### MAX1499 EV Kit Component List (continued)

REFERENCE	QTY	DESCRIPTION
JU10–JU14	5	3 pins
JU1–JU9	9	2 pins
R1	1	133k $\Omega$ 1% resistor (1206)
R2, R12	2	100k $\Omega$ 1% resistors (1206)
R3-R7	5	1k $\Omega$ 5% resistors (1206)
R8, R9	0	Do not install—shorted trace on PC board (1206)
R10	1	500k $\Omega$ potentiometer
R11	1	24k $\Omega$ 5% resistor (1206)
R13, R14	2	10 $\Omega$ 5% resistors (1206)
TB1	1	0.200in two-circuit screw terminal block
TP1-TP4	4	8 pins
U1	1	MAX1499ECJ
U2	1	MAX1659ESA
U3, U4	2	MAX1840EUB or MAX1841EUB
U5	1	MAX6062AEUR-T, FZFY
	1	PC board, MAX1499 EV kit
	13	Shunts

### \_Quick Start

### **Required Equipment**

Before you begin, you need the following equipment:

- MAX1499EVC16 (contains MAX1499EVKIT board and 68HC16MODULE-DIP)
- DC power supply, +7VDC to +20VDC at 0.5A
- Windows 95/98/2000/XP computer with an available serial (COM) port
- 9-pin I/O extension cable

#### Procedure

## Do not turn on the power until all connections are made.

- 1) Ensure that JU1–JU8 and JU10–JU14 have shunts installed, and that JU9 is open. See the jumper settings in Table 2.
- Carefully connect the boards by aligning the 40-pin header of the MAX1499 EV kit with the 40-pin connector of the 68HC16MODULE-DIP module. Gently press them together. The two boards should be flush against one another.
- 3) Connect a +7VDC to +20VDC power source to the  $\mu$ C module at the terminal block located next to the on/off switch, along the top edge of the  $\mu$ C module. Observe the polarity marked on the board.
- 4) Connect a cable from the computer's serial port to the μC module. If using a 9-pin serial port, use a straight-through, 9-pin female-to-male cable. If the only available serial port uses a 25-pin connector, a standard 25-pin to 9-pin adapter is required. The EV kit software checks the modem status lines (CTS, DSR, and DCD) to confirm that the correct port has been selected.
- 5) Install the evaluation software on your computer by running the INSTALL.EXE program on the disk. The program files are copied and icons are created for them in the Windows Start menu.
- 6) Turn on the power supply.
- 7) Start the MAX1499 program by opening its icon in the Start menu.
- 8) The program will prompt you to connect the μC module and to turn its power on. Slide SW1 to the ON position. Select the correct serial port, and click OK. The program automatically downloads its software to the module.

### Component Suppliers

///XI/M

SUPPLIER	PHONE	FAX	WEBSITE
Kingbright Corporation	909-468-0500 (ext 126)	909-468-0505	www.kingbright.com
Taiyo Yuden	800-348-2496	847-925-0899	www.t-yuden.com
TDK	847-803-6100	847-390-4405	www.component.tdk.com

Note: Indicate you are using the MAX1499 when contacting these component suppliers.



- Apply an input signal in the range -2V to +2V between AIN+ and AIN-. Observe the readout on the screen.
- 10) To view a graph of the measurements, pull down the **View** menu and click **Graph**.

### Detailed Description of Software

#### Measurement

The **Measurement** tab of the evaluation software mimics the behavior of a digital voltmeter (DVM). The status bits are polled approximately once per second. Whenever the **Data** status bit is one, the ADC result register is read and displayed as **Analog Input Code**. The MAX1499 also displays the result on its own LED display.

The EV kit is not a complete DVM. Additional input scaling and protection circuitry might be required.

Whenever the **Measurement** tab is activated, the software offers to clear the **spi/adc** and **seg\_sel** control bits to zero if they are not already clear.

#### Math Processing

The evaluation software implements several math functions found in physical systems. Whenever the **Math** tab is activated, the software offers to set the **spi/adc** control bit to one if it is not already set. The software also offers to clear the **seg\_sel** control bit to zero if it is not already clear.

The evaluation software intercepts the ADC result prior to display, calculating a new LED display value whenever the **Measurement** or **Math** tab is active and the **spi/adc** control bit is set to one. Math results are graphed as channel one data, alongside the raw ADC result as channel zero data.

The **Type K Thermocouple** function can be used along with a suitable cold junction connection to convert a type K thermocouple's measured Seebeck voltage into temperature in degrees centigrade. The **a0** coefficient 230 represents a cold junction temperature of +23°C.

#### **Control Register**

The **Control Register** tab provides access to all control register bits. Drop down the appropriate combo-box and then click **write**.

#### Limit Registers, ADC Offset, ADC Result, LED Display, and Peak

The **Results, Displays, Limits** tab provides access to the two's-complement data registers. Each register has a **read** button and a **write** button, except for **ADC** 

**RESULT1, ADC RESULT2**, and **PEAK RESULT**, which are read only.

Reading the **ADC RESULT1** or **ADC RESULT2** register automatically updates the LED display, regardless of the **seg\_sel** control register setting.

Writing to the ADC OFFSET register affects **ADC RESULT1** and **ADC RESULT2**, regardless of the **off-set\_cal1** control register setting.

#### **LED Segment Registers**

The **LED Segments** tab lets the user turn individual LED segments on and off by clicking them with the mouse.

Whenever the **LED Segments** tab is activated, the software offers to set the **seg\_sel** control bit to one if it is not already set.

The **Write LED Text** button translates a text string into approximate seven-segment characters, and then writes the character patterns to the LED display.

#### Graph

The evaluation software has two options for graphing data. A graph of recent data can be displayed by selecting the **View** menu and then **Graph**. Data can be viewed as a time sequence plot, a histogram plot, or as a table of raw numbers. To control the size and timing of the data runs, activate the sampling tool by clicking the main window's **Collect Samples** button.

Sampled data can be saved to a file in comma-delimited or tab-delimited format. Line numbers and a descriptive header line are optional.

Channel zero plots raw 16-bit ADC result data. Channel one plots LED display data, if math processing is enabled. If extended resolution is enabled, channel two plots raw 20-bit ADC result data.

#### **Diagnostics Window**

The diagnostics window is used for factory testing prior to shipping the evaluation kit. It is not intended for customer use.

### Detailed Description of Hardware

The MAX1499 device under test (U1) is a low-power, 4.5-digit ADC with integrated LED display drivers. The MAX6062 (U5) provides on-board 2.048V reference voltage. See the MAX1499 EV kit schematic in Figure 7 and refer to the MAX1499 data sheet.

The EV kit includes a MAX1659 high-current 5V linear regulator (U2) and a set of MAX1840/MAX1841 level shifters (U3 and U4) to support the use of 3V logic.

### Table 1. Graph Tool Buttons

TOOL	FUNCTION
<b>A</b>	Show the entire available input range.
<b>+</b>	Expand the graph data to fill the window.
<b>+ +</b>	Move the view left or right.
<b>++</b>	Move the view up or down.
↔++	Expand or contract the x-axis.
<b>‡</b> I	Expand or contract the y-axis.
7	Load data from a file.
<b>P</b>	Save data to a file.
Ln <u>de</u> 2. 22 2. 22	Option to write a header line when saving data.
2××× 3××× 4×××	Option to write line numbers when saving data.
$\sim$	View code vs. time plot.
ահ	View histogram plot (cumulative frequency of each code).
<u>××××</u> ×××× ××××	View table.
Min	Show minimum in tabular view.
Max	Show maximum in tabular view.
Span	Show span in tabular view. Span = maximum - minimum.
N	Show number of samples in tabular view.

TOOL	FUNCTION
Sum(x)	Show sum of the samples in tabular view.
Sum(x*x)	Show sum of the squares of the samples in tabular view.
Mean	Show arithmetic mean in tabular view: Mean = $\frac{\Sigma(x)}{n}$
StdDev	Show standard deviation in tabular view: Standard deviation = $\sqrt{\frac{n\Sigma(x^2) - (\Sigma \times )^2}{(n-1)n}}$
Rms	Show root of the mean of the squares (RMS) in tabular view: RMS = $\sqrt{\frac{\Sigma(x^2)}{n}}$
0	Channel 0 enable (16-bit ADC result).
1	Channel 1 enable (math result).
2	Channel 2 enable (20-bit ADC result).

