19-3145; Rev 0; 1/04

MAXIM

MAX1494评估板/评估系统

概述

MAX1494评估系统 (EV system) 由MAX1494评估板 (EV kit) 和Maxim 68HC16MODULE-DIP微控制器 (µC) 模块组成。MAX1494是集成了液晶显示 (LCD) 驱动器的低功耗、4位半模数转换器 (ADC)。评估软件可运行在Windows® 95/98/2000/XP操作系统下，为评估MAX1494的功能提供了方便的用户界面。

要在个人计算机上对MAX1494进行全面评估，请定购完整的评估系统 (MAX1494EVC16)。如果此前已经购买了Maxim评估系统中的68HC16MODULE-DIP模块，或是应用于其它µC系统，只需定购评估板(MAX1494EVKIT)。

该系统还可用来评估MAX1493CCJ和MAX1495CCJ。请联系厂商索取这些产品的免费样品。详细信息请参考*硬件详细说明*部分。

MAX1494评估板

MAX1494评估板提供经过验证的PC板布局，便于对MAX1494进行评估。为保证其正常工作，必须将评估板与正确的时序信号连接。将6V至26VDC电源以及地线连接到接线板TB1 (参见图7)。时序要求请参考MAX1494的数据资料。

MAX1494评估系统

MAX1494评估系统工作在用户提供的7VDC至20VDC电源下。评估软件在基于Windows 95/98/2000/XP操作系统的PC上运行，并通过计算机串行通信接口与评估系统电路板相连。设置及操作说明请参考*快速入门*部分。

特性

- ◆ 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
- ◆ 经过完全组装和测试

定购信息

PART	TEMP RANGE	INTERFACE TYPE
MAX1494EVKIT	0°C to +70°C	User supplied
MAX1494EVC16	0°C to +70°C	Windows software

***注意**：MAX1494评估软件需要和完整的评估系统 (MAX1494EVC16)配套使用。MAX1494EVC16包括68HC16MODULE-DIP模块以及MAX1494EVKIT。若不使用MAX1494评估软件进行评估，可购买不带µC模块的MAX1494评估板。*

MAX1494EV16系统

PART	QTY	DESCRIPTION
MAX1494EVKIT	1	MAX1494 EV kit
68HC16MODULE-DIP	1	68HC16 µC module

MAX1494评估板元件列表

DESIGNATION	QTY	DESCRIPTION
C1, C2	2	10µF ±20%, 10V X7R ceramic capacitors (1210) TDK C3225X7R1C106M
C3–C6	4	0.47µF ±10%, 16V X7R ceramic capacitors (0805) TDK C2012X7R1C474K
C7, C8, C9	3	0.1µF ±20%, 16V X7R ceramic capacitors (0603) TDK C1608X7R1C104K
CLK	1	BNC 50Ω PC board vertical mount A/D ELECTRONICS 580-002-00

DESIGNATION	QTY	DESCRIPTION
FB1	1	Ferrite bead (0805) Murata BLM21AH102SN1
J1	1	2 x 20 right angle socket, SamTec SSW-120-02-S-D-RA Methode Electronics RS2R-40-G
JU1	1	3-pin header
JU1–JU6	6	Shunts
JU2–JU6	5	2-pin headers
LCD1	1	Triplexed liquid crystal display (LCD), ICL7129 type DCI Inc. 04-0925-00 or Varitronix VIM-503-DP-FC-S-HV

Windows是Microsoft Corp.的注册商标。

MAX1494评估板元件列表 (续)

DESIGNATION	QTY	DESCRIPTION
LCD1 (2 rows)	2	15-pin socket strips
R1	1	133kΩ ±1% resistor (1206)
R2	1	100kΩ ±1% resistor (1206)
R3–R7	5	1kΩ ±5% resistors (1206)
TP1–TP4	4	8-pin headers
U1	1	MAX1494CCJ (32-pin TQFP)
U2	1	MAX1615EUK-T
U3, U4	2	MAX1840EUB or MAX1841EUB
U5	1	MAX6062AEUR-T
AIN+, AIN-, REF+, REF-	4	Noninsulated banana jacks Mouser 530-108-0740-1
None	1	MAX1494 EV kit PC board

快速入门

所需设备

在开始测试前，您需要下列设备：

- MAX1494EVC16 (包括MAX1494评估板和68HC16-MODULE-DIP)
- 0.25A，+7VDC至+20VDC电源
- 带有空闲串口 (COM)、操作系统为Windows 95/98/2000/XP的计算机
- 9针I/O扩展电缆

步骤

在完成所有连接之前，不要打开电源。

1) 确保JU1的1-2位置以及JU2–JU6都安装了短路器。请参考表2 (跳线设置)。
2) 将MAX1494评估板的40引脚插头对准68HC16-MODULE-DIP模块的40引脚连接器，并小心地连接两个电路板。轻按两块电路板使其连到一起。两块电路板应彼此对齐。
3) 将+7VDC至+20VDC电源连接至μC模块顶端边缘、位于ON/OFF开关旁边的接线端子板。注意板上所标的极性。
4) 用电缆将计算机串口与μC模块相连。若用到的是9针串口，就用直通式9针孔-针电缆。若只有25针连接器的串口，则需要标准的25针至9针转接器。评估软件检查调制解调器状态连线 (CTS、DSR和DCD)，以确认选择了正确的端口。
5) 运行磁盘上的INSTALL.EXE程序，将评估软件安装到您的计算机上。完成程序文件拷贝，并在Windows开始菜单里创建相应的图标。
6) 打开电源。
7) 点击开始菜单中的图标，运行MAX1494程序。
8) 程序会提示您连接μC模块，并打开其电源。将SW1拨到ON位置。选择正确的串口，并点击OK。程序会自动将其软件下载到模块中。
9) 在AIN+和AIN-之间加载-2V至+2V范围内的输入信号。观察屏幕上的读数显示。
10) 下拉View菜单并点击Graph选项，查看测量结果的图表显示。

软件详细说明

测量

评估软件的**Measurement**选项卡页模拟数字电压表 (DVM)的功能。状态位大约每秒刷新一次。当**Data**状态位为1时，读取ADC结果寄存器，并显示为**Analog Input Code**。MAX1494还在其LCD上显示结果。

评估板并不是一个完整的DVM。可能还需要额外的输入比例运算电路及保护电路。

Measurement选项卡页为活动状态时，若**spi/adc**和**seg_sel**控制位未清零，软件会将其清零。

数学运算

评估软件可实现物理层的几个数学函数功能。在**Math**选项卡页被激活后，若**spi/adc**控制位还未置1，软件会将其置为1。若**seg_sel**控制位尚未清零，软件会将其清零。

元件供应商

SUPPLIER	PHONE	FAX	WEBSITE
TDK	847-803-6100	847-390-4405	www.component.tdk.com

注意：当与这些供应商联系时，请指明您正在使用MAX1494。

表1. 图表工具按钮

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.
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.
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: $\text{Mean} = \frac{\sum(x)}{n}$
StdDev	Show standard deviation in tabular view: $\text{Standard deviation} = \sqrt{\frac{n\sum\left(x^2\right)-\left(\sum_n x\right)^2}{(n-1)n}}$
Rms	Show root of the mean of the squares (RMS) in tabular view: $\text{RMS} = \sqrt{\frac{\sum\left(x^2\right)}{n}}$
0	Channel 0 enable (ADC result)
1	Channel 1 enable (math result)
2	Channel 2 enable (20-bit ADC result)

