Application Note AN-014
Using External RT Addressing

abaco SYSTEMS

Copyrights

App Note 014: Using External RT Addressing

Document Revision: 1.20

Document Date: 1 November 2016

Abaco Systems, Inc.
26 Castilian Drive, Suite B
Goleta, CA 93117
Main +1 805-965-8000 or +1 805- 883-6101
Support +1 805-965-8000 or +1 805- 883-6097

support@abaco.com (email)

http://www.abaco.com/products/avionics

Additional Resources

For more information, please visit the Abaco Systems website at:

www.abaco.com

Contents

Introduction

Many MIL-STD-1553 terminals determine their RT address from external jumpers or signals. This allows the system designer to include the RT address assignment in the connector attached to the unit. Some remote terminals used in MIL-STD-1760 applications are required to respond with the correct status word and BUSY bit set before software has initialized the RT and assigned the RT address.

Many Abaco Systems MIL-STD-1553 products support the external RT address feature. This application note describes how to use this feature.

See the Abaco Systems "MIL-STD-1553 Hardware Installation and Reference Manual" (Publication No. 1500-046) for detailed information including which products support the external RT address feature, RT address pin assignments, and other product-specific information. Also, consult the section entitled "Hardwired RT Address" for additional information on RT addressing.

Note that many Abaco Systems MIL-STD-1553 products also support setting the RT address from values stored in on-board flash memory. See the MIL-STD-1553 Hardware Installation and Reference Manual (Section entitled "Flash Configurable 1553 Options") for more information on flash-based RT address initialization.

Hardware Settings for External RT Address

Setting the External RT Address Value

For each channel that supports the external RT address, you must set the five address lines and parity to select the desired RT address.

An external RT address is provided on six input signals – a 5-bit RT address plus parity (odd). The signals are internally pulled up, so all you have to do is tie selected lines to GND. Note that connecting a line to GND is equivalent to a logic "0" and leaving the line OPEN is equivalent to logic "1". The following table shows a few examples of external RT address settings:

	RT Address 0	RT Address 13	RT Address 23	RT Address 30
RTA_0	0 (GND)	1 (Open)	1 (Open)	0 (GND)
RTA_1	0 (GND)	0 (GND)	1 (Open)	1 (Open)
RTA_2	0 (GND)	1 (Open)	1 (Open)	1 (Open)
RTA_3	0 (GND)	1 (Open)	0 (GND)	1 (Open)
RTA_4	0 (GND)	0 (GND)	1 (Open)	1 (Open)
RTA_P	1 (Open)	0 (GND)	1 (Open)	1 (Open)

Enabling the External RT Address

The external RT address must be enabled before it will take effect. The method for enabling the external RT address is described for each product in the MIL-STD-1553 Hardware Installation and Reference Manual.

Testing the External RT Address

After you have made the appropriate settings to select and enable the external RT address you can install the board in the system and power it up. Connect the appropriate channel to a 1553 bus and use another 1553 device (or bus analyzer, like BusTools/1553) as the Bus Controller. Send a non-broadcast command word to the selected RT address. The RT should respond with a status word with the BUSY bit set. If you do not see a status word from the RT, then there is a problem with your board configuration for external RT address.

NOTE: *If you have problems here, check the firmware version on your board. If your firmware version is less than 3.88, update and try again.*

Software Usage of External RT Address

After power-up, the channel responds with a status word with the BUSY bit set for the RT address selected until the user software takes over operation of the RT.

Initialization

The channel is initialized using the API function **BusTools_API_InitExtended**. If the external RT address is enabled but the RT address signals are invalid (parity error), then this function will return the error code 45 (BTD_RTADDR_PARITY). The channel will NOT initialize unless a valid external RT address is set.

Reading the External RT Address Value

The API provides the function **BusTools_RT_GetRTAddr** to allow the user application to read the value of the external RT address lines. This value can then be used to setup the appropriate RT configuration.

Enabling the RT for Normal Operation

All RT setup functions work just as they would without the external RT address option, but you will want to use the RT address value returned from **BusTools_RT_GetAddr** for your RT programming.

This is demonstrated in the example program given below.