### **Table 2. Jumper Functions Table**

JUMPER	SHUNT POSITION	FUNCTION
	Closed*	LED displays are powered by U2.
JU1	Open	VLED must be supplied by an external power source.
1112	Closed*	VDISP = GND.
JU2	Open	Apply VDISP voltage at VDISP pad.
1112	Closed*	Banana jack AIN+ connects to AIN+ input pin.
103	Open	Insert custom filtering between JU3 pins 1 and 2.
	Closed*	Banana jack AIN- connects to AIN- input pin.
JU4	Open	Insert custom filtering between JU4 pins 1 and 2.
ILIE	Closed*	REF- = GND.
105	Open	REF- must be provided by user.
1116	Closed*	REF + = 2.048V from U5, MAX6062.
200	Open	REF+ must be provided by user.
11.17	Closed*	REF+ is bypassed by C6.
507	Open	C6 is disconnected.
	Closed*	GLED return current flows to $DV_{DD}$ .
3UC	Open	GLED return current flows to external power source.
	Closed	LED_EN = low; LED displays are blanked.
JUA	Open*	LED_EN = high; LED displays are enabled.
JU10-	1-2*	Display color = red.
JU14	2-3	Display color = green.

\*Asterisk indicates default configuration.

### Table 3. Stand-Alone Interface Pin Functions

U1 PIN	MAX1499 FUNCTION	MAX1498 FUNCTION
8	CLK	INTREF
9	EOC	RANGE
10	CS	DPSET1
11	DIN	DPSET2
12	SCLK	PEAK
13	DOUT	HOLD
30	LOWBATT	DPON

#### **Evaluating the MAX1498**

The MAX1499EVKIT supports stand-alone operation of the MAX1498; however, the evaluation software cannot be used because there is no  $\mu$ C interface.

The MAX1498 is the standalone version of the MAX1499. Refer to MAX1498/MAX1499 data sheet. Request a free sample of MAX1498ECJ.

- 1) The MAX1499EVKIT must be disconnected from the 68HC16MODULE.
- 2) With power disconnected, replace U1 with the MAX1498.
- 3) Connect DC power supply at terminal block TB1.
- 4) Turn on the power supply. The LED display should begin indicating measurement data.

After replacing U1 with the MAX1498, some of the pin functions are different. See Table 3.

### Example Code

Listings 1 shows the variable declarations needed in the EV kit software. Listing 2 contains the functions used in the EV kit software.

MAX1499 Evaluation Soft File Options View Help	ware
Measurement Math Control F Basic Analog Input M NOTE: Evaluation kit is no Analog Input Code 1 Hold mode 1	Register       Results, Display, Limits       LED Segments         Reasurement (software calculates display value if spi/adc=1)         ot a DVM. Requires additional input protection circuitry and resistor dividers.         Reference Input         REF voltage:       2.048         Use Internal 2.048V reference
<ul> <li>☐ Peak Detect mode</li> <li>AIN Range</li> <li>○ 200 mV</li> <li>○ 2 V</li> <li>○ 18888</li> </ul>	Collect Samples
Command DIN 1110 1000 DOUT	Low-Level Data reading 1111 1111 0011 0000 ADC_RESULT2

Figure 1. MAX1499 Evaluation Software—Measurement Tab

Measurement       Math       Control Register       Results, Display, Limits       LED Segments         Software calculates display value when measurement or math tab is active and spi/adc=1.         ADC Result:       12696.375       LED Display:       1100         AIN Range       Linear       Quadratic       Cubic       Exponential       Logarithmic       Type K Thermocouple         O 200 mV       Linear       Quadratic       Cubic       Exponential       Logarithmic       Type K Thermocouple         Decimal Point       LED = a0 + a1 * log (ADC + a2) assuming ADC > a2, otherwise -19999       log(10)=2.303; 1/log(10)=0.434       Reset to a0=-3000, a1=434, a2=0         O - 18888       O - 3000       a1       434       a2       O       a3         O - 18888       O - 3000       a1       434       a2       O       a3	MAX1499 Ev File Options	aluation Software
Software calculates display value when measurement or math tab is active and spi/adc=1.         ADC Result       12696.375       LED Display:       1100         AIN Range       Linear       Quadratic       Cubic       Exponential       Logarithmic       Type K Thermocouple         O 200 mV       LED = a0 + a1 * log (ADC + a2) assuming ADC > a2, otherwise ·19999       log(10)=2.303; 1/log(10)=0.434       Reset to a0=-3000, a1=434, a2=0         O -18888       0 -18888       a0 -3000       a1       434       a2       a3         Low-Level Data	Measurement Ma	ath Control Register Results, Display, Limits LED Segments
ADC Result:       12696.375 LED Display:       1100         AIN Range       Quadratic       Cubic       Exponential       Logarithmic       Type K Thermocouple         0 200 mV       QU       LED = a0 + a1 * log (ADC + a2) assuming ADC > a2, otherwise -19999       LeD = a0 + a1 * log (ADC + a2) assuming ADC > a2, otherwise -19999       Log(10)=2.303; 1/log(10)=0.434       Reset to a0=-3000, a1=434, a2=0         0 -18888       0 -18888       0 -3000       a1       434       a2       0       a3         0 -18888       0 -3000       a1       434       a2       0       a3       0         0 -18888       0 -18888       0 -3000       a1       434       a2       0       a3       0         Low-Level Data	Software cal	culates display value when measurement or math tab is active and spi/adc=1.
AIN Range       Linear       Quadratic       Cubic       Exponential       Logarithmic       Type K Thermocouple         0       200 mV       LED = a0 + a1 * log (ADC + a2) assuming ADC > a2, otherwise ·19999       log(10)=2.303; 1/log(10)=0.434       Reset to a0=-3000, a1=434, a2=0         0       -18888       Coefficients:       a1       434       a2       a3         0       -18888       a0       -3000       a1       434       a2       a3         0       -18888       a0       -3000       a1       434       a2       a3       a3	ADC Result:	12696.375 LED Display: 1100
O 200 mV       LED = a0 + a1 * log (ADC + a2) assuming ADC > a2, otherwise -19999         Decimal Point       log(10)=2.303; 1/log(10)=0.434       Reset to a0=-3000, a1=434, a2=0         O -18888       O -18888       O -18888         O -18888       a0       -3000       a1	AIN Range	Linear Quadratic Cubic Exponential Logarithmic Type K Thermocouple
Decimal Point       log(10)=2.303; 1/log(10)=0.434       Reset to a0=-3000, a1=434, a2=0         O -18888       Coefficients:       a1         O -18888       a0       -3000       a1         O -1.8888       a0       -3000       a1	O 200 mV ⊙ 2V	LED = a0 + a1 * log (ADC + a2) assuming ADC > a2, otherwise -19999
○     -18888       ○     -18888       ○     -18888       ○     -18888       ○     -18888       ○     -18888       ○     -18888       ○     -18888       ○     -18888       ○     -18888       □     -3000       a1     434       a2     □       a3     □		log(10)=2.303; 1/log(10)=0.434 Report to p0= 2000, p1=424, p2=0
Image: Coefficients:       Coefficients:         Image: O -18888       a0         Image: O -18888       a1         Image: O -18888       a1         Image: O -18888       a2         Image: O -18888       a3         Image: O -18888       a4         Image: O -18888       a5         Image: O -18888       a5         Image: O -18888       a5         Image: O -18888       a5         Image: O -18888	O -18888	
O -188.88       a0       -3000       a1       434       a2       0       a3       0         O -1.8888       a0       -3000       a1       434       a2       0       a3       0         O -1.8888       a0       -3000       a1       434       a2       0       a3       0         Low-Level Data	⊙ -1888.8	Coefficients:
C -1.8888	O -188.88	a0 -3000 a1 434 a2 0 a3 0
Low-Level Data	O -1.8888	
		Low-Level Data
Lommand LEU_UATA	Comma DIN 1001	and LED_DATA
DOUT #### #### ####	DOUT	#### #### ####
don't care		don't care

Figure 2. MAX1499 Evaluation Software—Math Tab

Evaluate: MAX1498/MAX1499

Evaluate: MAX1498/MAX1499

PPI MAX14	499 Evaluation S	Software			_ 🗆 X
File Op	tions Vie <del>w</del> H	lelp			
Measurem	ent Math Cont	rol Register Results, Disp	lay, Limits	s LED Segme	ents
	Enable	e advanced features u	sing the	control regis	ster directly
Read	d spi/adc	1 display LED Data Reg	-	hold	0 normal operation 💽
	extclk	0 clock = internal clock	-	peak	0 normal operation 📃 💌
Write	e intref	0 external reference	-	range	0 input range 2 V 🛛 💌
	dpon	1 decimal point on	-	clr	0 normal operation 📃 💌
	dpset2	0 DP1/DP3	-	seg_sel	0 normal operation 📃 💌
	dpset1	0 DP1/DP2	-	/offset_cal1	0 offset determined by auto 💌
	pd_dig	0 digital power on	-	offset_cal2	0 normal operation 📃 💌
	pd_ana	0 analog power on	-	enable	0 LED drivers disabled 🛛 💌
Reset	serial interface by c	locking in 32 one's			
		Low-Lo	evel Dat	ta	
DIN DOUT	Command 1100 0010	reading 1111 1111 1001 0000 CONTROL	1111 0000	1111 0000	