评估软件在显示ADC结果之前先读取结果，当**Measurement**或**Math**选项卡页被激活，且**spi/adc**控制位被置为1时，软件计算一个新的LCD显示值。数学计算结果作为通道1的数据用图形表示，而原始ADC结果作为通道0的数据。

Type K Thermocouple测量功能，当选择适当的冷端连接点时，可用来将K型热电偶测量的Seebeck电压转换成摄氏度。**a0**系数为230时代表冷端温度为23.0°C。

控制寄存器

Control Register选项卡页可对所有控制寄存器位进行访问。下拉相应组合框，并点击Write选项。

量程寄存器、ADC失调、ADC结果、LCD和峰值

Results、**Displays**和**Limits**选项卡页提供对二进制补码数据寄存器的访问。除了**ADC RESULT1**、**ADC RESULT2**和**PEAK RESULT**这些只读寄存器以外，每个寄存器均有**Read**按钮和**Write**按钮。

不管**seg_sel**控制位的设置如何，读取ADC RESULT1或ADC RESULT2寄存器将自动更新LCD显示。

不管**offset_cal1**控制位的设置如何，写入ADC OFFSET寄存器将会影响ADC RESULT1和ADC RESULT2。

LCD段寄存器

LCD Segments选项卡页允许用户通过点击鼠标来点亮和关闭独立的LCD段。

LCD Segments选项卡页被激活后，若**seg_sel**控制位尚未置1，软件会将其置为1。

表2. 跳线功能

JUMPER	SHUNT POSITION	FUNCTION
JU1	1-2*	DV_{DD} = +5V.
JU1	2-3	DV_{DD} = +3V.
JU2	Closed*	V_{DISP} = GND.
JU2	Open	Apply V_{DISP} voltage at VDISP pad.
JU3	Closed*	Banana jack AIN+ connects to AIN+ input pin.
JU3	Open	Insert custom filtering between JU3 pins 1 and 2.
JU4	Closed*	Banana jack AIN- connects to AIN- input pin.
JU4	Open	Insert custom filtering between JU4 pins 1 and 2.
JU5	Closed*	REF- = GND.
JU5	Open	REF- must be provided by user.
JU6	Closed*	REF+ = +2.048V from U5, MAX6062.
JU6	Open	REF+ must be provided by user.

** 表示缺省配置*

仅有12段的LCD (例如VIM503) 不支持**hold**或**peak**指示，然而，该器件和评估板支持**hold**或**peak**指示。

Write LCD Text按钮将文本字符串转换为近似的7段字符，然后将字符图案写入LCD。

图表

评估软件的图表数据有两种选项。最新数据可通过选择**View**菜单中的**Graph**选项来显示。可采用时序曲线图、柱状图或原始数据表的形式来查看数据。点击主窗口上的**Collect Samples**按钮，以便激活采样工具来控制数据的大小和采集时间。

采样数据可保存为用逗号或制表符分隔的文件。行编号和说明标题为可选项。

通道0显示原始的16位ADC结果。数学运算功能启用后，通道1显示LCD数据。若使能扩展分辨率功能，通道2显示20位ADC结果。

诊断窗口

诊断窗口用于评估板出厂前的测试。该功能不供用户使用。

硬件详细说明

所测试的MAX1494 (U1) 是集成了LCD驱动器的低功耗、4位半 ADC。MAX6062 (U5) 提供板上+2.048V基准电压。请参考图7和MAX1494数据资料。

表3. 单机接口引脚功能

U1 PIN	MAX1494 FUNCTION	MAX1493/MAX1495 FUNCTION
7	$\overline{EOC}$	RANGE
8	$\overline{CS}$	DPSET1
9	DIN	DPSET2
10	SCLK	PEAK
11	DOUT	HOLD
28	VDISP	DPON
30	CLK	INTREF

评估板包括MAX1615 +3V/+5V线性稳压器 (U2) 和一组电平转换器MAX1840/MAX1841 (U3和U4)，以便用+5V μC控制+3V MAX1494。

评估MAX1493/MAX1495

MAX1494评估板支持MAX1493/MAX1495的单机工作方式。但由于这些单机器件上没有微处理器接口，所以不能使用评估软件。

MAX1493是MAX1494的单机版本。MAX1495与MAX1493类似，但可根据要求启用失调校准功能。请参考MAX1491/MAX1493/MAX1495数据资料。可申请MAX1493CCJ或MAX1495CCJ免费样品。

1) MAX1494评估板必须与68HC16MODULE模块断开连接。

2) 在电源断开时，用MAX1493或MAX1495替换U1。用MAX1493或MAX1495替换U1之后，有些引脚功能是不同的。请参考表3。
3) 保证跳线JU1选择的是所期望的+3V或+5V逻辑电平。
4) 在接线板TB1上连接DC电源。
5) 打开电源。LCD应开始显示测量数据。

