

Tricolor Serial Communication ASCII Protocol



16900 FOLTZ PARKWAY - CLEVELAND, OH 44149
 Phone: (440) 238-2550 - Fax: (440) 238-0660
 www.weschler.com e-mail: sales@weschler.com

1.0 Overview.

BG Tricolor Series Bargraphs are designed to send and receive formatted strings of ASCII characters via RS-232/485. These message strings contain some combination of the following information: message header, unit id, address of a variable, variable data, length of the variable in bytes, length of the message in bytes, and a checksum. What this provides is flexible a mechanism for users to access information stored in the bargraphs memories. The disadvantage of the system is that PC base application must be programmed with knowledge of variable addresses and their length in bytes. Write messages simply replaces the existing information at the specified location with that in the data field. Read messages simply returns the data at the specified address in Motorola S format.

Variable addresses and lengths can be found in the files variable.c and eeprom.h. Those listed in eeprom.h are stored in non-volatile EEPROM memory and are used for the purpose of configuring the unit.

2.0 Serial Communication Parameters.

The bargraph is configured for the following parameters:

Baud Rate - 9600
 Data Bits - 8
 Stop Bits - 1
 Parity - None

Therefore a typical port configuration for a PC based application would be as follows:

Baud Rate - 9600
 Data Bits - 8
 Stop Bits - 1
 Parity - None
 Handshaking - None

Tricolor Serial Communication ASCII Protocol

3.0 Defining the Message Formats.

3.1 Identifying the Fields.

The formats for reading from a memory location, the bargraph responses, and writing to memory locations are defined in Tables 1, 2 and 3.

Table 1. Read from Memory Format

Byte Position	ASCII characters	Description
1	>R=	Message Header
2,3	>0-->9=, >A-->F=	Unit Id
4,5,6,7	>0-->9=, >A-->F=	Variable Address
8,9	>0-->9=, >A-->F=	Variable Length: Byte A01@, Word A02@, Long A04@, Float A04@
10,11	>0-->9=, >A-->F=	Checksum
12	<CR>	Message terminate

Tricolor Serial Communication ASCII Protocol

Table 2. BarGraph Response Format

Byte Position	ASCII characters	Description
Responses to Byte Reads		
1,2	AS1@	Message Header
3,4	A04@	Byte Count
5,6,7,8	>0-->9=, >A-->F=	Variable Address
9,10	>0-->9=, >A-->F=	Variable Data
11,12	>0-->9=, >A-->F=	Checksum
13	<CR>	Message terminate
Responses to Word Reads		
1,2	AS1@	Message Header
3,4	A05@	Byte Count
5,6,7,8	>0-->9=, >A-->F=	Variable Address
9,10	>0-->9=, >A-->F=	Variable Data, Most Significant Byte
11,12	>0-->9=, >A-->F=	Variable Data , Least Significant Byte
13,14	>0-->9=, >A-->F=	Checksum
15	<CR>	Message terminate
Responses to Long and Float Reads		
1,2	AS1@	Message Header
3,4	A07@	Bytes Count
5,6,7,8	>0-->9=, >A-->F=	Variable Address
9,10	>0-->9=, >A-->F=	Variable Data, Most Significant Word, Most Significant Byte
11,12	>0-->9=, >A-->F=	Variable Data, Most Significant Word, Least Significant Byte
13,14	>0-->9=, >A-->F=	Variable Data, Least Significant Word, Most Significant Byte
15,16	>0-->9=, >A-->F=	Variable Data, Least Significant Word, Least Significant Byte
17,18	>0-->9=, >A-->F=	Checksum
19	<CR>	Message terminate

Table 3. Write to Memory Format

Tricolor Serial Communication ASCII Protocol

Byte Position	ASCII characters	Description
Byte Writes		
1	AW@	Message Header
2,3	>0-->9=, >A-->F=	Unit Id
4,5	A04@	Byte Count
6,7,8,9	>0-->9=, >A-->F=	Variable Address
10,12	>0-->9=, >A-->F=	Variable Data
13,14	>0-->9=, >A-->F=	Checksum
15	<CR>	Message terminate
Word Writes		
1	AW@	Message Header
2,3	>0-->9=, >A-->F=	Unit Id
4,5	A05@	Byte Count
6,7,8,9	>0-->9=, >A-->F=	Variable Address
10,11	>0-->9=, >A-->F=	Variable Data, Most Significant Byte
12,13	>0-->9=, >A-->F=	Variable Data , Least Significant Byte
14,15	>0-->9=, >A-->F=	Checksum
16	<CR>	Message terminate
Long and Float Writes		
1	AW@	Message Header
2,3	>0-->9=, >A-->F=	Unit Id
4,5	A07@	Bytes Count
6,7,8,9	>0-->9=, >A-->F=	Variable Address
10,11	>0-->9=, >A-->F=	Variable Data, Most Significant Word, Most Significant Byte
12,13	>0-->9=, >A-->F=	Variable Data, Most Significant Word, Least Significant Byte
14,15	>0-->9=, >A-->F=	Variable Data, Least Significant Word, Most Significant Byte
16,17	>0-->9=, >A-->F=	Variable Data, Least Significant Word, Least Significant Byte
18,19	>0-->9=, >A-->F=	Checksum
20	<CR>	Message terminate