Figure 3. MAX1499 Evaluation Software—Control Register Tab

File Ontions	View Help		
Measurement N	1ath Control Regist	er Results, Display, Limits LED	Segments
Over/	Underrange thresh	olds, ADC offset, conversion	result, display, peak value
Read	OVERRANGE	9999	Write
Read	UNDERRANGE	20000	Write
Read	ADC OFFSET	)	Write
Read	ADC RESULT 1	2696.813	
Read	LED DATA	100	Write
Read	PEAK RESULT	)	
Read	ADC RESULT 2	208	
		Low-Level Data	
Com DIN 111 DOUT	mand read 0 1000 11 11 ADI	ling 11 1111 01 0000 C_RESULT2	

Figure 4. MAX1499 Evaluation Software—Results, Display, Limits Tab

Evaluate: MAX1498/MAX1499

MAX1499 Evaluation	Software
File Options View	Help
Measurement Math Co	Introl Register Results, Display, Limits LED Segments
	Drive LED segments directly
	RSS
Read all	Read LED Segment Reg 1 LED Seg 1 register = 0000 1111 0000 0111
All Segments On	Read LED Segment Reg 2 LED Seg 2 register = 0011 0000 1001 1101
All Segments Off	Read LED Segment Reg 3 LED Seg 3 register = 1110 1001
PASS	Write LED Text
	Low-Level Data
Command DIN 1000 1100 DOUT	LED_SEG_3 1110 1001 0000 0000 don't care

Figure 5. MAX1499 Evaluation Software—LED Segments Tab



Figure 6. MAX1499 Evaluation Software—Graph

Evaluate: MAX1498/MAX1499



Evaluate: MAX1498/MAX1499



12



Figure 7b. MAX1499 EV Kit Schematic (Sheet 2 of 2)