排查问题

问题：峰值检测模式在低于19,487个计数时不工作。

这是MAX1494限定的。请参考MAX1494数据资料。

问题：启动延迟

上电时MAX1494需要大约2秒来完成启动。

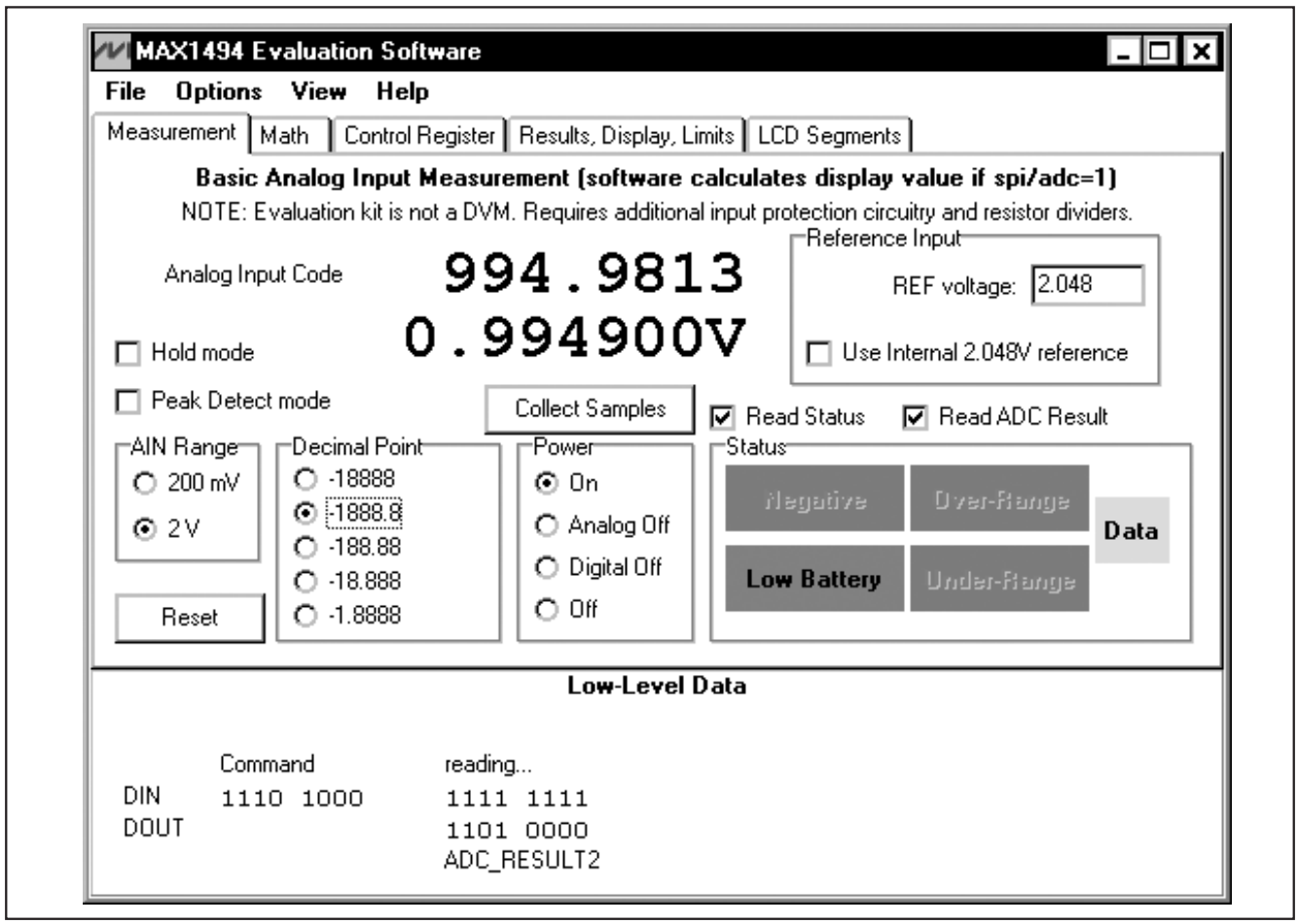

图1. 主窗口—MAX1494评估软件

