AN_1903_062 FEBRUARY 2009

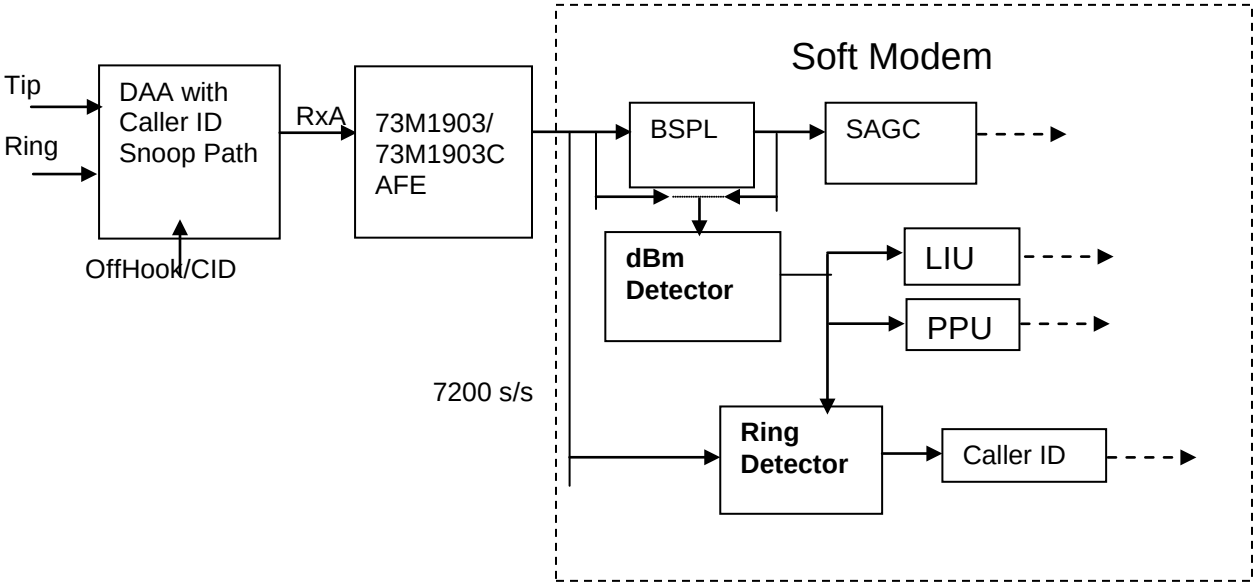
Software-Based Ring/LIU/PPU Detection

Introduction

This document describes implementation of Ring, Line-In-Use (LIU), and Parallel Pick-Up (PPU) functions using the Teridian "Energy Detect" software. This method, in conjunction with the 73M1903 and 73M1903C Analog Front End, reduces the overall bill of material cost associated with implementing embedded soft modems.

Fundamental to the technique is utilization of three system resources, constant analog Caller ID signal path, digital signal processing in host, and the 73M1903/73M1903C's internal dedicated signal boost capability.