Example Program

```
/*===========================================================================*
 * FILE:                E X A M P L E _ R T 5 . C                            *
 *===========================================================================*
 *
 * COPYRIGHT (C) 2004 BY
 *          CONDOR ENGINEERING, INC., SANTA BARBARA, CALIFORNIA
 *          ALL RIGHTS RESERVED.
 *
 *          THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND
 *          COPIED ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH
 *          THE INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR ANY
 *          OTHER COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE
 *          AVAILABLE TO ANY OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF THE
 *          SOFTWARE IS HEREBY TRANSFERRED.
 *
 *          THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT
 *          NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY CONDOR
 *          ENGINEERING.
 *
 *===========================================================================*
 *
 * FUNCTION:    EXAMPLE PROGRAM
 *              This is a basic example program that sets up a simple
 *              RT Simulation.  Simulates RTx with two subaddresses,
 *              SA1 RECEIVE and SA2 TRANSMIT.  The RT address is determined
 *              by the EXTERNAL RT ADDRESS inputs, or defaults to 1 if
 *              this is not available.
 *
 *              NOTE:  The external RT address feature is only available
 *              on the QPCI-1553, QPMC-1553, and QVME-1553 products.
 *
 *===========================================================================*/

/* $Revision:  X.xx Release $
   Date        Revision
  ----------   ----------------------------------------------------------------
   09/01/04    Initial version.  RSW
*/

#include "busapi.h"
#include <stdio.h>

// Constants for device information needed by BusTools_API_InitExtended.
// MODIFY THESE CONSTANTS TO MATCH YOUR CONFIGURATION, refer to documentation
// on the BusTools_API_InitExtended function for help.
#define MY_CARD_NUM         0
#define MY_BASE_ADDR        0
#define MY_IO_ADDR          0
#define MY_PLATFORM         PLATFORM_PC
#define MY_CARD_TYPE        QPCI1553
#define MY_CARRIER          NATIVE
#define MY_SLOT             CHANNEL_1
#define MY_MAPPING          CARRIER_MAP_DEFAULT

#define DEFAULT_RT_ADDR 1

// Main program
void main() {
  BT_INT                status;
  BT_UINT               flag = 1;
  API_RT_ABUF           Abuf_RTx;
  API_RT_CBUF           Cbuf_RTxSA1R;
  API_RT_CBUF           Cbuf_RTxSA2T;
  API_RT_MBUF_WRITE     msg_buffer_write;
```

```
API_RT_MBUF_READ      msg_buffer_read;
int                   i;
char                  c;
unsigned short        cmd, sts;
int                   rtaddr;
BT_UINT               wRev1, wRev2;

// Initialize API and board.
printf("Initializing API . . . ");
status = BusTools_API_InitExtended(MY_CARD_NUM, MY_BASE_ADDR, MY_IO_ADDR, &flag,
                          MY_PLATFORM, MY_CARD_TYPE, MY_CARRIER, MY_SLOT, MY_MAPPING);

if (status == API_SUCCESS) {
  printf("Success.\n");

  status = BusTools_GetRevision(MY_CARD_NUM, &wRev1, &wRev2);
  if (status != API_SUCCESS) printf("Error %d on BusTools_GetRevision.\n",status);
  printf("Microcode Revision = %x\n",wRev1);
  printf("API Revision       = %d\n",wRev2);

  printf("Check External RT Address . . . ");
  status = BusTools_RT_GetRTAddr(MY_CARD_NUM, &rtaddr);
  if (status != API_SUCCESS) printf("ERROR = %d\n",status);
  else printf("EXT RT ADDR = %d\n", rtaddr);

  if ((rtaddr < 0) || (rtaddr > 30)) {
      rtaddr = DEFAULT_RT_ADDR;
      printf("INVALID EXTERNAL RT ADDRESS, USING DEFAULT RT ADDR = %d\n", rtaddr);
  }

  // Initialize and reset memory.  Minimum BM setup.
  printf("BM Init . . . ");
  status = BusTools_BM_Init(MY_CARD_NUM, 1, 1);
  if (status != API_SUCCESS) printf("ERROR = %d\n",status);

  // Select External Bus.
  printf("Bus select . . . ");
  status = BusTools_SetInternalBus(MY_CARD_NUM, 0);