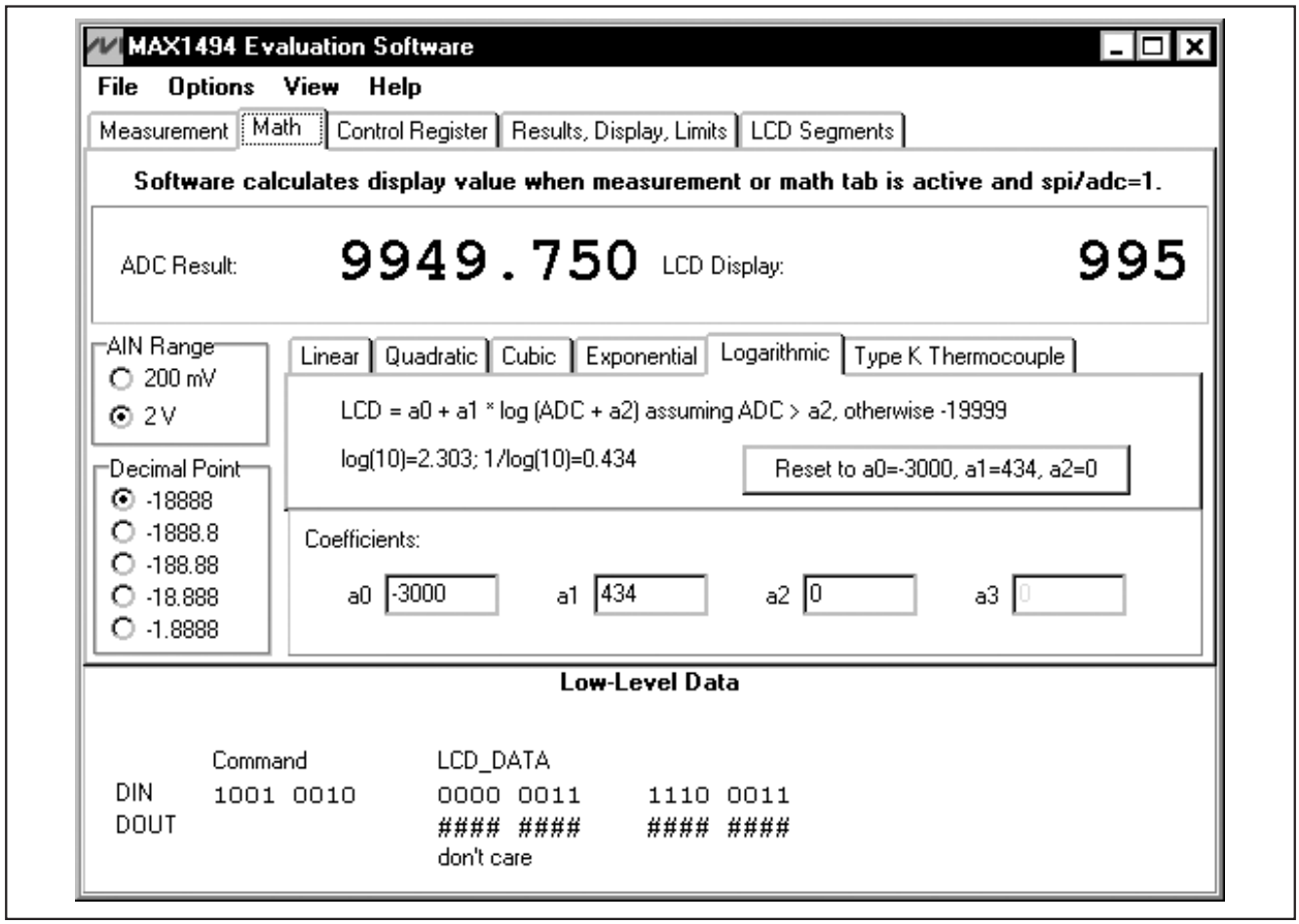

图2. Math选项卡页—MAX1494评估软件

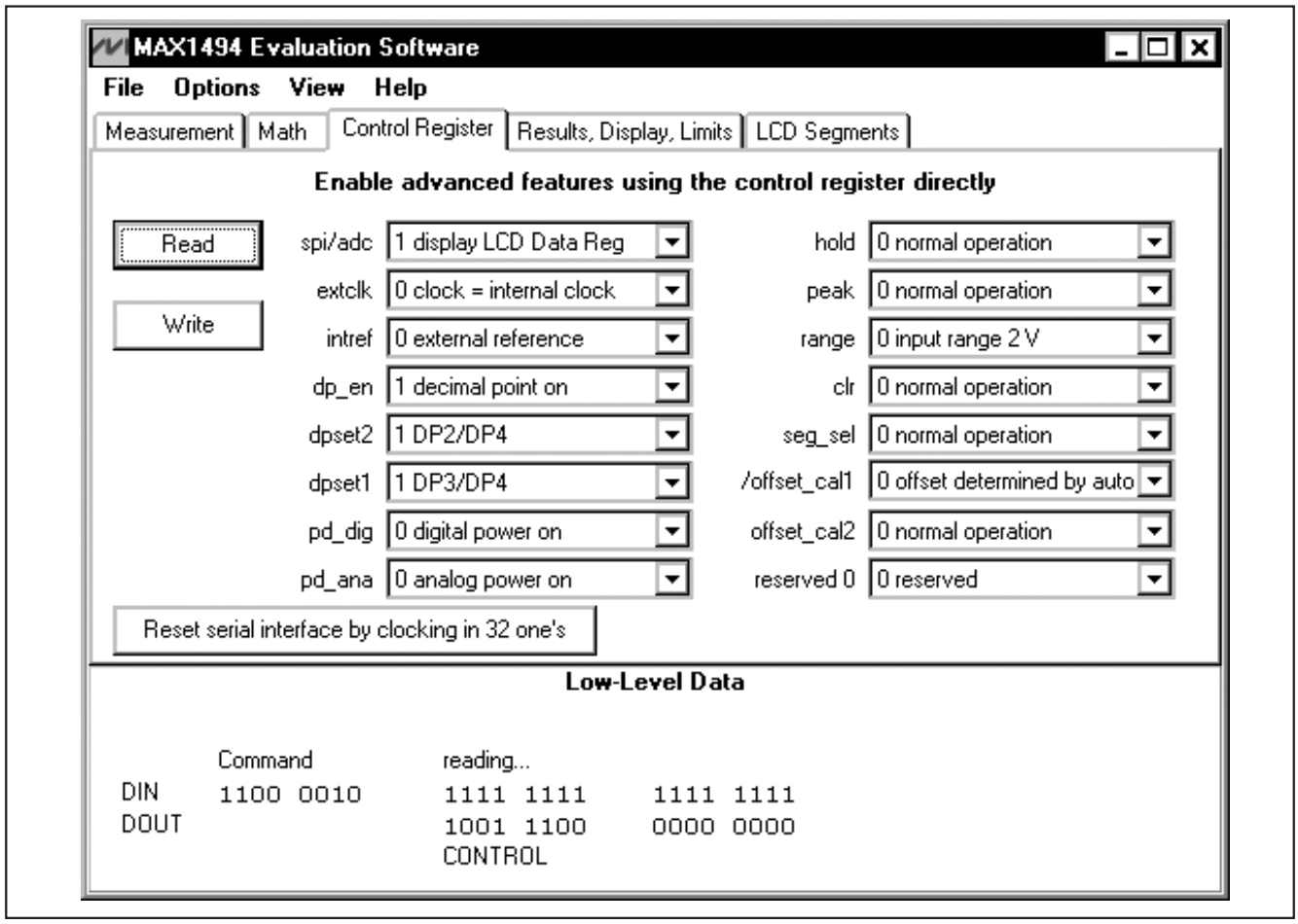

图3. Control Register选项卡页—MAX1494评估软件

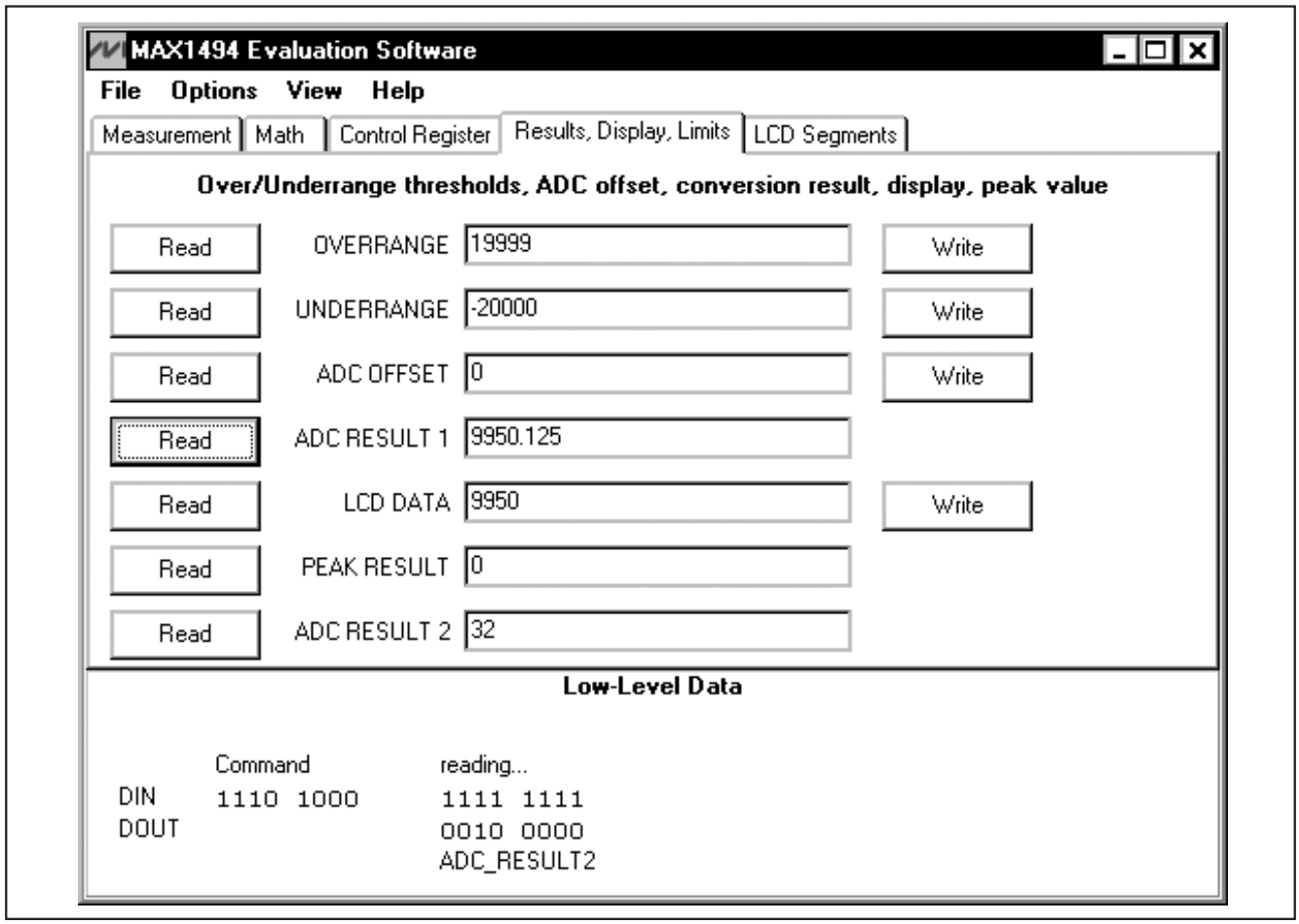