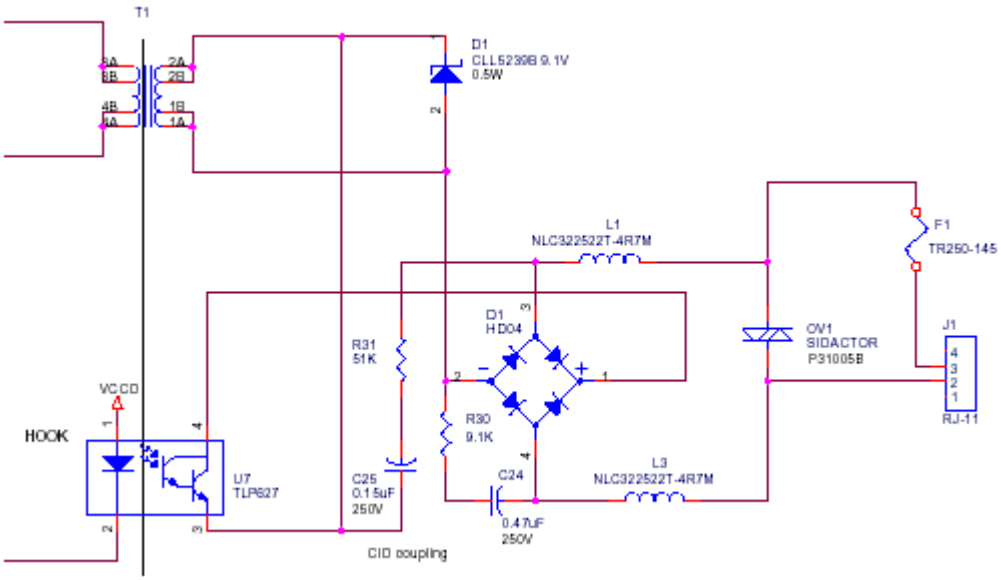
System Block Diagram



DAA, AFE

The 73M1903/73M1903C uses a transformer based DAA to interface to the telephone line. When the Modem is on hook, the DAA provides an alternate path for receiving Caller ID signals. This alternate path provides a relatively weak signal, therefore, to support this technique, the 73M1903/73M1903C AFE includes a dedicated gain block with 20dB of boost as means of compensation. The boost is activated by bit 0 of CTRL1 register (Address 00h)

The following schematic shows the CID path provided by R31 and C25.



Software Module Descriptions

DBm Detector

This software block measures the signal strength or power level of either the Raw receive signal (usually when modem is on hook) or the output of Band Split (BSPL) filter (usually when modem is off hook and connected) as needed. The output of this detector is then used by other higher-level functions. Sometimes gain value of Soft AGC (SAGC) block can also be used as an indication of the 'in Band' signal strength in place of or in addition to dBm detector. The following is a code sample that performs this function:

```
for (r0 = 0; r0 < rxd_size; r0++) /*  R0 points to data sample. */
{
     /* Use in_buf instead of a_buf (post bspl) for dBm detector,
    (1) To see Ring in CallerID mode or
    (2) In the presence of non-unity gain bspl filter. */
     tmp_s16_1 = (cid_idle == FALSE ||
                  (mode_flag == B202 && oa_mode == ANS))
                      ? in_buf[r0] : a_buf[r0];

    /* Do DC Offset calculation */
    dcoacc += tmp_s16_1;
    if (++dcoctr == 1024)
    {
        dco = dcoacc >> 10;
        dcoctr = 0;
        dcoacc = 0;
    }
    else if (tmp_s16_1<-2000 || tmp_s16_1>2000)   /* MAX_DCO */
    {
        dcoctr = 0;
        dcoacc = 0;
    }

    /*  Do dBm detection */
    dbmtmp = (tmp_s16_1 - dco) >> 2;
    dbmacc += dbmtmp * dbmtmp;/* accegy */
    if (++dbmctr == 64)
    {
        dbmtmp = 15*2;                /* REFDBM*2  */
        dbmcmp = 0x00240000;    /* REFDBMEGY */
        while ((dbmcmp > dbmacc) && (dbmtmp < 60*2))
        {
          dbmtmp += 3*2;        /* -3db*2     */
          dbmcmp >>= 1;               /* -3db       */
        }
        while ((dbmcmp < dbmacc) && (dbmtmp > 0*2))
        {
          dbmtmp -= 1;                /* +0.5db*2   */
          dbmcmp += dbmcmp>>3;  /* +0.5db     */
        }
        dbmraw = dbmtmp>>1;     /* /2 */
        dbm = (dbmraw + 3*dbm) >> 2;
        dbmegy = dbmacc;        /* for display */
        dbmctr = 0;
        dbmacc = 0;
    }
}
```

