```
/****************************************************************************/
/*                                                                          */
/*                                 MENUACT                                  */
/*                          Menu Action Functions                           */
/*                       Digital Oscilloscope Project                       */
/*                                 EE/CS 52                                 */
/*                                                                          */
/****************************************************************************/

/*
   This file contains the functions for carrying out menu actions for the
   Digital Oscilloscope project.  These functions are invoked when the <Left>
   or <Right> key is pressed for a menu item.  Also included are the functions
   for displaying the current menu option selection.  The functions included
   are:
      display_mode      - display trigger mode
      display_scale     - display the scale type
      display_sweep     - display the sweep rate
      display_trg_delay - display the tigger delay
      display_trg_level - display the trigger level
      display_trg_slope - display the trigger slope
      get_trigger_mode  - get the current trigger mode
      mode_down         - go to the "next" trigger mode
      mode_up           - go to the "previous" trigger mode
      no_display        - nothing to display for option setting
      no_menu_action    - no action to perform for <Left> or <Right> key
      scale_down        - go to the "next" scale type
      scale_up          - go to the "previous" scale type
      set_scale         - set the scale type
      set_sweep         - set the sweep rate
      set_trg_delay     - set the tigger delay
      set_trg_level     - set the trigger level
      set_trg_slope     - set the trigger slope
      set_trigger_mode  - set the trigger mode
      sweep_down        - decrease the sweep rate
      sweep_up          - increase the sweep rate
      trg_delay_down    - decrease the trigger delay
      trg_delay_up      - increase the trigger delay
      trg_level_down    - decrease the trigger level
      trg_level_up      - increase the trigger level
      trg_slope_toggle  - toggle the trigger slope between "+" and "-"

   The local functions included are:
      adjust_trg_delay  - adjust the trigger delay for a new sweep rate
      cvt_num_field     - converts a numeric field value to a string

   The locally global variable definitions included are:
      delay         - current trigger delay
      level         - current trigger level
      scale         - current display scale type
      slope         - current trigger slope
      sweep         - current sweep rate
      sweep_rates   - table of information on possible sweep rates
      trigger_mode  - current triggering mode


   Revision History
      3/8/94   Glen George       Initial revision.
      3/13/94  Glen George       Updated comments.
      3/13/94  Glen George       Changed all arrays of constant strings to be
                        static so compiler generates correct code.
      3/13/94  Glen George       Changed scale to type enum scale_type and
                        output the selection as "None" or "Axes".
                    This will allow for easier future expansion.
      3/13/94  Glen George       Changed name of set_axes function (in
                        tracutil.c) to set_display_scale.
      3/10/95  Glen George       Changed calculation of displayed trigger
                        level to use constants MIN_TRG_LEVEL_SET and
                    MAX_TRG_LEVEL_SET to get the trigger level
                    range.
      3/17/97  Glen George       Updated comments.
      5/3/06   Glen George       Changed sweep definitions to include new
                        sweep rates of 100 ns, 200 ns, 500 ns, and
                        1 us and updated functions to handle these
                    new rates.
      5/9/06   Glen George       Added new a triggering mode (automatic
                                 triggering) and a new scale (grid) and
                                 updated functions to implement these options.
      5/9/06   Glen George       Added functions for setting the triggering
                                 mode and scale by going up and down the list
                                 of possibilities instead of just toggling
                                 between one of two possibilities (since there
                 are more than two now).
      5/9/06   Glen George       Added accessor function (get_trigger_mode)
                                 to be able to get the current trigger mode.
      6/6/14   Santiago Navonne  Added fastest sweep rate and changed their
                                 values to reflect actual possible rates.
      6/11/14  Santiago Navonne  Modified delay set function to support faster
                                 sweep rates.