Tricolor Serial Communication ASCII Protocol

3.2 Message Headers.

The ASCII characters >R=, >W=, and >S= are used to indicate the beginning of read, write, and response messages respectively. Once received by the bargraph, the unit begins storing incoming bytes to the communication buffer. Before responding, the unit will parse the information and determine if the message received is a valid request.

3.3 Unit Id.

The unit id field is used to identify a particular bargraph for which the host device (i.e. PC) would like to communicate. The unit id can be any number from 00 to 99 and must match the unit id of the target bargraph. Each hex digit of the unit id is represented by its corresponding ASCII character.

3.4 Variable Address.

This field corresponds to the address of the variable being accessed. Each character represents the corresponding hex digit of the address location. Table 4. illustrates how the address of variables **Reading** and **NumReading** would be represented.

Table 4. Examples of Address Representation

Variable Name	Address	String Representation
Reading	0x0003	A0003@
NumReading	0x0007	A0007@

3.5 Variable Length.

This field, which is required only for read from memory commands, indicates the size of the variable being accessed. Table 5 contains valid lengths various variable types. When the unit receives a read from memory command, the unit loads into the data field of the response string, the bytes at that make up the variable at the specified location.

Table 5. Examples of Valid Variable Lengths

Variable Type	String Length Field
char	A01@
int, unsigned int	A02@
long	A04@
float	A04@

Tricolor Serial Communication ASCII Protocol

3.6 Byte Count.

This field, which is required only in response and write messages, denotes the number of bytes that make up the address, data, and checksum. Each ASCII character in the byte count, address, data, and checksum fields represent a hex digit. Thus, every 2 characters correspond to 1 byte.

Example 1: Assume the bargraph reading is 5123 and the unit id is A00@. The bargraph variable **Reading** is stored as a long variable at memory location 0x0003. Long variables are 4 bytes in length. The command string for retrieving the bargraph reading and the bargraph response is as follows:

Command	AR00000304F8<CR>@
Response	AS1 <u>07</u> 000300001403DE<CR>@

Underlined is the byte count field. The string A07@ corresponds to a hex value of 0x07 and a decimal value of 7. Verified below is that the sum of the address, data, and checksum field indeed total 7 bytes.

	<u>String</u>	<u>Hex Value</u>	<u>Number of Bytes</u>
Address:	A0003@	0x0003	2
Data:	A00001403@	0x00001403	4
Checksum	ADE@	0xDE	<u>1</u>
Byte Count	A07@	0x07	7

3.7 Data.

In response messages, this field contains that data the was requested. The number of bytes in the field is determined by the variable length field in the read form memory command. In write messages, this field contains that data that will be written to the specified memory location. The number of bytes in the field is determined by the type of variable being accessed. Variables of type *char*, *int*, *long* and *float*, will require fields of 1, 2, 4, and 4 bytes in length respectively. Each ASCII character corresponds to a hex digit. Table 6 provides some examples of typical data fields.

Tricolor Serial Communication ASCII Protocol

Table 6. Examples of Data Fields

Variable	No. Bytes	Decimal Value	Hex Value	String	Comments
ADCstatus	1	0	0x00	A00@	Valid Data
ADCstatus	1	1	0x01	A01@	Overrange
ADCavg	2	2500	0x09C4	A09C4@	
ADCavg	2	10000	0x2710	A2710@	
Reading	4	-19999	0xFFFFB1E1	AFFFFB1E1@	
Reading	4	-10000	0xFFFFD8F0	AFFFFD8F0@	
Reading	4	0	0x00000000	A00000000@	
Reading	4	10000	0x00002710	A00002710@	
Reading	4	99999	0x0001869F	A0001869F@	

3.8 Checksum.

The checksum provides a means of verifying the validity of the transmission. For read from a memory location, the checksum is the last byte of the 1=s complement of the sum of the variable address and length . For response and write messages the checksum is the last byte of the 1=s complement of the sum of the byte count, address, and data fields.

