



MAX1499 评估板/MAX1499 评估系统

概述

特性

MAX1499 评估系统 (EV system) 由 MAX1499 评估板 (EV kit) 和 Maxim 的 68HC16MODULE-DIP 微控制器 (μC) 模块组成。MAX1499 是集成了 LED 显示驱动器的低功耗 4.5 位模数转换器 (ADC)。评估软件可运行在 Windows[®] 95/98/2000/XP 操作系统下, 为评估 MAX1499 提供了方便的用户界面。

- ◆ 经过验证的 PC 板布局
- ◆ 完整的评估系统
- ◆ 方便的板上测试点
- ◆ 数据记录软件
- ◆ 经过完全安装和测试

如使用 PC 机对 MAX1499 进行全面评估, 请订购完整的 EV 系统 (MAX1499EVC16)。如此前已购买了 Maxim 评估系统中的 68HC16MODULE, 或者是在基于其它 μC 的系统中应用, 请订购评估板 (MAX1499EVKIT)。

该系统也可用来评估 MAX1498ECJ。如需索取免费样品, 请与厂商联系。详细信息请参考 [硬件详细说明](#)。

订购信息

PART	TEMP RANGE	INTERFACE TYPE
MAX1499EVKIT	0°C to +70°C	User supplied
MAX1499EVC16	0°C to +70°C	Windows software

注意: MAX1499 评估软件设计用于完整的评估系统 MAX1499EVC16 (包括 68HC16MODULE-DIP 模块以及 MAX1499EVKIT)。如果不使用 MAX1499 评估软件, 可单独购买 MAX1499EVKIT, 不需要购买 μC 模块。

MAX1499 评估板

MAX1499 评估板是经过验证的 PC 板, 可方便实现对 MAX1499 的评估。但需提供合适的时序信号保证其正常运行。将 6V 至 26V DC 电源以及地线连至 TB1 (参见图 7)。时序要求请参考 MAX1499 的数据资料。

MAX1499 评估系统

MAX1499 评估系统由用户提供 7V DC 至 20V DC 电源。评估软件运行在 PC 机 Windows95/98/2000/XP 操作系统下, 通过串口实现与 EV 系统的接口。参见 [快速入门](#) 一节, 了解安装和操作说明。

MAX1499EVC16 系统

元件列表

PART	QTY	DESCRIPTION
MAX1499EVKIT	1	MAX1499 EV kit
68HC16MODULE-DIP	1	68HC16 μC module

MAX1499 评估板元件列表

DESIGNATION	QTY	DESCRIPTION
AIN+, AIN-, REF+, REF-	4	Noninsulated banana jacks Mouser 530-108-0740-1
C1, C2, C11	3	10 μF $\pm 20\%$, 10V X7R ceramic capacitors (1210) TDK C3225X7R1C106M Taiyo Yuden LMK325BJ106MN
C3, C4, C5, C7-C10, C12	8	0.1 μF $\pm 20\%$, 16V X7R ceramic capacitors (0603) TDK C1608X7R1C104K Taiyo Yuden EMK107BJ104MA

DESIGNATION	QTY	DESCRIPTION
C6	1	4.7 μF $\pm 10\%$, 16V X7R ceramic capacitor (1206) TDK C3216X7R1C475K
CLK	1	BNC 50 Ω PC board vertical mount A/D Electronics 580-002-00
DIG0-DIG4	5	Bicolor seven-segment LED displays, common cathode (DIP-10-0.600in) Kingbright Corporation SBC56-21EGWA
J1	1	2 x 20 right-angle socket

Windows 是 Microsoft Corp. 的注册商标。

评估板: MAX1498/MAX1499



MAX1499 评估板/MAX1499 评估系统

MAX1499 评估板元件列表 (续)

REFERENCE	QTY	DESCRIPTION
JU10-JU14	5	3 pins
JU1-JU9	9	2 pins
R1	1	133k Ω 1% resistor (1206)
R2, R12	2	100k Ω 1% resistors (1206)
R3-R7	5	1k Ω 5% resistors (1206)
R8, R9	0	Do not install—shorted trace on PC board (1206)
R10	1	500k Ω potentiometer
R11	1	24k Ω 5% resistor (1206)
R13, R14	2	10 Ω 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

快速入门

所需设备

在您开始之前，您需要以下设备：

- MAX1499EVC16 (包括 MAX1499EVKIT 板和 68HC16-MODULE-DIP)。
- 0.5A、+7V DC 至 +20V DC 直流电源。

- Windows 95/98/2000/XP 计算机，带有串口 (COM)。
- 9 针 I/O 扩展电缆。

操作过程

在所有连接完成前，不要打开电源。

- 1) 确定 JU1-JU8 和 JU10-JU14 安装了短路器，JU9 开路。参见表 2 的跳线设置。
- 2) 仔细连接电路板，将 MAX1499 评估板的 40 针插头和 68HC16MODULE-DIP 模块的 40 针连接器对齐，把二者轻轻按在一起。两块电路板应彼此贴合。
- 3) 将 +7V DC 至 +20V DC 电源连至 μ C 模块顶部边缘通/断开关旁的接线柱。注意板上标出的极性。
- 4) 用一根电缆连接计算机串口与 μ C 模块。如使用 9 针串口，可采用 9 针直通式孔 - 针电缆。如果串口只有 25 针连接器，则需要标准的 25 针至 9 针转接器。评估板软件检查调制解调器状态连线 (CTS、DSR 和 DCD)，以确保选择了正确的端口。
- 5) 运行 INSTALL.EXE 程序，在您的计算机上安装评估软件。程序文件将被复制，并在 Windows 开始菜单栏产生相关图标。
- 6) 打开电源。
- 7) 点击开始菜单栏中的图标，运行 MAX1499 程序。
- 8) 程序将提示您连接 μ C 模块，并打开电源。将 SW1 拨到 ON 的位置。选择正确的串口，点击 OK。程序自动将软件下载到模块中。

元件供应商

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

注意：同这些供货商联系时，请说明您使用的是 MAX1499。

MAX1499 评估板/MAX1499 评估系统

- 9) 在 AIN+ 和 AIN- 之间接入范围在 -2V 至 +2V 的输入信号。注意屏幕读数。
- 10) 下拉 **View** 菜单，点击 **Graph**，查看测量图表。

软件详细说明

测量

评估软件的 **Measurement** 页模拟数字电压表 (DVM) 功能。状态位大约每秒轮询一次。当 **Data** 状态位为 1 时，ADC 结果寄存器被读出，并显示为 **Analog Input Code**。MAX1499 也在 LED 上显示结果。

评估板并不是一个完整的 DVM。可能需要其它校准和保护电路。

当 **Measurement** 页被激活后，如果 **spi/adc** 和 **seg_sel** 控制位没有清零，则软件将其清零。

数学计算

评估软件可实现物理系统中的几个数学函数功能。当 **Math** 页激活后，如果 **spi/adc** 控制位没有置“1”，则软件将其置位。同样，如果 **seg_sel** 控制位没有清零，则软件将其清零。

评估软件在结果显示前，获取 ADC 结果，当 **Measurement** 或 **Math** 页激活，并且 **spi/adc** 控制位置“1”时，软件计算新的 LED 显示值。数学计算结果做为通道 1 的数据进行显示，原始 ADC 结果则做为通道 0 的数据。

Type K Thermocouple 功能与适当的冷端连接可将 K 型热电偶测得的塞贝克电压转换为摄氏温度。**a0** 系数 230 代表冷端温度为 +23°C。

控制寄存器

Control Register 页可实现对全部控制寄存器位的访问。下拉相应组合框，单击 **write**。

限制寄存器、ADC 失调、ADC 结果、LED 显示和峰值

Results、Displays、Limits 页提供对二的补码数据寄存器的访问。除只读的 **ADC RESULT1**、**ADC RESULT2** 和 **PEAK RESULT** 之外，每个寄存器还具有 **read** 和 **write** 按钮。

不论 **seg_sel** 控制寄存器如何设置，读取 **ADC RESULT1** 或 **ADC RESULT2** 寄存器会自动刷新 LED 显示。

不论 **offset_cal1** 控制寄存器如何设置，写入 **ADC OFFSET** 寄存器将会影响 **ADC RESULT1** 和 **ADC RESULT2**。