*/



/* library include files */
  /* none */

/* local include files */
#include  "interfac.h"
#include  "scopedef.h"
#include  "lcdout.h"
#include  "menuact.h"
#include  "tracutil.h"




/* local function declarations */
static void  adjust_trg_delay(int, int);       /* adjust the trigger delay for new sweep */
static void  cvt_num_field(long int, char *);   /* convert a number to a string */




/* locally global variables

/* trace parameters */
static enum trigger_type   trigger_mode; /* current triggering mode */
static enum scale_type     scale;    /* current scale type */
static int         sweep;        /* sweep rate index */
static int         level;    /* current trigger level */
static enum slope_type     slope;    /* current trigger slope */
static long int            delay;    /* current trigger delay */

/* sweep rate information */
static const struct sweep_info  sweep_rates[] =
    { { 19000000L, " 52 ns " },
      {  9500000L, " 104 ns" },
      {  4750000L, " 208 ns" },
      {  2000000L, " 500 ns" },
      {  1000000L, " 1 \004s  " },
      {   500000L, " 2 \004s  " },
      {   200000L, " 5 \004s  " },
      {   100000L, " 10 \004s " },
      {    50000L, " 20 \004s " },
      {    20000L, " 50 \004s " },
      {    10000L, " 100 \004s" },
      {     5000L, " 200 \004s" },
      {     2000L, " 500 \004s" },
      {     1000L, " 1 ms  "    },
      {      500L, " 2 ms  "    },
      {      200L, " 5 ms  "    },
      {      100L, " 10 ms "    },
      {       50L, " 20 ms "    } };




/*
   no_menu_action

   Description:      This function handles a menu action when there is nothing
                     to be done.  It just returns.

   Arguments:        None.
   Return Value:     None.

   Input:            None.
   Output:           None.

   Error Handling:   None.

   Algorithms:       None.
   Data Structures:  None.

   Global Variables: None.

   Author:           Glen George
   Last Modified:    Mar. 8, 1994

*/

void  no_menu_action()
{
    /* variables */
      /* none */



    /* nothing to do - return */
    return;

}




/*
   no_display

   Description:      This function handles displaying a menu option's setting
                     when there is nothing to display.  It just returns,
		     ignoring all arguments.

   Arguments:        x_pos (int) - x position (in character cells) at which to
			   display the menu option (not used).
		     y_pos (int) - y position (in character cells) at which to
			   display the menu option (not used).
		     style (int) - style with which to display the menu option
			       (not used).
   Return Value:     None.

   Input:            None.
   Output:           None.

   Error Handling:   None.

   Algorithms:       None.
   Data Structures:  None.

   Global Variables: None.

   Author:           Glen George
   Last Modified:    Mar. 8, 1994

*/

void  no_display(int x_pos, int y_pos, int style)
{
    /* variables */
      /* none */



    /* nothing to do - return */
    return;
```

```c

}




/*
   set_trigger_mode

   Description:      This function sets the triggering mode to the passed
                     value.

   Arguments:        m (enum trigger_type) - mode to which to set the
                         triggering mode.
   Return Value:     None.

   Input:            None.
   Output:           None.

   Error Handling:   None.

   Algorithms:       None.
   Data Structures:  None.

   Global Variables: trigger_mode - initialized to the passed value.

   Author:           Glen George
   Last Modified:    Mar. 8, 1994

*/

void  set_trigger_mode(enum trigger_type m)
{
    /* variables */
      /* none */



    /* set the trigger mode */
    trigger_mode = m;

    /* set the new mode */
    set_mode(trigger_mode);


    /* all done setting the trigger mode - return */
    return;

}




/*
   get_trigger_mode

   Description:      This function returns the current triggering mode.

   Arguments:        None.
   Return Value:     (enum trigger_type) - current triggering mode.

   Input:            None.
   Output:           None.

   Error Handling:   None.

   Algorithms:       None.
   Data Structures:  None.

   Global Variables: trigger_mode - value is returned (not changed).

   Author:           Glen George
   Last Modified:    May 9, 2006

*/
```

```

enum trigger_type  get_trigger_mode()
{
    /* variables */
      /* none */



    /* return the current trigger mode */
    return  trigger_mode;

}




/*
   mode_down

   Description:      This function handles moving down the list of trigger
                     modes.  It changes to the "next" triggering mode and
                     sets that as the current mode.

   Arguments:        None.
   Return Value:     None.

   Input:            None.
   Output:           None.

   Error Handling:   None.

   Algorithms:       None.
   Data Structures:  None.

   Global Variables: trigger_mode - changed to "next" trigger mode.

   Author:           Glen George
   Last Modified:    May 9, 2006

*/

void  mode_down()
{
    /* variables */
      /* none */



    /* move to the "next" triggering mode */
    if (trigger_mode == NORMAL_TRIGGER)
        trigger_mode = AUTO_TRIGGER;
    else if (trigger_mode == AUTO_TRIGGER)
        trigger_mode = ONESHOT_TRIGGER;
    else
        trigger_mode = NORMAL_TRIGGER;

    /* set the new mode */
    set_mode(trigger_mode);


    /* all done with the trigger mode - return */
    return;

}




/*
   mode_up

   Description:      This function handles moving up the list of trigger
                     modes.  It changes to the "previous" triggering mode and
                     sets that as the current mode.

