XStream Module Addresses and Masks

Contents

Abstract 2

Packet Radios.....2

Pseudo Code for receiving 3

Pseudo code for transmitting ...4

Examples.....5

Application Note

XST-AN004b-Masks

June 2012

Website: www.digi.com
Email: rf-experts@digi.com

Abstract

This document explains how to use Digi's Module Address and Module Address Mask to set up global or local addresses for establishing module groups, subnets, etc. It includes examples illustrating point-to-point and point-to-multipoint network configurations with Reliable Delivery modes.

Packet Radios

XStream modules are packet based. This means all data shifted into one module is packetized and sent out the antenna port. Because XStream modules use a peer-to-peer architecture, all modules on the same channel (or network) (ATHP) will receive the packet and decide whether to pass it to the host or to throw it away. Each transmitted packet contains information about the transmitting module in the following structure:

<VID><ATHP><ATDT><PSN><Pay Load><CRC>

<VID>	Factory assigned Vendor ID number
<ATHP>	Channel (or network) number
<ATDT>	Module Address
<PSN >	Packet serial Number - 8-bit number that uniquely identifies each packet
<Pay Load>	Data shifted into module for transmission
<CRC>	16-bit CRC (like a checksum) for error detection

Any module that receives a packet will check the address values and decide what to do with the packet. The options are: receive it as a global packet, receive it as a local packet or throw it away. All packets that do not have an exact match for the <VID> and <ATHP> fields are discarded. The difference between receiving the module address as a global or local packet affects only the Reliable Delivery mode (when retries <ATRR> is enabled).

The mask parameter can be used to allow a base module to receive data from a range of addresses. It may also be used to configure "subnets" of modules that communicate in a group together.

See below for the Pseudo 'C' Code that qualifies the module addresses and address masks.

The Pseudo Code uses the bit-wise "AND" operation: "&". This operation is performed bit by bit on each of the 16 bits in the TXDT, RXDT and RXMK parameters.

Here is the bit-wise AND truth table:

Bit-wise AND operation ("&")				
Operand 1	&	Operand 2	=	Result
0		0		0
0		1		0
1		0		0
1		1		1

Example: hexadecimal: 0x3 & 0x9 = 0x1
binary: 0011 & 1001 = 0001

Pseudo Code for receiving

```
/* ********************************************************
*  Function: Receive_Data()                                *
*                                                          *
*  Description: Algorithm used by XStream modules          *
*               to qualify incoming data packets.          *
*                                                          *
*  Variables:                                              *
*  (parameter types: short = 16 bits, char = 8 bits)       *
*                                                          *
*    short TXDT = Transmitter's Module Address   (ATDT)    *
*    short RXDT = Receiver's Module Address      (ATDT)    *
*    short RXMK = Receiver's Module Address Mask (ATMK)    *
*    char  RXRR = Receiver's Retry setting       (ATRR)    *
*                                                          *
**********************************************************/

Function Receive_Data (TXDT, RXDT, RXMK, RXRR)
{
  if((TXDT & RXMK) == RXMK) /* Is incoming address a global address? */
  {
    Send_data_out_port(); /* Call to function that Sends data out port */
  }

  else if((TXDT & RXMK) == (RXDT & RXMK)) /* Is TXDT a local address? */
  {
    Send_data_out_port(); /* Call to function that Sends data out port */

    if(RXRR > 0)     /* Is Reliable Delivery mode enabled? */
    {
      Send_Ack();   /* Call to some function that Sends Ack */
    }

  }

  else     /* neither global nor local address */
  {
    Purge_buffer();  /* Call to some function that throws data away */
  }

}  /* End Function Receive_Data() */
```

Pseudo code for transmitting

```
/* ************************************************************
*  Function: RF_Transmit_Control()                            *
*                                                             *
*  Description: Algorithm used by XStream modules             *
*               to packetize and transmit data packets.       *
*               This procedure only runs if there is          *
*               data in the data buffer and the               *
*               communication channel is clear.               *
*                                                             *
*  Variables:                                                 *
*  (parameter types: short = 16 bits, char = 8 bits)          *
*                                                             *
*    char  DINC = Number of bytes in Data In Buffer           *
*    short TXDT = Transmitter's Module Address      (ATDT)*
*    short TXMK = Transmitter's Module Address Mask (ATMK)*
*    short TXHP = Transmitter's Channel (network)   (ATHP)*
*    short TXVD = Transmitter's Vendor ID number             *
*    char  TXRR = Transmitter's Retry setting       (ATRR)*
*                                                             *
************************************************************/

Function RF_Transmit_Control (DINC, TXDT, TXMK, TXHP, TXVD, TXRR)
{

  Initialize_RF_Channel(); /* This process takes 35ms */
  while(DINC > 0)    /* Data In Buffer is not empty */
  {
    Assemble_RF_Packet(); /* Packet contains TXDT, TXVD and TXHP params*/

    if((TXDT & TXMK) == TXMK) /* Is DT address a global address? */
    {
      Transmit_Data(); /* Call function that shifts data out antenna */
    }                  /* Global packets not subject to TXRR */

    else               /* TXDT is local address */
    {
      Transmit_Data(); /* Call to function that Sends data out port */
      while(TXRR-- > 0)     /* Retransmit up to TXRR times  */
      {
        if(Receive_ACK())   /* TRUE if ACK is received on local address*/
        {
          TXRR=0;           /* Condition to exit while loop */
        }
        else                /* If ACK not received  */
        {
          Retransmit_Data();/* Function that resends previous packet */
        }

      }  /* End while loop for retransmitting */

    }  /* End TXDT is local address */

  }  /* End while Data In Buffer is not empty */

  Close_RF_Channel();   /* Allows other modules to communicate */

}  /* End Function RF_Transmit_Control() */
```