LED 段寄存器

LED Segments 页使用户能够通过鼠标点击实现对每个 LED 段的开和关。

LED Segments 激活后，如果 **seg_sel** 控制位没有置位，则软件将其置位。

Write LED Text 按钮将文本字符串转换为相近的七段字符，然后将这些字符图案写入 LED 显示。

图表

评估软件的图表数据有两个选项。近期数据可通过选择 **View** 菜单中的 **Graph** 显示。数据可以显示为时序曲线图、柱状图或者是原始数据表。点击主窗口的 **Collect Samples** 按钮，激活采样工具来控制数据尺寸和时序。

采样数据可以保存到用逗号或制表符分割的文件中。可以加入行标号和说明标题行。

通道 0 显示 16 位 ADC 原始结果数据。数学计算有效后，通道 1 显示 LED 数据。如扩展精度有效，通道 2 显示 20 位 ADC 原始结果。

诊断窗口

诊断窗口用于评估板出厂前的测试，不面向用户使用。














硬件详细说明

被测试的 MAX1499 (U1) 是集成了 LED 显示驱动器的低功耗 4.5 位 ADC。MAX6062 (U5) 提供板上 2.048V 基准电压。参见图 7 的 MAX1499 评估板原理图和 MAX1499 数据资料。

评估板包括 MAX1659 大电流 5V 线性稳压器 (U2) 和一组 MAX1840/MAX1841 电平转换器 (U3 和 U4) 以支持 3V 逻辑。

MAX1499 评估板/MAX1499 评估系统

表 1. 图形工具按钮

TOOL	FUNCTION
	Show the entire available input range.
	Expand the graph data to fill the window.
	Move the view left or right.
	Move the view up or down.
	Expand or contract the x-axis.
	Expand or contract the y-axis.
	Load data from a file.
	Save data to a file.
	Option to write a header line when saving data.
	Option to write line numbers when saving data.
	View code vs. time plot.
	View histogram plot (cumulative frequency of each code).
	View table.
<input type="text" value="Min"/>	Show minimum in tabular view.
<input type="text" value="Max"/>	Show maximum in tabular view.
<input type="text" value="Span"/>	Show span in tabular view. Span = maximum - minimum.
<input type="text" value="N"/>	Show number of samples in tabular view.

TOOL	FUNCTION
<input type="text" value="Sum(x)"/>	Show sum of the samples in tabular view.
<input type="text" value="Sum(x*x)"/>	Show sum of the squares of the samples in tabular view.
<input type="text" value="Mean"/>	Show arithmetic mean in tabular view: $\text{Mean} = \frac{\sum(x)}{n}$
<input type="text" value="StdDev"/>	Show standard deviation in tabular view: $\text{Standard deviation} = \sqrt{\frac{n\sum(x^2) - \left(\sum x\right)^2}{(n-1)n}}$
<input type="text" value="Rms"/>	Show root of the mean of the squares (RMS) in tabular view: $\text{RMS} = \sqrt{\frac{\sum(x^2)}{n}}$
<input type="checkbox" value="0"/>	Channel 0 enable (16-bit ADC result).
<input type="checkbox" value="1"/>	Channel 1 enable (math result).
<input type="checkbox" value="2"/>	Channel 2 enable (20-bit ADC result).

MAX1499 评估板/MAX1499 评估系统

表 2. 跳线功能表

JUMPER	SHUNT POSITION	FUNCTION
JU1	Closed*	LED displays are powered by U2.
	Open	VLED must be supplied by an external power source.
JU2	Closed*	VDISP = GND.
	Open	Apply VDISP voltage at VDISP pad.
JU3	Closed*	Banana jack AIN+ connects to AIN+ input pin.
	Open	Insert custom filtering between JU3 pins 1 and 2.
JU4	Closed*	Banana jack AIN- connects to AIN- input pin.
	Open	Insert custom filtering between JU4 pins 1 and 2.
JU5	Closed*	REF- = GND.
	Open	REF- must be provided by user.
JU6	Closed*	REF+ = 2.048V from U5, MAX6062.
	Open	REF+ must be provided by user.
JU7	Closed*	REF+ is bypassed by C6.
	Open	C6 is disconnected.
JU8	Closed*	GLLED return current flows to DVDD.
	Open	GLLED return current flows to external power source.
JU9	Closed	LED_EN = low; LED displays are blanked.
	Open*	LED_EN = high; LED displays are enabled.
JU10– JU14	1-2*	Display color = red.
	2-3	Display color = green.

* 星号为默认设置。

表 3. 独立接口引脚功能

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

评估 MAX1498

MAX1499EVKIT 支持 MAX1498；但由于没有 μC 接口，因此不能使用评估软件。

MAX1498 是独立于 MAX1499 之外的另一版本。请参考 MAX1498/MAX1499 数据资料。可申请 MAX1498ECJ 免费样片。

- 1) MAX1499EVKIT 必须与 68HC16MODULE 断开。
- 2) 断开电源后，以 MAX1498 替代 U1。
- 3) 在 TB1 上连接直流电源。
- 4) 打开电源。LED 应开始显示测量数据。

