CPE-bus interface circuirity

Overview

CPE-bus communication protocol is designed for high speed, high resolution encoder or linear scale interfacing. The key features of CPE-bus is high reliability and easy to use.

CPE-bus is the China's recommended national industrial standard (JB/T 11505-2013) for rotary encoders / linear scales communication which was pulished by the China Ministry of Industrial & Information on 2014, CPE-bus is a public standard which is totally royality & patent free, therefore, CPE-bus users are free from any unpleasant legal harassment from dominating companies.

Applications :

- Rotary Encoders
- Linear Scales
- Robots
- Auto vehicles

Main Features :

- Small in size and low cost, very efficient use of FPGA logic gate, only about 32,500 logic gates used.
- Auto CRC verifications, minimum effort to achieve the highest communication reliability.
- Flexible readout configuration, work for both interrupt or polling mode.
- High data update rate, 12.5K updates / sec.
- Flexible hardware configuration, it can work with CPE_TXD.v for bi-directional bus communication

CPE_RXD.v

CPE_RXD.v is a CPE-bus communication ASIC module IP core (receiver side) that designed and developed by Easson Measurement Technology Ltd. The purpose of CPE_RXD.v is to provide an easy, proven reliable interfaces to our customer to interface their CNC controller/servo driver to our linear scale or rotary encoder products.

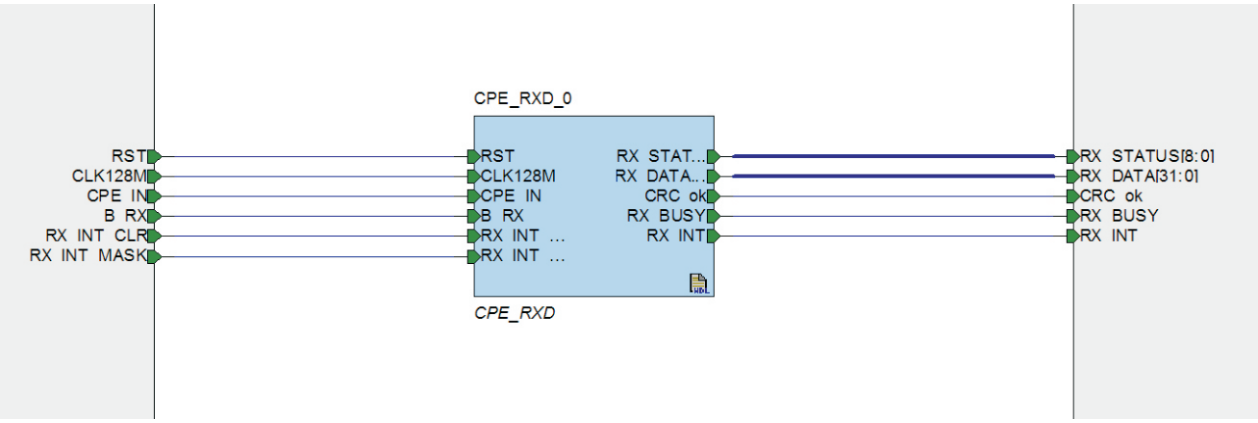
Easson provides the CPE_RXD.v in FPGA source program (Verilog) , so that our customer can have it intergated and implemented in all types and all brands of FPGA/CPLD devices available in the market with only very small effort.

Despite the CPE-bus communication have CRC mechanism, the normal engineering practise is to have such protocols being implemented by serveral independent modules which is simple in structure. However, for the consideration in ease of implementation and to reduce number of logic gates used, one complete module design concept is used in CPE_RXD.v to make it very simple and easy to integrate into customers' hardware system.

CPE_RXD.v is an proven and superior reliable communication module IP core that have AUTO CRC verification. All CRC generations and verifcations are done by hardware in background without any involvement of the system CPU. Only the verified correct data shall appear in the STATUS[9:0] output bus and DATA[31:0] output bus.