  if (status != API_SUCCESS) printf("ERROR = %d\n",status);

  // Now lets set up an RT.
  printf("Initializing RT functionality . . . ");
  status = BusTools_RT_Init(MY_CARD_NUM, 0);
  if (status == API_SUCCESS) {
    printf("Success.\n");

    // Setup RT address buffer for our RT (RTx)
    Abuf_RTx.enable_a = 1;               // Respond on bus A
    Abuf_RTx.enable_b = 1;               // Respond on bus B
    Abuf_RTx.inhibit_term_flag = 1;      // Inhibit terminal flag in status word
    Abuf_RTx.status = rtaddr << 11;
    Abuf_RTx.bit_word = 0x0000;          // Set BIT word (for mode code 19)
    status = BusTools_RT_AbufWrite(MY_CARD_NUM, rtaddr, &Abuf_RTx);
    if (status != API_SUCCESS) printf("Error %d on BusTools_RT_AbufWrite.\n",status);

    // Setup a control buffer - RTx, SA1, Receive, 1 buffer.
    Cbuf_RTxSA1R.legal_wordcount = 0xFFFFFFFF;  // any word count is legal.
    status = BusTools_RT_CbufWrite(MY_CARD_NUM, rtaddr, 1, 0, 1, &Cbuf_RTxSA1R);
    if (status != API_SUCCESS) printf("Error %d on BusTools_RT_CbufWrite.\n",status);

    // Setup a control buffer - RTx, SA2, Transmit, 1 buffer.
    Cbuf_RTxSA2T.legal_wordcount = 0xFFFFFFFF;  // any word count is legal.
    status = BusTools_RT_CbufWrite(MY_CARD_NUM, rtaddr, 2, 1, 1, &Cbuf_RTxSA2T);
    if (status != API_SUCCESS) printf("Error %d on BusTools_RT_CbufWrite.\n",status);

    // Put some data in our transmit buffer
    msg_buffer_write.enable = 0;         // No interrupts enabled
```

```
      msg_buffer_write.error_inj_id = 0;    // No error injection
      for (i=0; i<32; i++)
        msg_buffer_write.mess_data[i] = 0xAB00 + i;

      status = BusTools_RT_MessageWrite(MY_CARD_NUM, rtaddr, 2, 1, 0, &msg_buffer_write);
      if (status != API_SUCCESS)
        printf("Error %d on BusTools_RT_MessageWrite.\n",status);

      // Now lets turn on our RT
      status = BusTools_RT_StartStop(MY_CARD_NUM, 1);
      if (status != API_SUCCESS) printf("Error %d on BusTools_RT_StartStop.\n",status);

      do {
        printf("\nInput Q to quit, anything else to read RTx SA1 RCV data buffer.\n");
        scanf("%c",&c);

        if (c != 'Q') {
          printf("\nReading data for RTx SA1 RCV . . . \n");

          // Read the data buffer
          status = BusTools_RT_MessageRead(MY_CARD_NUM, rtaddr, 1, 0, 0,
                                &msg_buffer_read);
          if (status != API_SUCCESS) {
            printf("ERROR READING RT MESSAGE, Error = %d\n",status);
          }
          else {
            // Print the data buffer.
            printf("RTx SA1 RCV Data Buffer:\n");
            memcpy(&cmd, &msg_buffer_read.mess_command,
                        sizeof(msg_buffer_read.mess_command));
            printf("Command word = %04X\n",cmd);
            for (i=0; i<32; i++) {
              printf("%04x ",msg_buffer_read.mess_data[i]);
              if (!((i+1)%8)) printf("\n");
            } // End of for loop to print data
            memcpy(&sts, &msg_buffer_read.mess_status,
                        sizeof(msg_buffer_read.status));
            printf("Status word  = %04X\n",sts);
          } // End of if (error reading message) else
        } // End of if (c != 'Q')
      } while (c != 'Q'); // End of do-while

      printf("Stopping RT simulation . . . ");
      status = BusTools_RT_StartStop(MY_CARD_NUM, 0);
      if (status != API_SUCCESS) printf("Error = %d.\n", status);
      else printf("Stopped.\n");

    } // End of if (RT init successful)
    else printf("ERROR = %d\n",status);

    // We're done.  Close API and board
    printf("\nClosing API . . . ");
    status = BusTools_API_Close(MY_CARD_NUM);
    if (status == API_SUCCESS)
      printf("Success.\n");
    else
      printf("FAILURE, error = %d\n", status);
  } // End of if (initialization successful)
  else {
    printf("FAILURE, error = %d\n", status);
    if (status == BTD_RTADDR_PARITY) {
        printf("INVALID EXTERNAL RT ADDRESS, BAD PARITY.\n");
    }
  }

  printf("FINISHED.\n");
} // End of main
```