   Arguments:        None.
   Return Value:     None.

   Input:            None.
   Output:           None.

   Error Handling:   None.

   Algorithms:       None.
   Data Structures:  None.

   Global Variables: trigger_mode - changed to "previous" trigger mode.

   Author:           Glen George
   Last Modified:    May 9, 2006

*/

void  mode_up()
{
    /* variables */
      /* none */



    /* move to the "previous" triggering mode */
    if (trigger_mode == NORMAL_TRIGGER)
        trigger_mode = ONESHOT_TRIGGER;
    else if (trigger_mode == AUTO_TRIGGER)
        trigger_mode = NORMAL_TRIGGER;
    else
        trigger_mode = AUTO_TRIGGER;

    /* set the new mode */
    set_mode(trigger_mode);


    /* all done with the trigger mode - return */
    return;

}




/*
   display_mode

   Description:      This function displays the current triggering mode at the
                     passed position, in the passed style.

   Arguments:        x_pos (int) - x position (in character cells) at which to
                   display the trigger mode.
             y_pos (int) - y position (in character cells) at which to
                   display the trigger mode.
             style (int) - style with which to display the trigger
                       mode.
   Return Value:     None.

   Input:            None.
   Output:           The trigger mode is displayed at the passed position on
             the screen.

   Error Handling:   None.

   Algorithms:       None.
   Data Structures:  None.

   Global Variables: trigger_mode - determines which string is displayed.

   Author:           Glen George
   Last Modified:    May 9, 2006

*/
```

```
void  display_mode(int x_pos, int y_pos, int style)
{
    /* variables */

    /* the mode strings (must match enumerated type) */
    const static char * const  modes[] =  {  " Normal   ",
                                             " Automatic",
                                             " One-Shot "  };



    /* display the trigger mode */
    plot_string(x_pos, y_pos, modes[trigger_mode], style);


    /* all done displaying the trigger mode - return */
    return;

}




/*
   set_scale

   Description:      This function sets the scale type to the passed value.

   Arguments:        s (enum scale_type) - scale type to which to initialize
                       the scale status.
   Return Value:     None.

   Input:            None.
   Output:           The new trace display is updated with the new scale.

   Error Handling:   None.

   Algorithms:       None.
   Data Structures:  None.

   Global Variables: scale - initialized to the passed value.

   Author:           Glen George
   Last Modified:    Mar. 13, 1994

*/

void  set_scale(enum scale_type s)
{
    /* variables */
      /* none */



    /* set the scale type */
    scale = s;

    /* output the scale appropriately */
    set_display_scale(scale);


    /* all done setting the scale type - return */
    return;

}




/*
   scale_down

   Description:      This function handles moving down the list of scale
                     types.  It changes to the "next" type of scale and sets
              this as the current scale type.

   Arguments:        None.
   Return Value:     None.

   Input:            None.
   Output:           The new scale is output to the trace display.

   Error Handling:   None.

   Algorithms:       None.
   Data Structures:  None.

   Global Variables: scale - changed to the "next" scale type.

   Author:           Glen George
   Last Modified:    May 9, 2006

*/

void  scale_down()
{
    /* variables */
      /* none */



    /* change to the "next" scale type */
    if (scale == SCALE_NONE)
        scale = SCALE_AXES;
    else if (scale == SCALE_AXES)
        scale = SCALE_GRID;
    else
        scale = SCALE_NONE;

    /* set the scale type */
    set_display_scale(scale);


    /* all done with toggling the scale type - return */
    return;

}




/*
   scale_up

   Description:      This function handles moving up the list of scale types.
                     It changes to the "previous" type of scale and sets this
		     as the current scale type.

   Arguments:        None.
   Return Value:     None.

   Input:            None.
   Output:           The new scale is output to the trace display.

   Error Handling:   None.

   Algorithms:       None.
   Data Structures:  None.

   Global Variables: scale - changed to the "previous" scale type.