CPE_RXD.v provides two modes of readout operation, they are Interrupt mode or Polling mode.
For slow data rate (data rate less than 1K/sec), Interrupt mode is prefered in most caes.
For high data rate (data rate up to 12.5K/sec) polling mode seems to be the only feasible way for data readout.

Block diagram



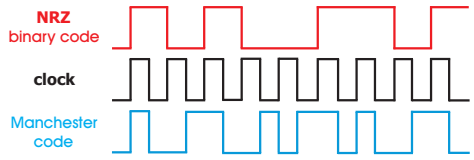
CPE-bus interface circuirity

Theory of operation

CPE-bus is a serial communication protocol based on Manchester encoded serial data frame which very similar to the data frame used in normal Ethernet communication.

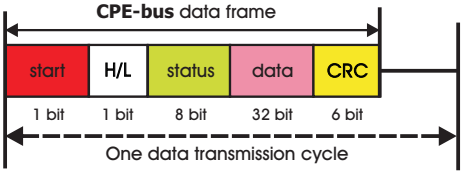
CPE-bus is a simple self clocking serial data communication protocol which is very suitable for high speed and long distance serial data transmission using low cost physical hardware interface.

CPE-bus featured with CRC error detection mechanism which enable very reliable communication under very heavy electricially noisy industry environment.



Manchester Encoded Serial Data

CPE-bus data frame

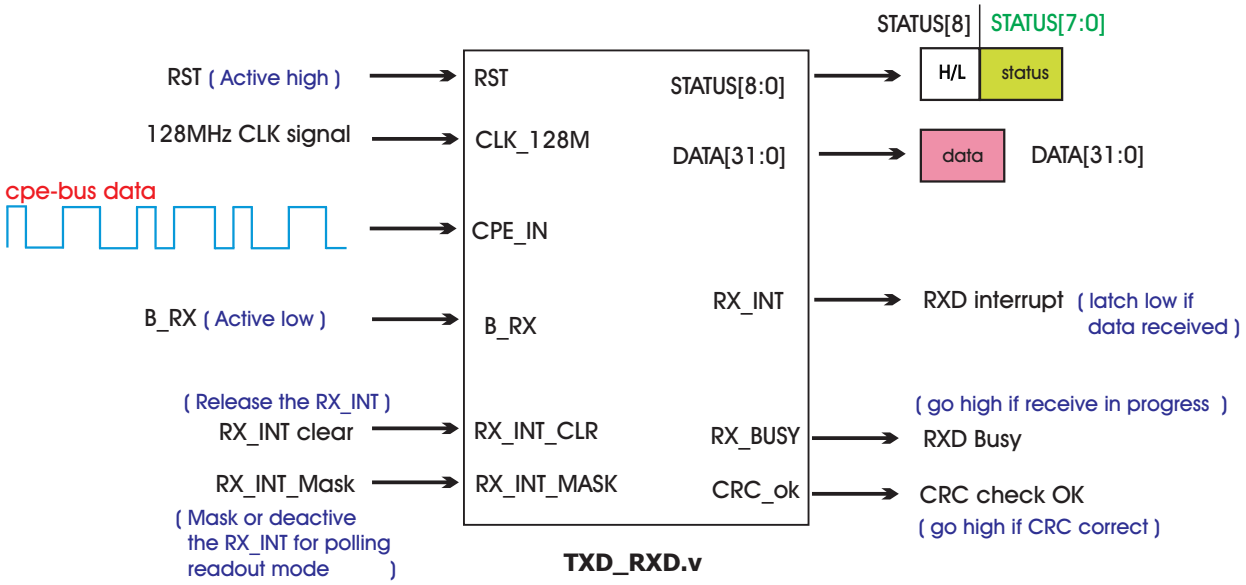


CPE-bus serial communication protocol

CPE-bus have a very simple 48 bit data frame as per follows

- start bit (always 0)
- H/L or R/W bit (reserved for 64 bit operation or master/ slave configuration transmission)
- STATUS byte (user defined data byte for status transmission)
- DATA word (32 bit data word for encoder or scale position transmission)
- CRC (6 bit CRC, polynomial = "1000011", start CRC = "000000")

