```
        initTimeout = initTimeout - RAMP_DELAY;
      }
    }
  }
  // if no switch is pressed, timeout is cleared and initial timeout is set to initial value.
  else
  {
    switchTimeout = 0;
    initTimeout   = INIT_DELAY;
  }

  // when alarm tone is not 0, the alarm is on, decrement alarm tone to change the current tone
  played.
  if(alarm_tone != 0)
  {
    if((milliseconds - prev_milliseconds) > TONE_TIME)
    {
      prev_milliseconds = milliseconds;
      alarm_tone = ((alarm_tone <= 1) ? 7 : alarm_tone - 1);
    }
  }

  // move digit selection by one on each millisecond.
  digitSelect = (digitSelect < (1 << 3) ? digitSelect << 1 : 1);

}

/// @brief Keep track of time in seconds as precisely as possible.
void timer_isr (void) __interrupt (TF1_VECTOR)
{
  // reset timer overflow, though it does this anyways.
  TF1 = 0;

  // reset timer counters start point.
  TH1 = TH1_START;
  TL1 = TL1_START;

  // check if the time set switch is pressed. If so keep seconds at and hold.
  if(!SET_T_SWITCH)
  {
    seconds = 0;
    return;
  }

  // increment seconds on each timer overflow.
  seconds++;

  // once over 59 seconds, increment minutes and reset seconds
  if(seconds > 59)
  {
    gs_timeKeeper.one_minutes++;
    seconds = 0;
  }

  // once over 9 minutes, increment ten minutes and reset minutes
  if(gs_timeKeeper.one_minutes > 9)
  {
    gs_timeKeeper.ten_minutes++;
    gs_timeKeeper.one_minutes = 0;
  }
```