图4. Results, Display, Limits选项卡页—MAX1494评估软件

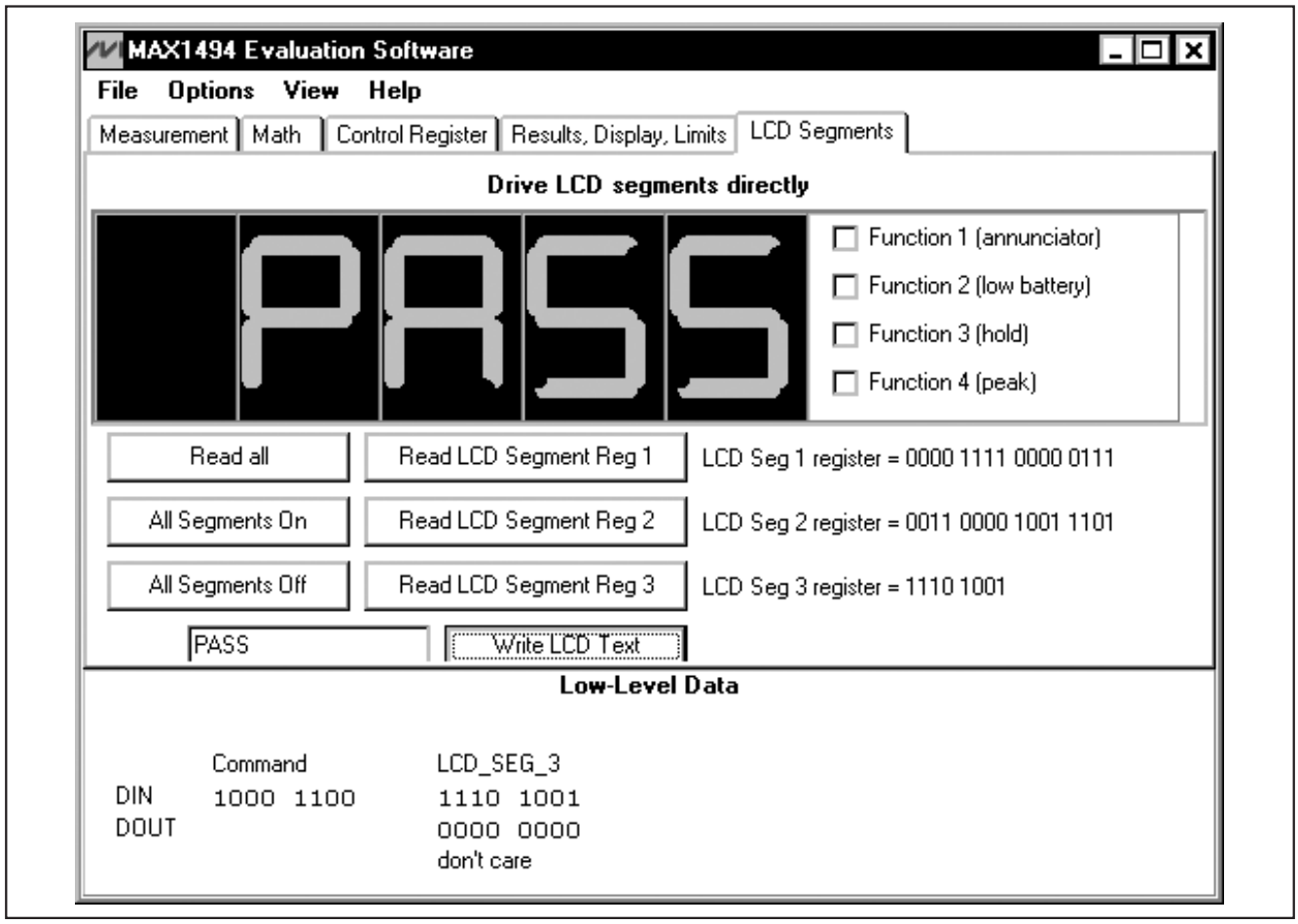

图5. LCD Segments选项卡页—MAX1494评估软件

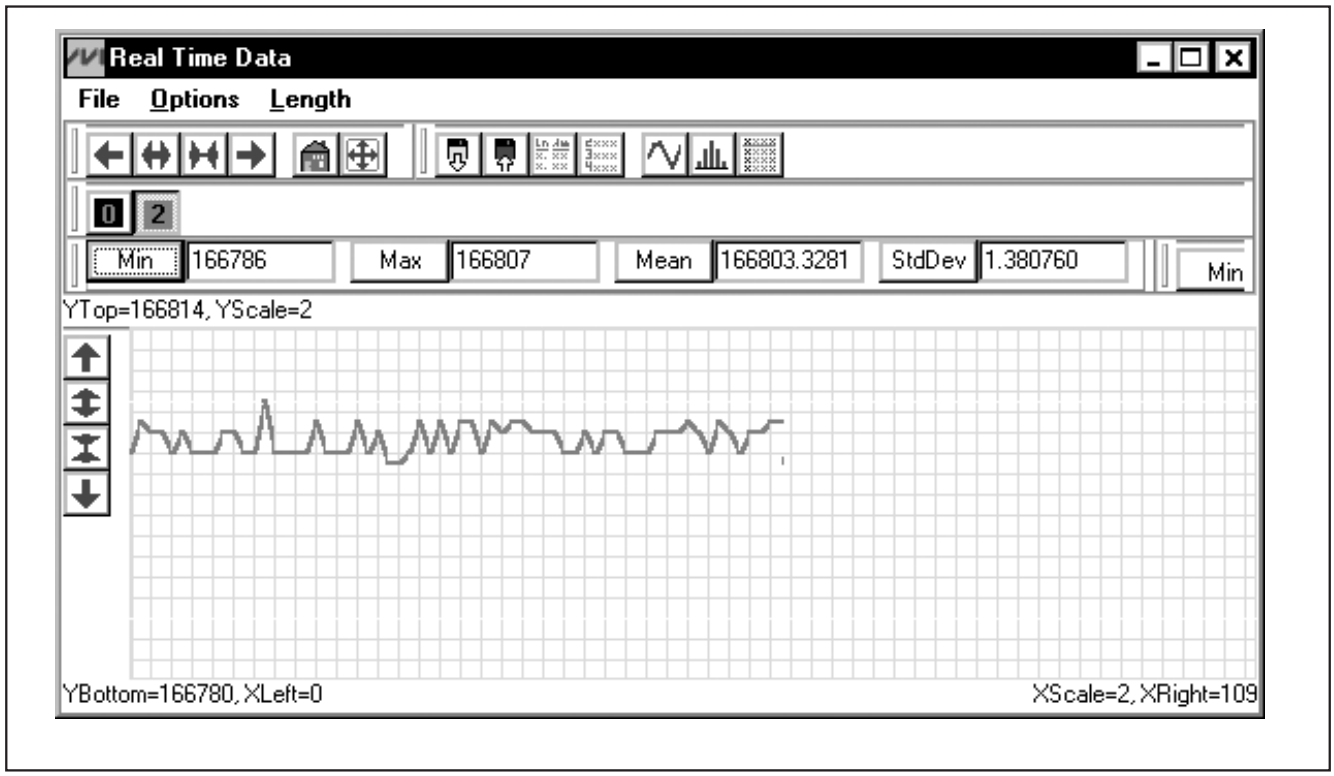