CPE_RXD.v Pins layout and Signal Connection Diagram



TXD_RXD.v

CPE-bus interface circuirity

CPE_RXD.v Pin Function Description Table

Pin	Function	Input/Output	Description
RST	Reset Signal	Input	Reset CPE_RXD.v , Active high signal
CLK_128M	Clock	Input	Clock signal, 16X the CPE-bus signal Frequency, i.e., for 8 Mhz CPE-bus data frequency, 128MHz clock is required
CPE_IN	CPE data input	input	CPE-bus serial data input
B_RX	Receive Enable	input	Active Low, work together with Easson's CPE_TXD.v module for bi-directional data transmission, if only receiver module (CPE_RXD.v) is used, please tie this pin to ground.
RX_INT_CLR	RX_INT clear	Input	When data received and verified correct, the RX_INT will latch low to flag CPU to readout the received data (unless the RX_INT_MASK tie high) RX_INT CLR is to clear the latched RX_INT
RX_INT_MASK	RX_INT_MASK	Input	RX_INT_MASK is to mask out the RX_INT, so that it won't go low after CPE-bus data received. This Pin is to let the CPE_RXD.v work in polling mode.
STATUS[8:0]	STATUS byte	Output	The H/L and Status data received STATUS[8] : H/L or R/W bit of the CPE-bus data frame Status[7:0] : Status byte of the CPE-bus data frame
DATA[31:0]	DATA word	Output	The Encoder/Linear scale position data received DATA[31:0] : 32 bit DATA of the CPE-bus data frame
RX_INT	RX_INT	Output	Go low output latch to interrupt CPU to inform data have received correctly The STATUS[8:0] and DATA[31:0] are ready for readout RX_INT can only be released by "go high" trigger of RX_INT_CLR pin RX_INT function can be disabled by RX_INT_MASK tie high, to have CPE_RXD.v work in readout polling mode
RX_BUSY	RX_BUSY flag	Output	RX_BUSY flag high when data receive is in progress
CRC_ok	CRC correct	Output	CRC check correct flag latch, it latch high is CRC check is correct

Points to notice for CPE-bus CRC generation.

In the published CPE-bus standard, the CRC polynoimal = "1000011" and starting CRC "000000" is stated, Please follow this parameters 100% to avoid CRC checking mistake.

Easson's CPE_RXD.v CRC verifcation and check follows the above CRC parameters 100%.

Please notice that 42 bit (including the start bit) data CRC generation is used in CPE-bus, please don't ignore the start bit (always 0) of the CPE-bus data frame during the CRC generation.

Points to notice for CPE-bus transmission frequency.

In the published CPE-bus standard, the data frequency of CPE-bus can be 1MHz to 8MHz.

