Applications of the HRA and Energy Detect Blocks of the MT90812 Integrated Digital Switch

AN5207 ISSUE 1 September 1999

Contents

1.0 Introduction

The MT90812 Integrated Digital Switch (IDX) provides the integration of several functions required in a telecom application. The purpose of this Application Note is to provide supplementary information on the application of the HRA and Energy Detect blocks of the IDX. A programming sequence for the multiplexed mode of the HRA is provided to assist the system designer with the steps required to program the IDX. The software algorithm for the Energy Detect block provides the system designer with a sample algorithm to interpret the cadence of supervisory signals.

2.0 HRA Programming Sequence for Multiplexed Mode

The HLDC Resource Allocator (HRA) module in the MT90812 supports communications over the D-channel and provides an interface to the MT8952 HDLC Protocol Controller. The RX section of the HRA can operate in Dedicated Receive mode or Multiplexed Receive mode (RX channel Auto-hunt).

The primary function of the RX channel Auto-hunt circuit is to provide the RX circuit with the next receive channel number. This section of the Application Note provides the programming sequence for the Auto-hunt mode. For the hardware architecture refer to the diagram “Typical Application Using the HDLC Resource Allocator” in the MT90812 data sheet.

The steps provided here are intended to guide the system designer with some of the programming aspects of the Auto-hunt mode. To exercise the Auto-hunt mode a few assumptions are made as follows:

- The system contains 16 DNICs
- The C-channels occur before D-channels on the STi/o1 stream
- The D-channel bit rate is at 16 kb/s on the ST-BUS. Hence the DNICs need to operate at 160 kb/s
- The peripheral connected to DNIC 2 sends the first RTS flags
- While the HRA and the HDLC receiver are busy with the D-channel slot for DNIC 2, the

peripheral connected to DNIC 14 sends RTS flags

The following steps provide the initial programming steps:

- Enable the HDLC Controller for External Timing mode and idle state
- Enable the Interrupt Flag bits of the HDLC Controller as desired
- Enable the HRA by setting the EN bit high in the HRA Control register 1
- Allow C channels to occur before D channels by setting the CD bit high in the HRA Control register 1
- To enable the D channel bit rate at 16 kb/s, set the BRSEL1-0 bits to 01 in the HRA Control register 1
- In order for the Auto-hunt circuit to monitor all 16 D channels, set LOC0-7 and LOC8-15 bits to low in HRA Lock Out Registers 1 and 2

Figure 1 summarizes the Auto-hunt algorithm. Figure 2 illustrates the sequence of events in this example. The Auto-hunt circuit will start monitoring channels 16 to 31 (bit CD=1 in the HC1 register) of stream STi1. It is assumed that the peripheral connected to DNIC 2 sends the first RTS flags. Upon detection of the RTS flag by the auto-hunt circuit, it will send a GA flag (CTS flag = 011111110 = 7F +'0') and enable $\overline{\text{RxCEN}}$ for 2 bits per frame during channel 17 which is allocated for the DNIC 2 D-channel. A read of HRA Status register 1 should confirm that the present receive channel is channel 17 (PRX4-1 = 0001) and the receive channel is latched (RXCHNL is set high).

Once the $\overline{\text{RxCEN}}$ is enabled for 2 bits per frame during channel 17, the HDLC Controller will start receiving packets from the DNIC 2 peripheral. The receive section of the HDLC Controller searches the incoming packets for flags on a bit-by-bit basis and establishes frame synchronization. The flag is a unique pattern of 8 bits (01111110 = 7E) defining the data packets frame boundary.

It is assumed that while the HRA RX and HDLC receiver are busy with DNIC 2 peripheral, DNIC 14 peripheral will start sending RTS flags. Since the RX is busy with DNIC 2 peripheral, then no GA flag will be sent to the DNIC 14 peripheral and therefore the FLAG bit in the HRA Status register 1 will be held high until the RX becomes inactive by receiving a REOP strobe (1 bit long) from the HDLC Controller and the HRA Status register 3 is read. The FLAG bit indicates that the Auto-hunt circuitry has detected an