图6. 图表窗口—MAX1494评估软件

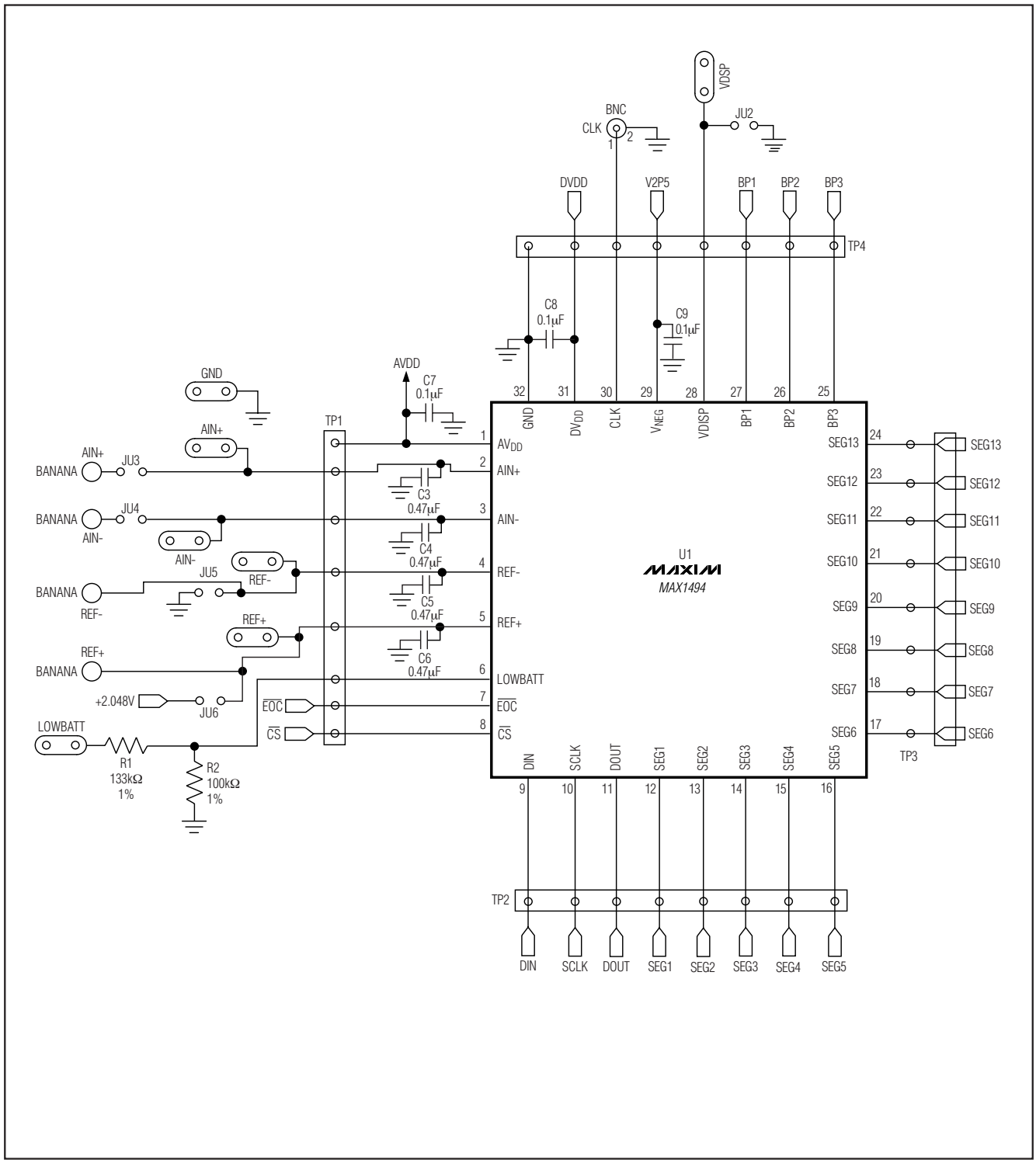

图7a. MAX1494评估板原理图

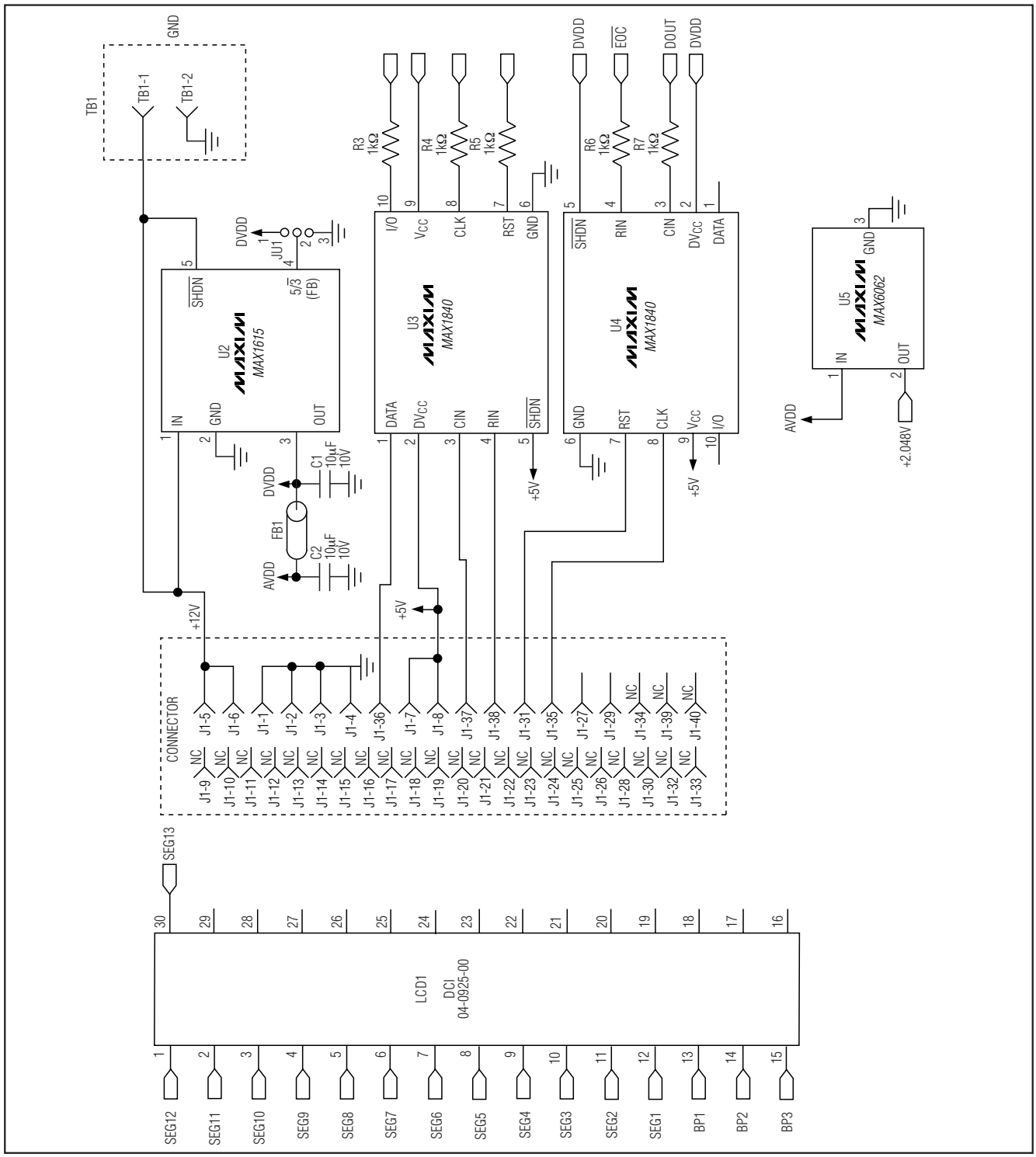

图7b. MAX1494评估板原理图 (续)

