed.

```
N       .set    03FFh           ; N equals number of values to convert - 1
EXTD    .word   0800000h        ; put external address here
SADR    .word   0809800h        ; put source data address here
*
        LDI     @EXTD,AR0       ; load AR0 w/address of external device
        LDI     @SADR,AR1       ; load AR1 w/source data address
        LDI     2,IOF           ; XF0=output=0, select the converter
        LDF     *AR0++,R0       ; read 1st converted value for 1st STF
        RPTS    N               ; repeat next instruction N+1 (0400h) times
        LDF     *AR1++,R0       ; read converted values into R0
||      STF     R0,*AR1++       ; store converted values into on-chip
*                               ;   memory, 1st store will save junk
        LDI     6,IOF           ; XF0=output=1, deselect the converter
```

Using the TMS320C30 DMA Capability

The built-in TMS320C30 DMA controller can be used to read converted IEEE values. The TMS320C30 assembly code to set up the DMA is shown below. Non-pipelined mode is assumed.

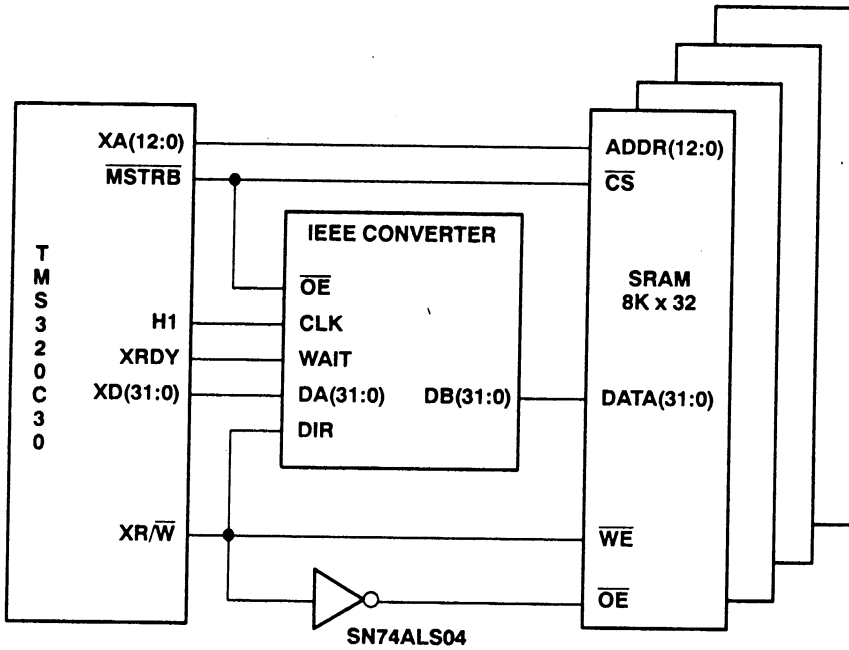
```
DMA     .word   0808000h        ; base address of DMA registers
GLBL    .word   0C53h           ; DMA global register init value
N       .set    0400h           ; N equals number of values to convert
EXTD    .word   0800000h        ; put external address here
DADR    .word   0809800h        ; put destination data address here
*
*    DMA controller setup
*
        LDI     @DMA,AR0        ; AR0 -> DMA control registers
        LDI     @EXTD,R0        ; R0 = address of IEEE data
        LDI     @DADR,R1        ; R1 = converted data destination address
        LDI     N,R2            ; R2 = DMA transfer count
        LDI     @GLBL,R3        ; R3 = DMA Global register initial value
        STI     R0,*+AR0(4)     ; DMA will transfer from external device
        STI     R1,*+AR0(6)     ; DMA will transfer to RAM block 0
        STI     R2,*+AR0(8)     ; DMA will transfer N values
        STI     R3,*AR0         ; start the DMA
```

Hardware Application Examples

IEEE Data Stored in TMS320C30 External MSTRB Memory

Below is shown an example of interfacing the converter to TMS320C30 external memory containing only IEEE formatted data. In this configuration, it is likely that the memory would be dual bus RAM to enable a second processor to share data with the TMS320C30 through this memory. Figure 19 shows an interface to a static RAM (SRAM) bank.