Figure 8. MAX1499 EV Kit Component Placement Guide—Component Side

Evaluate: MAX1498/MAX1499

M/IXI/M



Figure 9. MAX1499 EV Kit PC Board Layout—Component Side







M/IXI/M

<pre>// Revision I // 01/07/200 // 12/04/200 // 09/15/200 // 09/15/200 // 09/15/200 // 09/13/200 // 08/13/200 // 08/13/200 // #ifndef #define //</pre>	history: 14: modify drv 1 4 9 4 driver to become dr 15: fix indentation 13: add double Voltage(void) 13: add SPI_Transfer_After_EOC() 13: add SPI_Transfer_After_EOC() 13: add class MAX1499 dependent on extr 13: preliminary draft of reusable code drv1499H drv1499H	v1499 ernal SPI_Interface()
<pre>// 12/04/200 // 09/15/200 // 09/12/200 // 09/09/200 // 09/09/200 // 08/13/200 // #ifndef #define //</pre>	<ul> <li>G: fix indentation</li> <li>G: add double Voltage(void)</li> <li>add SPI_Transfer_After_EOC()</li> <li>add class MAX1499 dependent on extra strain and the strain</li></ul>	ernal SPI_Interface()
// 09/13/200 // 09/02/200 // 08/13/200 // #ifndef #define //	3: add SBI_Transfer_After_EOC() 3: add Class MAX1499 dependent on extr 3: preliminary draft of reusable code drv1499H drv1499H	ernal SPI_Interface()
<pre>// 09/09/200 // 08/13/200 // #ifndef #define // //</pre>	33: add class MAX1499 dependent on ext 33: preliminary draft of reusable code drv1499H drv1499H drv1499H	ernal SPI_Interface()
// #ifndef #define //	drv1499H drv1499H	
#define //	drv1499H	
//		
//		
// The follo	wing interface protocols must be provided	by
// the appro	priate low-level interface code.	03
//		
/* SPI inter	face:	
** mos	si[] = array of master-out, slave-in data by	tes
** mis	o_buf[] = receive buffer for master-in, sla	ve-out data bytes
extern h	ool SPI Transfer(int byte c	count,
cons	st unsignedint8 mosi[], u	<pre>insignedint8 miso_buf[]</pre>
/* SPI inter	face with data transfer immediately after	FOC is asserted:
** byte	e_count = transfer length	Loe is asserted.
** mos	si[] = array of master-out, slave-in data by  buffl = reasive buffer for mester in slave-	tes
*/	o_bur[] = receive burrer for master-m, sia	ive-out data bytes
extern h	ool SPI Transfer After EOC(	int byte count,
cons	st unsignedint8 mosi[], u	<pre>insignedint8 miso_buf[]</pre>
11		
// Define the	e bits in the COMMS register.	
// START R	/W RS4 RS3 RS2 RS1 RŠ0 0	
#define	MAX1499_COMMS_START	0x80
#define	MAX1499_COMMS_RW_MASK	0×40
#define	MAX1499_COMMS_RW_WRITE	0×40
#define	MAX1400 COMMS DS MASK	0.0-
	MAAI499 COMMS KS MASK	UXJE
#define	MAX1499 COMMS RS 00000	0x3E 0x00
#define #define	MAX1499_COMMS_KS_MASK MAX1499_COMMS_RS_00000 MAX1499_COMMS_RS_STATUS	0x3E 0x00 0x00
#define #define #define	MAX1499_COMMS_RS_0000 MAX1499_COMMS_RS_STATUS MAX1499_COMMS_RS_STATUS	0x00 0x00 0x00 0x02
<pre>#define #define #define #define</pre>	MAX1499_COMMS_RS_00000 MAX1499_COMMS_RS_STATUS MAX1499_COMMS_RS_STATUS MAX1499_COMMS_RS_00001 MAX1499_COMMS_RS_CONTROL	0x00 0x00 0x02 0x02
<pre>#define #define #define #define #define</pre>	MAX1499_COMMS_RS_00000 MAX1499_COMMS_RS_STATUS MAX1499_COMMS_RS_STATUS MAX1499_COMMS_RS_00001 MAX1499_COMMS_RS_CONTROL MAX1499_COMMS_RS_00010	0x3E 0x00 0x00 0x02 0x02 0x02 0x04
<pre>#define #define #define #define #define #define</pre>	MAX1499_COMMS_RS_DADK MAX1499_COMMS_RS_STATUS MAX1499_COMMS_RS_STATUS MAX1499_COMMS_RS_00001 MAX1499_COMMS_RS_00010 MAX1499_COMMS_RS_00010 MAX1499_COMMS_RS_0VERRANGE	0x3E 0x00 0x02 0x02 0x02 0x04 0x04 0x04
<pre>#define #define #define #define #define #define #define #define</pre>	MAX1499_COMMS_RS_D0000 MAX1499_COMMS_RS_STATUS MAX1499_COMMS_RS_STATUS MAX1499_COMMS_RS_00001 MAX1499_COMMS_RS_00010 MAX1499_COMMS_RS_0VERRANGE MAX1499_COMMS_RS_00011 MAX1499_COMMS_RS_00011	0x3E 0x00 0x00 0x02 0x02 0x04 0x04 0x04 0x06
<pre>#define #define #define #define #define #define #define #define #define</pre>	MAX1499_COMMS_RS_D0000 MAX1499_COMMS_RS_STATUS MAX1499_COMMS_RS_00001 MAX1499_COMMS_RS_00001 MAX1499_COMMS_RS_00010 MAX1499_COMMS_RS_00010 MAX1499_COMMS_RS_00011 MAX1499_COMMS_RS_00011 MAX1499_COMMS_RS_00011 MAX1499_COMMS_RS_00011 MAX1499_COMMS_RS_00011 MAX1499_COMMS_RS_00011 MAX1499_COMMS_RS_00011 MAX1499_COMMS_RS_00011	0x3E 0x00 0x02 0x02 0x04 0x04 0x04 0x06 0x06 0x06
<pre>#define #define #define #define #define #define #define #define #define</pre>	MAX1499_COMMS_RS_D0000 MAX1499_COMMS_RS_D0000 MAX1499_COMMS_RS_D0001 MAX1499_COMMS_RS_00001 MAX1499_COMMS_RS_00010 MAX1499_COMMS_RS_00010 MAX1499_COMMS_RS_00011 MAX1499_COMMS_RS_UNDERRANGE MAX1499_COMMS_RS_UNDERRANGE MAX1499_COMMS_RS_0100 MAX1490_COMMS_RS_0100	0x3E 0x00 0x02 0x02 0x04 0x04 0x06 0x06 0x06 0x08 0x08
<pre>#define #define #define</pre>	MAX1499_COMMS_RS_D0000 MAX1499_COMMS_RS_D0000 MAX1499_COMMS_RS_D0001 MAX1499_COMMS_RS_00001 MAX1499_COMMS_RS_00010 MAX1499_COMMS_RS_00010 MAX1499_COMMS_RS_0011 MAX1499_COMMS_RS_UNDERRANGE MAX1499_COMMS_RS_UNDERRANGE MAX1499_COMMS_RS_LED_SEG_1 MAX1499_COMMS_RS_00101	0x3E 0x00 0x02 0x02 0x04 0x04 0x06 0x06 0x06 0x08 0x08 0x08
<pre>#define #define #define</pre>	MAX1499_COMMS_RS_DABR MAX1499_COMMS_RS_STATUS MAX1499_COMMS_RS_STATUS MAX1499_COMMS_RS_OONTROL MAX1499_COMMS_RS_OONTROL MAX1499_COMMS_RS_OVERRANGE MAX1499_COMMS_RS_OVERRANGE MAX1499_COMMS_RS_OO101 MAX1499_COMMS_RS_LED_SEG_1 MAX1499_COMMS_RS_LED_SEG_1 MAX1499_COMMS_RS_LED_SEG_2	UX3E 0x00 0x00 0x02 0x02 0x04 0x04 0x04 0x06 0x06 0x08 0x08 0x08 0x08 0x0A
<pre>#define #define #define</pre>	MAX1499_COMMS_RS_DABK MAX1499_COMMS_RS_STATUS MAX1499_COMMS_RS_STATUS MAX1499_COMMS_RS_ONTROL MAX1499_COMMS_RS_ONTROL MAX1499_COMMS_RS_OVERRANGE MAX1499_COMMS_RS_OVERRANGE MAX1499_COMMS_RS_UDDERRANGE MAX1499_COMMS_RS_LED_SEG_1 MAX1499_COMMS_RS_LED_SEG_1 MAX1499_COMMS_RS_LED_SEG_2 MAX1499_COMMS_RS_LED_SEG_2 MAX1499_COMMS_RS_00110	0x3E 0x00 0x02 0x02 0x04 0x04 0x04 0x06 0x06 0x08 0x08 0x08 0x08 0x08 0x0A 0x0A
<pre>#define #define #</pre>	MAX1499_COMMS_RS_D0000 MAX1499_COMMS_RS_STATUS MAX1499_COMMS_RS_STATUS MAX1499_COMMS_RS_O0001 MAX1499_COMMS_RS_O0010 MAX1499_COMMS_RS_OVERRANGE MAX1499_COMMS_RS_00011 MAX1499_COMMS_RS_00101 MAX1499_COMMS_RS_00101 MAX1499_COMMS_RS_LED_SEG_1 MAX1499_COMMS_RS_LED_SEG_2 MAX1499_COMMS_RS_LED_SEG_2 MAX1499_COMMS_RS_00110 MAX1499_COMMS_RS_0010	0x3E 0x00 0x02 0x02 0x04 0x04 0x04 0x06 0x06 0x08 0x08 0x08 0x08 0x08 0x00 0x00 0x02
<pre>#define #define #</pre>	MAX1499_COMMS_RS_D0000 MAX1499_COMMS_RS_STATUS MAX1499_COMMS_RS_STATUS MAX1499_COMMS_RS_CONTROL MAX1499_COMMS_RS_CONTROL MAX1499_COMMS_RS_00010 MAX1499_COMMS_RS_0011 MAX1499_COMMS_RS_00101 MAX1499_COMMS_RS_00101 MAX1499_COMMS_RS_LED_SEG_1 MAX1499_COMMS_RS_LED_SEG_2 MAX1499_COMMS_RS_LED_SEG_2 MAX1499_COMMS_RS_LED_SEG_3 MAX1499_COMMS_RS_00111	0x3E 0x00 0x02 0x02 0x04 0x04 0x06 0x06 0x06 0x08 0x08 0x08 0x08 0x0A 0x0A 0x0A 0x0C 0x0C 0x0E