   Author:           Glen George
   Last Modified:    May 9, 2006

*/

void  scale_up()
{
    /* variables */
      /* none */
```

```c



    /* change to the "previous" scale type */
    if (scale == SCALE_NONE)
        scale = SCALE_GRID;
    else if (scale == SCALE_AXES)
        scale = SCALE_NONE;
    else
        scale = SCALE_AXES;

    /* set the scale type */
    set_display_scale(scale);


    /* all done with toggling the scale type - return */
    return;

}




/*
   display_scale

   Description:      This function displays the current scale type at the
                     passed position, in the passed style.

   Arguments:        x_pos (int) - x position (in character cells) at which to
                   display the scale type.
             y_pos (int) - y position (in character cells) at which to
                   display the scale type.
             style (int) - style with which to display the scale type.
   Return Value:     None.

   Input:            None.
   Output:           The scale type is displayed at the passed position on the
             display.

   Error Handling:   None.

   Algorithms:       None.
   Data Structures:  None.

   Global Variables: scale - determines which string is displayed.

   Author:           Glen George
   Last Modified:    Mar. 13, 1994

*/

void  display_scale(int x_pos, int y_pos, int style)
{
    /* variables */

    /* the scale type strings (must match enumerated type) */
    const static char * const  scale_stat[] =  {  " None",
                                                  " Axes",
                                                  " Grid"  };



    /* display the scale status */
    plot_string(x_pos, y_pos, scale_stat[scale], style);


    /* all done displaying the scale status - return */
    return;

}
```

```
/*
   set_sweep

   Description:      This function sets the sweep rate to the passed value.
                     The passed value gives the sweep rate to choose from the
             list of sweep rates (it gives the list index).

   Arguments:        s (int) - index into the list of sweep rates to which to
                   set the current sweep rate.
   Return Value:     None.

   Input:            None.
   Output:           None.

   Error Handling:   The passed index is not checked for validity.

   Algorithms:       None.
   Data Structures:  None.

   Global Variables: sweep - initialized to the passed value.

   Author:           Glen George
   Last Modified:    Mar. 8, 1994

*/

void  set_sweep(int s)
{
    /* variables */
    int  sample_size;       /* sample size for this sweep rate */



    /* set the new sweep rate */
    sweep = s;

    /* set the sweep rate for the hardware */
    sample_size = set_sample_rate(sweep_rates[sweep].sample_rate);
    /* also set the sample size for the trace capture */
    set_trace_size(sample_size);


    /* all done initializing the sweep rate - return */
    return;

}




/*
   sweep_down

   Description:      This function handles decreasing the current sweep rate.
             The new sweep rate (and sample size) is sent to the
             hardware (and trace routines).  If an attempt is made to
             lower the sweep rate below the minimum value it is not
             changed.  This routine also updates the sweep delay based
             on the new sweep rate (to keep the delay time constant).

   Arguments:        None.
   Return Value:     None.

   Input:            None.
   Output:           None.

   Error Handling:   None.

   Algorithms:       None.
   Data Structures:  None.

   Global Variables: sweep - decremented if not already 0.
             delay - increased to keep delay time constant.

   Known Bugs:       The updated delay time is not displayed.  Since the time
```

```
             is typically only rounded to the new sample rate, this is
             not a major problem.

   Author:           Glen George
   Last Modified:    Mar. 8, 1994

*/

void  sweep_down()
{
    /* variables */
    int  sample_size;       /* sample size for the new sweep rate */



    /* decrease the sweep rate, if not already the minimum */
    if (sweep > 0)  {
        /* not at minimum, adjust delay for new sweep */
    adjust_trg_delay(sweep, (sweep - 1));
    /* now set new sweep rate */
        sweep--;
    }

    /* set the sweep rate for the hardware */
    sample_size = set_sample_rate(sweep_rates[sweep].sample_rate);
    /* also set the sample size for the trace capture */
    set_trace_size(sample_size);


    /* all done with lowering the sweep rate - return */
    return;

}




/*
   sweep_up

   Description:      This function handles increasing the current sweep rate.
             The new sweep rate (and sample size) is sent to the
             hardware (and trace routines).  If an attempt is made to
             raise the sweep rate above the maximum value it is not
             changed.  This routine also updates the sweep delay based
             on the new sweep rate (to keep the delay time constant).

   Arguments:        None.
   Return Value:     None.

   Input:            None.
   Output:           None.

   Error Handling:   None.

   Algorithms:       None.
   Data Structures:  None.

