IrDA Control Errata to
IrDA Control Specification version 1.0

Last Modified: July 2, 1998

Following are a list of corrections/clarifications and suggested amendments or changes to the IrDA Control specification Version 1.0 dated February 25, 1998. The closed errata are organized by the date they were closed. Open errata are at the end.
The closed errata should be considered an integral part of the IrDA Control version 1.0 specification. No change to the version number is required based on these changes.
The points are classified according to the following scheme:

- CORRECTION: a change required to correct an error in the existing document corrections may change the specified behavior of the protocol to match that which was originally intended by the authors
- CLARIFICATION: textual enhancement that provides clearer explanation of a protocol element without changing any behavior
- MODIFICATION: a modification of the currently specified behavior that adds but does not delete any function from the protocol
- CHANGE: a modification of the currently specified protocol that may add and or delete function from the protocol
- PROBLEM: a known problem for which an alteration to the document has yet to be proposed.

1. Open Errata

1.1 CRC Field (3.2.3)

CLARIFICATION
An example of VHDL implementation follows to describes the process.

For better understanding of CRC calculation algorithm, an example of VHDL implementation is described.

```
library IEEE;
use IEEE.std_logic_1164.all;
use WORK.all;

entity TxCRC is
  port(
    TxRESET :in          std_logic;
    TxLongFrame :in std_logic;
    TX_CRCCLK :in        std_logic;
    TXCRCIN :in          std_logic;
    ADDTXCRC :in std_logic;
```

```
   CRCOUT :out                std_logic
-------------------------------------------------------------
-- TxRESET          ... Reset signal for this module
-- TxLongFrame      ... H for long frame, L for short frame
-- TX_CRCCLK        ... Clock signal for this module
-- TXCRCIN          ... Input data sequence for this module
-- ADDTXCRC         ... This signal is L while calculating CRC value and H when transmitting CRC value
-- CRCOUT           ... Output signal from this module
-------------------------------------------------------------
);
end TxCRC;

architecture ARC of TxCRC is
  signal CRCFF                :std_logic_vector(15 downto 0);
  signal CRC_LONGIN           :std_logic;
  signal CRC_SHORTIN          :std_logic;

begin
-------------------------------------------------------------
-- Compute CRC8   (x8+x7+x2+1) for Short Frame
-- Compute CRC16  (x16+x15+x2+1) for Long Frame
-------------------------------------------------------------
  CRC_LONGIN    <= (CRCFF(15) xor TXCRCIN) and (not (ADDTXCRC));
  CRC_SHORTIN   <= (CRCFF(7) xor TXCRCIN) and (not (ADDTXCRC));

  process(TxRESET,TX_CRCCLK)
  begin
    if TxRESET='1' then
      CRCFF<="111111111111111";
-------------------------------------------------------------
-- The CRC register is preset to all "1"s
-- prior to calculation of the CRC value.
-------------------------------------------------------------
    elsif TX_CRCCLK='1' and TX_CRCCLK'event then
      if TxLongFrame='1' then
            CRCFF(0)          <= CRC_LONGIN;
            CRCFF(1)          <= CRCFF(0);
        CRCFF(2)      <= CRCFF(1) xor CRC_LONGIN;
        CRCFF(3)      <= CRCFF(2);
        CRCFF(4)      <= CRCFF(3);
        CRCFF(5)      <= CRCFF(4);
        CRCFF(6)      <= CRCFF(5);
        CRCFF(7)      <= CRCFF(6);
        CRCFF(8)      <= CRCFF(7);
        CRCFF(9)      <= CRCFF(8);
        CRCFF(10)     <= CRCFF(9);
        CRCFF(11)     <= CRCFF(10);
        CRCFF(12)     <= CRCFF(11);
        CRCFF(13)     <= CRCFF(12);
        CRCFF(14)     <= CRCFF(13);
        CRCFF(15)     <= CRCFF(14) xor CRC_LONGIN;
```

```
    elsif TxLongFrame='0' then
      CRCFF(0)      <= CRC_SHORTIN;
      CRCFF(1)      <= CRCFF(0);
      CRCFF(2)      <= CRCFF(1) xor CRC_SHORTIN;
      CRCFF(3)      <= CRCFF(2);
      CRCFF(4)      <= CRCFF(3);
      CRCFF(5)      <= CRCFF(4);
      CRCFF(6)      <= CRCFF(5);
      CRCFF(7)      <= CRCFF(6) xor CRC_SHORTIN;
    end if;
  end if;
 end process;

 process(TxRESET,TX_CRCCLK)
 begin
  if TxRESET='1' then
    CRCOUT<='0';
  elsif TX_CRCCLK='1' and TX_CRCCLK'event then
    if ADDTXCRC='0' then
      CRCOUT<=TxCRCIN;
-------------------------------------------------------------
-- This module outputs the input data while ADDTXCRC is L
-------------------------------------------------------------
    else
      CRCOUT<= ( TxLongFrame and not(CRCFF(15)) ) or
            ( not TxLongFrame and not(CRCFF(7)) );
-------------------------------------------------------------
-- This module outputs the CRC value when ADDTXCRC is H
-- CRC value is inverted as it is sent out
-------------------------------------------------------------
    end if;
  end if;
 end process;

end ARC;
```

1.2 Overall Link Specification (3.4)

MODIFICATION

Version 1.0 defined the shape of subcarrier as rectangular infrared pulse (Figure 3.8). This does not intend that subcarrier have to be rectangular infrared pulse. For example, sinusoidal infrared signal also can be used as subcarrier. In addition, DC biased subcarrier (rectangular pulse or sinusoidal signal) also can be used. In other words IrDA Control system by rectangular pulse, sinusoidal signal, DC biased rectangular pulse, and DC biased sinusoidal signal are interoperable.

For reducing spurious infrared emission beyond 1MHz - 2MHz band, it is better to use sinusoidal signal than rectangular pulse. For reducing spurious infrared emission below 1MHz – 2MHz band, it is effective to add DC bias infrared emission to subcarrier at the expense of power consumption. In the implementation of an IrDA Control system,

the shape of subcarrier should be chosen carefully with consideration of the environment in which the system is used.