以 MAX1498 替代 U1 后，一些引脚功能发生变化，如表 3 所示。

代码实例

列表 1 是评估板软件变量的说明。列表 2 是评估板软件使用的函数。

MAX1499 评估板/MAX1499 评估系统

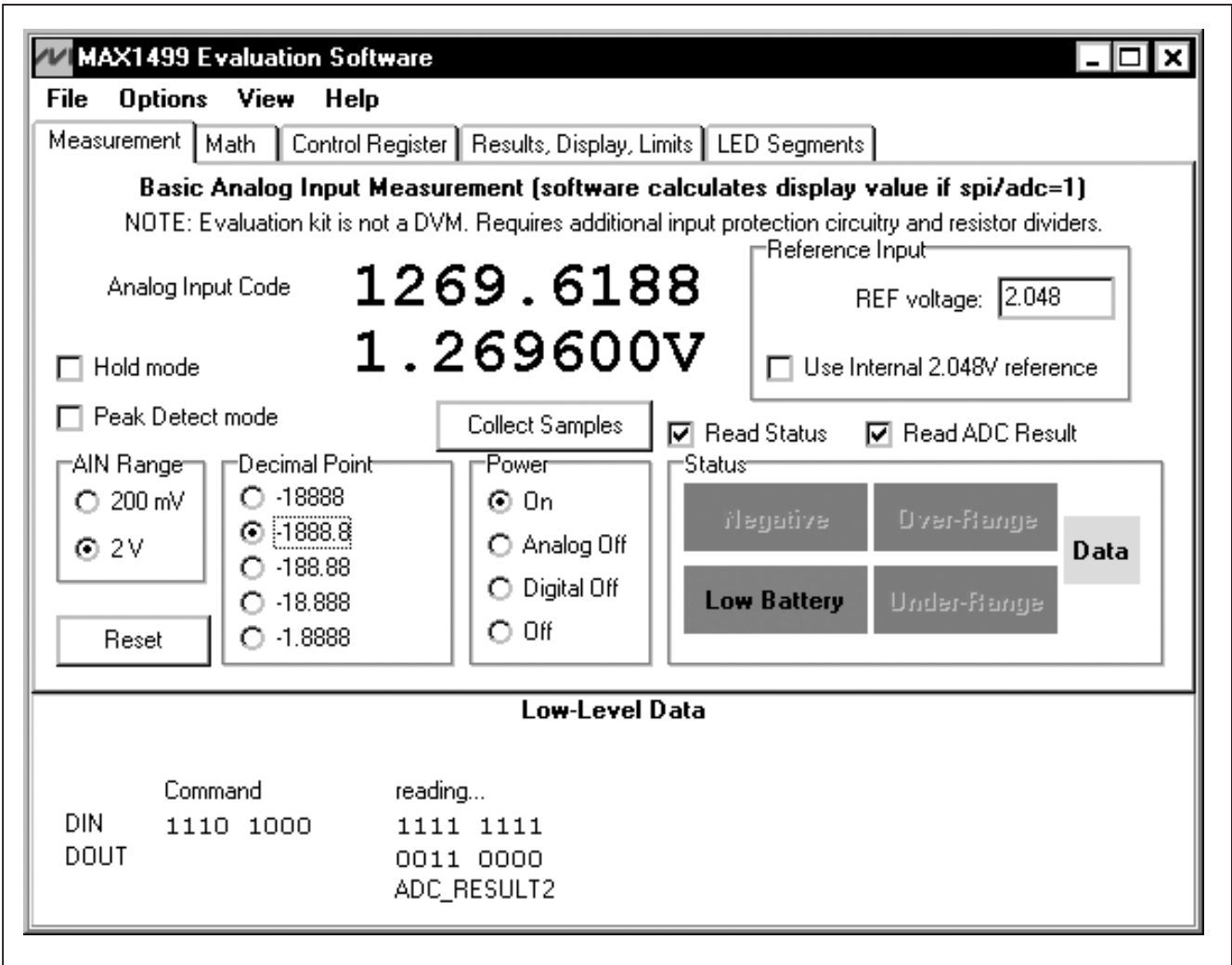


图 1. MAX1499 评估软件 — Measurement 页

MAX1499 评估板/MAX1499 评估系统

评估板: MAX1498/MAX1499

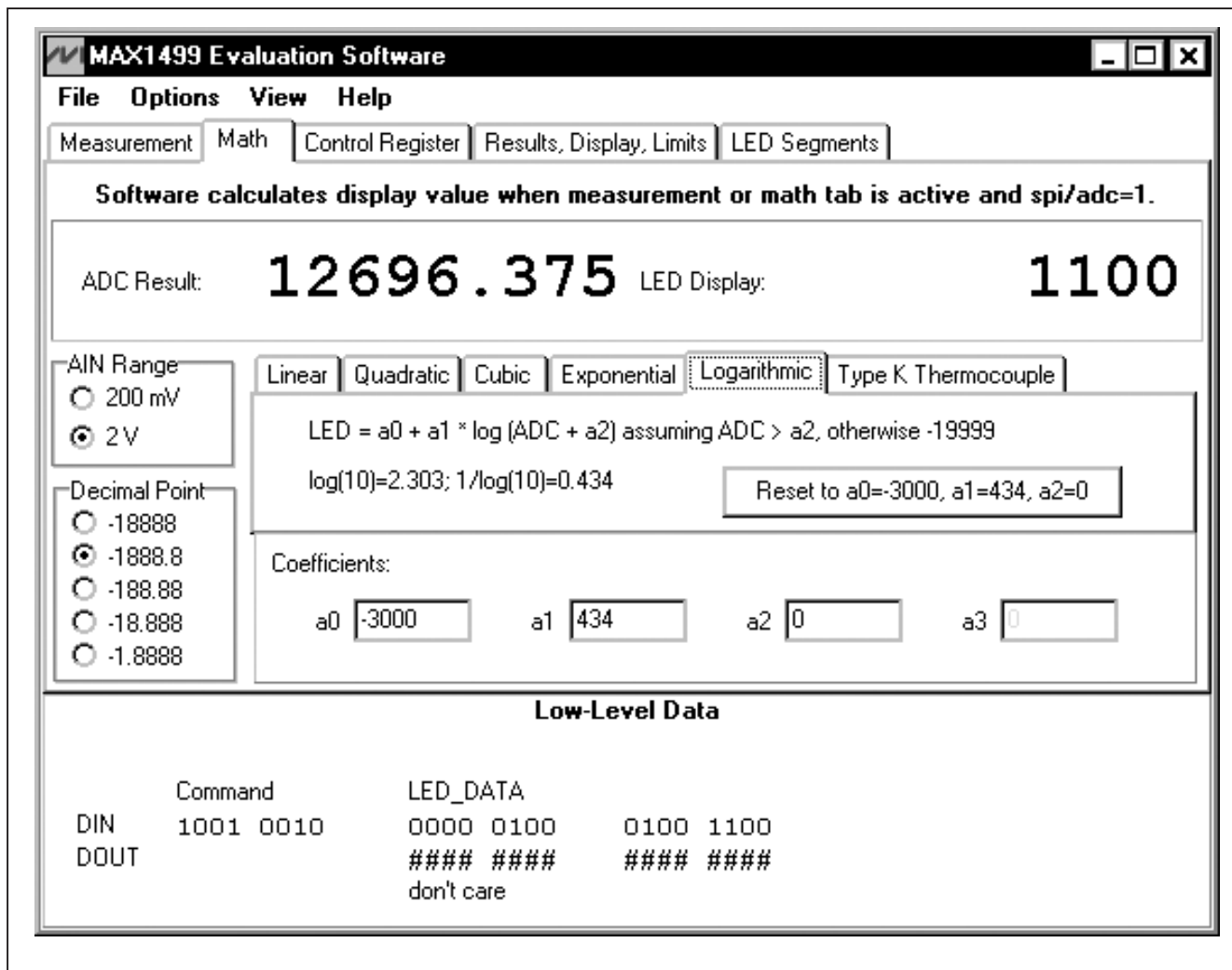


图 2. MAX1499 评估软件 — Math 页