Examples

Example 1 – Global address

A transmitter can send data to any remote module regardless of the remote module's address by sending a packet with a DT of 0xFFFF. Consider the following parameters:

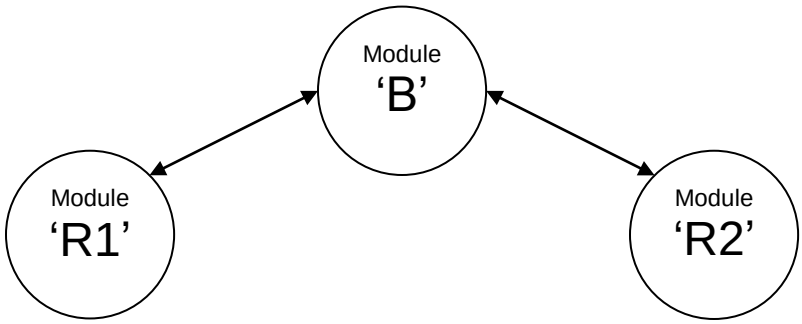
Module Parameters		
Command	Transmitter	Receiver
ATDT	0xFFFF	0x0000
ATMK	0xFFFF	0xFFFF

With these parameters we look at the Pseudo Code for the Receive_Data Function. TXDT=0xFFFF and RXMK=0xFFFF. Is the incoming address a global address?

```
If((TXDT & RXMK) == RXMK)  /* Is incoming address a global address? */
```

Yes. The incoming address is a global address. The receiving module will send the data out the port and will not send an ACK even if RXRR is set.

Example 2 – Reliable Delivery with polling



Scenario: Module 'B', the base node will communicate first with 'R1' then with 'R2' with reliable delivery mode enabled to overcome transmission errors.

Module Parameters			
Command	'B'	'R1'	'R2'
ATDT	0x0001	0x0001	0x0002
ATMK	0xFFFF	0xFFFF	0xFFFF
ATRR	0x0A	0x0A	0x0A

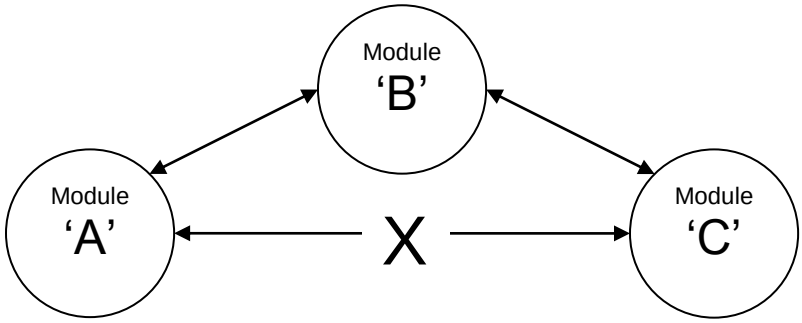
With these settings the module 'B' is configured to communicate point-to-point with 'R1'. Module 'B' will attempt 10 (0x0A) times to deliver any packets and will only move on to subsequent packets after either the RR parameter expires or a positive ACK is received.

With module 'B's DT=0x0001, 'B' will not be able to receive data from 'R2'. To communicate with 'R2', the host attached to module 'B' will use either binary or AT commands to configure the DT parameter of module 'B' to 0x0002 to establish a point to point connection with 'R2'.

If the DT address will be changing frequently (for a polling application), binary commands are more suitable as the address may be changed in less than 10ms. AT commands may be used to change the DT address though this will take 2 seconds by default. The time required to

enter AT command mode may be set to as little as 300ms by changing the before and after silence guard times (ATBT=2 and ATAT=1) for module 'B'.

Example 3 – Repeater function

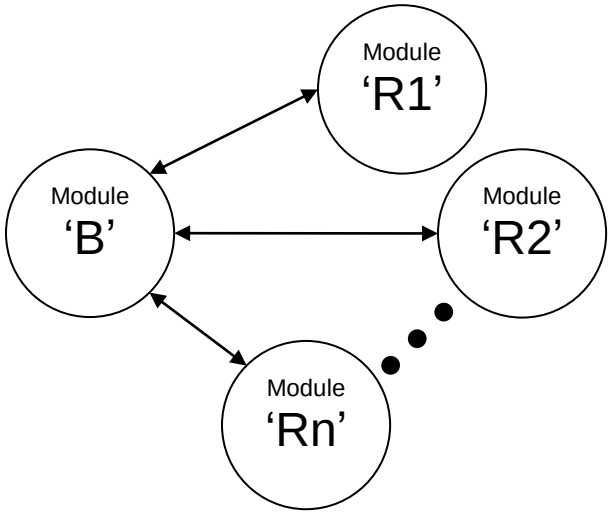


Scenario: Modules 'A' and 'C' communicate through store & forward repeater 'B' (see app note XST-AN001 on repeaters) though they may be within range of each other at times. We configure the addresses and masks to ensure that 'A' and 'C' communicate only through 'B', not directly.