   Global Variables: sweep - incremented if not already the maximum value.
             delay - decreased to keep delay time constant.

   Known Bugs:       The updated delay time is not displayed.  Since the time
             is typically only rounded to the new sample rate, this is
             not a major problem.

   Author:           Glen George
   Last Modified:    Mar. 8, 1994

*/

void  sweep_up()
{
    /* variables */
    int  sample_size;       /* sample size for the new sweep rate */



    /* increase the sweep rate, if not already the maximum */
    if (sweep < (NO_SWEEP_RATES - 1))  {
        /* not at maximum, adjust delay for new sweep */
    adjust_trg_delay(sweep, (sweep + 1));
    /* now set new sweep rate */
        sweep++;
    }

    /* set the sweep rate for the hardware */
    sample_size = set_sample_rate(sweep_rates[sweep].sample_rate);
    /* also set the sample size for the trace capture */
    set_trace_size(sample_size);


    /* all done with raising the sweep rate - return */
    return;

}




/*
   display_sweep

   Description:      This function displays the current sweep rate at the
                     passed position, in the passed style.

   Arguments:        x_pos (int) - x position (in character cells) at which to
                   display the sweep rate.
             y_pos (int) - y position (in character cells) at which to
                   display the sweep rate.
             style (int) - style with which to display the sweep rate.
   Return Value:     None.

   Input:            None.
   Output:           The sweep rate is displayed at the passed position on the
             display.

   Error Handling:   None.

   Algorithms:       None.
   Data Structures:  None.

   Global Variables: sweep - determines which string is displayed.

   Author:           Glen George
   Last Modified:    Mar. 8, 1994

*/

void  display_sweep(int x_pos, int y_pos, int style)
{
    /* variables */
      /* none */



    /* display the sweep rate */
    plot_string(x_pos, y_pos, sweep_rates[sweep].s, style);


    /* all done displaying the sweep rate - return */
    return;

}




/*
   set_trg_level

   Description:      This function sets the trigger level to the passed value.

   Arguments:        l (int) - value to which to set the trigger level.
   Return Value:     None.

   Input:            None.
   Output:           None.

   Error Handling:   The passed value is not checked for validity.

   Algorithms:       None.
   Data Structures:  None.

   Global Variables: level - initialized to the passed value.

   Author:           Glen George
   Last Modified:    Mar. 8, 1994

*/

void  set_trg_level(int l)
{
    /* variables */
      /* none */



    /* set the trigger level */
    level = l;

    /* set the trigger level in hardware too */
    set_trigger(level, slope);


    /* all done initializing the trigger level - return */
    return;

}




/*
   trg_level_down

   Description:      This function handles decreasing the current trigger
             level.  The new trigger level is sent to the hardware.
             If an attempt is made to lower the trigger level below
             the minimum value it is not changed.

   Arguments:        None.
   Return Value:     None.

   Input:            None.
   Output:           None.

   Error Handling:   None.

   Algorithms:       None.
   Data Structures:  None.

   Global Variables: level - decremented if not already at the minimum value.

   Author:           Glen George
   Last Modified:    Mar. 8, 1994

*/

void  trg_level_down()
{
    /* variables */
      /* none */
    /* decrease the trigger level, if not already the minimum */
    if (level > MIN_TRG_LEVEL_SET)
        level--;

    /* set the trigger level for the hardware */
    set_trigger(level, slope);


    /* all done with lowering the trigger level - return */
    return;

}




/*
   trg_level_up

   Description:      This function handles increasing the current trigger
             level.  The new trigger level is sent to the hardware.
             If an attempt is made to raise the trigger level above
             the maximum value it is not changed.

   Arguments:        None.
   Return Value:     None.

   Input:            None.
   Output:           None.

   Error Handling:   None.

   Algorithms:       None.
   Data Structures:  None.

   Global Variables: level - incremented if not already the maximum value.

   Author:           Glen George
   Last Modified:    Mar. 8, 1994

*/

void  trg_level_up()
{
    /* variables */
      /* none */



    /* increase the trigger level, if not already the maximum */
    if (level < MAX_TRG_LEVEL_SET)
        level++;

    /* tell the hardware the new trigger level */
    set_trigger(level, slope);


    /* all done raising the trigger level - return */
    return;

}




/*
   display_trg_level

   Description:      This function displays the current trigger level at the
                     passed position, in the passed style.