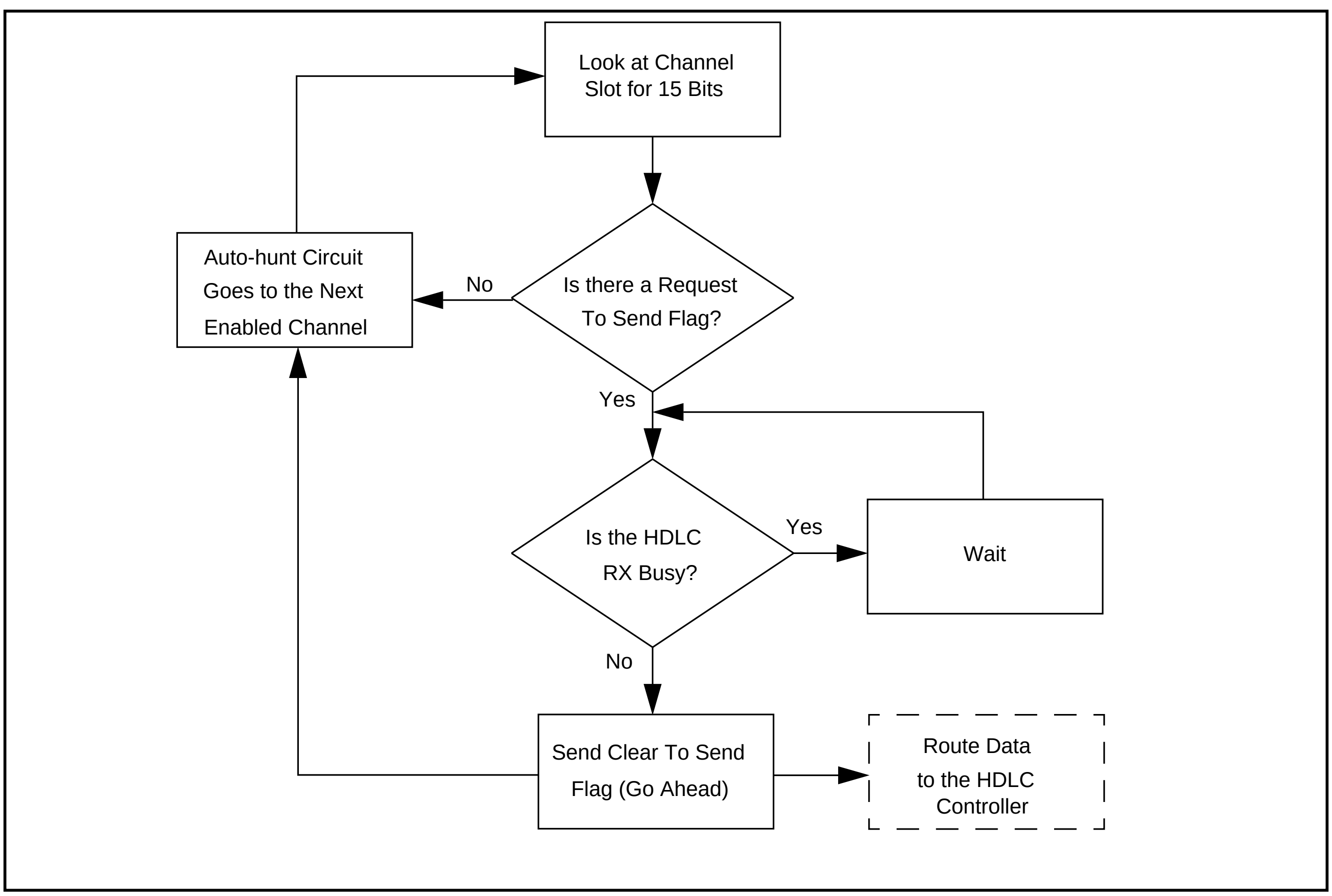

Figure 1 - The Auto-hunt Algorithm

RTS flag from the peripheral but has not acted upon it yet. A read of HRA Status register 3 must also be done by the system to initiate receiving of packets from the DNIC 14 peripheral.

Once an end-of-packet flag is received from the DNIC 2 peripheral (the EOPD bit in the Interrupt Flag register of the HDLC controller will be held high) the HDLC controller will generate an interrupt. The REOP strobe is generated by the HDLC controller once it receives a closing flag or the incoming packet is aborted or an invalid packet of 24 or more bits is detected. The system may also deactivate the HRA RX by enabling the SREOP bit in the HRA Control register 2.