Module Parameters			
Command	'A'	'B'	'C'
ATDT	0x8000	0x8001	0x0001
ATMK	0xFFFE	0x7FFE	0x7FFF

Note that 'B' sends out packets that either 'A' or 'C' can accept while 'A' and 'C' can 't communicate with each other directly. Also keep in mind that when 'B' repeats a packet, a transmitter will receive a copy of its own packet.

Example 4 – Multi-point reliable delivery



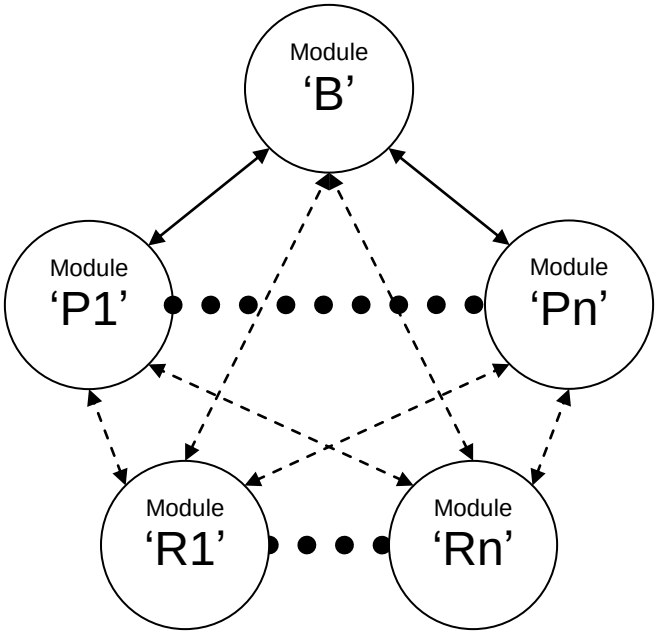
Scenario: Module 'B', the base unit communicates with remote modules 'R1', 'R2'. . .'Rn' using a reliable delivery mode of transmitting every packet multiple times so each remote has multiple opportunities to receive it. When any module receives one copy of the packet it will ignore subsequent copies (only one copy is sent out its serial port). Remote modules respond using the same technique so that the base has multiple opportunities to receive each packet

from the remote modules. Remote modules also do not receive packets from each other, only from the base.

Module Parameters					
Command	'B'	'R1'	'R2'	. . .	'Rn'
ATDT	0x7FFF	0x8001	0x8002	. . .	0x800n
ATMK	0x8000	0x7FFF	0x7FFF	. . .	0x7FFF
ATRR	3	3	3	. . .	3

Note that the ATDT parameter of the base module will be accepted as a global address by all remote modules so they will not send ACK; and the base module will expire the retries with every packet. When a remote module has data to transmit, it will be ignored by other remotes while it likewise is accepted as a global packet by the base and will be retransmitted ATRR times. In this example, an ATRR = 3 results in one transmission and 3 retransmissions for a total of four copies being sent. This mode of operation effectively decreases the available data throughput by a factor of four. Higher ATRR values can increase reliability while decreasing throughput.

Example 5 – Roaming area network



Scenario: Remote modules 'R1' … 'Rn' communicate back to base module 'B' either directly or through store & forward repeaters 'P1' … 'Pn' (see app note XST-AN001 on repeaters). This allows the remote modules to roam over an extended area covered by the modules 'B', 'P1', …, 'Pn'. More store & forward repeaters can be added to give the remote modules a larger roaming area.

Module Parameters							
Command	'B'	'P1'	. . .	'Pn'	'R1'	. . .	'Rn'
ATDT	0xFFFF	0x0001	. . .	0x000n	0xFFFF	. . .	0xFFFF
ATMK	0	0xFFFF	. . .	0xFFFF	0	. . .	0

With these settings base module 'B' is configured with global address 0xFFFF to send data *to* every repeater module 'Pn' and remote module 'Rn'; we also configure 'B' with global address mask 0 to receive data *from* every repeater module and remote module. Remote modules 'R1' … 'Rn' are configured with the same global address and global address mask to communicate

back with the base or repeater modules. Repeater modules 'P1' … 'Pn' are configured to *not* communicate with each other, thus avoiding an infinite loop back situation.

Note that a roaming remote module 'Rn' will occasionally be within range of base module 'B' and one or more of the repeater modules, and will likely receive multiple copies of the same message. The base module 'B' will also likely receive multiple copies of each message originated by 'Rn'. Care should be taken to plan for these repeated messages.