   Arguments:        x_pos (int) - x position (in character cells) at which to
                   display the trigger level.
             y_pos (int) - y position (in character cells) at which to
                   display the trigger level.
             style (int) - style with which to display the trigger
                       level.
   Return Value:     None.

   Input:            None.
   Output:           The trigger level is displayed at the passed position on
             the display.

   Error Handling:   None.

   Algorithms:       None.
   Data Structures:  None.

   Global Variables: level - determines the value displayed.

   Author:           Glen George
   Last Modified:    Mar. 10, 1995

*/

void  display_trg_level(int x_pos, int y_pos, int style)
{
    /* variables */
    char      level_str[] = "        "; /* string containing the trigger level */
    long int  l;            /* trigger level in mV */



    /* compute the trigger level in millivolts */
    l = ((long int) MAX_LEVEL - MIN_LEVEL) * level / (MAX_TRG_LEVEL_SET - MIN_TRG_LEVEL_SET) + MIN_LEV

    /* convert the level to the string (leave first character blank) */
    cvt_num_field(l, &level_str[1]);

    /* add in the units */
    level_str[7] = 'V';


    /* now finally display the trigger level */
    plot_string(x_pos, y_pos, level_str, style);


    /* all done displaying the trigger level - return */
    return;

}




/*
   set_trg_slope

   Description:      This function sets the trigger slope to the passed value.

   Arguments:        s (enum slope_type) - trigger slope type to which to set
                       the locally global slope.
   Return Value:     None.

   Input:            None.
   Output:           None.

   Error Handling:   None.

   Algorithms:       None.
   Data Structures:  None.

   Global Variables: slope - set to the passed value.

   Author:           Glen George
   Last Modified:    Mar. 8, 1994

*/

void  set_trg_slope(enum slope_type s)
```

```
{
    /* variables */
      /* none */



    /* set the slope type */
    slope = s;

    /* also tell the hardware what the slope is */
    set_trigger(level, slope);


    /* all done setting the trigger slope - return */
    return;

}




/*
   trg_slope_toggle

   Description:      This function handles toggling (and setting) the current
                     trigger slope.

   Arguments:        None.
   Return Value:     None.

   Input:            None.
   Output:           None.

   Error Handling:   None.

   Algorithms:       None.
   Data Structures:  None.

   Global Variables: slope - toggled.

   Author:           Glen George
   Last Modified:    Mar. 8, 1994

*/

void  trg_slope_toggle()
{
    /* variables */
      /* none */



    /* toggle the trigger slope */
    if (slope == SLOPE_POSITIVE)
        slope = SLOPE_NEGATIVE;
    else
        slope = SLOPE_POSITIVE;

    /* set the new trigger slope */
    set_trigger(level, slope);


    /* all done with the trigger slope - return */
    return;

}




/*
   display_trg_slope

   Description:      This function displays the current trigger slope at the
                     passed position, in the passed style.

   Arguments:        x_pos (int) - x position (in character cells) at which to
                   display the trigger slope.
             y_pos (int) - y position (in character cells) at which to
                   display the trigger slope.
             style (int) - style with which to display the trigger
                       slope.
   Return Value:     None.

   Input:            None.
   Output:           The trigger slope is displayed at the passed position on
             the screen.

   Error Handling:   None.

   Algorithms:       None.
   Data Structures:  None.

   Global Variables: slope - determines which string is displayed.

   Author:           Glen George
   Last Modified:    Mar. 13, 1994

*/

void  display_trg_slope(int x_pos, int y_pos, int style)
{
    /* variables */

    /* the trigger slope strings (must match enumerated type) */
    const static char * const  slopes[] =  {  " +", " -"  };



    /* display the trigger slope */
    plot_string(x_pos, y_pos, slopes[slope], style);


    /* all done displaying the trigger slope - return */
    return;

}




/*
   set_trg_delay

   Description:      This function sets the trigger delay to the passed value.

   Arguments:        d (long int) - value to which to set the trigger delay.
   Return Value:     None.

   Input:            None.
   Output:           None.

   Error Handling:   The passed value is not checked for validity.

   Algorithms:       None.
   Data Structures:  None.

   Global Variables: delay - initialized to the passed value.