Example 2: The bargraph variable **Reading** is stored as a long variable at memory location 0x0003. Long variables are 4 bytes in length. Assuming the unit id is 00, the command string for retrieving the bargraph reading is as follows:

Command AR00000304F8<CR>@

The checksum of AF8@ was determined in the following manner:

Address	A0003@	0x00	Most significant byte
		0x03	Least significant byte
Length	A04@	<u>0x04</u>	
Sum		0x07	
1=s Complement		0xF8	
Checksum string		AF8@	

Tricolor Serial Communication ASCII Protocol

Example 3: Assume the response to the read from memory command in example 2 is:

AS107000300001403DE<CR>@

The checksum of ADE@ was determined in the following manner:

Byte count	A07@	0x07	
Address	A0003@	0x00	Most significant byte
		0x03	Least significant byte
Data bytes	A00@	0x00	Most significant word, most significant byte
	A00@	0x00	Most significant word, least significant byte
	A14@	0x14	Least significant word, most significant byte
	A03@	<u>0x03</u>	Least significant word, least significant byte
Sum		0x21	
1=s Complement		0xDE	
Checksum string		ADE@	

3.9 String Terminator.

The <CR> character is used to indicate to the bargraph the end of a message.

Tricolor Serial Communication ASCII Protocol

4.0 Read from Memory Examples.

Table 7. Examples of Read from Memory Commands Assuming Unit ID of A00@.

Variable Name	Type	Address	Command String	Description
BGmode	int	0x0000	R00000002FD<CR>	Mode of Operation
EELock	char	0x0002	R00000201FC<CR>	EEPROM lock
Reading	long	0x0003	R00000304F8<CR>	Bar Display Reading
NumReading	long	0x0007	R00000704F4<CR>	Digital Display Reading
Peak	long	0x000B	R00000B04F0<CR>	Peak Value
Valley	long	0x000F	R00000F04EC<CR>	Valley Value
DecPoint	int	0x0013	R00001302EA<CR>	Decimal Point Location
Alarms	unsigned char	0x0015	R00001501E9<CR>	Status of Alarms
Leds	char	0x0016	R00001601E8<CR>	Trend Leds
ADCstatus	char	0x0017	R00001701E7<CR>	A/D Status
AC_avg	int	0x0018	R00001802E5<CR>	A/D average
CurrentADCavg	int	0x001A	R00001A02E3<CR>	Current A/D average
noZones	int	0x001C	R00001C02E1<CR>	Number of Zones
Zones[0].start	long	0x001E	R00001E04DD<CR>	Zone 0 start value
Zones[0].color	char	0x0022	R00002201DC<CR>	Zone 0 color
Zones[0].segment	int	0x0023	R00002302DA<CR>	Zone 0 begin segment
BarDpy2	15 char	0x0048	R0000480FA8<CR>	Bar Display Buffer
NumStr2	5 char	0x0057	R00005705A3<CR>	Digital Display Buffer

5.0 Write to Memory Example.

Example 4: Assume the bargraph is in standard ASCII mode and it is desired to have the unit display a reading of 5123. Also assume that the unit id is A01@. To accomplish this, a value of 5123 needs to be written to the variable **Reading**. Given that this is a long variable stored at memory location 0x0003, the command string would be as follows:

Command AW0107000300001403DE<CR>@

Tricolor Serial Communication ASCII Protocol

6.0 VARIABLE.C

```

/*
 * File: variable.c
 * Version: 1.
 *
 */

/*
 * Revision History
 * 1. Initial Issue. RRS
 */

/*
 * This file contains the list of global variables whose address must remain * in
fixed. To accomplish this, this file is to be the first file listed in * the link
list. The order in which the variables are listed is the order in * which memory
locations are assigned.
 */

#include "ad7715.h" /* AD7715 f()'s, defines, and variables */
#include "display.h" /* Display f()'s, defines. and variables */
#include "max7219.h" /* Display driver f()'s, defines, and variables */

/* Address */ /* Description*/
@dir int BGmode=0; /* 0x0000 */ /* Bargraph mode of operation */
/* 0 = RUN */
/* 1 = PRECONFIG mode */
/* 2 = PEAK display peak */
/* 3 = VALLEY display valley */
/* 4 = RESETALARMS */
/* 5 = OPERATOR Setup */
/* 6 = SUPERSET Supervisor Setup */
/* 7 = FACTORY Setup */
@dir char EElock=1; /* 0x0002 */ /* 0 = EEPROM Writes allowed */
/* 1 = EEPROM Writes not allowed */
@dir long Reading=0; /* 0x0003 */ /* Std. (Bar) Display Reading */
@dir long NumReading=0; /* 0x0007 */ /* Digital Display Reading */
@dir long Peak=0; /* 0x000B */ /* Peak reading */
@dir long Valley=0; /* 0x000F */ /* Valley reading */
@dir int DecPoint=0; /* 0x0013 */ /* Decimal point location */
@dir unsigned char Alarms=0; /* 0x0015 */ /* Alarm setpoint status */
@dir char Leds=0; /* 0x0016 */ /* Trend Leds bit mask */
@dir char ADCstatus=0; /* 0x0017 */ /* A/D status:
/* 0 = valid data */
/* 1 = overrange */
/* 2 = underrange */
/* 3 = corrupted data */
@dir int ADC_avg=0; /* 0x0018 */ /* A/D running average */
@dir int CurrentADCavg=0; /* 0x001A */ /* Current delay avg. use to
/* calc. reading */
@dir int noZones=0; /* 0x001C */ /* Total Number of zones */
@dir struct zonestruct Zones[6]=0; /* 0x001E */ /* Zone details
/* Zones[i]=long start */
/* char color */
/* int segment */
@dir char BarDpy2 [BARBYTES]=0; /* 0x0048 */ /* Bar display buffer */
@dir char NumStr2 [NUMBYTES+2]=0; /* 0x0057 */ /* Numeric display buffer */

```

Tricolor Serial Communication ASCII Protocol

7.0 EEPROM.H

```

/*
 * File: eeprom.h
 * Version: 7
 */

/*
 * Revision History
 */

/*
 * 1.0 Initial Isse. RRS.
 * ZF. Added eedata.eebytes.dpydelay. RRS 1/26/96
 * 2. Made ZF option a standard feature. RRS 1/29/96.
 * 3. Added eedata.eebytes.sample_size.
 *    Added eedata.eebytes.signal
 *    Reworked EEPROM structure.
 *    RRS 2/5/96.
 * 4. Changed for readability:
 *
 *           fullscale to digFull
 *           zeroscale to digZero
 *           numfull to barFull
 *           numzero to barZero
 *
 *    RRS 3/28/96
 *
 * 5. Changed the order of the bit fields. RRS 4/18/96.
 * 6. Added RtxZero and RtxFull to eedata.eebytes. These variables
 *    are to be using in conjunction with the AD420 D/A converter
 *    for the new isolated retransmit.
 *    Added calNo to eedata.eebytes. for the purpose of storing a calNo
 *    which can be used for tracking purposes.
 *    RRS 6/17/96.
 * 7. Added variable eedata.eebytes.supervisor2 to allocate space for
 *    future features requiring setup steps.
 *    RRS 7/8/96.
 */

#define EEBASE      0x0E00          /* eeprom base */
#define EEBASE2    0x0E28

struct alarm_tbl
{
    long    trip;    /* alarm trip point */
    char    type;    /* alarm type (hi,lo) */
    char    mode;    /* Std or FailSafe */
    int     seg;     /* alarm segment */
} @eeprom alarmtbl[4] @EEBASE;

union eeunion
{
    struct options
    { /* 1 = allow / 0 = disallow */

        /* Bargraph Features */

        unsigned    :7; /* Not assigned */ /* LSB */
        unsigned    f_super:1; /* Supervisor setup enable */

        unsigned    f_comm:1; /* ASCII communication enable */
        unsigned    f_aRtx:1; /* Analog retransmit enable */
        unsigned    f_peak:1; /* Peak enable */
        unsigned    f_trend:1; /* Trend enable */
        unsigned    f_alarm4:1; /* Alarm 4 enable */
        unsigned    f_alarm3:1; /* Alarm 3 enable */
        unsigned    f_alarm2:1; /* Alarm 2 enable */
    }