Ring Detector

While on hook, the modem gets signals through the DAA using Caller ID Snoop path. Since ring frequency is very low, a simple time domain method such as zero crossing can be used for ring detection. Amplitude, frequency and cadence checks can be used to narrowly qualify proper ring signal. The following is sample code for a ring detector.

```
/* Following code feeds every sixth raw sample to ring_detect */
if (get_ie_ring())
{
      if (!ring_flag && dbmraw < S_ring_amp)
      {
            ring_interrupt ();
            ring_zc_cnt = ring_zc_flg = 0;
      }
      else if (ring_flag && dbmraw < 50)
      {
            ring_detect (in_buf[0] - dco);
            ring_detect (in_buf[6] - dco);
      }
}

static void ring_detect (S16 sample)
{
      if ((ring_zc_flg && sample > 10) ||
          (!ring_zc_flg && sample < 10))
      {
            if (++ring_zc_cnt >= 5)
            {
                  if (ring_zc_flg)
                  {
                        ring_interrupt ();
                       /* Ring Interrupt on Rising edge */
                        ring_cycles++;
                  }

                  ring_zc_cnt = 0;
                  ring_zc_flg ^= 1;       /* Look for the other edge */
            }
      }
      else
            ring_zc_cnt = 0;
}
```

Line In-Use Energy (on hook LIU-E) Detector

While on hook, the modem gets signals through the DAA using Caller ID Snoop path. The LIU-E detector can use dBm detector to check if a parallel phone is currently in use by measuring signal energy on the line. A timer is usually used to qualify this energy detection for a user programmable duration of time. Using a programmable threshold, the modem then decides whether it is appropriate to go off hook or not. The following are sample code fragments.

```
if (liu_e_enb)
{
     if (cid_usr11)
        let_CID_sensing (FALSE); /* Enable DAA Energy sensing.*/

     wait (S_liu_settle  /* Wait before sampling On-Hook LIU.*/
     c = wait_egy_liu ();
     p10_p11_off  ();   /* Disable DAA for Energy sensing.*/

     if (!c)
     {
        cli_result = ON_HOOK_ELIU_ID; /* Line in use..*/
        jmpstate (DISCONNECT);        /* ..disconnect.*/
     }
}

U01 wait_egy_liu ()
{
    tmr_100ms_1 = S_wait_egy; /* Start max wait for Energy detection.*/

    while ((dbm > S_dbm_thld) && tmr_100ms_1)
       background ();                    /**/

    return (!tmr_100ms_1);
}
```

Parallel Pick-Up Energy (off hook PPU-E) Detector

Off hook PPU-E detection starts by measuring a reference energy using dBm detector after the modem is connected and on-line with another modem. Throughout the rest of the connection the modem compares this reference with the current energy level and if it detects significant amount of drop in signal power due to a parallel phone being picked up, it then hangs up and sends a PPU-E message to the host. If modem stays online for extended period of time the logic can be modified to track slow changes in signal energy not caused by parallel pick-up. A special timer is used to prevent the modem from sending PPU-E message prematurely when a sudden loss in energy is due to a loss of carrier. The following are sample code fragments.

```
ppu_ref_dbm = dbm;                    /* PPU Reference dbm. */
ppu_ref_agc = agc_gain+(agc_gain>>2); /* PPU Reference agc_gain. */
jmpstate (ONLINE);      /* Go to (or back to) ONLINE state.*/

if (ppu_e_enb && OFF_HOOK ())
{
   if (tmr_ppu)
   {
      if (tmr_ppu == 1)
      {                              /* PPU timer expired. */
         tmr_ppu = 0;                /* Deactivate it. */

         if (carrier_detect)         /* Carrier still present.. */
               disconnect_rslt (OFF_HOOK_ELIU_ID);
             /* ..line in use, energy sensing. */
      }
   }
   else if ((dbm > ppu_ref_dbm + S_ppu_thld) &&
            (agc_gain > ppu_ref_agc))
   {
      if (S_ppu_debounce)
         tmr_ppu = S_ppu_debounce + 2;
      else
         disconnect_rslt (OFF_HOOK_ELIU_ID);
         /* Line in use, energy sensing. */
   }
}
```

END.

Revision History

Revision	Date	Description
Rev. 1.0	2/4/09	First publication.

Maxim Integrated Products, 120 San Gabriel Drive, Sunnyvale, CA 94086 408-737-7600