MAX1499 评估板/MAX1499 评估系统

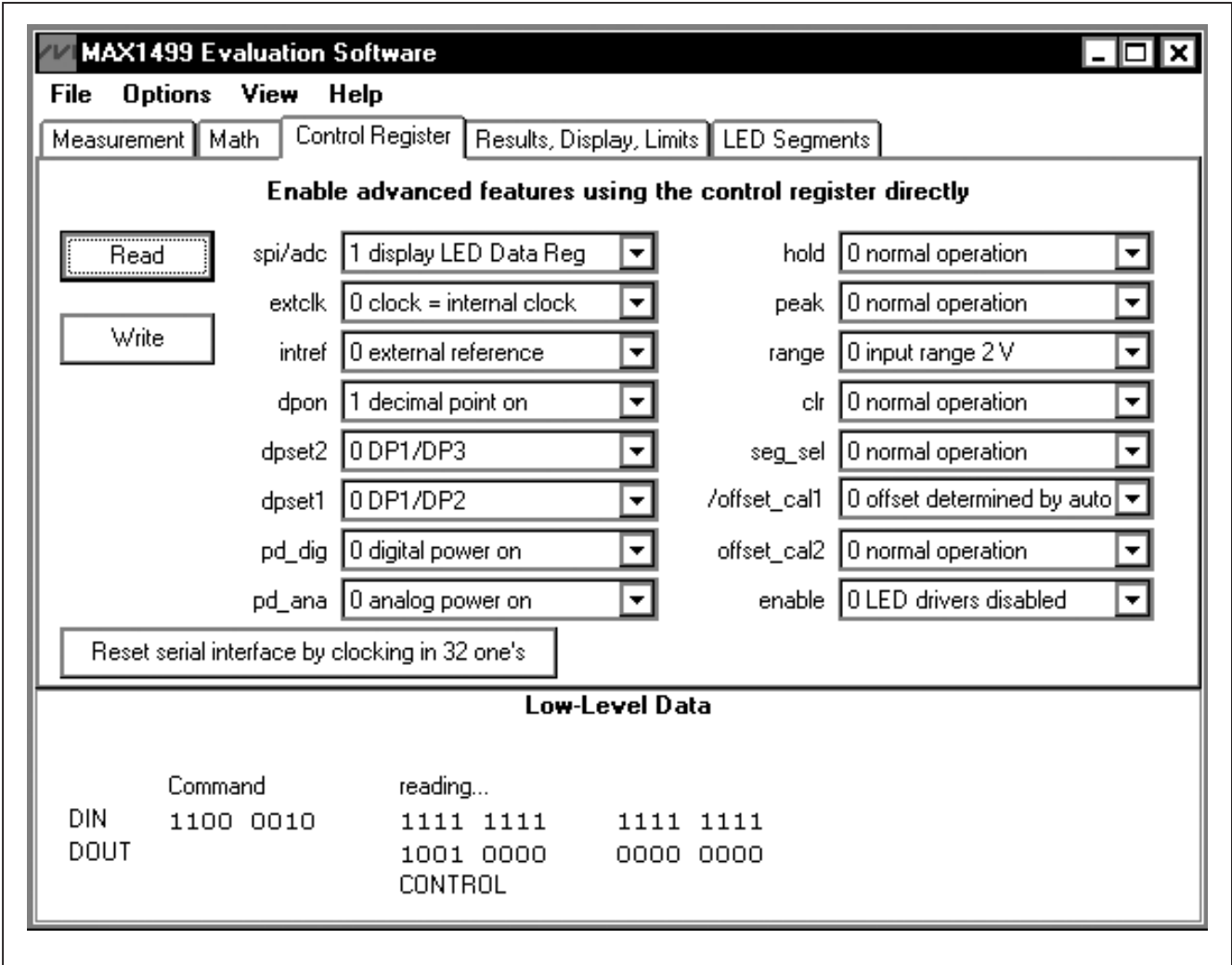


图 3. MAX1499 评估软件 — Control Register 页

MAX1499 评估板/MAX1499 评估系统

评估板: MAX1498/MAX1499

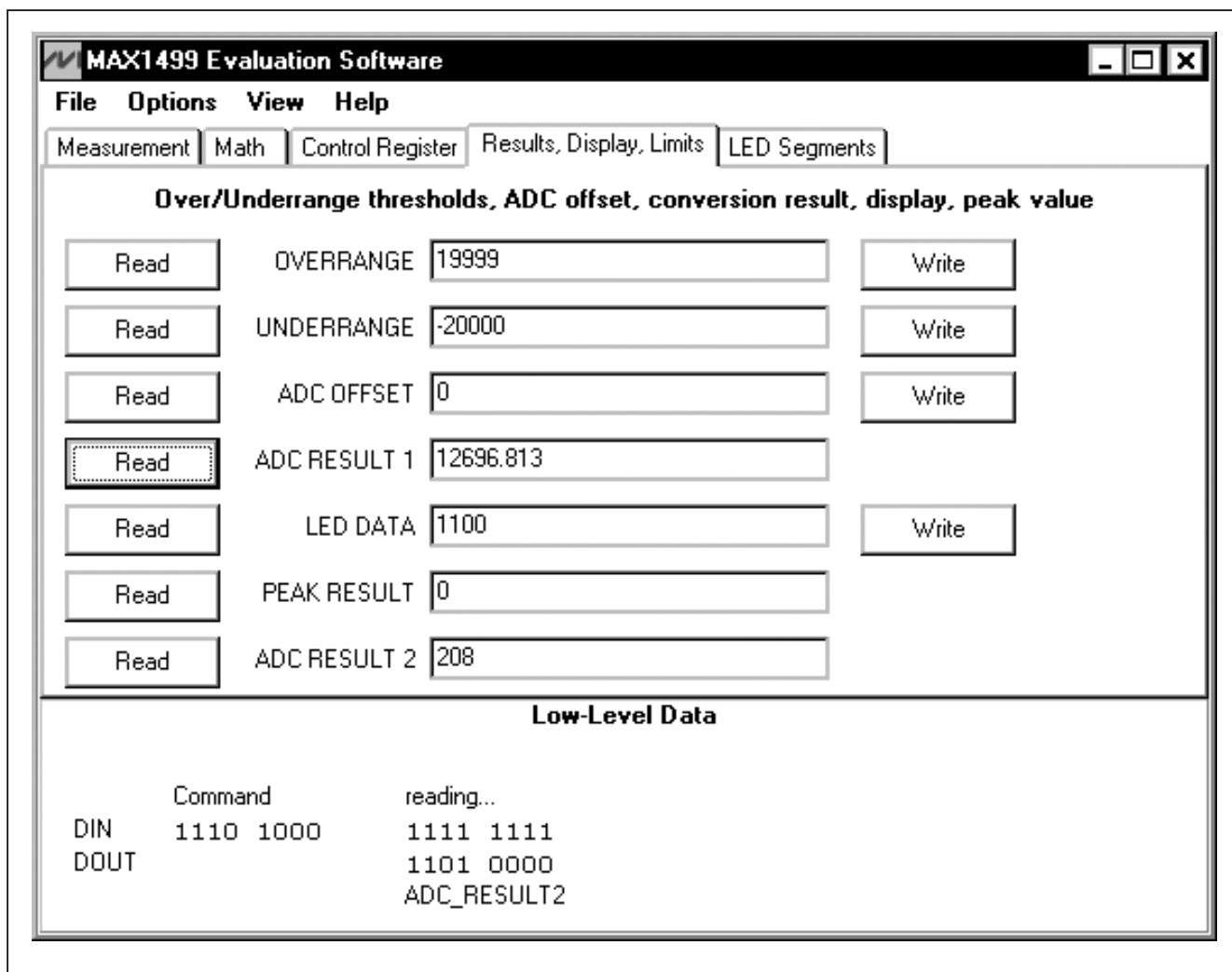


图 4. MAX1499 评估软件 — Results, Display, Limits 页

MAX1499 评估板/MAX1499 评估系统