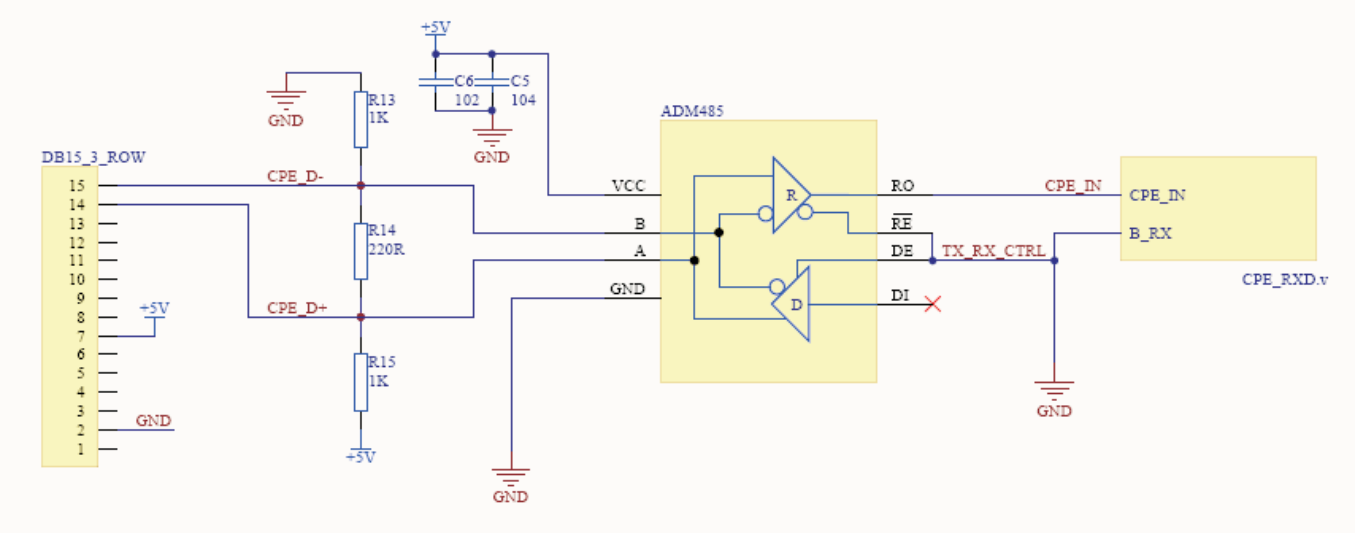
Since the serial data of CPE-bus data frame is based on Manchester encoded data, it is an one signal only and self clocking serial data of which the data transmission delay due to conductor wire capactance is self compensated. Easson have conducted many many tests on data transmission relaibility and concluded from our practical experiment, it seems to us there is no any difference in data transmission reliability to have the CPE-bus work in 1MHz or 8MHz, even under very extreme and unlikely happen man made nosiy environment, CPE-bus data drop out rate is as low as few parts per million. Therefore, for all Easson's product, only the 8MHz (the maximum) data transmission is used.