<pre>#define #define #</pre>	MAX1499_COMMS_RS_0000 MAX1499_COMMS_RS_0000 MAX1499_COMMS_RS_STATUS MAX1499_COMMS_RS_STATUS MAX1499_COMMS_RS_00010 MAX1499_COMMS_RS_00010 MAX1499_COMMS_RS_00010 MAX1499_COMMS_RS_00101 MAX1499_COMMS_RS_00101 MAX1499_COMMS_RS_00101 MAX1499_COMMS_RS_00101 MAX1499_COMMS_RS_00110 MAX1499_COMMS_RS_00110 MAX1499_COMMS_RS_00110 MAX1499_COMMS_RS_00110 MAX1499_COMMS_RS_00111	UX3E 0x00 0x00 0x02 0x02 0x04 0x04 0x04 0x06 0x06 0x08 0x08 0x08 0x08 0x08 0x08 0x08 0x02 0x04 0x04 0x06 0x06 0x08 0x08 0x08 0x08 0x08 0x08 0x08 0x02 0x02 0x02 0x02 0x02 0x02 0x02 0x02 0x02 0x02 0x02 0x02 0x04 0x06 0x08 0x08 0x08 0x08 0x08 0x08 0x08 0x02 0x02 0x02 0x02 0x08 0x08 0x08 0x02 0x02 0x02 0x08
<pre>#define #define #</pre>	MAX1499_COMMS_RS_DABR MAX1499_COMMS_RS_STATUS MAX1499_COMMS_RS_STATUS MAX1499_COMMS_RS_ONTROL MAX1499_COMMS_RS_ONTROL MAX1499_COMMS_RS_OVERRANGE MAX1499_COMMS_RS_OVERRANGE MAX1499_COMMS_RS_UDERRANGE MAX1499_COMMS_RS_LED_SEG_1 MAX1499_COMMS_RS_LED_SEG_2 MAX1499_COMMS_RS_UD10 MAX1499_COMMS_RS_UD10 MAX1499_COMMS_RS_UD10 MAX1499_COMMS_RS_UD10 MAX1499_COMMS_RS_UD10 MAX1499_COMMS_RS_UD110 MAX1499_COMMS_RS_UD110 MAX1499_COMMS_RS_UD110 MAX1499_COMMS_RS_UD100 MAX1499_COMMS_RS_UD100 MAX1499_COMMS_RS_UD100	0x3E 0x00 0x02 0x02 0x04 0x04 0x04 0x06 0x06 0x08 0x08 0x08 0x08 0x08 0x08 0x02 0x04 0x06 0x06 0x08 0x08 0x08 0x08 0x02 0x02 0x02 0x02 0x02 0x02 0x02 0x02 0x02 0x02 0x02 0x02 0x02 0x02 0x02 0x02 0x04 0x06 0x08 0x08 0x08 0x00 0x02 0x02 0x02 0x02 0x02 0x02 0x02 0x02 0x06 0x08 0x08 0x00 0x00 0x02 0x02 0x02 0x08 0x02 0x02 0x02 0x02 0x08 0x08 0x00 0x00 0x02 0x02 0x02 0x02 0x02 0x08 0x02
<pre>#define #define #</pre>	MAX1499_COMMS_RS_DOUD MAX1499_COMMS_RS_STATUS MAX1499_COMMS_RS_STATUS MAX1499_COMMS_RS_STATUS MAX1499_COMMS_RS_OONTROL MAX1499_COMMS_RS_OVERRANGE MAX1499_COMMS_RS_OVERRANGE MAX1499_COMMS_RS_OUDIN MAX1499_COMMS_RS_UDDERRANGE MAX1499_COMMS_RS_LED_SEG_1 MAX1499_COMMS_RS_LED_SEG_2 MAX1499_COMMS_RS_UDIN MAX1499_COMMS_RS_UDIN MAX1499_COMMS_RS_00110 MAX1499_COMMS_RS_UDIN MAX1499_COMMS_RS_00111 MAX1499_COMMS_RS_00111 MAX1499_COMMS_RS_00111 MAX1499_COMMS_RS_D0111 MAX1499_COMMS_RS_00101 MAX1499_COMMS_RS_00100 MAX1499_COMMS_RS_0000 MAX1499_COMMS_RS_0000 MAX1499_COMMS_RS_0000	UX3E 0x00 0x02 0x02 0x04 0x04 0x06 0x06 0x08 0x08 0x08 0x08 0x08 0x08 0x08 0x02 0x02 0x02 0x02 0x02 0x02 0x04 0x06 0x06 0x08 0x08 0x08 0x08 0x08 0x08 0x08 0x08 0x08 0x08 0x08 0x08 0x08 0x08 0x08 0x08 0x08 0x02 0x02 0x04 0x08
<pre>#define #define #</pre>	MAX1499_COMMS_RS_D0000 MAX1499_COMMS_RS_STATUS MAX1499_COMMS_RS_STATUS MAX1499_COMMS_RS_O0001 MAX1499_COMMS_RS_O0010 MAX1499_COMMS_RS_OVERRANGE MAX1499_COMMS_RS_O0011 MAX1499_COMMS_RS_00101 MAX1499_COMMS_RS_00101 MAX1499_COMMS_RS_00101 MAX1499_COMMS_RS_00101 MAX1499_COMMS_RS_LED_SEG_2 MAX1499_COMMS_RS_00110 MAX1499_COMMS_RS_00111 MAX1499_COMMS_RS_00111 MAX1499_COMMS_RS_00111 MAX1499_COMMS_RS_00111 MAX1499_COMMS_RS_00111 MAX1499_COMMS_RS_00100 MAX1499_COMMS_RS_0000 MAX1499_COMMS_RS_0000 MAX1499_COMMS_RS_0000 MAX1499_COMMS_RS_00101	UX3E 0x00 0x00 0x02 0x02 0x04 0x04 0x06 0x06 0x08 0x08 0x08 0x08 0x08 0x08 0x08 0x08 0x02 0x02 0x02 0x02 0x02 0x02 0x02 0x02 0x04 0x04 0x04 0x06 0x06 0x06 0x08 0x08 0x08 0x08 0x08 0x08 0x08 0x02 0x02 0x02 0x02 0x04 0x06 0x06 0x06 0x08 0x08 0x08 0x08 0x08 0x08 0x08 0x02 0x02 0x02 0x08 0x08 0x08 0x08 0x08 0x08 0x08 0x08 0x08 0x02 0x02 0x02 0x08
<pre>#define #define #</pre>	MAX1499_COMMS_RS_D0000 MAX1499_COMMS_RS_STATUS MAX1499_COMMS_RS_STATUS MAX1499_COMMS_RS_STATUS MAX1499_COMMS_RS_O0010 MAX1499_COMMS_RS_00010 MAX1499_COMMS_RS_00101 MAX1499_COMMS_RS_00101 MAX1499_COMMS_RS_00101 MAX1499_COMMS_RS_00101 MAX1499_COMMS_RS_00110 MAX1499_COMMS_RS_00110 MAX1499_COMMS_RS_00110 MAX1499_COMMS_RS_00110 MAX1499_COMMS_RS_00111 MAX1499_COMMS_RS_00111 MAX1499_COMMS_RS_00111 MAX1499_COMMS_RS_01001 MAX1499_COMMS_RS_01000 MAX1499_COMMS_RS_01000 MAX1499_COMMS_RS_01001 MAX1499_COMMS_RS_01001 MAX1499_COMMS_RS_01001 MAX1499_COMMS_RS_01001 MAX1499_COMMS_RS_01001 MAX1499_COMMS_RS_01001 MAX1499_COMMS_RS_1001 MAX1499_COMMS_RS_1001 MAX1499_COMMS_RS_1001 MAX1499_COMMS_RS_1001	UX3E 0x00 0x00 0x02 0x02 0x04 0x04 0x06 0x06 0x08 0x08 0x08 0x08 0x08 0x08 0x08 0x00 0x00 0x02 0x02 0x02 0x02 0x02 0x02 0x04 0x06 0x06 0x06 0x06 0x06 0x08 0x12 0x12 0x12 0x12 0x18
<pre>#define #define</pre>	MAX1499_COMMS_RS_D0000 MAX1499_COMMS_RS_STATUS MAX1499_COMMS_RS_STATUS MAX1499_COMMS_RS_STATUS MAX1499_COMMS_RS_ONTROL MAX1499_COMMS_RS_OVERRANGE MAX1499_COMMS_RS_OVERRANGE MAX1499_COMMS_RS_UDERRANGE MAX1499_COMMS_RS_LED_SEG_1 MAX1499_COMMS_RS_LED_SEG_2 MAX1499_COMMS_RS_LED_SEG_2 MAX1499_COMMS_RS_00110 MAX1499_COMMS_RS_00110 MAX1499_COMMS_RS_LED_SEG_3 MAX1499_COMMS_RS_LED_SEG_3 MAX1499_COMMS_RS_LED_SEG_3 MAX1499_COMMS_RS_D0110 MAX1499_COMMS_RS_00111 MAX1499_COMMS_RS_00100 MAX1499_COMMS_RS_01001 MAX1499_COMMS_RS_D001 MAX1499_COMMS_RS_D0101 MAX1499_COMMS_RS_LED_DATA MAX1499_COMMS_RS_0101	UX3E 0x00 0x02 0x02 0x04 0x04 0x04 0x06 0x06 0x08 0x08 0x08 0x08 0x08 0x08 0x02 0x02 0x02 0x02 0x02 0x02 0x02 0x02 0x04 0x06 0x06 0x08 0x10 0x10 0x12 0x14
<pre>#define #define #</pre>	MAX1499_COMMS_RS_D0000 MAX1499_COMMS_RS_STATUS MAX1499_COMMS_RS_STATUS MAX1499_COMMS_RS_STATUS MAX1499_COMMS_RS_ONTROL MAX1499_COMMS_RS_OVERRANGE MAX1499_COMMS_RS_OVERRANGE MAX1499_COMMS_RS_UDERRANGE MAX1499_COMMS_RS_LED_SEG_1 MAX1499_COMMS_RS_LED_SEG_1 MAX1499_COMMS_RS_LED_SEG_1 MAX1499_COMMS_RS_LED_SEG_1 MAX1499_COMMS_RS_LED_SEG_2 MAX1499_COMMS_RS_LED_SEG_3 MAX1499_COMMS_RS_LED_SEG_3 MAX1499_COMMS_RS_LED_SEG_3 MAX1499_COMMS_RS_LED_SEG_3 MAX1499_COMMS_RS_LED_SEG_3 MAX1499_COMMS_RS_D0101 MAX1499_COMMS_RS_ADC_OFFSET MAX1499_COMMS_RS_ADC_RESULT: MAX1499_COMMS_RS_1LED_DATA MAX1499_COMMS_RS_1LED_DATA MAX1499_COMMS_RS_1010 MAX1499_COMMS_RS_1010 MAX1499_COMMS_RS_1010 MAX1499_COMMS_RS_1010 MAX1499_COMMS_RS_1010 MAX1499_COMMS_RS_1010	UX3E 0x00 0x00 0x02 0x04 0x04 0x06 0x06 0x08 0x08 0x08 0x08 0x08 0x08 0x08 0x02 0x02 0x02 0x02 0x02 0x02 0x02 0x04 0x04 0x04 0x06 0x08 0x10 0x10 0x12 0x14 0x14 0x14
<pre>#define #define</pre>	MAX1499_COMMS_RS_DABK MAX1499_COMMS_RS_STATUS MAX1499_COMMS_RS_STATUS MAX1499_COMMS_RS_STATUS MAX1499_COMMS_RS_ONTROL MAX1499_COMMS_RS_ONTROL MAX1499_COMMS_RS_OVERRANGE MAX1499_COMMS_RS_ONDIN MAX1499_COMMS_RS_UDDERRANGE MAX1499_COMMS_RS_UDDERRANGE MAX1499_COMMS_RS_LED_SEG_1 MAX1499_COMMS_RS_LED_SEG_1 MAX1499_COMMS_RS_UDDIN MAX1499_COMMS_RS_UDDIN MAX1499_COMMS_RS_UDDIN MAX1499_COMMS_RS_UDDIN MAX1499_COMMS_RS_UDDIN MAX1499_COMMS_RS_D0111 MAX1499_COMMS_RS_D0111 MAX1499_COMMS_RS_D0101 MAX1499_COMMS_RS_D1000 MAX1499_COMMS_RS_D1001 MAX1499_COMMS_RS_D1001 MAX1499_COMMS_RS_D1010 MAX1499_COMMS_RS_UDDIN MAX1499_COMMS_RS_D1010 MAX1499_COMMS_RS_D1010 MAX1499_COMMS_RS_D1010 MAX1499_COMMS_RS_D1010 MAX1499_COMMS_RS_D1010 MAX1499_COMMS_RS_D1010 MAX1499_COMMS_RS_D1010 MAX1499_COMMS_RS_D1010 MAX1499_COMMS_RS_D1010	0x3E 0x00 0x02 0x02 0x04 0x04 0x06 0x06 0x08 0x08 0x08 0x08 0x08 0x00 0x02 0x0C 0x0C 0x0C 0x0C 0x0C 0x0C 0x10 1 0x10 0x12 0x14 0x14 0x28