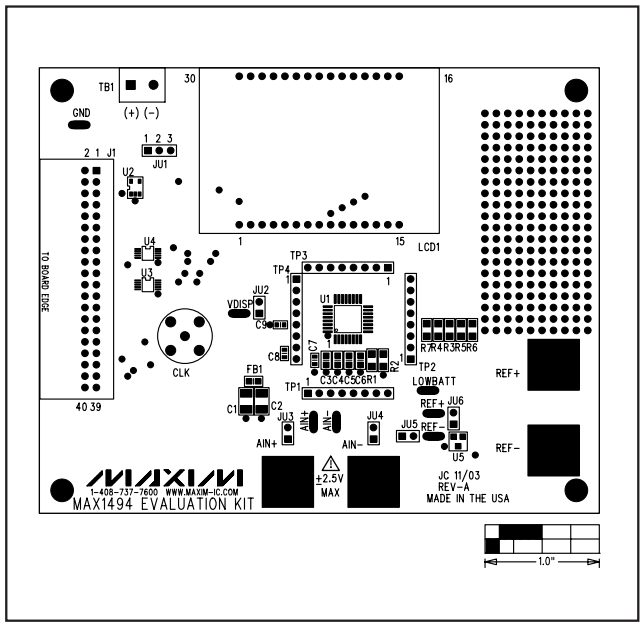

图8. MAX1494评估板元件摆放指南—顶层丝印层

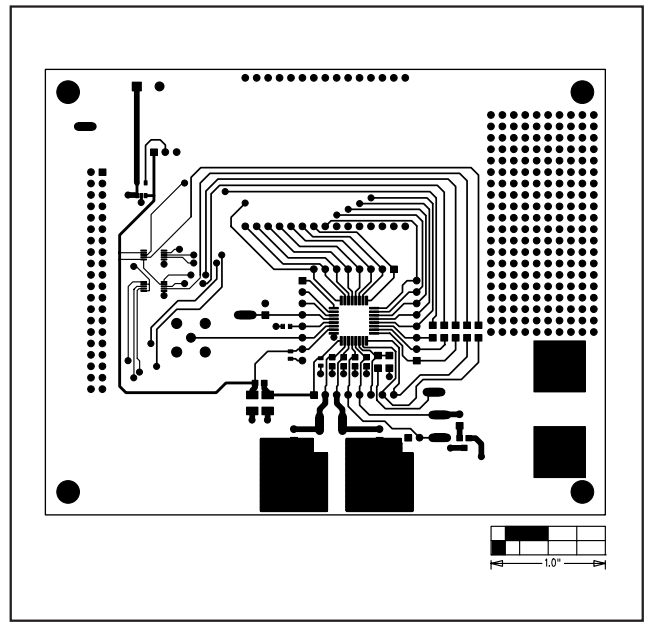

图9. MAX1494评估板PC板布局—元件层

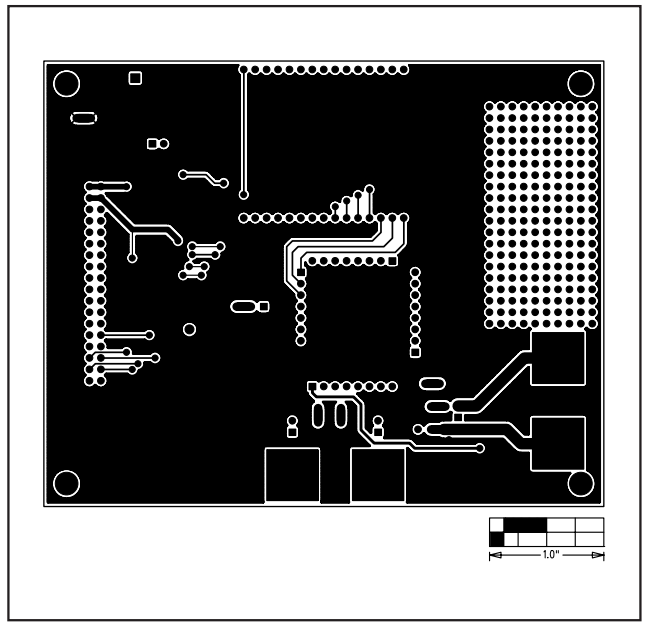

图10. MAX1494评估板PC板布局—焊接层

```
// Drv1494.h
// MAX1494-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)
// 09/12/2003: add SPI_Transfer_After_EOC()
// 09/09/2003: add class MAX1494 dependent on external SPI_Interface()
// 08/13/2003: preliminary draft of reuseable code
//---------------------------------------------------------------------------
#ifndef drv1494H
#define drv1494H
//---------------------------------------------------------------------------

//---------------------------------------------------------------------------
// 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 MAX1494_COMMS_START          0x80
#define MAX1494_COMMS_RW_MASK        0x40
#define MAX1494_COMMS_RW_WRITE           0x00
#define MAX1494_COMMS_RW_READ            0x40
#define MAX1494_COMMS_RS_MASK        0x3E
#define MAX1494_COMMS_RS_00000           0x00
#define MAX1494_COMMS_RS_STATUS          0x00
#define MAX1494_COMMS_RS_00001           0x02
#define MAX1494_COMMS_RS_CONTROL         0x02
#define MAX1494_COMMS_RS_00010           0x04
#define MAX1494_COMMS_RS_OVERRANGE       0x04
#define MAX1494_COMMS_RS_00011           0x06
#define MAX1494_COMMS_RS_UNDERRANGE      0x06
#define MAX1494_COMMS_RS_00100           0x08
#define MAX1494_COMMS_RS_LCD_SEG_1       0x08
#define MAX1494_COMMS_RS_00101           0x0A
#define MAX1494_COMMS_RS_LCD_SEG_2       0x0A
#define MAX1494_COMMS_RS_00110           0x0C
#define MAX1494_COMMS_RS_LCD_SEG_3       0x0C
#define MAX1494_COMMS_RS_00111           0x0E
#define MAX1494_COMMS_RS_ADC_OFFSET      0x0E
#define MAX1494_COMMS_RS_01000           0x10
#define MAX1494_COMMS_RS_ADC_RESULT1     0x10
#define MAX1494_COMMS_RS_01001           0x12
#define MAX1494_COMMS_RS_LCD_DATA        0x12
```

图11. 程序清单1 (第1页，共4页)

```
#define MAX1494_COMMS_RS_01010          0x14
#define MAX1494_COMMS_RS_PEAK           0x14
#define MAX1494_COMMS_RS_10100          0x28
#define MAX1494_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
#define MAX1494_STATUS_POL_MASK         0x80
#define MAX1494_STATUS_POL_POSITIVE         0x00
#define MAX1494_STATUS_POL_NEGATIVE         0x80
#define MAX1494_STATUS_OVER_RANGE       0x40
#define MAX1494_STATUS_UNDER_RANGE      0x20
#define MAX1494_STATUS_LOW_BATTERY      0x10
#define MAX1494_STATUS_DATA_READY       0x08

//-------------------------------------------------------------------------
// Define the bits in the CONTROL register.
// SPI_ADC EXTCLK INTREF DP_EN DPSET2 DPSET1   PD_DIG    PD_ANA
// HOLD    PEAK   RANGE CLR  LCD   OFFSET_CAL1 OFFSET_CAL2 0
#define MAX1494_CONTROL_SPI_ADC         0x8000
#define MAX1494_CONTROL_EXTCLK          0x4000
#define MAX1494_CONTROL_INTREF          0x2000
#define MAX1494_CONTROL_DPMASK          0x1C00
#define MAX1494_CONTROL_DP_EN           0x1000
#define MAX1494_CONTROL_DPSET2          0x0800
#define MAX1494_CONTROL_DPSET1          0x0400
// (DPSET2 is the LSB and DPSET1 is the MSB)
#define MAX1494_CONTROL_DP1ON           0x1000 /* -1888.8 */
#define MAX1494_CONTROL_DP2ON           0x1800 /* -188.88 */
#define MAX1494_CONTROL_DP3ON           0x1400 /* -18.888 */
#define MAX1494_CONTROL_DP4ON           0x1C00 /* -1.8888 */
#define MAX1494_CONTROL_PD_DIG          0x0200
#define MAX1494_CONTORL_PD_ANA          0x0100
#define MAX1494_CONTROL_PD_ALL          0x0300
#define MAX1494_CONTROL_HOLD            0x0080
#define MAX1494_CONTROL_PEAK            0x0040
#define MAX1494_CONTROL_RANGE_200mV     0x0020
#define MAX1494_CONTROL_CLR             0x0010
#define MAX1494_CONTROL_SEG_SEL         0x0008
#define MAX1494_CONTROL_OFFSET_CAL1     0x0004
#define MAX1494_CONTROL_OFFSET_CAL2     0x0002

//-------------------------------------------------------------------------
// Define the bits in the LCD SEGMENT 1 register.
// A2 G2 D2 F2 E2 DP2 ANNUNCIATOR B1
// C1 A1 G1 D1 F1 E1 DP1 0
//
#define MAX1494_LCD_SEG1_A2             0x8000
#define MAX1494_LCD_SEG1_G2             0x4000
#define MAX1494_LCD_SEG1_D2             0x2000
#define MAX1494_LCD_SEG1_F2             0x1000
#define MAX1494_LCD_SEG1_E2             0x0800
#define MAX1494_LCD_SEG1_DP2            0x0400
#define MAX1494_LCD_SEG1_ANNUNCIATOR    0x0200
#define MAX1494_LCD_SEG1_B1             0x0100
#define MAX1494_LCD_SEG1_C1             0x0080
#define MAX1494_LCD_SEG1_A1             0x0040
#define MAX1494_LCD_SEG1_G1             0x0020
#define MAX1494_LCD_SEG1_D1             0x0010
#define MAX1494_LCD_SEG1_F1             0x0008
#define MAX1494_LCD_SEG1_E1             0x0004
#define MAX1494_LCD_SEG1_DP1            0x0002

//-------------------------------------------------------------------------
// Define the bits in the LCD SEGMENT 2 register.
// F4 E4 DP4 MINUS B3 C3 A3 G3
// D3 F3 E3 DP3 LOWBATT B2 C2 0
```