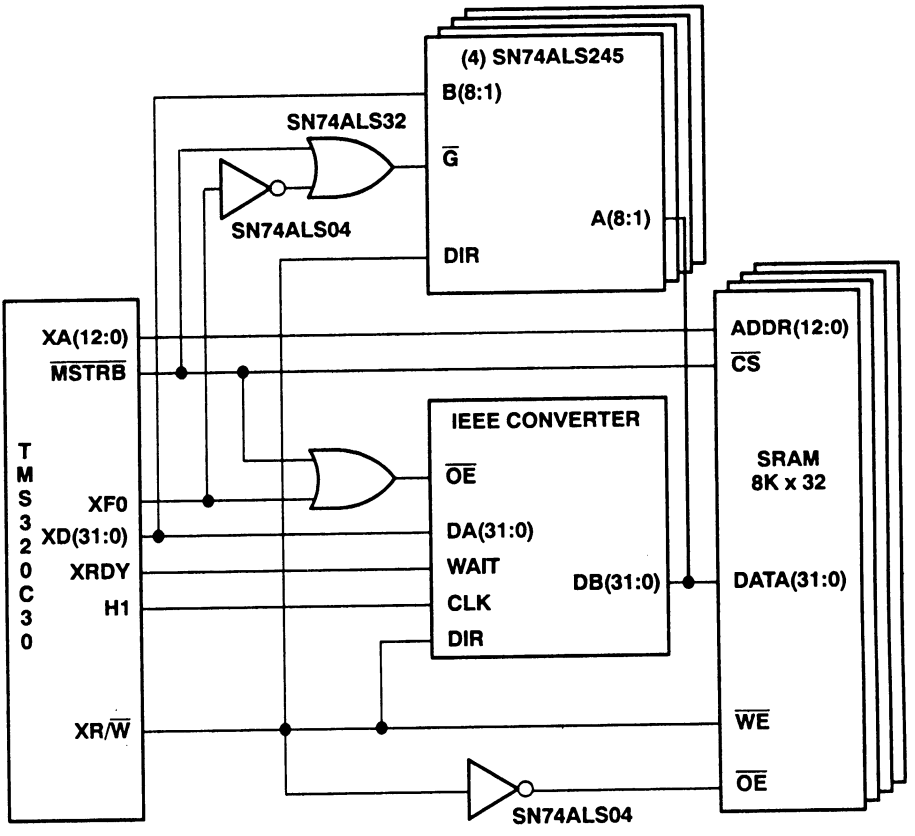
Figure 19. Interface to Static RAM



Bypassing the Converter

A previous subsection (Using TMS320C30 External Flag 0) showed TMS320C30 assembly code that used the TMS320C30 XF0 pin either to steer data through the converter or to bypass the converter for direct, or unconverted, access to that memory. Figure 20 shows a circuit that can be used with that code.

Figure 20. Steered Access to the Memory



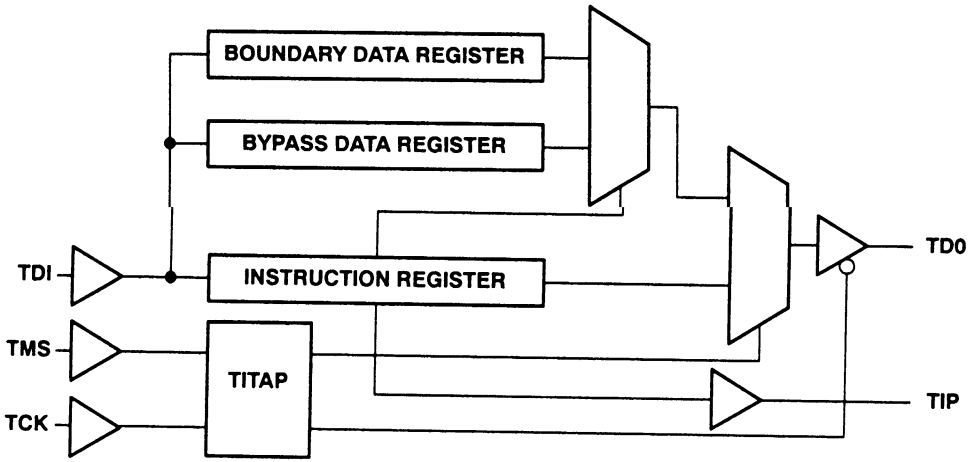
JTAG/IEEE-1149.1 Scan Interface

Integrated circuit and board-level testing is increasingly important. JTAG or IEEE-1149.1 is a standard test methodology. It is based on a 4-wire connection to a device and provides access to all I/O buffers (boundary scan) of a device. This permits stimulation and observation of internal logic. By allowing stimulation of output pins and observation of input pins, external circuitry can also be tested. If implemented completely, this can eliminate "bed of nails" test rigs.

The TMS320C30-IEEE Floating-Point Format Converter is equipped with a JTAG/ IEEE-1149.1 compatible scan interface. The internal architecture is based on Texas Instruments' SCOPEtm design specifications. This provides for boundary-scanning of the device and inclusion of an eight-bit instruction register.

Figure 21 shows the internal scan architecture and gives the naming conventions used to describe the device blocks:

Figure 21. Scan Architecture



I/O Pin Description

TCK

The TCK input clock signal is the scan clock. It typically will be generated off-board by a test controller. All tests of the device are controlled by an external controller and proceed at the scan clock (TCK) speed.