   Author:           Glen George
   Last Modified:    Mar. 8, 1994

*/

void  set_trg_delay(long int d)
{
    /* variables */
      /* none */



    /* set the trigger delay */
    delay = d;

    /* set the trigger delay in hardware too */
    set_delay(delay);


    /* all done initializing the trigger delay - return */
    return;

}




/*
   trg_delay_down

   Description:      This function handles decreasing the current trigger
             delay.  The new trigger delay is sent to the hardware.
             If an attempt is made to lower the trigger delay below
             the minimum value it is not changed.

   Arguments:        None.
   Return Value:     None.

   Input:            None.
   Output:           None.

   Error Handling:   None.

   Algorithms:       None.
   Data Structures:  None.

   Global Variables: delay - decremented if not already at the minimum value.

   Author:           Glen George
   Last Modified:    Mar. 8, 1994

*/

void  trg_delay_down()
{
    /* variables */
      /* none */



    /* decrease the trigger delay, if not already the minimum */
    if (delay > MIN_DELAY)
        delay--;

    /* set the trigger delay for the hardware */
    set_delay(delay);


    /* all done with lowering the trigger delay - return */
    return;

}




/*
   trg_delay_up

   Description:      This function handles increasing the current trigger
             delay.  The new trigger delay is sent to the hardware.
             If an attempt is made to raise the trigger delay above
             the maximum value it is not changed.

   Arguments:        None.
   Return Value:     None.

   Input:            None.
   Output:           None.

   Error Handling:   None.

   Algorithms:       None.
   Data Structures:  None.

   Global Variables: delay - incremented if not already the maximum value.

   Author:           Glen George
   Last Modified:    Mar. 8, 1994

*/

void  trg_delay_up()
{
    /* variables */
      /* none */



    /* increase the trigger delay, if not already the maximum */
    if (delay < MAX_DELAY)
        delay++;

    /* tell the hardware the new trigger delay */
    set_delay(delay);


    /* all done raising the trigger delay - return */
    return;

}




/*
   adjust_trg_delay

   Description:      This function adjusts the trigger delay for a new sweep
             rate.  The factor to adjust the delay by is determined
             by looking up the sample rates in the sweep_rates array.
             If the delay goes out of range, due to the adjustment it
             is reset to the maximum or minimum valid value.

   Arguments:        old_sweep (int) - old sweep rate (index into sweep_rates
                       array).
             new_sweep (int) - new sweep rate (index into sweep_rates
                       array).
   Return Value:     None.

   Input:            None.
   Output:           None.

   Error Handling:   None.

   Algorithms:       The delay is multiplied by 10 times the ratio of the
             sweep sample rates then divided by 10.  This is done to
             avoid floating point arithmetic and integer truncation
             problems.
   Data Structures:  None.

   Global Variables: delay - adjusted based on passed sweep rates.

   Known Bugs:       The updated delay time is not displayed.  Since the time
             is typically only rounded to the new sample rate, this is
             not a major problem.

   Author:           Glen George
   Last Modified:    Mar. 8, 1994

*/
```

```c

static void  adjust_trg_delay(int old_sweep, int new_sweep)
{
    /* variables */
      /* none */



    /* multiply by 10 times the ratio of sweep rates */
    delay *= (10 * sweep_rates[new_sweep].sample_rate) / sweep_rates[old_sweep].sample_rate;
    /* now divide the factor of 10 back out */
    delay /= 10;

    /* make sure delay is not out of range */
    if (delay > MAX_DELAY)
        /* delay is too large - set to maximum */
        delay = MAX_DELAY;
    if (delay < MIN_DELAY)
        /* delay is too small - set to minimum */
    delay = MIN_DELAY;


    /* tell the hardware the new trigger delay */
    set_delay(delay);


    /* all done adjusting the trigger delay - return */
    return;

}




/*
   display_trg_delay

   Description:      This function displays the current trigger delay at the
                     passed position, in the passed style.

   Arguments:        x_pos (int) - x position (in character cells) at which to
                   display the trigger delay.
             y_pos (int) - y position (in character cells) at which to
                   display the trigger delay.
             style (int) - style with which to display the trigger
                       delay.
   Return Value:     None.

   Input:            None.
   Output:           The trigger delay is displayed at the passed position on
             the display.

   Error Handling:   None.

   Algorithms:       None.
   Data Structures:  None.

   Global Variables: delay - determines the value displayed.