Listing 1 (Sheet 1 of 4)

#define MAX	1499 STATUS POL MASK	0×80				
#define MAX	(1499 STATUS POL POSITIVE	0x00				
#define MAX	K1499 STATUS POL NEGATIVE	0x80				
#define MAX	(1499 STATUS OVER RANGE	0x40				
#define MAX	K1499 STATUS UNDER RANGE	0x20				
#define MAX	K1499 STATUS LOW BATTERY	0x10				
#define MAX	<1499_STATUS_DATA_READY	0x08				
//						
// Define the bit // SPI ADC EX	s in the CONTROL register. TCLK INTREF DPON DPSET2 DP	SET1 PD DIG	PD ANA			
// HOLD PE.	AK RANGE CLR LED OFFSET	CAL1 OFFSET_CA	AL2 ENABLE			
#define MAX	<pre>K1499_CONTROL_SPI_ADC</pre>	0x8000				
#define MAX	(1499_CONTROL_EXTCLK	0 x 4 0 0 0				
#define MAX	<pre>K1499_CONTROL_INTREF</pre>	0x2000				
#define MAX	(1499_CONTROL_DPMASK	0x1C00				
#define MAX	(1499_CONTROL_DPON	0x1000				
#define MAX	<1499_CONTROL_DPSET2	0x0800				
#define MAX	(1499 CONTROL DPSET1	0x0400				
Hofine Mar	LOD AND DESENTIS THE MODE	01000 /.	_1000 0 ↓	/		
#define MAX	(1499 CONTROL DPION	0x1000 /^ 0x1000 /*	-188 88 *	, /		
#define MAX	(1499 CONTROL DESON	0x1000 /^ 0x1/00 /*	-18 888 *	, /		
#define MAX	(1499 CONTROL DESON	0x1000 /*	-1.8888 *	/		
#define MAY	(1499 CONTROL PD DIG	0x0200	1.0000 ")			
#define MAY	(1499 CONTROL PD ANA	0x0100				
#define MAY	(1499 CONTROL PD ALL	0x0300				
#define MAX	(1499 CONTROL HOLD	0x0080				
#define MAX	(1499 CONTROL PEAK	0x0040				
#define MAX	K1499 CONTROL RANGE 200mV	0x0020				
#define MAX	(1499 CONTROL CLR	0x0010				
#define MAX	(1499 CONTROL SEG SEL	0x0008				
#define MAX	K1499 CONTROL OFFSET CAL1	0x0004				
#define MAX	K1499 CONTROL OFFSET CAL2	0x0002				
#define MAX	(1499_CONTROL_ENABLE	0x0001				
/ #define MAX #define MAX #define MAX #define MAX	(1499_LED_SEG1_A1 (1499_LED_SEG1_G1 (1499_LED_SEG1_D1 (1499_LED_SEG1_F1 (1499_LED_SEG1_E1	0x8000 0x4000 0x2000 0x1000 0x0800				
define MAX define MAX define MAX define MAX define MAX define MAX define MAX define MAX	<pre>k1499_LED_SEG1_DP2 k1499_LED_SEG1_B0 k1499_LED_SEG1_C0 k1499_LED_SEG1_A0 k1499_LED_SEG1_G0 k1499_LED_SEG1_D0 k1499_LED_SEG1_F0 k1499_LED_SEG1_F0 k1499_LED_SEG1_E0 k1499_LED_SEG1_DP1</pre>	0x0400 0x0100 0x0080 0x0040 0x0020 0x0010 0x0008 0x0004 0x0004				
#define MAX #define MAX #define MAX #define MAX #define MAX #define MAX #define MAX #define MAX #define MAX //	(1499_LED_SEG1_DP2 (1499_LED_SEG1_B0 (1499_LED_SEG1_C0 (1499_LED_SEG1_C0 (1499_LED_SEG1_G0 (1499_LED_SEG1_G0 (1499_LED_SEG1_F0 (1499_LED_SEG1_E0 (1499_LED_SEG1_DP1 s in the LED SEGMENT 2 register. HB3 C3 A3 C3 0 B2 C2 0	0x0400 0x0100 0x0080 0x0040 0x0020 0x0010 0x0008 0x0004 0x0002				
Weetine     MAX       #define     MAX       //     // <td>(1499_LED_SEG1_DP2 (1499_LED_SEG1_DP2 (1499_LED_SEG1_C0 (1499_LED_SEG1_C0 (1499_LED_SEG1_G0 (1499_LED_SEG1_G0 (1499_LED_SEG1_F0 (1499_LED_SEG1_E0 (1499_LED_SEG1_DP1 s in the LED SEGMENT 2 register. B B3 C3 A3 C3 6 0 B2 C2 0 (1499_LED_SEG2_F3 (1499_LED_SEG2_F3</td> <td>0x0400 0x0100 0x0080 0x0020 0x0010 0x0008 0x0004 0x0002</td> <td></td> <td></td> <td></td> <td></td>	(1499_LED_SEG1_DP2 (1499_LED_SEG1_DP2 (1499_LED_SEG1_C0 (1499_LED_SEG1_C0 (1499_LED_SEG1_G0 (1499_LED_SEG1_G0 (1499_LED_SEG1_F0 (1499_LED_SEG1_E0 (1499_LED_SEG1_DP1 s in the LED SEGMENT 2 register. B B3 C3 A3 C3 6 0 B2 C2 0 (1499_LED_SEG2_F3 (1499_LED_SEG2_F3	0x0400 0x0100 0x0080 0x0020 0x0010 0x0008 0x0004 0x0002				
Adefine MAX Hdefine MAX Hdefine MAX Hdefine MAX Hdefine MAX Hdefine MAX Hdefine MAX Hdefine MAX /	(1499_LED_SEG1_DP2 (1499_LED_SEG1_DP2 (1499_LED_SEG1_C0 (1499_LED_SEG1_C0 (1499_LED_SEG1_A0 (1499_LED_SEG1_G0 (1499_LED_SEG1_D0 (1499_LED_SEG1_F0 (1499_LED_SEG1_DP1 	0x0400 0x0100 0x0080 0x0040 0x0020 0x0010 0x0008 0x0004 0x0002				
<pre>% A state of the state of</pre>	(1499_LED_SEG1_DP2 (1499_LED_SEG1_DP2 (1499_LED_SEG1_C0 (1499_LED_SEG1_C0 (1499_LED_SEG1_G0 (1499_LED_SEG1_G0 (1499_LED_SEG1_F0 (1499_LED_SEG1_E0 (1499_LED_SEG1_DP1 s in the LED SEGMENT 2 register. H B3 C3 A3 C3 60 B2 C2 0 (1499_LED_SEG2_F3 (1499_LED_SEG2_P4 (1499_LED_SEG2_PP4 (1499_LED_SEG2_MINUS	0x0400 0x0100 0x0080 0x0040 0x0020 0x0010 0x0008 0x0004 0x0002 	TODO: is	this documented	i? minus sign	
Weetine     MAX       #define     MAX       //     Define       //     //       //     Dofine       //     // </td <td>(1499_LED_SEG1_DP2 (1499_LED_SEG1_DP2 (1499_LED_SEG1_C0 (1499_LED_SEG1_C0 (1499_LED_SEG1_G0 (1499_LED_SEG1_G0 (1499_LED_SEG1_F0 (1499_LED_SEG1_E0 (1499_LED_SEG1_DP1 </td> <td>0x0400 0x0100 0x0080 0x0020 0x0010 0x0008 0x0004 0x0002 </td> <td>TODO: is</td> <td>this documented</td> <td>ž? minus sign</td> <td></td>	(1499_LED_SEG1_DP2 (1499_LED_SEG1_DP2 (1499_LED_SEG1_C0 (1499_LED_SEG1_C0 (1499_LED_SEG1_G0 (1499_LED_SEG1_G0 (1499_LED_SEG1_F0 (1499_LED_SEG1_E0 (1499_LED_SEG1_DP1 	0x0400 0x0100 0x0080 0x0020 0x0010 0x0008 0x0004 0x0002 	TODO: is	this documented	ž? minus sign	
#define MAX #define MAX #define MAX #define MAX #define MAX #define MAX #define MAX #define MAX //	<pre>(1499_LED_SEG1_DP2 (1499_LED_SEG1_DP (1499_LED_SEG1_C0 (1499_LED_SEG1_G0 (1499_LED_SEG1_G0 (1499_LED_SEG1_D0 (1499_LED_SEG1_F0 (1499_LED_SEG1_DP1  s in the LED SEGMENT 2 register. 4B3 C3 A3 G3 0 B2 C2 0 (1499_LED_SEG2_F3 (1499_LED_SEG2_DP4 (1499_LED_SEG2_MINUS (1499_LED_SEG2_B2 (1490_LED_SEG2_B2 (140_LED_SEG2_B2 (140_LED_SEG2_B2 (140_LED_SEG2_B2 (140_LED_SEG2_B2 (140_LED_SEG2_B2 (140_LED_SEG2_B2 (140_LED_SEG2_B2 (140_LED_SEG2_B2 (140_L</pre>	0x0400 0x0100 0x0080 0x0020 0x0010 0x0008 0x0004 0x0002 	TODO: is	this documented	3? minus sign	
#define MAX #define MAX	11499_LED_SEG1_DP2 (1499_LED_SEG1_B0 (1499_LED_SEG1_C0 (1499_LED_SEG1_G0 (1499_LED_SEG1_G0 (1499_LED_SEG1_F0 (1499_LED_SEG1_F0 (1499_LED_SEG1_E0 (1499_LED_SEG1_E0 (1499_LED_SEG1_DP1 s in the LED SEGMENT 2 register. 183 C3 A3 G3 0 B2 C2 0 (1499_LED_SEG2_F3 (1499_LED_SEG2_F3 (1499_LED_SEG2_DP4 (1499_LED_SEG2_MINUS (1499_LED_SEG2_B2 (1499_LED_SEG2_C2 (1490_LED_SEG2_C2 (140_LED_SEG2_C2 (140_LED_SEG2_C2 (140_LED_SEG2_C2 (140_LED_SEG2_C2 (140_LED_SEG2_C2 (140_LED_SEG2_C2 (140_LED_SEG2_C2 (140_LED_SEG2_C2 (140_LED_SEG2 (140_LED_SEG3_C2 (140_LED_SEG3 (140_LED_SEG3 (14	0x0400 0x0100 0x0080 0x0040 0x0010 0x0010 0x0008 0x0004 0x0002 0x0002 0x0000 0x0000 0x0000 0x1000 /* 0x0800 0x0400 0x0200	TODO: is	this documented	d? minus sign	
<pre>#define MAX #define MAX #define MAX #define MAX #define MAX #define MAX #define MAX #define MAX #define MAX //</pre>	(1499_LED_SEG1_DP2 (1499_LED_SEG1_DP2 (1499_LED_SEG1_CO (1499_LED_SEG1_CO (1499_LED_SEG1_GO (1499_LED_SEG1_GO (1499_LED_SEG1_FO (1499_LED_SEG1_EO (1499_LED_SEG1_EO (1499_LED_SEG1_DP1 	0x0400 0x0100 0x0080 0x0040 0x0020 0x0010 0x0008 0x0004 0x0002 0x0002 0x0000 0x0002 0x0000 0x1000 /* 0x0800 0x0400 0x0400 0x0400 0x0400 0x0400 0x0400 0x0400 0x0400 0x0400 0x0400 0x0400 0x0100 0x0100 0x0100 0x0100 0x0100 0x0100 0x0100 0x0000 0x0000 0x0000 0x0000 0x000 0x00000 0x0000 0x00000 0x000000	TODO: is	this documented	3? minus sign	
<pre>#define MAX #define MAX</pre>	(1499_LED_SEG1_DP2 (1499_LED_SEG1_DP2 (1499_LED_SEG1_CO (1499_LED_SEG1_CO (1499_LED_SEG1_GO (1499_LED_SEG1_GO (1499_LED_SEG1_FO (1499_LED_SEG1_EO (1499_LED_SEG1_EO (1499_LED_SEG1_DP1 ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) ) )	0x0400 0x0100 0x0080 0x0040 0x0020 0x0010 0x0008 0x0004 0x0004 0x0002 0x0004 0x0002 0x0000 0x4000 0x2000 0x1000 /*	TODO: is	this documented	d? minus sign	
<pre>#define MAX #define MAX</pre>	<pre>(1499_LED_SEG1_DP2 (1499_LED_SEG1_DP (1499_LED_SEG1_C0 (1499_LED_SEG1_G0 (1499_LED_SEG1_G0 (1499_LED_SEG1_G0 (1499_LED_SEG1_F0 (1499_LED_SEG1_E0 (1499_LED_SEG1_DP1</pre>	0x0400 0x0100 0x0080 0x0040 0x0020 0x0010 0x0008 0x0004 0x0002 0x0004 0x0002 0x0000 0x1000 /* 0x0800 0x0400 0x0200 0x0400 0x0200 0x0100 0x0100 0x0080 0x0040	TODO: is	this documented	3? minus sign	
<pre>#define MAX #define MAX #</pre>	(1499_LED_SEG1_DP2 (1499_LED_SEG1_DP2 (1499_LED_SEG1_B0 (1499_LED_SEG1_C0 (1499_LED_SEG1_G0 (1499_LED_SEG1_D0 (1499_LED_SEG1_F0 (1499_LED_SEG1_F0 (1499_LED_SEG1_E0 (1499_LED_SEG1_DP1 	0x0400 0x0100 0x0080 0x0040 0x0020 0x0010 0x0008 0x0004 0x0002 0x0004 0x0002 0x1000 0x1000 0x1000 0x0000 0x000 0x000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x000000	TODO: is	this documented	i? minus sign	