Once the RX becomes inactive then the system needs to read the HRA Status register 3 for the most recently active receive channel number (channel 17) in order to initiate receiving packets from the DNIC 14 peripheral. The Auto-hunt circuit will then send a GA flag to the DNIC 14 peripheral and move the $\overline{\text{RxCEN}}$ strobe to channel 29 on the STi1 stream. A read of HRA Status register 1 should confirm that the present receive channel is channel 29 (PRX4-1 = 1101) and the receive channel is latched, i.e. RXCHNL is set high. The HDLC receiver is now ready to receive packets from the peripheral.

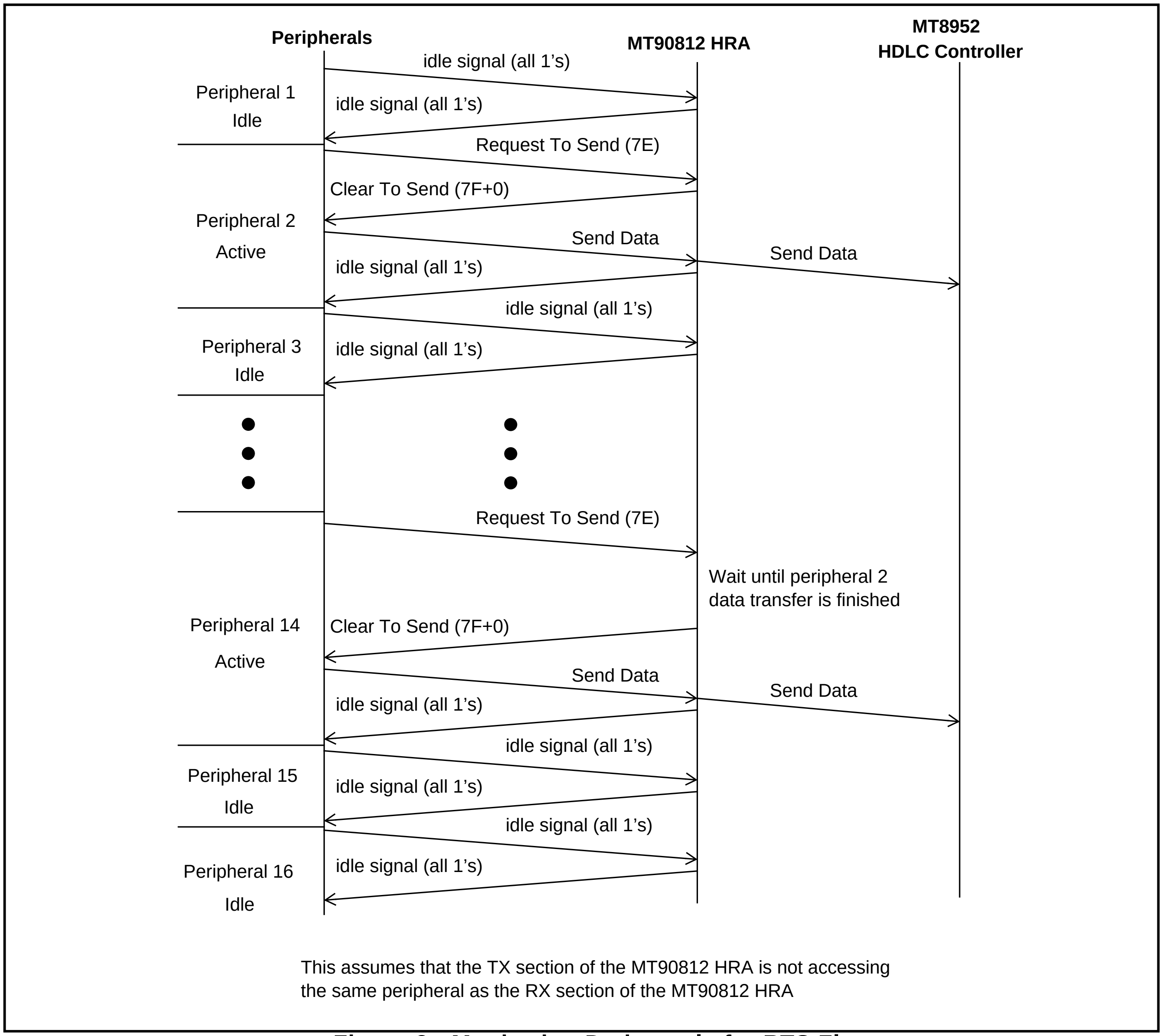

Figure 2 - Monitoring D-channels for RTS Flags

3.0 Implementing an Algorithm for Interpreting The Measured Cadence of a Call Progress Signal

The cadence of a call progress signal can be measured by the Energy Detect block of the MT90812. This section of the Application Note provides a software algorithm to interpret the measured cadence of the call progress signal.