   Author:           Glen George
   Last Modified:    June 11, 2014

*/

void  display_trg_delay(int x_pos, int y_pos, int style)
{
    /* variables */
    char      delay_str[] = "         "; /* string containing the trigger delay */
    long int  units_adj;         /* adjustment to get to microseconds */

    long int  d;                         /* delay in appropriate units */
    float     temp_d;                    /* delay in float to avoid overflows */

    /* compute the delay in the appropriate units */
    /* have to watch out for overflow, so use float temp */
    if (sweep_rates[sweep].sample_rate > 1000000L)  {
        /* have a fast sweep rate  */
        /* first compute with float to avoid overflow */
        temp_d = delay * (1000000000L / sweep_rates[sweep].sample_rate);

    /* now convert to int */
    d = (int) temp_d;
    /* need to divide by 1000 to get to microseconds */
    units_adj = 1000;
    }
    else  {
        /* slow sweep rate, don't have to worry about overflow */
        d = delay * (1000000L / sweep_rates[sweep].sample_rate);
    /* already in microseconds, so adjustment is 1 */
    units_adj = 1;
    }

    /* convert it to the string (leave first character blank) */
    cvt_num_field(d, &delay_str[1]);

    /* add in the units */
    if (((d / units_adj) < 1000) && ((d / units_adj) > -1000) && (units_adj == 1000)) {
        /* delay is in microseconds */
    delay_str[7] = '\004';
    delay_str[8] = 's';
    }
    else if (((d / units_adj) < 1000000) && ((d / units_adj) > -1000000)) {
        /* delay is in milliseconds */
    delay_str[7] = 'm';
    delay_str[8] = 's';
    }
    else if (((d / units_adj) < 1000000000) && ((d / units_adj) > -1000000000))  {
        /* delay is in seconds */
    delay_str[7] = 's';
    delay_str[8] = ' ';
    }
    else  {
        /* delay is in kiloseconds */
    delay_str[7] = 'k';
    delay_str[8] = 's';
    }


    /* now actually display the trigger delay */
    plot_string(x_pos, y_pos, delay_str, style);


    /* all done displaying the trigger delay - return */
    return;

}




/*
   cvt_num_field

   Description:      This function converts the passed number (numeric field
                     value) to a string and returns that in the passed string
             reference.  The number may be signed, and a sign (+ or -)
             is always generated.  The number is assumed to have three
             digits to the right of the decimal point.  Only the four
             most significant digits of the number are displayed and
             the decimal point is shifted appropriately.  (Four digits
             are always generated by the function).

   Arguments:        n (long int) - numeric field value to convert.
             s (char *)   - pointer to string in which to return the
                        converted field value.
   Return Value:     None.

   Input:            None.
   Output:           None.

   Error Handling:   None.

   Algorithms:       The algorithm used assumes four (4) digits are being
             converted.
   Data Structures:  None.

   Global Variables: None.

   Known Bugs:       If the passed long int is the largest negative long int,
             the function will display garbage.

   Author:           Glen George
   Last Modified:    Mar. 8, 1994

*/

static void  cvt_num_field(long int n, char *s)
{
    /* variables */
    int  dp = 3;         /* digits to right of decimal point */
    int  d;         /* digit weight (power of 10) */

    int  i = 0;         /* string index */



    /* first get the sign (and make n positive for conversion) */
    if (n < 0)  {
        /* n is negative, set sign and convert to positive */
    s[i++] = '-';
    n = -n;
    }
    else  {
        /* n is positive, set sign only */
    s[i++] = '+';
    }


    /* make sure there are no more than 4 significant digits */
    while (n > 9999)  {
        /* have more than 4 digits - get rid of one */
    n /= 10;
    /* adjust the decimal point */
    dp--;
    }

    /* if decimal point is non-positive, make positive */
    /* (assume will take care of adjustment with output units in this case) */
    while (dp <= 0)
       dp += 3;


    /* adjust dp to be digits to the right of the decimal point */
    /* (assuming 4 digits) */
    dp = 4 - dp;


    /* finally, loop getting and converting digits */
    for (d = 1000; d > 0; d /= 10)  {

        /* check if need decimal the decimal point now */
    if (dp-- == 0)
        /* time for decimal point */
        s[i++] = '.';

    /* get and convert this digit */
    s[i++] = (n / d) + '0';
    /* remove this digit from n */
    n %= d;
    }


    /* all done converting the number, return */
    return;

}

```