```

Tricolor Serial Communication ASCII Protocol

```

unsigned   f_alarm1:1; /* Alarm 1 enable */ /* MSB */

/* Supervisor setup access */
/* MSW */
unsigned   s_barCol:1; /* Bar zone colors setup step */ /* LSB */
unsigned   s_hiCol:1; /* High setpoint default color setup step */
unsigned   s_loCol:1; /* Low setpoint default color setup step */
unsigned   s_bright:1; /* LED brightness control setup step */
unsigned   s_barFlash:1; /* Bar overrange flash enable setup step */
unsigned   s_numFlash:1; /* Numeric overrange flash enable setup step */
unsigned   s_rtxTrack:1; /* Analog retransmit tracking setup step */
unsigned   s_table:1; /* Linearization table setup step */

unsigned   s_sGravity:1; /* Specific Gravity setup step */
unsigned   s_lampTest:1; /* LED LampTest setup step */
unsigned   s_uRate:1; /* Display update rate setup step */
unsigned   s_signal:1; /* Signal input type setup step */
unsigned   s_bsize:1; /* A/D input buffer size setup step */
unsigned   s_unitId:1; /* Unit ID setup step */
unsigned   :2; /* Not assigned */ /* MSB */

/* LSW */
unsigned   s_passwd:1; /* Password setup step */ /* LSB */
unsigned   s_alarm1:1; /* Alarm 1 setup step */
unsigned   s_alarm2:1; /* Alarm 2 setup step */
unsigned   s_alarm3:1; /* Alarm 3 setup step */
unsigned   s_alarm4:1; /* Alarm 4 setup step */
unsigned   s_almHys:1; /* Alarm hysteresis setup step */
unsigned   s_almDelay:1; /* Alarm delay setup step */
unsigned   s_numZero:1; /* Zeroscale setup step */

unsigned   s_numFull:1; /* Fullscale setup step */
unsigned   s_barform:1; /* Barform setup step */
unsigned   s_barZero:1; /* Bar zeroscale setup step */
unsigned   s_barFull:1; /* Bar fullscale setup step */
unsigned   s_oper:1; /* Operator access enable setup step */
unsigned   s_peak:1; /* Peak enable setup step */
unsigned   s_decimal:1; /* Decimal point setup step */
unsigned   s_barFill:1; /* Bar fill mode setup step */ /* MSB */

/* Supervisor setup access - supervisor2 */
unsigned   :8; /* Not assigned */
unsigned   :8; /* Not assigned */
unsigned   :8; /* Not assigned */
unsigned   :8; /* Not assigned */

/* Bit Data Fields */
unsigned   d_peak:1; /* Peak Supervisor Enable Status: 1 = Enabled / 0 = Disabled */
unsigned   d_oper:1; /* Operator Setup Status: 1 = Enabled / 0 = Disabled */
unsigned   d_latch:1; /* Alm. latch mode status: 1 = Latch / 0 = No Latch */
unsigned   d_numFlash:1; /* Numeric overrange flashing: 1 = Enable / 0 = Disable */
unsigned   d_barFlash:1; /* Bar overrange flashing: 1 = Enable / 0 = Displable */
unsigned   d_barFill:1; /* Bar fill mode: 1 = Tricolor / 0 = Banded */
unsigned   d_rtxTrack:1; /* Analog Retransmit Tacking: 1 = bar / 0 = digit */
unsigned   d_table:1; /* 1 = Use linearization table / 0 = two point cal */

unsigned   d_mode:1; /* 0 = Standard / 1 = Spit mode */
unsigned   :7; /* Not Assigned */

} eebits;

struct eememory
{
/* Address */
int   features; /* 0x0E28 */ /* factory enabled features */
long  supervisor; /* 0x0E2A */ /* supervisor setup access */
long  supervisor2; /* 0x0E2E */ /* additional supervisor steps */
int   bitData; /* 0x0E32 */ /* Bit field data */
int   version; /* 0x0E34 */ /* Version Number */
long  calNo; /* 0x0E36 */ /* Unique Calibration number */
char  unitid; /* 0x0E3A */ /* unit id */
char  barform; /* 0x0E3B */ /* bargraph display format */
char  deciplace; /* 0x0E3C */ /* decimal point placement */
int   zeroseg; /* 0x0E3D */ /* zero segment of barform */
}