CPE-bus interface circuirity

Application Examples

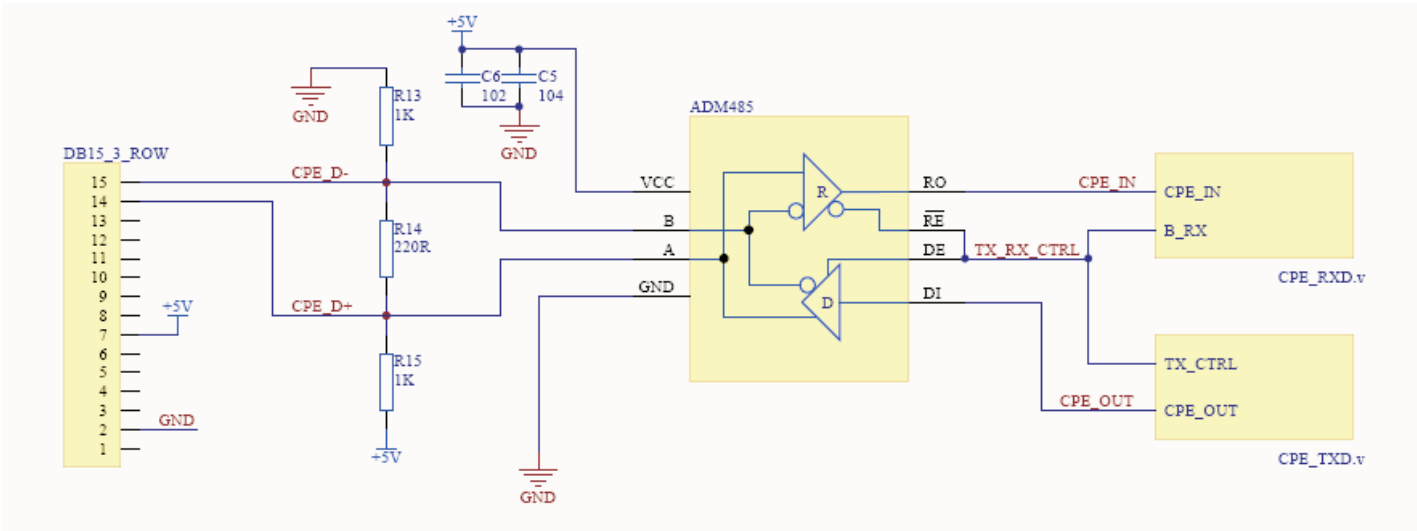
1) Only Receiver Side

For most applications in which high speed and the reliability in communications are fundamental or the most preferred requirement, then polling readout mode is normally the best solution, such as CNC controller, servo driver, rotary tables, touch probe measurements.. and etc.. In such case, each transducer (encoder, linear scale ..) have its own cable to the system, and only the receiver side (CPE_RXD.v) is needed, the schematic of application example is per follows.



2) Bi-directional Communcations (both Transmitter and Receiver sides are required)

For those applications in which there are many Transducers (encoders, linear scales..) working together synchronously, for the ease of system integration and to reduce the number of cables in the system, CPE-bus allows the cascade of all transducers into two pairs of two wires cables { +5V, GND, D+, D- }. In such case, only the interrupt readout mode is feasible. Easson allows the max. of 31 transducers working together in one CPE-bus system, such as in robot, specialized automatic machines, transportation vehicles and etc..., then both transmitter side (CPE_TXD.v) and receiver side (CPE_RXD.v) are needed, the schematic of application example is per follows.



CPE-bus interface circuirity

CRC Generation Algorithm

Easson's CPE_RXD.v and CPE_TXD.v CPE-bus interface modules already included the automatic CRC generation and verification by hardware. No need for user to generate the CRC by the system CPU.

However, there may be some cases user might have to generate the CRC by themselves to incorporate our transducers in their CPE-bus system, such as to reduce the hardware system resource of the receiving circuit or etc.. Followings is the C source programs that implement our CRC generation algorithm.

```
// ***********************************************
// * Polynoimal defaulted by CPE-bus standard   *
// ***********************************************
#define polynomial    0x43   // ** default value in CPE-bus standard
#define CRC_init_val  0x00   // ** default value in CPE-bus standard
#define CRC_msb       0x20


// ***************************************************************************
// *  CRC calculation routine for CPE-bus (optimized for 32bit ARM CPU) *
// ***************************************************************************
unsigned char CRC_cpe( unsigned long STATUS, unsigned long DATA )
{
   unsigned long cpe_data  ;
   unsigned char CRC       ;
   unsigned long xbit      ;

   cpe_data = STATUS & 0x1FF ;   // ** STATUS[8]=HL, STATUS[7:0]=STATUS byte
   CRC      = CRC_init_val   ;

   // ** 42 bit CRC generation  ** //
   for ( xbit = 0x00000200 ; xbit != 0x00000000 ; xbit >>= 1 )
     {
         if ( CRC & CRC_msb )
           {
              CRC <<= 1;
              CRC  ^= polynomial ;
           }
          else
           CRC <<= 1;

        if ( cpe_data & xbit ) CRC ^= polynomial;
     }

   cpe_data = DATA ;  // ** 32 bit Encoder/Scale Data

   for ( xbit = 0x80000000 ; xbit != 0x00000000 ; xbit >>= 1 )
     {
         if ( CRC & CRC_msb )
           {
              CRC <<= 1;
              CRC  ^= polynomial ;
           }
          else
           CRC <<= 1;

        if ( cpe_data & xbit ) CRC ^= polynomial;
     }

  return ( CRC & 0x3F ) ;
}
```