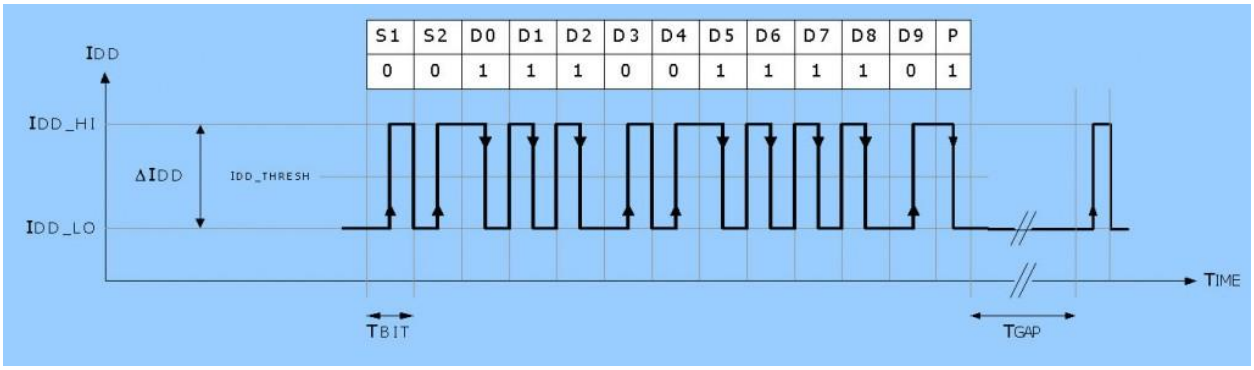
CAS DATALOGGERS

Implementation Of PSI-5 Serial Sensor Interface On ADwin Real-Time Data Acquisition Systems

CHESTERLAND OH—August 3, 2011

The PSI-5 communications interface is a 2 wire serial communications interface which is used to interface sensors to Electronic Control Units (ECU) in applications such as accelerometers for airbag systems. Promoted by major auto suppliers such as Bosch and Continental, the PSI-5 interface has been in use for more than 10 years and has shown itself to be both robust and reliable. **Figure 1** shows the PSI-5 telegram definition.

Figure 1: PSI-5 Telegram Definition



To minimize the number of wires going to the sensor, the physical layer of PSI-5 is implemented as a 2 wire interface that carries both the signal and power. The sensor data is sent as a series of current pulses which ride on top of the normal sensor supply current. This allows the interface to communicate at up to 250K bits/second and also provides EMC compatibility for conducted and radiated emissions. The standard data packet consists of 13 bits: 10 bits of data, 2 start bits and a parity bit (see figure 1). The data is encoded using a Manchester coding scheme where a 0 corresponds to a rising edge in the middle of the bit transition time and a 1 by a falling edge in the middle of the bit. Typically a dedicated receiver IC is used to read current signal and extract the data.

A pressure sensor manufacturer was looking for a system to simultaneously test multiple sensors using this interface. Rather than try to build a custom solution around the dedicated interface chips, they were looking for an off-the-shelf data acquisition solution that could be programmed to read the raw signal and extract the data. For this project, an **ADwin-Light-16 Real-Time Data Acquisition System** was selected. A simple comparator circuit was used to convert the current signal to a digital waveform that could be read with standard TTL logic. The ADwin system featured 6 high speed digital inputs allowing up to 6 sensors to be read at the same time. For each digital input, a simple state machine was

implemented in software to monitor the incoming data, look for the start bits, read and decode the data.

Figure 2 shows the state machine to read one input. The minimum bit time corresponding to a 0 was 40 useconds; therefore, to allow for small variations in the timing, the state machine was setup to read the inputs at 8 times the expected clock rate, or 5 useconds. A watchdog timer was also provided to allow the state machine to reset in the case of a partial or corrupted transmission. **Figure 3** lists the ADBasic code for the state machine to read one sensor. For testing, a second ADwin system was configured to generate various data packets to verify that they could be read correctly.

For this project, it was quite straightforward to create a simple state machine using the CASE structure available in the ADBasic programming environment. The deterministic, real-time capability of the ADwin system allowed the data to be read reliably without the need for specialized receiver IC's, and the flexibility of the software enabled the different scenarios and failures mechanisms to be tested quickly and easily.