Background Information

Call progress or supervisory signals are audible signals sent to a caller during the process of setting up, holding, or transferring a call in order to inform the caller about the status of the call. Call progress signals can be heard by users located in the following systems:

- On-premise and off-premise PBX stations
- PBX attendants
- The public switched telephone network (PSTN)
- Users of other PBXs or other switching systems accessing this PBX via tie trunks or other facilities

Call progress signals are either a single-frequency or the combination of dual-frequency of sinusoidal voltages applied in specific cadence or ON-OFF patterns. The signals which are most frequently used in PBXs are Dial, Reorder, Busy, and Audible Ring tones. The following table provides the frequency combination, permissible power level and the ON-OFF or cadence pattern.

The recognition of the call progress tones can be done either with frequency or cadence detection. As shown in Table 1 each tone has a distinct cadence pattern. This Application Note provides a software algorithm to interpret the cadence information measured by the energy detect block of the MT90812 (IDX).

Name	Frequency (Hz)	Permissible Power Level Per Frequency				Cadence or ON-OFF Pattern
		CO Trunk Interface		All Other Interfaces		
		Max (dBm)	Min (dBm)	Max (dBm)	Min (dBm)	
Dial Tone	350 + 440	-16	-17.5	-16	-26	No Interruption or ON-OFF Pattern (Continuous Tone)
Reorder Tone	480 + 620	-19.5	-22.5	-19.5	-35	Repetition of tone ON for 0.25 +/- 0.025 s and tone OFF for 0.25 +/- 0.025 s
Busy Tone	480 + 620	-19.5	-22.5	-19.5	-35	Repetition of tone ON for 0.5 +/- 0.05 s and tone OFF for 0.5 +/- 0.05 s
Audible Ring Tone	440 + 480	-14.5	-17.5	-14.5	-30	Repetition of tone ON for 2.0 s and tone OFF for 4.0 s

Table 1 - Most Frequently Used Call Progress Signals

4.0 Energy Detect Program Algorithm

The MT90812 provides two energy detect blocks for monitoring call progress tones during trunk calls. For a detailed description of the energy detect blocks refer to the MT90812 (IDX) data sheet. The energy detect block follows the envelope of the monitored call progress signal and generates an interrupt once the programmed threshold level (content of EDLTA/B or EDHTA/B registers of the IDX) is crossed by the envelope of the signal or a maximum count of 508 msec is reached. Refer to Figure 3. When an interrupt is generated the SSCRA/B register is updated with the cadence and position of the signal envelope.

In order to interpret the measured cadence time and position of the signal envelope, the attached algorithm can be implemented. Refer to the Algorithm Source Code section. This software algorithm assumes that the cadence and position information of the monitored supervisory signal is stored in a 2x16 array. The software algorithm runs through 16 consecutive rows of the cadence and position array, adds consecutive ON or OFF times and rejects noise intervals. When the reported cadence time reaches the maximum count of 508 msec and the position remains the same as the previous position, the ON/OFF interval continues and the interval times get added. Table 2 provides the result generated by the software algorithm from a sample data reported by the IDX. The cadence time column is the content of SSCRA/B cadence bits converted to msec.

The software algorithm also rejects noise intervals during ON time or OFF time. Refer to Figure 3. It should be noted that the 25 msec leaky hold time of the energy detect block provides noise rejection during ON time. This software algorithm provides additional noise rejection during ON time as well as OFF time. As shown in Figure 3 at the beginning of the ON time noise interval the present position is low (end of T3) and the next and previous positions are both high (end of T2 and T4, respectively). The software algorithm interprets the dip in the envelope of the signal as noise and therefore complements the

Interrupt Count	Position_Cadence Array (Content of SSCRA/B register over 16 interrupts)		Cadence Time Interpreted By the Software Algorithm		Comments
	Position bit	Cadence time (msec)	Interval	Duration (msec)	
1	1	50	OFF Time	50	
2	1	508			Maximum Count
3	0	472			
4	1	12			Noise during ON time
5	1	508			Maximum Count
6	0	500	ON Time	2,000	
7	1	52			
8	0	36			Noise during OFF time
9	0	508			Maximum Count
10	0	508			Maximum Count
11	0	508			Maximum Count
12	0	508			Maximum Count
13	0	508			Maximum Count
14	0	508			Maximum Count
15	0	508			Maximum Count
16	1	356	OFF Time	4,000	

Table 2 - Interpreting the Cadence and Position Bits of the SSCRA/B

present position to high. This will ensure that T4 is added to T2 and T3 and reported as ON time. In the second case the present position is high (end of T7) and the next and previous positions are both low (end of T6 and T8, respectively). The software algorithm interprets the spike in the envelope of the signal as noise and therefore complements the present position to low. This will ensure that T8 is added to T7 and T9 and reported as OFF time.