Listing 1 (Sheet 2 of 4)

```
#define MAX1499 LED SEG2 DP3
                                                                   0x0010
#define MAX1499 LED SEG2 B1
                                                                   0x0004
#define MAX1499 LED SEG2 C1
                                                                  0x0002
                                                   -----
// Define the bits in the LED SEGMENT 3 register.
// 0 0 BC5 B4 C4 A4 G4 D4
#define MAX1499_LED_SEG3_BC_
                                                                  0x20
#define MAX1499_LED_SEG3_B3
#define MAX1499_LED_SEG3_C3
                                                                  0x10
                                                                  0x08
#define MAX1499_LED_SEG3_A3
#define MAX1499_LED_SEG3_G3
                                                                  0x04
                                                                  0x02
#define MAX1499_LED_SEG3_D3
                                                                  0x01
//-----
class MAX1499
ł
public:
       MAX1499(void);
     // Enumerated type describing the register select bits.
enum RegisterSelect_t {
    RS_STATUS = MAX1499_COMMS_RS_STATUS,
    RS_CONTROL = MAX1499_COMMS_RS_CONTROL,
    RS_OVERRANGE = MAX1499_COMMS_RS_UNDERRANGE,
    RS_LED_SEG_1 = MAX1499_COMMS_RS_LED_SEG_1,
    RS_LED_SEG_2 = MAX1499_COMMS_RS_LED_SEG_2,
    RS_LED_SEG_3 = MAX1499_COMMS_RS_LED_SEG_3,
    RS_ADC_OFFSET = MAX1499_COMMS_RS_ADC_OFFSET,
    RS_ADC_RESULT1 = MAX1499_COMMS_RS_ADC_OFFSET,
    RS_LED_DATA = MAX1499_COMMS_RS_LED_DATA,
    RS_FEAR = MAX1499_COMMS_RS_PEAR,
    RS_ADC_RESULT2 = MAX1499_COMMS_RS_ADC_RESULT2
};

       // Enumerated type describing the register select bits.
       1;
       // Reference voltage
       double vref;
       //-----
       // Status Register
       // POL OVR_RNG UNDR_RNG LOW_BATT ADD(data available) 0 0 0
       int STATUS_REG;
       bool Read STATUS(void);
       //-----
       // Control Register
       // Control Register
// SPI_ADC_EXTCLK_INTREF DPON_DPSET2_DPSET1_PD_DIG_PD_ANA
// HOLD_PEAK_RANGE_CLR_LED_OFFSET_CAL1_OFFSET_CAL2_ENABLE
       int CONTROL REG;
       bool Write CONTROL(int data);
       bool Read CONTROL(void);
       //-----
       // Data Registers
       int ADC_RESULT1;
       unsigned int ADC_RESULT2;
      bool Read_ADC_RESULT1 (void);
bool Read_ADC_RESULT2 (void);
long int DATA_REG; // 16-bit or 24-bit result from A/D converter
       bool extended_resolution;
       long Read_DATA(void);
       double Voltage(void);
       // Other registers, having 16-bit 2's complement data format
       bool Write_2s_complement(int reg, int data);
int Read_2s_complement(int reg);
```

```
Listing 1 (Sheet 3 of 4)
```

Evaluate: MAX1498/MAX1499

// Other registers, having 8 bit data format
bool Write\_8bit\_reg(int reg, int data);
int Read\_8bit\_reg(int reg);

};