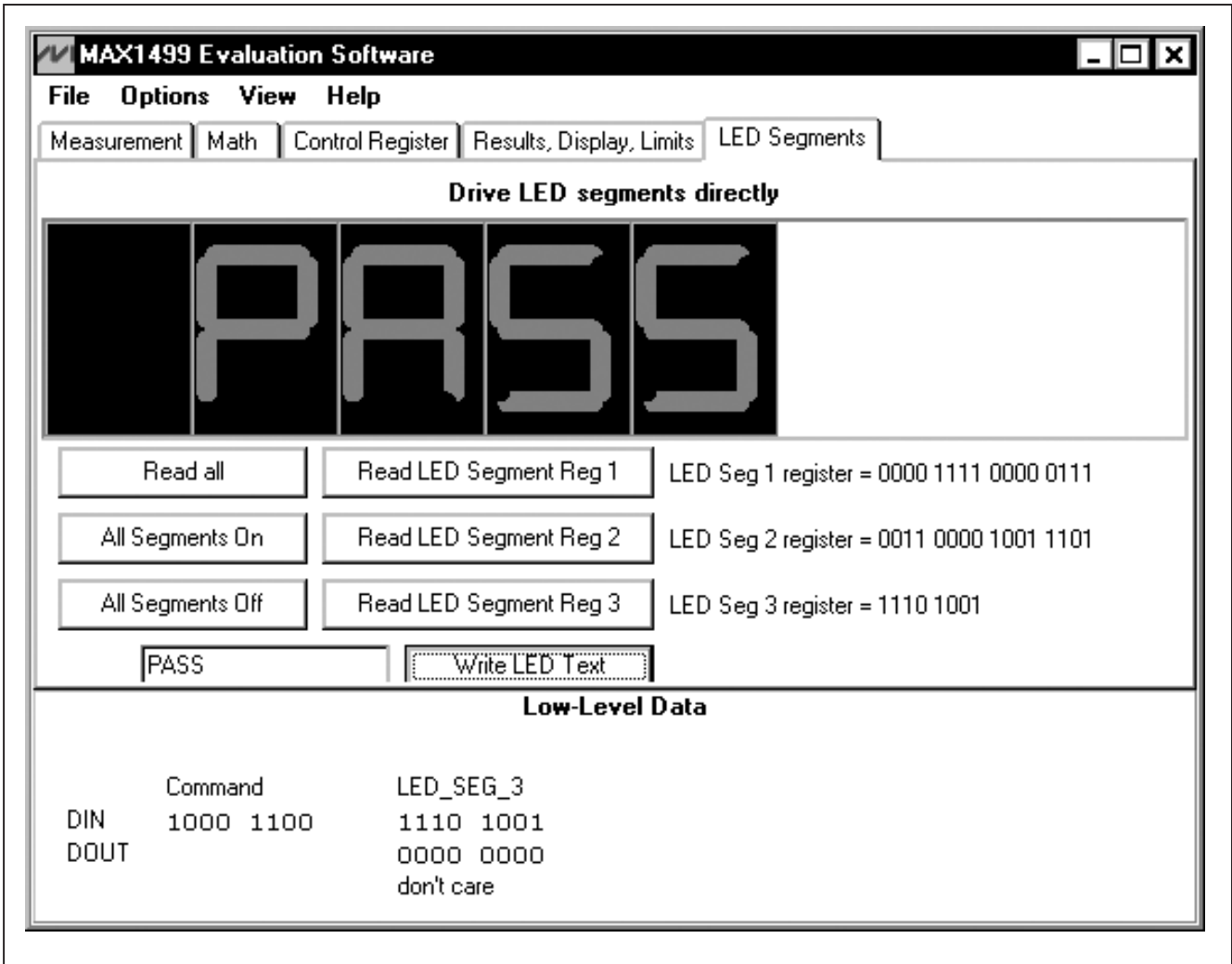


图 5. MAX1499 评估软件 — LED Segments 页

MAX1499 评估板/MAX1499 评估系统

评估板: MAX1498/MAX1499

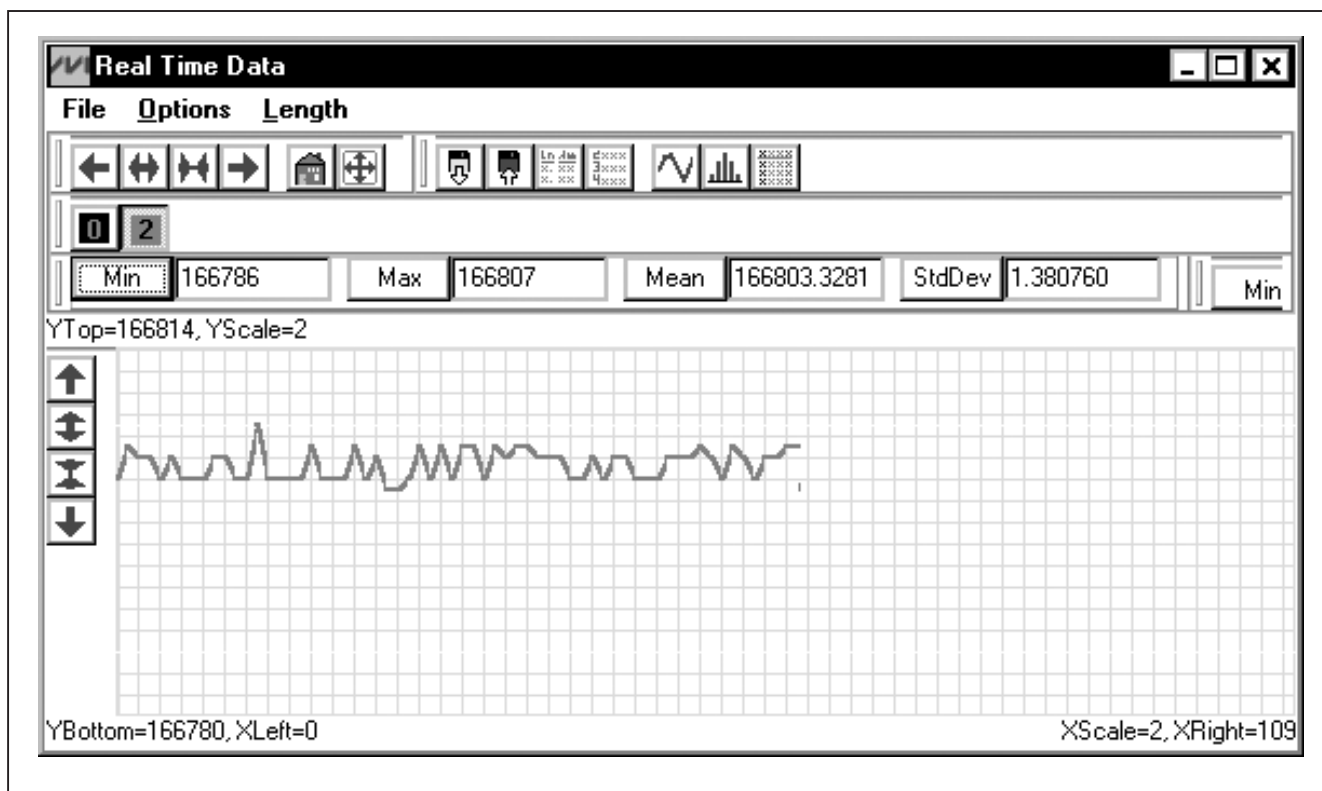


图 6. MAX1499 评估软件 — 图表

MAX1499 评估板/MAX1499 评估系统

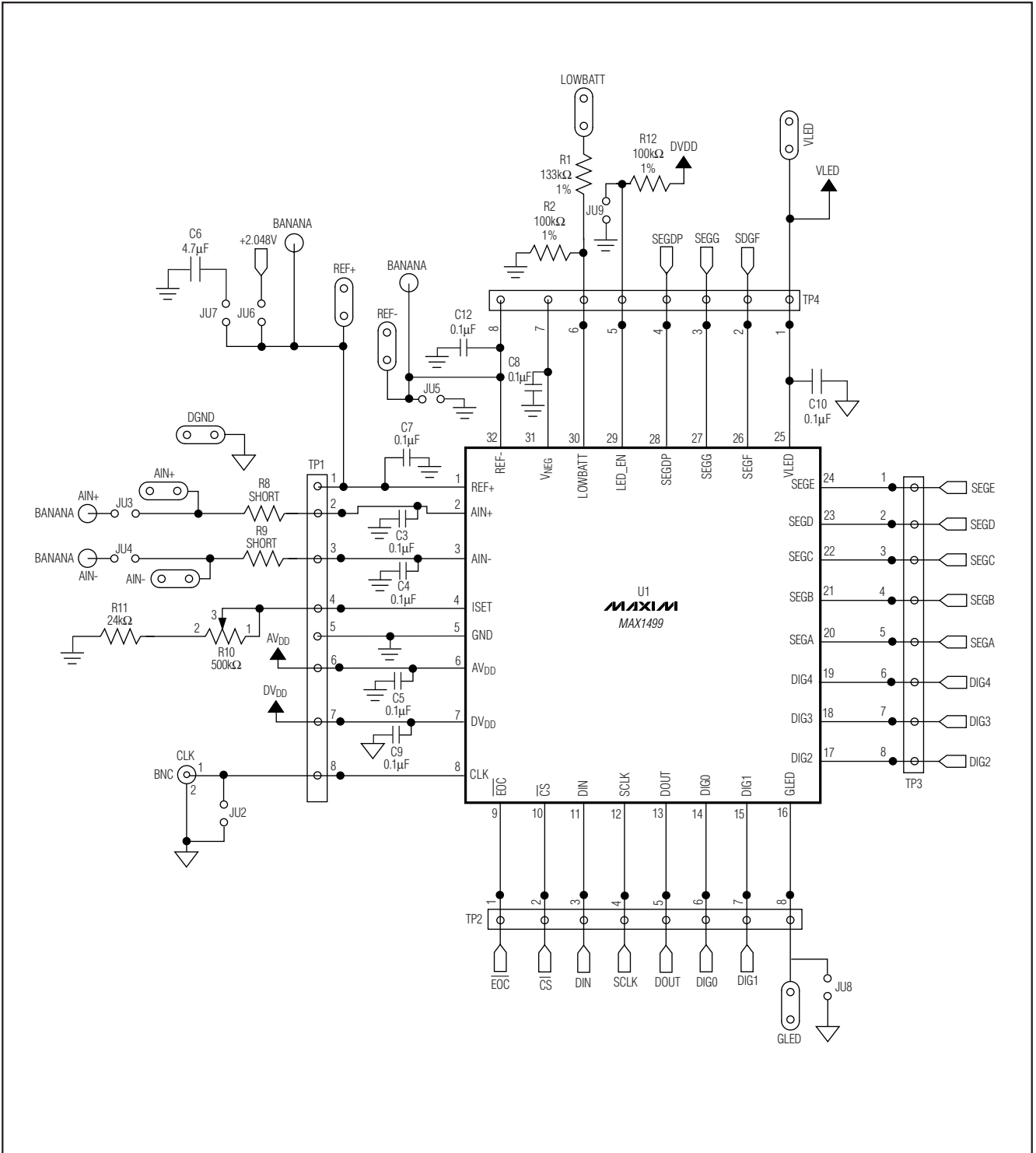


图 7a. MAX1499 评估板原理图 (图 2 之 1)