There are two variables used to specify the duration of the noise intervals to be rejected, *MaxReported-ONTimeNoiseInterval* and *MaxReportedOFFTime-NoiseInterval*.

The *MaxReportedONTimeNoiseInterval* variable is assigned to 15 msec in the initialization portion of the software algorithm. With the *MaxReported-ONTimeNoiseInterval* assigned to 15 msec, noise intervals of 40 msec or less during ON time are rejected by the software algorithm. Refer to Figure 4. The maximum interval of 40 msec is reported by the energy detect block as 15 msec due to the leaky hold time of 25 msec. This hold time is used for the peak detector of the energy detect block to bridge between the envelope peaks.

The *MaxReportedOFFTimeNoiseInterval* variable is assigned to 40 msec in the initialization portion of the software algorithm. Hence the algorithm rejects noise intervals up to and including 40 msec during OFF time. This value is compared with the reported time by the energy detect block. If the reported time is greater than 40 msec then the interval is considered as ON Time. Otherwise the interval is considered as noise and will be added to OFF time. The reported time will also include the 25 msec leaky hold time. Refer to Figure 4.

The software algorithm source code for interpreting the cadence and position information is provided in the following pages.

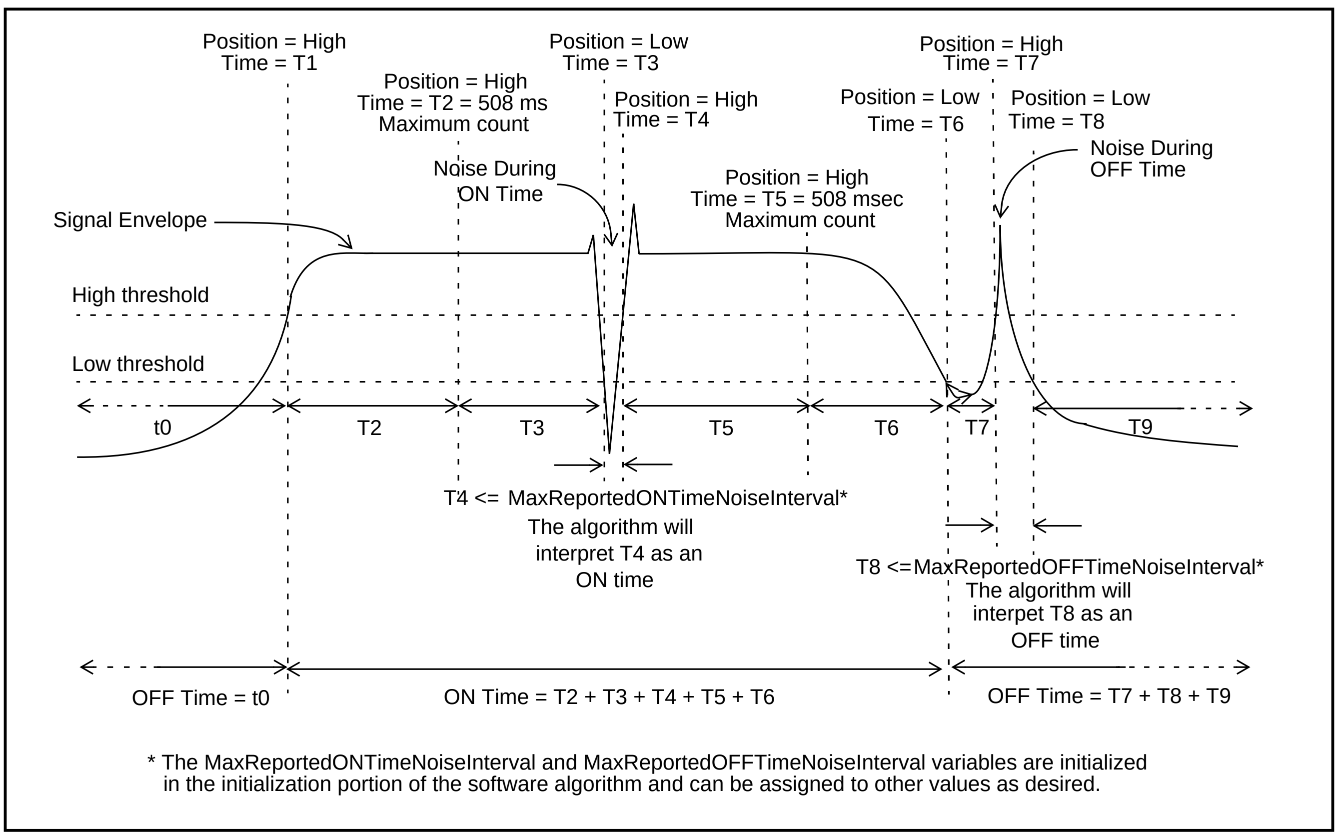

Figure 3 - Two Noise Secenarios Considered by the Algorithm

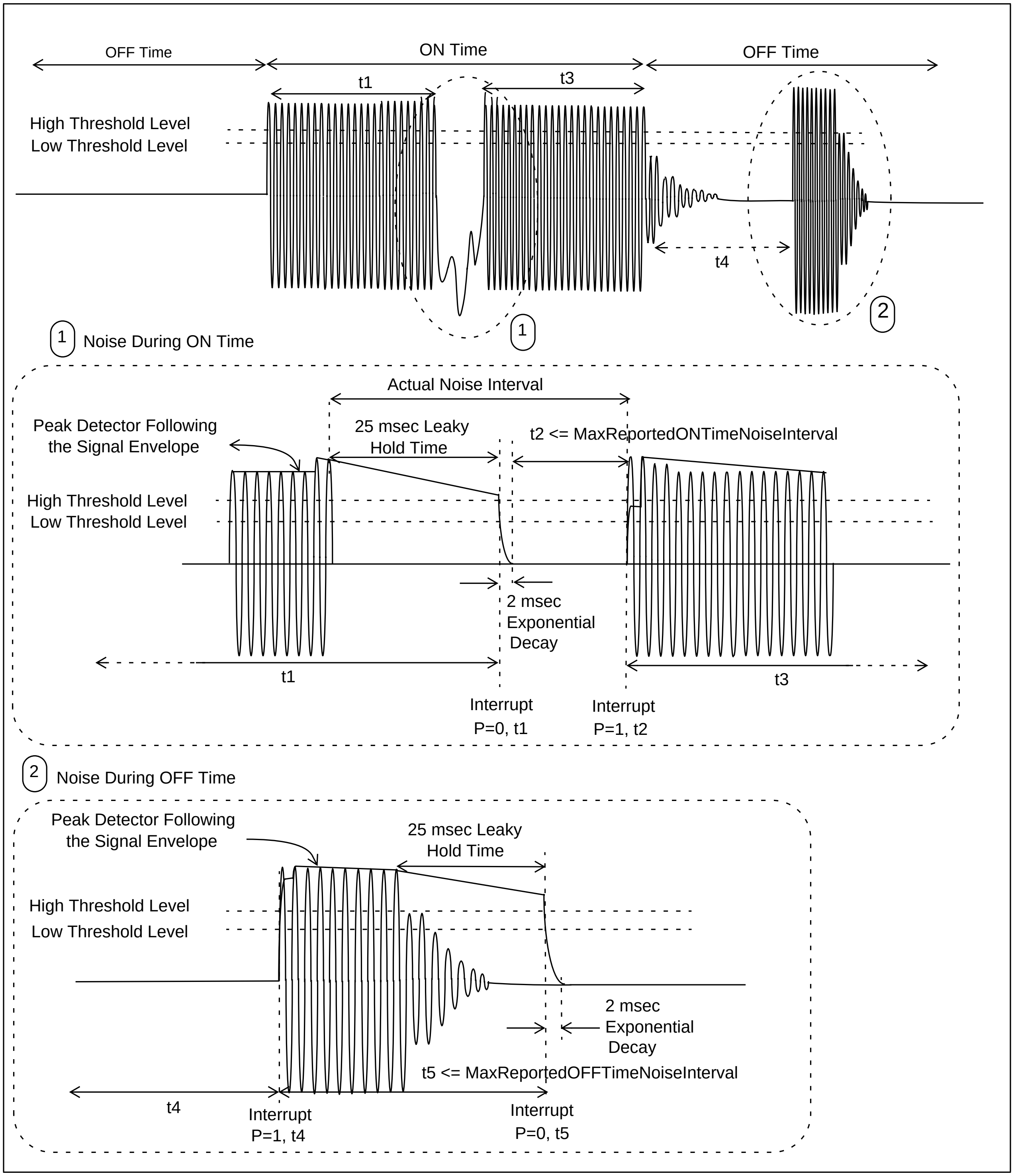

Figure 4 - Noise During ON-OFF Intervals

5.0 Algorithm Source Code

The program assumes that the Position and Cadence information is stored in Position_Cadence [] array. The program stores the results in the Interval_Cadence[] array, i.e. "ON/OFF Time" string and the ON/OFF time interval in msec.