Figure 2: State Machine to Read PSI-5 Data

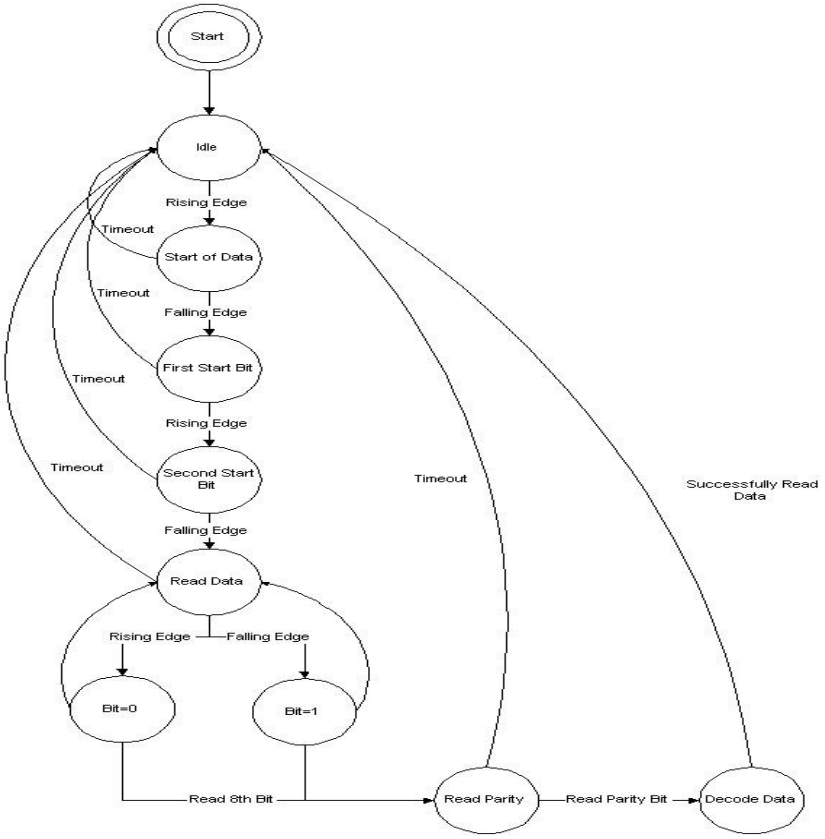


Figure 3: ADBasic Code For The State Machine to Read One Sensor

```
EVENT:
  newbit = digin(0)AND 01B
  inc counter
  SelectCase(State)
    CASE 1       ' wait for high to indicate start of transmission
      if (newbit = 01b) then
        startcnt = startcnt+1
        IF (startcnt > 10) then   ' need to get >50 usec high to start
          state = 2
          startcnt=0
        endif
      endif
    CASE 2         ' waiting for low start bit
      if (newbit = 00b) then  'first low is start of word
        state = 3
        numbits = 0
        bit_val = 01b
        numUnusedbits = 0
        counter = 1
        fall_time = counter
        i=i+1
        data_1[i]=0
      endif
    CASE 3         'waiting for low to high transition to mark bit
      if (newbit = 01b) then
        rise_time = counter
        if ((rise_time - fall_time) > 8) then    ' > 40 usec = 1
          data_1[i]= data_1[i] + 1 * bit_val
        else
          data_1[i]= data_1[i] + 0 * bit_val ' < 40 usec = 0
        endif
        numbits = numbits + 1
        bit_val = bit_val * 2
        if (numbits > 12) then
          state = 5
          par_2 = Data_1[i]
          fpar_1 = Data_1[i]*2.0
        else
          state = 4
        endif
      endif
    CASE 4       'waiting for high to low transition to start next bit
```

```
    if (newbit = 00b) then
      fall_time = counter
      state = 3
    endif
  case 5                    ' 3 unused bits
    if (newbit = 00b) then   ' start of unused bit
      state = 6
    endif
  case 6
    if (newbit = 01b) then       ' end of unused bit
      numUnusedBits = numUnusedBits+1
      if (numUnusedbits > 1) then
        state = 7
      ELSE
        state = 5
      endif
    endif
  case 7
    if (newbit = 00b) then       ' start of error flag
      fall_time = counter
      state = 8
    endif
  case 8 ' read error flag
    if (newbit = 01b) then
      rise_time = counter
      if ((rise_time - fall_time) > 8) then    '>60 usec = 1
        errorflag = 1
      else
        errorflag = 0
      endif
      state = 9
      par_3 = errorflag
    endif
  case 9
    IF (counter > 1000) Then
      state = 1
      counter = 0
    endif
ENDSELECT
'watchdog
IF (counter > 1000) Then
  state = 1
  counter = 0
```

```
  endif
  par_1 = counter
```

For further information on ADwin real-time data acquisition and control systems, other data acquisition system product lines, or to find the ideal solution for your application-specific needs, contact a CAS Data Logger Applications Analyst at (800) 956-4437 or visit the website at www.DataLoggerInc.com.

Contact Information:

CAS DataLoggers, Inc.
12628 Chillicothe Road
Chesterland, Ohio 44026
(440) 729-2570
(800) 956-4437
sales@dataloggerinc.com
http://www.dataloggerinc.com