图11. 程序清单1 (第2页，共4页)

```
//
#define MAX1494_LCD_SEG2_F4          0x8000
#define MAX1494_LCD_SEG2_E4      0x4000
#define MAX1494_LCD_SEG2_DP4     0x2000
#define MAX1494_LCD_SEG2_MINUS     0x1000
#define MAX1494_LCD_SEG2_B3          0x0800
#define MAX1494_LCD_SEG2_C3          0x0400
#define MAX1494_LCD_SEG2_A3          0x0200
#define MAX1494_LCD_SEG2_G3          0x0100
#define MAX1494_LCD_SEG2_D3          0x0080
#define MAX1494_LCD_SEG2_F3          0x0040
#define MAX1494_LCD_SEG2_E3      0x0020
#define MAX1494_LCD_SEG2_DP3     0x0010
#define MAX1494_LCD_SEG2_LOWBATT    0x0008
#define MAX1494_LCD_SEG2_B2          0x0004
#define MAX1494_LCD_SEG2_C2          0x0002

//-------------------------------------------------------------------------
// Define the bits in the LCD SEGMENT 3 register.
// ?PEAK? ?HOLD? BC5 B4 C4 A4 G4 D4
//
#define MAX1494_LCD_SEG3_PEAK          0x80
#define MAX1494_LCD_SEG3_HOLD          0x40
#define MAX1494_LCD_SEG3_BC5         0x20
#define MAX1494_LCD_SEG3_B4             0x10
#define MAX1494_LCD_SEG3_C4             0x08
#define MAX1494_LCD_SEG3_A4             0x04
#define MAX1494_LCD_SEG3_G4             0x02
#define MAX1494_LCD_SEG3_D4             0x01

//-------------------------------------------------------------------------
class MAX1494
{
public:
    MAX1494(void);

    // Enumerated type describing the register select bits.
    enum RegisterSelect_t {
        RS_STATUS       = MAX1494_COMMS_RS_STATUS,
        RS_CONTROL      = MAX1494_COMMS_RS_CONTROL,
        RS_OVERRANGE    = MAX1494_COMMS_RS_OVERRANGE,
        RS_UNDERRANGE   = MAX1494_COMMS_RS_UNDERRANGE,
        RS_LCD_SEG_1    = MAX1494_COMMS_RS_LCD_SEG_1,
        RS_LCD_SEG_2    = MAX1494_COMMS_RS_LCD_SEG_2,
        RS_LCD_SEG_3    = MAX1494_COMMS_RS_LCD_SEG_3,
        RS_ADC_OFFSET   = MAX1494_COMMS_RS_ADC_OFFSET,
        RS_ADC_RESULT1  = MAX1494_COMMS_RS_ADC_RESULT1,
        RS_LCD_DATA     = MAX1494_COMMS_RS_LCD_DATA,
        RS_PEAK         = MAX1494_COMMS_RS_PEAK,
        RS_ADC_RESULT2  = MAX1494_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 DP_EN DPSET2 DPSET1   PD_DIG   PD_ANA
    // HOLD   PEAK  RANGE CLR LCD   OFFSET_CAL1 OFFSET_CAL2 0
    int CONTROL_REG;
```

图11. 程序清单1 (第3页，共4页)

```
        //
        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);

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

};

//---------------------------------------------------------------------------
#endif
```

图11. 程序清单1 (第4页，共4页)

```
// Drv1494.cpp
// MAX1494-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 MAX1494 dependent on external SPI_Interface()
// 08/13/2003: preliminary draft of reuseable code

#include "drv1494.h"

//---------------------------------------------------------------------------
MAX1494::MAX1494(void)
{
    vref = 2.048;
    extended_resolution = false;
}
//---------------------------------------------------------------------------
bool MAX1494::Read_STATUS(void)
{
    const unsigned __int8 mosi[] = {
        (unsigned __int8)(MAX1494_COMMS_START |
            MAX1494_COMMS_RW_READ | MAX1494_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 MAX1494::Write_CONTROL(int data)
{
    data = data & 0xFFFF;       // validate the data
    const unsigned __int8 mosi[] = {
        (unsigned __int8)(MAX1494_COMMS_START |
            MAX1494_COMMS_RW_WRITE | MAX1494_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 &=~ MAX1494_CONTROL_CLR;
    return result;
}
//---------------------------------------------------------------------------
bool MAX1494::Read_CONTROL(void)
{
    const unsigned __int8 mosi[] = {
        (unsigned __int8)(MAX1494_COMMS_START |
            MAX1494_COMMS_RW_READ | MAX1494_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
    }
```

图12. 程序清单2 (第1页，共4页)

```
    return result;
}
//-------------------------------------------------------------------
bool MAX1494::Read_ADC_RESULT1(void)
{
    const unsigned __int8 mosi[] = {
        (unsigned __int8)(MAX1494_COMMS_START |
            MAX1494_COMMS_RW_READ | MAX1494_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 MAX1494::Read_ADC_RESULT2(void)
{
    const unsigned __int8 mosi[] = {
        (unsigned __int8)(MAX1494_COMMS_START |
            MAX1494_COMMS_RW_READ | MAX1494_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 MAX1494::Read_DATA(void)
{
    // Read the DATA register
    const unsigned __int8 mosi[] = {
        (unsigned __int8)(MAX1494_COMMS_START |
            MAX1494_COMMS_RW_READ | MAX1494_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)(MAX1494_COMMS_START |
                MAX1494_COMMS_RW_READ | MAX1494_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
```

图12. 程序清单2 (第2页，共4页)

```
            }
            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;
        }
        return DATA_REG;
    }
    //---------------------------------------------------------------------
    double MAX1494::Voltage(void)
    {
        if ((CONTROL_REG & MAX1494_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 MAX1494::Write_2s_complement(int reg, int data)
    {
        // Write one of the 2's complement registers
        reg = (reg & MAX1494_COMMS_RS_MASK);
        data = data & 0xFFFF;       // validate the data

        const unsigned __int8 mosi[] = {
            (unsigned __int8)(MAX1494_COMMS_START | MAX1494_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 MAX1494::Read_2s_complement(int reg)
    {
        // Read one of the 2's complement registers
        reg = (reg & MAX1494_COMMS_RS_MASK);

        const unsigned __int8 mosi[] = {
            (unsigned __int8)(MAX1494_COMMS_START | MAX1494_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 MAX1494::Write_8bit_reg(int reg, int data)
    {
        // Write one of the 8 bit registers
        reg = (reg & MAX1494_COMMS_RS_MASK);
        const unsigned __int8 mosi[] = {
```

图12. 程序清单2 (第3页，共4页)

```
            (unsigned __int8)(MAX1494_COMMS_START | MAX1494_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 MAX1494::Read_8bit_reg(int reg)
    {
        // Read one of the 8 bit registers
        reg = (reg & MAX1494_COMMS_RS_MASK);
        const unsigned __int8 mosi[] = {
            (unsigned __int8)(MAX1494_COMMS_START | MAX1494_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;
    }
    //---------------------------------------------------------------------
```

图12. 程序清单2 (第4页，共4页)

MAXIM北京办事处

北京 8328信箱 邮政编码 100083

免费电话：800 810 0310

电话：010-6201 0598

传真：010-6201 0298