```

Tricolor Serial Communication ASCII Protocol

```

long    barFull;          /* 0xE3F */ /* bar fullscale value */
long    barZero;         /* 0xE43 */ /* bar zeroscale value */
int     adcfull;        /* 0xE47 */ /* ADC full scale value */
int     adczero;        /* 0xE49 */ /* ADC value @ zero set */
long    digZero;        /* 0xE4B */ /* Digital zeroscale value */
long    digFull;        /* 0xE4F */ /* Digital fullscale value*/
long    hysteresis;     /* 0xE53 */ /* alarm hysteresis value */
long    trendhys;       /* 0xE57 */ /* trend (.5% of numfull) */
float   numfactor;      /* 0xE5B */ /* scaling factor */
float   barfactor;      /* 0xE5F */ /* bargraph scaling factor */
float   pwmfactor;      /* 0xE63 */ /* PWM scaling factor */
float   hystfactor;     /* 0xE67 */ /* hysteresis divide factor */
float   multiplier;     /* 0xE6B */ /* Reading multiplier factor */
long    centerpoint;    /* 0xE6F */ /* value for deviation */
long    barspan;        /* 0xE73 */ /* numeric span from bottom to top of bar */
char    ledctl;         /* 0xE77 */ /* led brightness control */
long    password;       /* 0xE78 */ /* Supervisor password */
char    zonecolor[6];   /* 0xE7C */ /* default bar zone colors */
char    hicolor;        /* 0xE82 */ /* high set point color */
char    locolor;        /* 0xE83 */ /* low set point color */
unsigned int delay;     /* 0xE84 */ /* Delay in milliseconds */
unsigned int dpydelay;  /* 0xE86 */ /* Display upate rate delay */
int     sample_size;    /* 0xE88 */ /* ADC input buffer size */
char    signal;         /* 0xE8A */ /* Signal type */
unsigned int RtxZero;   /* 0xE8B */ /* A. Rtx. zero offset value */
unsigned int RtxFull;   /* 0xE8D */ /* A. Rtx. Full scale value */
int     totalpoints;    /* 0xE8F */ /* Points in scaletable */
int     scaletableIn[50]; /* 0xE91 */ /* Scale table input */
long    scaletableOut[50]; /* 0xEF5 */ /* Scaled Output */
} eebytes;              /* 0xFBD */ /* Next avail. loc */
} @eeprom eedata @EEBASE2; /* 43 bytes remaining */

```

8.0 Configuring the Unit via Serial Communications Port.

Configuring the unit via the serial communications ports involves writing to memory locations in non-volatile EEPROM memory. To provide protection against unintentional writes to configuration parameters, the variable EElock was added. When EElock is cleared, then writes to configurations parameters are allowed. When EElock is set to 1, writes to configuration parameters are inhibited. Thus the configuration sequence is as follows:

1. Clear EElock = 0.
2. Issue write command for configuration parameter.
3. Set EElock = 1.

Table 8 shows several examples of write commands for setting and clearing the EElock variable.

Table 8. Examples of Setting and Clearing EElock.

Operation	Unit Id	Write Command
Setting EElock	00	AW0004000201F8<CR>@
Clearing EElock	00	AW0004000200F9<CR>@
Setting EElock	12	AW1204000201F8<CR>@
Clearing EElock	12	AW1204000200F9<CR>@

Tricolor Serial Communication ASCII Protocol

8.1 Configuring the Alarms Setpoints.

The unit has available up to 4 alarm setpoints. Associated with each setpoint is a trip value, a type, a mode, and a segment parameter. All but the alarmtbl[*i*].seg field is programmable. Table 9 indicates the address locations for these fields.

8.1.1 Setting Alarm Trip Values.

The alarm trip value is the reading at which the alarm will engage. A standard, normally open high alarm will close when the reading equals or exceeds the trip value. In contrast, a standard, normally open low alarms will close when the reading equals or falls below the setpoint value. Table 10 provides examples of commands strings for assigning alarm trip values.

Table 9. Alarm Addresses

Alarm No.	Field	Address	Type
alarmtbl[0]	trip	0x0E00	long
	type	0x0E04	char
	mode	0x0E05	char
	seg	0x0E06	int
alarmtbl[1]	trip	0x0E08	long
	type	0x0E0C	char
	mode	0x0E0D	char
	seg	0x0E0E	int
alarmtbl[2]	trip	0x0E10	long
	type	0x0E14	char
	mode	0x0E15	char
	seg	0x0E16	int
alarmtbl[3]	trip	0x0E18	long
	seg	0x0E1E	int
	mode	0x0E1D	char
	seg	0x0E1E	int

Tricolor Serial Communication ASCII Protocol

Table 10. Examples of Alarm Trip Value Assignments

Operation	Address	Command String
Unit 00, Set alarm 0 to 8000	0x0E00	AW00070E0000001F408B<CR>@
Unit 00, Set alarm 1 to 6000	0x0E08	AW00070E08000017705B<CR>@
Unit 00, Set alarm 2 to 4000	0x0E10	AW00070E1000000FA02B<CR>@
Unit 00, Set alarm 3 to 2000	0x0E18	AW00070E18000007D0FB<CR>@
Unit 00, Set alarm 2 to -6000	0x0E10	AW00070E10FFFFE89064<CR>@
Unit 00, Set alarm 3 to -8000	0x0E18	AW00070E18FFFFE0C034<CR>@

8.1.2. Setting Alarm Types.

Alarms are classified as either high or low setpoints. As previously mentioned, a high setpoint is an alarm that trips when the reading meets or exceeds the setpoint value. In contrast, the low setpoint trips when the reading meets or falls below the setpoint value. The setpoint type is stored in the variable `alarmtbl[].type`. When `alarmtbl[].type = 1`, then the setpoint is of type high. When `alarmtbl[].type = 0`, then the setpoint is of type low. Table 11 contains several examples of configuring the setpoint type via the serial communication link.