MAX1499 评估板/MAX1499 评估系统

评估板: MAX1498/MAX1499

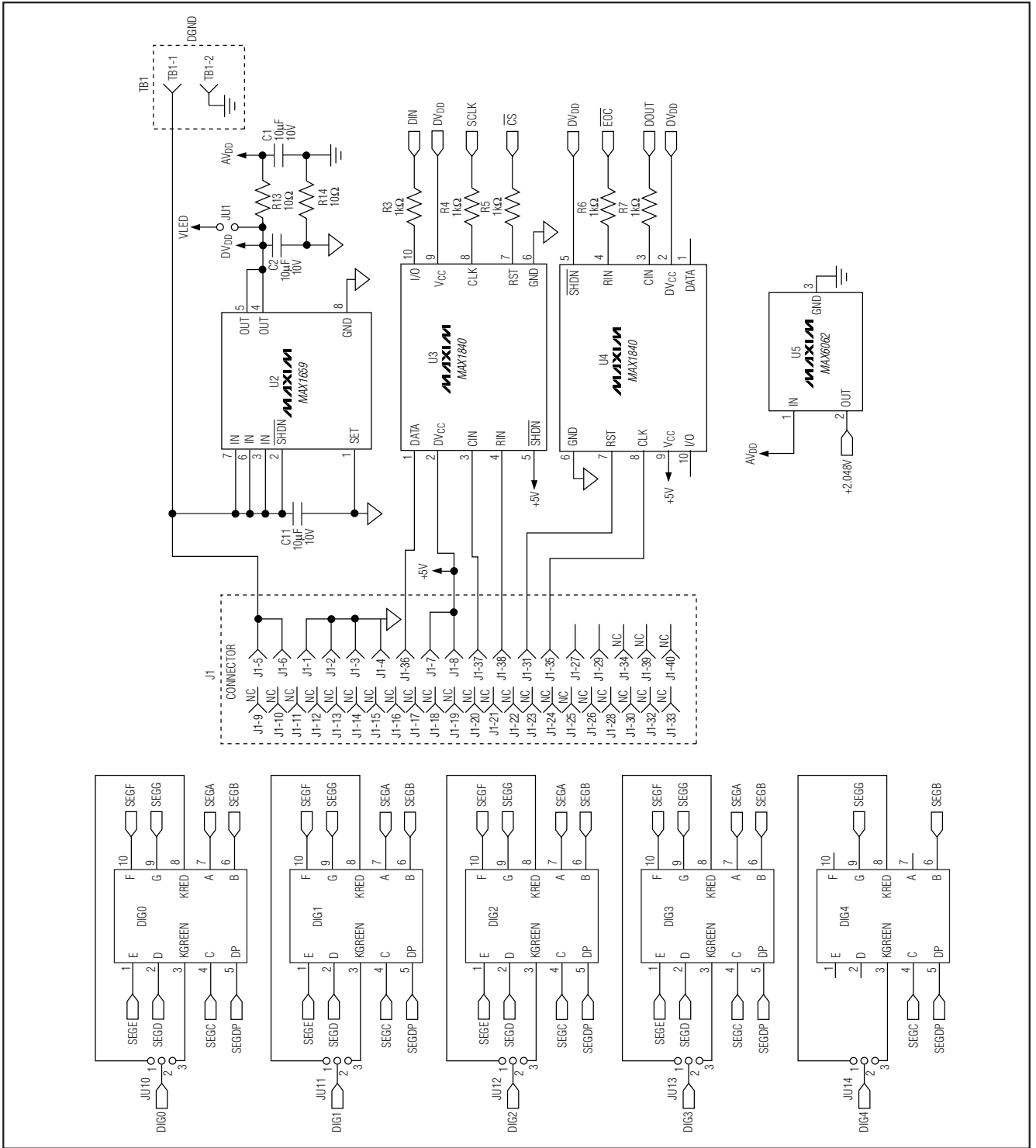


图 7b. MAX1499 评估板原理图 (图 2 之 2)