```
.......
// Initialization
#define Low 0;
#define High 1;
#define MaxReportedONTimeNoiseInterval  15;          // 15 msec
#define MaxReportedOFFTimeNoiseInterval  40;         // 40 msec = 25 msec Leaky Hold Time + 15 msec
#define Min_Row 1;
#define Max_Row 16;
T_previous = 0;                                      // Previous cadence time in msec
T_present = 0;                                       // Present cadence time in msec
P_previous = Low;                                    // Previous position bit
P_present = Low;                                     // Present position bit
.......
for(Row_Number = Min_Row ; Row_Number < Max_Row ; Row_Number++)   // First and only for() loop
{
  // ON time noise rejection
  // if(present_position = 0 AND next_position = 1 AND previous_position = 1)
  if(((Position_Cadence [Row_Number].position == Low)&&
     (Position_Cadence [Row_Number + 1].position == High))&&
     (Position_Cadence [Row_Number - 1].position == High))
  {
          if(Position_Cadence[Row_Number + 1].cadence <= MaxReportedONTimeNoiseInterval)
                  // complement the P bit:  present_position = 0 -> 1
                  P_present = !( Position_Cadence [Row_Number].position);
          else
                  P_present = Position_Cadence [Row_Number].position;
  }

  // OFF time noise rejection
  // elseif(present_position = 1 AND next_position = 0 AND previous_position = 0)
  else if(((Position_Cadence [Row_Number].position == High)&&
       (Position_Cadence [Row_Number + 1].position == Low))&
       (Position_Cadence [Row_Number - 1].position == Low))
  {
```

```
        if(Position_Cadence[Row_Number + 1].cadence <= MaxReportedOFFTimeNoiseInterval)
                // complement the P bit: present_position = 1 -> 0
                P_present = !( Position_Cadence [Row_Number].position);