Table 11. Examples of Alarm Type Assignments

Operation	Address	Command String
Unit 00, Set alarm 0 to high	0x0E04	AW00040E0401E8<CR>@
Unit 00, Set alarm 1 to high	0x0E0C	AW00040E0C01E0<CR>@
Unit 00, Set alarm 2 to low	0x0E14	AW00040E1400D9<CR>@
Unit 00, Set alarm 3 to low	0x0E1C	AW00040E1C00D1<CR>@

8.1.3. Setting Alarm Mode.

The alarm mode parameter is used in determining how alarms will be tripped. The two modes are standard and failsafe. When the alarm mode is set to standard, they operate as described in section 8.3.2. When the alarm mode is set to failsafe, high setpoints act as lows and low setpoints act as high. Thus a high setpoint in failsafe mode will trip when the reading meets or falls below the setpoint value. Table 12 contains examples of setting alarm modes.

Tricolor Serial Communication ASCII Protocol

Table 12. Examples of Alarm Mode Assignments

Operation	Address	Command String
Unit 00, Set alarm 0 to standard	0x0E05	AW00040E0501E7<CR>@
Unit 00, Set alarm 1 to standard	0x0E0D	AW00040E0D01E0<CR>@
Unit 00, Set alarm 0 to failsafe	0x0E05	AW00040E0500E8<CR>@
Unit 00, Set alarm 1 to failsafe	0x0E0D	AW00040E0D00E0<CR>@

8.1.4. Alarm Segment.

This variable contains segment number corresponding to the setpoint value and is recalculated every time the setpoint value changes. Thus DO NOT WRITE TO THIS VARIABLE !!!

8.2 Setting the Unit Id.

The unit id, which is displayed as a decimal value between 0 and 99 is stored as a binary value in the variable eedata.ebytes.unitid. For example a unit id of 99 is stored as 0x63. Since the unit id is part of the read and write strings, writing to the unit id should occur only when changing the unit id number of a particular bargraph. Table 13 contains examples command strings for changing the unit id.

Table 13. Examples of Unit Id Write Commands

Operation	Write Command String
Change unit 0 to unit 10	AW00040E3A0AA9<CR>@
Change unit 10 to unit 99	AW0A040E3A6350<CR>@
Change unit 99 to unit 15	AW63040E3A0FA4<CR>@
Change unit 15 to unit 1	AW0F040E3A01B2<CR>@
Change unit 1 to unit 0	AW01040E3A00B3<CR>@

8.3 Setting the Barform.

The bargraph can be configured for the following five types of barforms: 0%, 50%, 100%, variable, and deviation. The barform parameter is stored in the variable eedata.ebytes.barform and can be any value between 0 and 4. Table 14 describes the barforms whereas table 15 contains examples of read and writes to eedata.ebytes.barform.

Tricolor Serial Communication ASCII Protocol

Table 14. Description of Barforms

Barform	eedata.ebytes.barform	Description
0%	0	Bar grows from the bottom segment to the top segment as the reading increases from zeroscale to fullscale.
50%	1	Bar grows from the midscale segment to the top segment as the reading increases from zeroscale to fullscale.
100%	2	Bar grows from the top segment to the bottom segment as the reading increases form zeroscale to fullscale.
Variable	3	Bar grows from the bottom segment to the top segment as the reading increases from the bar zeroscale value to the bar fullscale value.
Deviation	4	Bar grows from the midscale segment to the top segment as the reading increases from the the bar midscale value to the bar fullscale value.

Table 15. Examples of Reading and Writing to the Barform Parameter.

Operation	Command String
Read unit 00 barform	AR000E3B01B5<CR>@
Change barform to 0%	AW00040E3B00B2<CR>@
Change barform to 50%	AW00040E3B01B1<CR>@
Change barform to 100%	AW00040E3B02B0<CR>@
Change barform to Variable	AW00040E3B03AF<CR>@
Change barform to Deviation	AW00040E3B04AE<CR>@

8.4 Setting the Decimal Place Location.

The decimal place parameter is stored in the variable eedata.ebytes.deciplace. Table 16 defines valid decimal place values. To change the decimal point location, simply write the appropriate value to the address of eedata.ebytes.deciplace. Table 17 contains several examples retrieving and changing the decimal point location.