MAX1499 评估板/MAX1499 评估系统

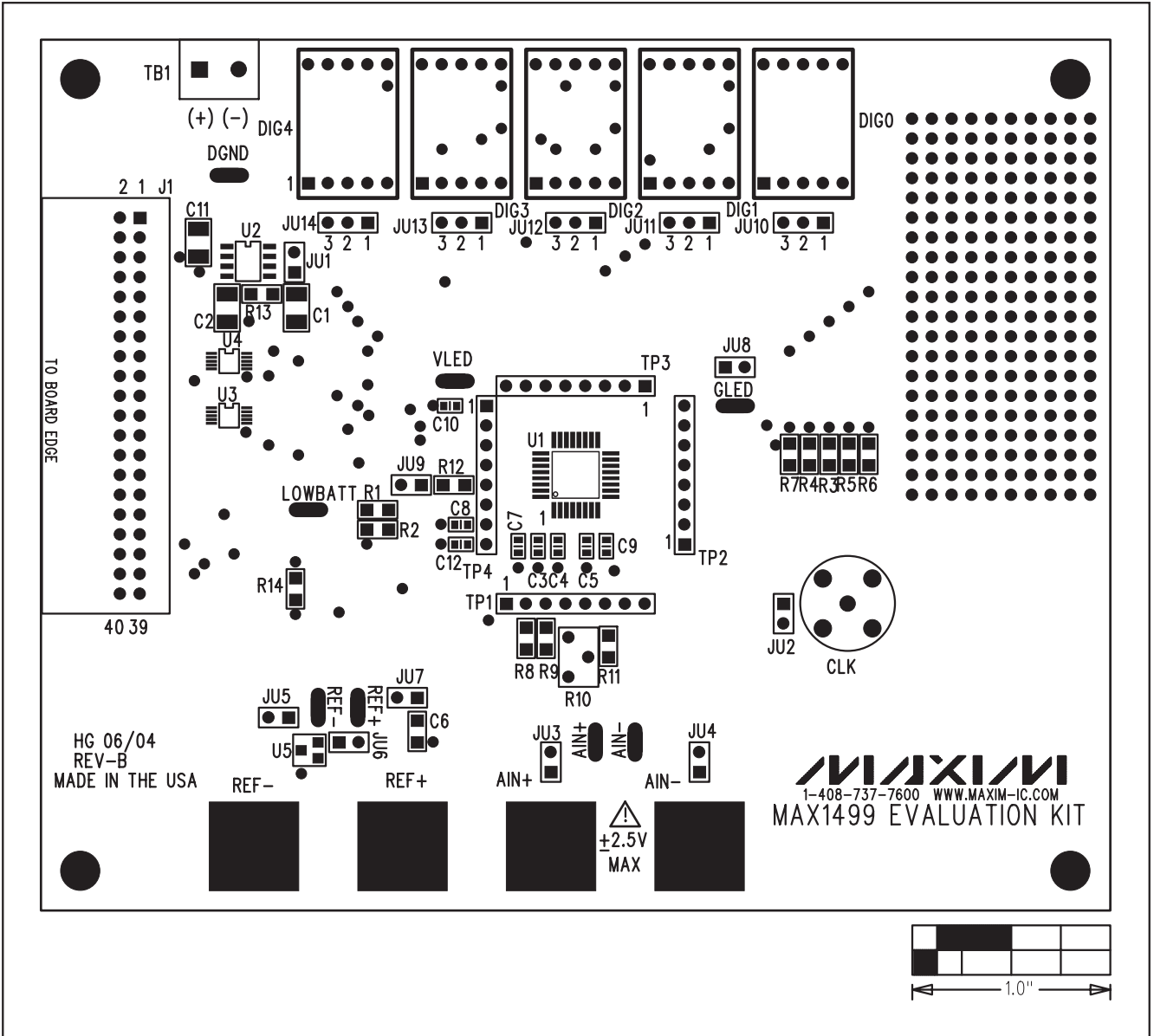


图 8. MAX1499 评估板元件布置指南——元件面

MAX1499 评估板/MAX1499 评估系统

评估板：MAX1498/MAX1499

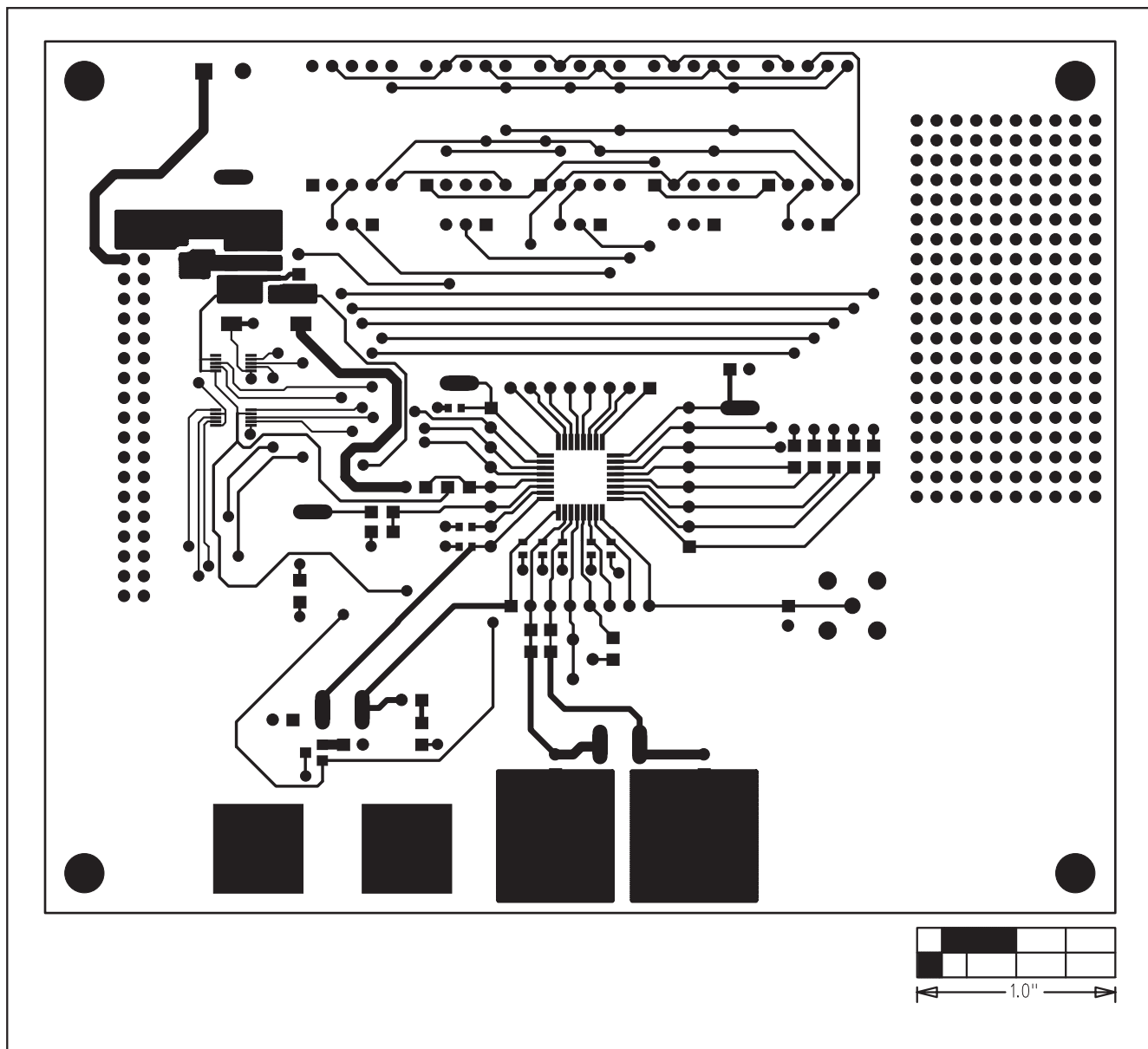


图9. MAX1499 评估板PC 板布板 —— 元件面

MAX1499 评估板/MAX1499 评估系统

评估板: MAX1498/MAX1499

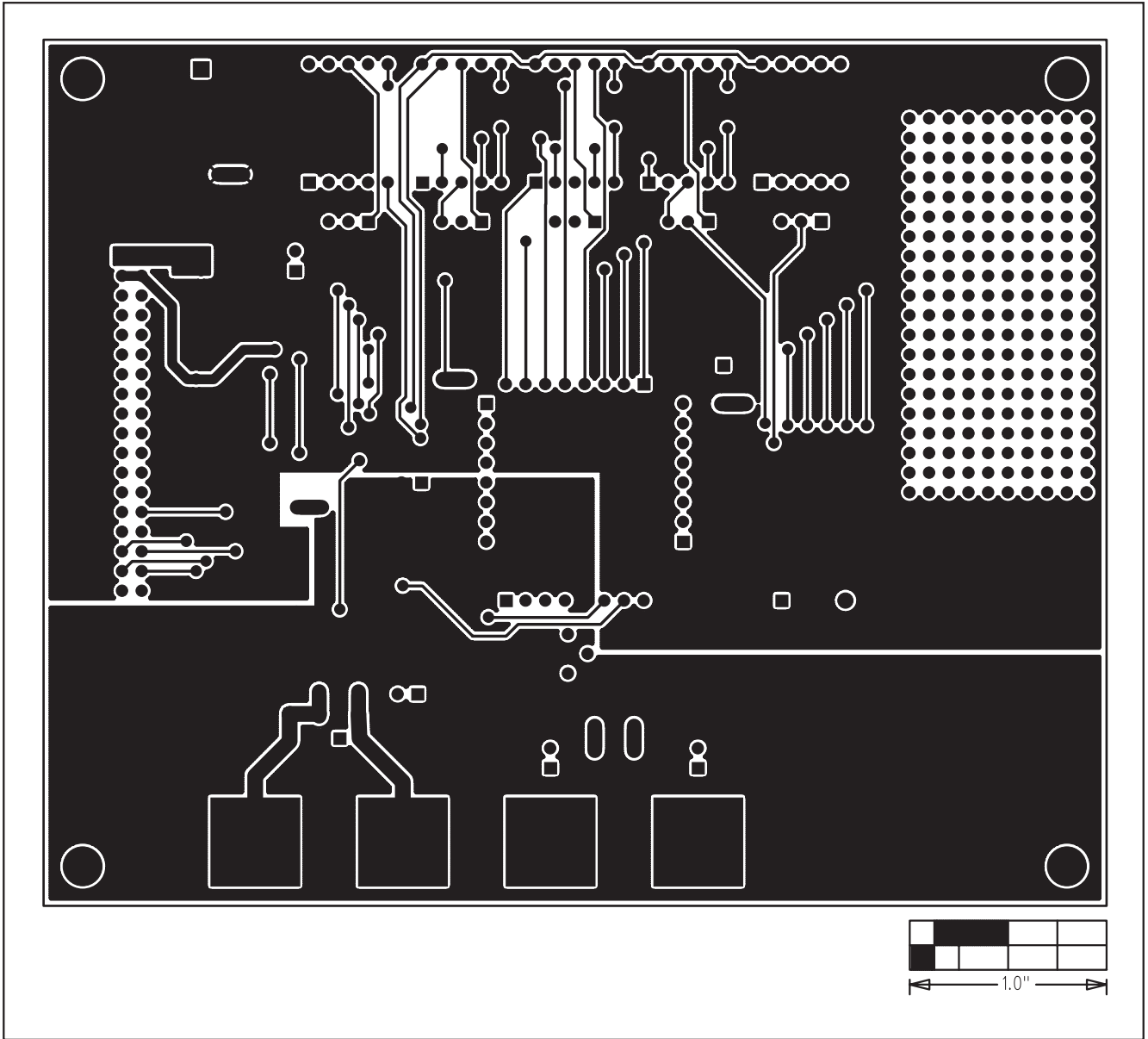


图 10. MAX1499 评估板PC 板布板 —— 焊接面

MAX1499 评估板/MAX1499 评估系统

评估板: MAX1498/MAX1499

```
// Drv1499.h
// MAX1499-specific driver.
// mku 01/07/2004
// (C) 2004 Maxim Integrated Products
// For use with Borland C++ Builder 3.0
//-----
// Revision history:
// 01/07/2004: modify drv 1 4 9 4 driver to become drv1499
// 12/04/2003: fix indentation
// 09/15/2003: add double Voltage(void)
// 09/12/2003: add SPI_Transfer_After_EOC()
// 09/09/2003: add class MAX1499 dependent on external SPI_Interface()
// 08/13/2003: preliminary draft of reusable code
//-----
#ifdef drv1499H
#define drv1499H
//-----

//-----
// The following interface protocols must be provided by
// the appropriate low-level interface code.
//

/* SPI interface:
** byte_count = transfer length
** mosi[] = array of master-out, slave-in data bytes
** miso_buf[] = receive buffer for master-in, slave-out data bytes
*/
extern bool SPI_Transfer(int byte_count,
    const unsigned __int8 mosi[], unsigned __int8 miso_buf[]);

/* SPI interface, with data transfer immediately after EOC is asserted:
** byte_count = transfer length
** mosi[] = array of master-out, slave-in data bytes
** miso_buf[] = receive buffer for master-in, slave-out data bytes
*/
extern bool SPI_Transfer_After_EOC(int byte_count,
    const unsigned __int8 mosi[], unsigned __int8 miso_buf[]);

//-----
// Define the bits in the COMMS register.
// START R/W RS4 RS3 RS2 RS1 RS0 0
#define MAX1499_COMMS_START 0x80
#define MAX1499_COMMS_RW_MASK 0x40
#define MAX1499_COMMS_RW_WRITE 0x00
#define MAX1499_COMMS_RW_READ 0x40
#define MAX1499_COMMS_RS_MASK 0x3E
#define MAX1499_COMMS_RS_00000 0x00
#define MAX1499_COMMS_RS_STATUS 0x00
#define MAX1499_COMMS_RS_00001 0x02
#define MAX1499_COMMS_RS_CONTROL 0x02
#define MAX1499_COMMS_RS_00010 0x04
#define MAX1499_COMMS_RS_OVERRANGE 0x04
#define MAX1499_COMMS_RS_00011 0x06
#define MAX1499_COMMS_RS_UNDERRANGE 0x06
#define MAX1499_COMMS_RS_00100 0x08
#define MAX1499_COMMS_RS_LED_SEG_1 0x08
#define MAX1499_COMMS_RS_00101 0x0A
#define MAX1499_COMMS_RS_LED_SEG_2 0x0A
#define MAX1499_COMMS_RS_00110 0x0C
#define MAX1499_COMMS_RS_LED_SEG_3 0x0C
#define MAX1499_COMMS_RS_00111 0x0E
#define MAX1499_COMMS_RS_ADC_OFFSET 0x0E
#define MAX1499_COMMS_RS_01000 0x10
#define MAX1499_COMMS_RS_ADC_RESULT1 0x10
#define MAX1499_COMMS_RS_01001 0x12
#define MAX1499_COMMS_RS_LED_DATA 0x12
#define MAX1499_COMMS_RS_01010 0x14
#define MAX1499_COMMS_RS_PEAK 0x14
#define MAX1499_COMMS_RS_10100 0x28
#define MAX1499_COMMS_RS_ADC_RESULT2 0x28

//-----
// Define the bits in the STATUS register.
// POL OVR_RNG UNDR_RNG LOW_BATT ADD(data available) 0 0 0
```