        else
                P_present = Position_Cadence [Row_Number].position;
}
else
        P_present = Position_Cadence [Row_Number].position;

// Compute ON or OFF time
T_present = Position_Cadence [Row_Number].cadence;
// Low to High, Calculate OFF Time
if(P_present == High)           // P_present = 1
{
        if(P_previous == High)          // P_previous = 1, Add consecutive ON time intervals
                T_previous = T_present + T_previous;
        else    // P_previous = 0
        {
                T_off = T_previous + T_present;
                T_previous = 0;
                T_present = 0;
                // Store “OFF Time” in the Interval_Cadence array
                Interval_Cadence[Row_Number].interval = "OFF Time”;
                // Store the OFF time value in the Interval_Cadence array
                Interval_Cadence[Row_Number].cadence = T_off
        }
}
// High to Low, Calculate ON Time
else     // P_present = 0
{
        if(P_previous == High)          // P_previous = 1
        {
                T_on = T_previous + T_present;
                T_previous = 0;
                T_present = 0;
                // Store “ON Time” in the Interval_Cadence array
                Interval_Cadence[Row_Number].interval = "On Time";
                // Store the ON time value in the Interval_Cadence array
```

```
                Interval_Cadence[Row_Number].Cadence = T_on;
            }
            else        // P_previous = 0
                T_previous = T_present + T_previous;
    }
    P_previous = P_present;
}   // End of the for() loop
. . . . . . .
```

Notes:

For more information about all Zarlink products visit our Web Site at

www.zarlink.com

For more information about all Zarlink products visit our Web Site at

www.zarlink.com