Tricolor Serial Communication ASCII Protocol

Table 16. Decimal Point Definitions

Description	eedata.eebytes.decipalce
XXXXX.	0
XXXX.X	1
XXX.XX	2
XX.XXX	3
X.XXXX	4
XXXXX	5

Table 17. Examples of Reading and Writing to the Decimal Point Parameter.

Operation	Command String
Read unit 00 decimal place	AR000E3C01B4<CR>@
Change decimal place to XXXXX.	AW00040E3C00B1<CR>@
Change decimal place to XXXX.X	AW00040E3C01B0<CR>@
Change decimal place to XXX.XX	AW00040E3C02AF<CR>@
Change decimal place to XX.XXX	AW00040E3C03AE<CR>@
Change decimal palce to X.XXXX	AW00040E3C04AD<CR>@
Change to no decimal point	AW00040E3C05AC<CR>@

Tricolor Serial Communication ASCII Protocol

Appendix. EEPROM SETUP FIELD

Address	Data Type	Function
0x0E00	Long	Alarmtbl[0].trip
0x0E04	Char	Alarmtbl[0].type; High/Low
0x0E05	Char	Alarmtbl[0].mode; Standard/failsafe
0x0E06	Int	Alarmtbl[0].seg
0x0E08	Long	Alarmtbl[1].trip
0x0E0C	Char	Alarmtbl[1].type; High/Low
0x0E0D	Char	Alarmtbl[1].mode; Standard/failsafe
0x0E0E	Int	Alarmtbl[1].seg
0x0E10	Long	Alarmtbl[2].trip
0x0E14	Char	Alarmtbl[2].type; High/Low
0x0E15	Char	Alarmtbl[2].mode; Standard/failsafe
0x0E16	Int	Alarmtbl[2].seg
0x0E18	Long	Alarmtbl[3].trip
0x0E1C	Char	Alarmtbl[3].type; High/Low
0x0E1D	Char	Alarmtbl[3].mode; Standard/failsafe
0x0E1E	Int	Alarmtbl[3].seg
0x0E28	Int	Factory enabled features; features
0x0E2A	Long	Supervisor setup access; supervisor
0x0E2E	Long	Additional supervisor steps; supervisor2
0x0E32	Int	Bit field data; bitdata
0x0E34	Int	Version Number; version
0x0E36	Long	Unique Calibration number; calNo
0x0E3A	Char	Unit id; unitid
0x0E3B	Char	Bargraph display format; barform
0x0E3C	Char	Decimal point placement; deciplace
0x0E3D	Int	Zero segment of barform; zeroseg
0x0E3F	Long	Bar fullscale value; barFull
0x0E43	Long	Bar zeroscale value; barZero

Tricolor Serial Communication ASCII Protocol

0x0E47	Int	ADC full scale value; adcfull
0x0E49	Int	ADC value @ zero set; adczero;
0x0E4B	Long	Digital zeroscale value; digZero
0x0E4F	Long	Digital fullscale value; digFull
0x0E53	Long	Alarm hysteresis value; hysteresis
0x0E57	Long	Trend (.5% of numfull); trendhys
0x0E5B	Float	Scaling factor; numfactor
0x0E5F	Float	Bargraph scaling factor; barfactor
0x0E63	Float	PWM scaling factor; pwmfactor
0x0E67	Float	Hysteresis divide factor; hystfactor
0x0E6B	Float	Reading multiplier factor; multiplier
0x0E6F	Long	Value for deviation; center point
0x0E73	Long	Numeric span from bottom to top of bar; barspan
0x0E77	Char	Led brightness control; ledctl
0x0E78	Long	Supervisor password; password
0x0E7C	Char	Default bar zone colors; zonecolor[6]
0x0E82	Char	High set point color; hicolor
0x0E83	Char	Low set point color; locolor
0x0E84	Unsigned int	Delay in milliseconds; delay
0x0E86	Unsigned int	Display update rate delay; dpydelay
0x0E88	Int	ADC input buffer size; sample_size
0x0E8A	char	Signal type; signal
0x0E8B	Unsigned int	Rtx. zero offset value; RtxZero
0x0E8D	Unsigned int	A. Rtx. Full scale value; RtxFull
0x0E8F	Int	Points in scale table; total points
0x0E91	Int	Scale table input; scaletableIn[50]
0x0EF5	Long	Scaled output; scaletableOut[50]