列表 1 (表 4 之 1)

MAX1499 评估板/MAX1499 评估系统

```

#define MAX1499_STATUS_POL_MASK          0x80
#define MAX1499_STATUS_POL_POSITIVE     0x00
#define MAX1499_STATUS_POL_NEGATIVE     0x80
#define MAX1499_STATUS_OVER_RANGE       0x40
#define MAX1499_STATUS_UNDER_RANGE      0x20
#define MAX1499_STATUS_LOW_BATTERY      0x10
#define MAX1499_STATUS_DATA_READY       0x08

//-----
// Define the bits in the CONTROL register.
// SPI_ADC EXTCLK INTREF DPON DPSET2 DPSET1 PD_DIG PD_ANA
// HOLD PEAK RANGE CLR LED OFFSET_CAL1 OFFSET_CAL2 ENABLE
#define MAX1499_CONTROL_SPI_ADC          0x8000
#define MAX1499_CONTROL_EXTCLK           0x4000
#define MAX1499_CONTROL_INTREF           0x2000
#define MAX1499_CONTROL_DPMASK          0x1C00
#define MAX1499_CONTROL_DPON             0x1000
#define MAX1499_CONTROL_DPSET2           0x0800
#define MAX1499_CONTROL_DPSET1           0x0400
// (DPSET2 is the LSB and DPSET1 is the MSB)
#define MAX1499_CONTROL_DP1ON            0x1000 /* -1888.8 */
#define MAX1499_CONTROL_DP2ON            0x1800 /* -188.88 */
#define MAX1499_CONTROL_DP3ON            0x1400 /* -18.888 */
#define MAX1499_CONTROL_DP4ON            0x1C00 /* -1.8888 */
#define MAX1499_CONTROL_PD_DIG           0x0200
#define MAX1499_CONTROL_PD_ANA           0x0100
#define MAX1499_CONTROL_PD_ALL           0x0300
#define MAX1499_CONTROL_HOLD              0x0080
#define MAX1499_CONTROL_PEAK              0x0040
#define MAX1499_CONTROL_RANGE_200mV      0x0020
#define MAX1499_CONTROL_CLR               0x0010
#define MAX1499_CONTROL_SEG_SEL           0x0008
#define MAX1499_CONTROL_OFFSET_CAL1       0x0004
#define MAX1499_CONTROL_OFFSET_CAL2       0x0002
#define MAX1499_CONTROL_ENABLE            0x0001

//-----
// Define the bits in the LED SEGMENT 1 register.
// A2 G2 D2 F2 E2 DP2 0 B1
// C1 A1 G1 D1 F1 E1 DP1 0
//
#define MAX1499_LED_SEG1_A1               0x8000
#define MAX1499_LED_SEG1_G1               0x4000
#define MAX1499_LED_SEG1_D1               0x2000
#define MAX1499_LED_SEG1_F1               0x1000
#define MAX1499_LED_SEG1_E1               0x0800
#define MAX1499_LED_SEG1_DP2              0x0400
#define MAX1499_LED_SEG1_B0               0x0100
#define MAX1499_LED_SEG1_C0               0x0080
#define MAX1499_LED_SEG1_A0               0x0040
#define MAX1499_LED_SEG1_G0               0x0020
#define MAX1499_LED_SEG1_D0               0x0010
#define MAX1499_LED_SEG1_F0               0x0008
#define MAX1499_LED_SEG1_E0               0x0004
#define MAX1499_LED_SEG1_DP1              0x0002

//-----
// Define the bits in the LED SEGMENT 2 register.
// F4 E4 DP4 G4 B3 C3 A3 G3
// D3 F3 E3 DP3 0 B2 C2 0
//
#define MAX1499_LED_SEG2_F3               0x8000
#define MAX1499_LED_SEG2_E3               0x4000
#define MAX1499_LED_SEG2_DP4              0x2000
#define MAX1499_LED_SEG2_MINUS             0x1000 /* TODO: is this documented? minus sign */
#define MAX1499_LED_SEG2_B2               0x0800
#define MAX1499_LED_SEG2_C2               0x0400
#define MAX1499_LED_SEG2_A2               0x0200
#define MAX1499_LED_SEG2_G2               0x0100
#define MAX1499_LED_SEG2_D2               0x0080
#define MAX1499_LED_SEG2_F2               0x0040
#define MAX1499_LED_SEG2_E2               0x0020

```

列表 1 (表 4 之 2)

MAX1499 评估板/MAX1499 评估系统

评估板: MAX1498/MAX1499

```
#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      0x10
#define MAX1499_LED_SEG3_C3      0x08
#define MAX1499_LED_SEG3_A3      0x04
#define MAX1499_LED_SEG3_G3      0x02
#define MAX1499_LED_SEG3_D3      0x01

//-----
class MAX1499
{
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_OVERRANGE,
        RS_UNDERRANGE   = 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_RESULT1,
        RS_LED_DATA     = MAX1499_COMMS_RS_LED_DATA,
        RS_PEAK         = MAX1499_COMMS_RS_PEAK,
        RS_ADC_RESULT2  = MAX1499_COMMS_RS_ADC_RESULT2
    };

    // 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
    // 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);
};
```

列表 1 (表 4 之 3)

MAX1499 评估板/MAX1499 评估系统

```
//-----  
// Other registers, having 8 bit data format  
bool Write_8bit_reg(int reg, int data);  
int Read_8bit_reg(int reg);  
  
};  
  
//-----  
#endif
```

MAX1499 评估板/MAX1499 评估系统

评估板: MAX1498/MAX1499

```
// 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 reusable code

#include "drv1499.h"

//-----
MAX1499::MAX1499(void)
{
    vref = 2.048;
    extended_resolution = false;
}
//-----
bool MAX1499::Read_STATUS(void)
{
    const unsigned __int8 mosi[] = {
        (unsigned __int8)(MAX1499_COMMS_START |
            MAX1499_COMMS_RW_READ | MAX1499_COMMS_RS_STATUS),
        (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];
        STATUS_REG = data;           // remember the value we just received
    }
    return result;
}
//-----
bool MAX1499::Write_CONTROL(int data)
{
    data = data & 0xFFFF;           // validate the data
    const 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)(0xFF),
        (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] * 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[] = {
```

列表 2 (表 4 之 1)

MAX1499 评估板/MAX1499 评估系统

```

        (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)];
    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)
{
    const unsigned __int8 mosi[] = {
        (unsigned __int8) (MAX1499_COMMS_START |
            MAX1499_COMMS_RW_READ | MAX1499_COMMS_RS_ADC_RESULT2),
        (unsigned __int8) (0xFF)
    };
    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) (MAX1499_COMMS_START |
                MAX1499_COMMS_RW_READ | MAX1499_COMMS_RS_ADC_RESULT2),
            (unsigned __int8) (0xFF)
        };
        unsigned __int8 miso_buf[sizeof(mosi)];
        if (SPI_Transfer(sizeof(mosi), mosi, miso_buf) == false) {
            return 0; // failure
        }
        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;
    }
}

```

列表 2 (表 4 之 2)

MAX1499 评估板/MAX1499 评估系统

评估板: MAX1498/MAX1499

```
    }
    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)
{
    // Write one of the 2's complement registers
    reg = (reg & MAX1499_COMMS_RS_MASK);
    data = data & 0xFFFF; // 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)
{
    // 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;
    }
    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 & 0xFF)
    };
    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);
```

列表 2 (表 4 之 3)

MAX1499 评估板/MAX1499 评估系统

```
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;
}
//-----
```

列表 2 (表 4 之 4)

Maxim 不对 Maxim 产品以外的任何电路使用负责, 也不提供其专利许可。Maxim 保留在任何时间、没有任何通报的前提下修改产品资料和规格的权利。