//-----#endif

Listing 1 (Sheet 4 of 4)

```
// Drv1499.cpp
// MAX1499-specific driver.
// mku 09/15/2003
// (C) 2003 Maxim Integrated Products
// For use with Borland C++ Builder 3.0
// Revision history:
// 09/15/2003: add double Voltage(void)
// 09/09/2003: add class MAX1499 dependent on external SPI_Interface()
// 08/13/2003: preliminary draft of reuseable code
#include "drv1499.h"
//-----
MAX1499::MAX1499(void)
{
           vref = 2.048;
           extended_resolution = false;
}
//-----
bool MAX1499::Read STATUS(void)
          (unsigned __int8) (0xFF)
          };
         };
unsigned __int8 miso_buf[sizeof(mosi)];
bool result = SPI_Transfer(sizeof(mosi), mosi, miso_buf);
if (result) {
    int data = miso_buf[1];
    // remember the value we just received
    //
                     STATUS_REG = data;
                                                                                            // remember the value we just received
          }
          return result;
.//-----
bool MAX1499::Write CONTROL(int data)
{
          data = data & OxFFFF;
                                                                                 // validate the data
         const unsigned __int8 mosi[] = {
   (unsigned __int8 mosi[] = {
      (unsigned __int8) (MAX1499 COMMS_START |
      MAX1499_COMMS_RW_WRITE | MAX1499_COMMS_RS_CONTROL),
      (unsigned __int8) ( data >> 8) & 0xFF),
      (unsigned __int8) ( data & 0xFF)
          };
          unsigned _
                                    __int8 miso_buf[sizeof(mosi)];
          bool result = SPI Transfer(sizeof(mosi), mosi, miso buf);
          CONTROL REG = data; // remember the value we just wrote // The CLR bit is self-clearing, and should not be kept high.
           CONTROL REG &=~ MAX1499 CONTROL CLR;
          return result;
.//-----
bool MAX1499::Read CONTROL (void)
{
          const unsigned __int8 mosi[] = {
  (unsigned __int8) (MAX1499_COMMS_START |
      MAX1499_COMMS_RW_READ | MAX1499_COMMS_RS_CONTROL),
                     (unsigned __int8) (OxFF),
(unsigned __int8) (OxFF)
          };
          unsigned
                                     int8 miso buf[sizeof(mosi)];
          bool result = SPI Transfer(sizeof(mosi), mosi, miso buf);
          if (result) {
                     int data = miso_buf[1] * 0x100 + miso_buf[2];
                     CONTROL_REG = data;
                                                                                                 // remember the value we just wrote
          }
          return result;
}
//-----
bool MAX1499::Read_ADC_RESULT1(void)
          const unsigned __int8 mosi[] = {
```

Listing 2 (Sheet 1 of 4)

M/XI/M

```
Evaluate: MAX1498/MAX1499
```

```
igned __int8) (MAX1499_COMMS_START |
MAX1499 COMMS RW READ | MAX1499 COMMS RS ADC RESULT1),
            (unsigned
            (unsigned __int8) (0xFF),
(unsigned __int8) (0xFF)
      };
      unsigned __int8 miso_buf[sizeof(mosi)];
     bool result = SPI_Transfer_After_EOC(sizeof(mosi), mosi, miso_buf);
if (result) {
           ADC_RESULT1 = (miso_buf[1] * 0x100L) + miso_buf[2];
long data = (miso_buf[1] * 0x100L) + miso_buf[2];
if (data >= 32768) {
                 data -= 65536;
            DATA REG = data;
                                                // remember the value we just received
     }
     return result:
         .....
bool MAX1499::Read_ADC_RESULT2 (void)
ł
     const unsigned __int8 mosi[] = {
  (unsigned __int8) (MAX1499_COMMS_START |
      MAX1499_COMMS_RW_READ | MAX1499_COMMS_RS_ADC_RESULT2),
           (unsigned __int8) (0×FF)
     };
     unsigned __int8 miso_buf[sizeof(mosi)];
bool result = SPI_Transfer(sizeof(mosi), mosi, miso_buf);
     if (result) {
            ADC RESULT2 = miso buf[1];
           long data 24 = (long) ADC_RESULT1 * 0x100L) + ADC_RESULT2;
DATA_REG = data_24;
      return result;
     _____
//----
long MAX1499::Read_DATA(void)
{
     // Read the DATA register
     const unsigned __int8 mosi[] = {
    (unsigned __int8) (MAX1499_COMMS_START |

           MAX1499_COMMS_RW_READ | MAX1499_COMMS_RS_ADC_RESULT1),
(unsigned __int8)(0xFF),
(unsigned __int8)(0xFF)
     };
     unsigned __int8 miso_buf[sizeof(mosi)];
if (SPI_Transfer_After_EOC(sizeof(mosi), mosi, miso_buf) == false) {
           return 0; // failure
     }
     ADC_RESULT1 = (miso_buf[1] * 0x100L) + miso_buf[2];
long data = (miso_buf[1] * 0x100L) + miso_buf[2];
if (data >= 32768) {
           data -= 65536:
     }
      DATA REG = data;
                                             // remember the value we just received
     if (extended_resolution) {
    // Read the ADC_RESULT2 register
           const unsigned __int8 mosi[] = {
    (unsigned __int8 mosi[] = {
        MAX1499_COMMS_START |
        MAX1499_COMMS_RW_READ | MAX1499_COMMS_RS_ADC_RESULT2),
                  (unsigned int8) \overline{(0 \times FF)}
           };
            unsigned _
                          _int8 miso_buf[sizeof(mosi)];
           if (SPI_Transfer(sizeof(mosi), mosi, miso_buf) == false) {
                 return 0; // failure
            ł
           ADC_RESULT2 = miso_buf[1];
long data_24 = ((long)ADC_RESULT1 * 0x100L) + ADC_RESULT2;
double data_16 = data_24 / 256.0;
           if (data_16 >= 32768) {
                 data_16 = data_16 - 65536;
            DATA_REG = data_24;
```

Listing 2 (Sheet 2 of 4)

```
}
     return DATA REG;
}
double MAX1499::Voltage(void)
     if ((CONTROL REG & MAX1499 CONTROL RANGE 200mV) == 0) {
          // Input range 2V
          return DATA REG * (vref / 2.048) * 10e-6 * 10;
     } else {
// Input range 200mV
          return DATA REG * (vref / 2.048) * 10e-6;
     }
bool MAX1499::Write_2s_complement(int reg, int data)
ł
     // Write one of the 2's complement registers
     reg = (reg & MAX1499_COMMS_RS_MASK);
data = data & 0xFFFF; // validate the
                                       // validate the data
     const unsigned __int8 mosi[] = {
   (unsigned __int8) (MAX1499 COMMS_START | MAX1499_COMMS_RW_WRITE | reg),
   (unsigned __int8) ((data >> 8) & 0xFF),
   (unsigned __int8) (data & 0xFF)
     };
     unsigned __int8 miso_buf[sizeof(mosi)];
bool result = SPI_Transfer(sizeof(mosi), mosi, miso_buf);
     return result;
//-----
int MAX1499::Read_2s_complement(int reg)
ł
     // Read one of the 2's complement registers
     reg = (reg & MAX1499_COMMS_RS_MASK);
     const unsigned __int8 mosi[] = {
   (unsigned __int8) (MAX1499_COMMS_START | MAX1499_COMMS_RW_READ | reg),
   (unsigned __int8) (0xFF),
   (unsigned __int8) (0xFF)
     };
     unsigned
                  _int8 miso buf[sizeof(mosi)];
     bool result = SPI Transfer(sizeof(mosi), mosi, miso_buf);
if (result == false) {
          return 0; // failure
     }
     int data = miso buf[1] * 0x100 + miso buf[2];
     if (data >= 32768) {
          data -= 65536;
     if (data >= 32768) {
          data -= 65536;
     ł
     return data;
bool MAX1499::Write_8bit_reg(int reg, int data)
{
     // Write one of the 8 bit registers
     reg = (reg & MAX1499_COMMS_RS_MASK);
     const unsigned __int8 mosi[] = {
   (unsigned __int8) (MAX1499_COMMS_START | MAX1499_COMMS_RW_WRITE | reg),
   (unsigned __int8) (data & OxFF)
     };
     unsigned __int8 miso_buf[sizeof(mosi)];
     bool result = SPI_Transfer(sizeof(mosi), mosi, miso_buf);
     return result;
//---
int MAX1499::Read 8bit reg(int reg)
{
     // Read one of the 8 bit registers
     reg = (reg & MAX1499 COMMS RS MASK);
```

Listing 2 (Sheet 3 of 4)

```
Evaluate: MAX1498/MAX1499
```

const unsigned \_\_int8 mosi[] = {
 (unsigned \_\_int8) (MAX1499\_COMMS\_START | MAX1499\_COMMS\_RW\_READ | reg),
 (unsigned \_\_int8) (0xFF)
};
unsigned \_\_int8 miso\_buf[sizeof(mosi)];
bool result = SPI\_Transfer(sizeof(mosi), mosi, miso\_buf);
if (result == false) {
 return 0; // failure
}
int data = miso\_buf[1];
return data;

} //\_\_\_\_\_

Listing 2 (Sheet 4 of 4)

Maxim cannot assume responsibility for use of any circuitry other than circuitry entirely embodied in a Maxim product. No circuit patent licenses are implied. Maxim reserves the right to change the circuitry and specifications without notice at any time.

Printed USA

24 \_

\_\_\_\_\_Maxim Integrated Products, 120 San Gabriel Drive, Sunnyvale, CA 94086 408-737-7600

© 2004 Maxim Integrated Products

**MAXIM** is a registered trademark of Maxim Integrated Products.