TMS

The TMS input signal is clocked in by TCK. TMS controls the test mode of the device. Using TMS and TCK, a test controller can scan registers through the device, perform tests, or place the device in a normal functional mode.

TDI

The TDI input signal is used to input serial data through the registers in the device. All data is clocked in by TCK and shifts according to the state of the test logic set up by an external test controller using TMS and TCK.

TDO

The TDO output signal is used to scan serial test data out of the device under the control of the test host. While shifting data, TDO is active-shifting data out on the falling edge of TCK. When through shifting data, TDO is tri-stated.

TIP

TIP is an output indicating good or bad parity in the instruction register. The indication defaults to good if the external controller does not check for parity. To check parity, the test controller places the device in the instruction register pause state. While in this state, the device will output the actual (i.e., hardware-determined) parity of the device's instruction register. A high logic level indicates good parity, while a low logic level indicates bad parity.

Architectural Elements

TITAP

The Texas Instruments' Test Access Port (TITAP) is a 16-state state-machine designed according to the JTAG and IEEE-1149.1 specifications. The TITAP controls the test logic and is controlled by the TMS and TCK inputs to the device from an external test host controller.

Instruction Register

The Instruction Register is eight bits in length. Table 10 lists the instructions available for this device.

Table 10. Test Instructions

msb –> lsb	Instruction
00000000	Boundary Scan
10000001	ID Register Scan
10000010	Sample Boundary Scan
00000011	Boundary Scan
00000110	Control Boundary HI-Z
10000111	Control Boundary 1/0
00001010	Read Boundary-Normal
10001011	Read Boundary-Test
00001100	Boundary Selftest
11111111	Bypass Scan
All Others	Bypass Scan

The Instruction Register is preloaded with 00000001 (msb–lsb) in the instruction register capture state of the TITAP. This is not per the JTAG/IEEE–1148.1 standards.

Boundary Scan Instruction

This instruction places the device in test mode: all function inputs and outputs are controlled by the test logic. Function inputs and outputs are sampled in the data register capture state of the TITAP, and the boundary data register is selected in the data register scan path during data register scans.

ID Register Scan Instruction

This instruction places the device in normal mode: all function inputs and outputs operate in their normal modes. The bypass data register is selected in the data register scan path during data register scans.

Sample Boundary Scan Instruction

This instruction places the device in normal mode: all function inputs and outputs operate in their normal modes. Function inputs and outputs are sampled in the data register capture state of the TITAP, and the boundary data register is selected in the data register scan path during data register scans.

Control Boundary HI-Z Instruction

This instruction places the device in test mode: all function outputs are tri-stated (if possible), while all function inputs operate in their normal mode. The bypass data register is selected in the data register scan path during data register scans.

Control Boundary 1/0 Instruction

This instruction places the device in test mode: all function inputs and outputs are controlled by the test logic. The bypass data register is selected in the data register scan path during data register scans.

Read Boundary – Normal Instruction

This instruction places the device in normal mode: all function inputs and outputs operate in their normal modes. The boundary data register retains its current state in the data register capture state of the TITAP, and the boundary data register is selected in the data register scan path during data register scans.

Read Boundary – Test Instruction

This instruction places the device in test mode: all function inputs and outputs are controlled by the test logic. The boundary data register retains its current state in the data register capture state of the TITAP, and the boundary data register is selected in the data register scan path during data register scans.

Boundary Self–Test Instruction

This instruction places the device in normal mode: all function inputs and outputs operate in their normal modes. The boundary data register contents are toggled, and the data register captures the state of the TITAP. Also, the boundary data register is selected in the data register scan path during data register scans.

Bypass Scan Instruction

This instruction places the device in normal mode: all function inputs and outputs operate in their normal modes. The bypass data register is selected in the data register scan path during data register scans.

Boundary Data Register

The boundary data register contains 70 bits and is ordered according to Figure 22.

Figure 22. Scan Path Bit Order

TDI —> DIR —> PIPE —> CLK —> OEZ —> NAN —> WAIT —>
DA31 —> DA30 —> ... —> DA1 —> DA0 —>
DB31 —> DB30 —> ... —> DB1 —> DB0 ————————> TDO

Bypass Data Register

The Bypass Data Register is one bit in length and is operated in accordance with the JTAG/ IEEE-1149.1 specifications.

Scan References

Refer to the following documents for further descriptions of the test logic of this device:

1) A Test Access Port and Boundary Scan Architecture; Technical Sub-Committee of the Joint Test Action Group (JTAG).
2) IEEE Standard 1149.1 – IEEE Standard Test Access Port and Boundary-Scan Architecture.