Example Name	Description	BusTools/1553-API Functions Used
example_auto_sync_mode.c	This example shows how to setup the time tag auto sync mode. This mode synchronizes the time tag to an external pulse. The example also sets up a dead man timer to detect if the external 1PPS pulse does not occur on time. There is optional code to have the 1553 interface board generate the 1PPS pulse instead of external 1PPS.	BusTools_API_OpenChannel, BusTools_API_Close, BusTools_SetInternalBus, BusTools_BoardIsV6, BusTools_SetV6TrigIn, BusTools_SetV6TrigOut, BusTools_DiscreteSetIO, BusTools_DiscreteTriggerOut, BusTools_DiscreteTriggerIn, BusTools_TimeTagMode, BusTools_ExtTrigIntEnable, BusTools_BM_StartStop, BusTools_BC_StartStop, BusTools_RT_StartStop, BusTools_RegisterFunction, BusTools_BC_Init, BusTools_BC_MessageAlloc, BusTools_BC_MessageWrite, BusTools_BC_MessageRead, BusTools_TimeGetString, BusTools_DiscreteWrite, BusTools_RT_Init, BusTools_RT_CbufWrite, BusTools_RT_MessageWrite, BusTools_RT_AbufWrite, BusTools_BM_Init, BusTools_BM_MessageAlloc

example_bcrt_2ch.c	This is a 2 CHANNEL example program that sets up one channel as a Remote Terminal and another channel as Bus Controller on a multi-channel board. The first channel is setup as RT1 with two subaddresses,SA1 RECEIVE and SA2 TRANSMIT. BusTools_RegisterFunction is called to implement a user callback function to process the RT tranmit message data. The second channel is setup as BC with a list of two messages, 1-R-1-32 and 1-T-2-32, in a 500ms minor frame. This BC list runs until stopped by user input. User can either display the data for the 1-T-2-32 message or quit. A interrupt callback function is setup by calling BusTools_RegisterFunction. The rt_intFunction displays the RT data. The user can also switch between Transformer or direct couple or dump the channels memory contents through command-line input. This test requires a multi-channel board and either a physical connection between channel 1 and channel 2 or if you are using a QPCX-1553 or QPCI-1553 you can use the test bus.	BusTools_API_OpenChannel, BusTools_API_Close, BusTools_SetInternalBus, BusTools_TimeTagMode, BusTools_GetBoardType, BusTools_SetTestBus, BusTools_SetVoltage, BusTools_RT_StartStop, BusTools_BC_StartStop, BusTools_RegisterFunction, BusTools_RT_MessageRead, BusTools_TimeGetString, BusTools_RT_Init, BusTools_StatusGetString, BusTools_RT_AbufWrite, BusTools_RT_CbufWrite, BusTools_RT_MessageWrite, BusTools_BC_Init, BusTools_BC_MessageAlloc, BusTools_BC_MessageWrite,
example_bc_1Shot.c	This console application shows how to setup BC messages that transact just once and then stop the Bus Controller. This setup is referred to as a 1Shot message frame. Three one shot frames are created and BusTools_BC_Start is used to start the BC at the beginning one of the selected bus list.	BusTools_API_OpenChannel, BusTools_API_Close, BusTools_SetInternalBus, BusTools_TimeTagMode, BusTools_RT_StartStop, BusTools_BC_Start, BusTools_RegisterFunction, BusTools_BC_Init, BusTools_BC_MessageAlloc, BusTools_BC_MessageWrite, BusTools_BC_MessageRead, BusTools_RT_Init, BusTools_RT_CbufWrite, BusTools_RT_MessageWrite, BusTools_RT_AbufWrite, BusTools_RegisterFunction

example_bc_aperiodic.c	This example shows how to setup and run aperiodic messages. This example sets up a periodic bus list running at 1hz and two aperiodic bust lists. The user can command either of the aperiodic list to run at low or priorty or high priority.	BusTools_API_OpenChannel, BusTools_API_Close, BusTools_SetInternalBus, BusTools_BC_AperiodicRun, BusTools_BC_StartStop, BusTools_RegisterFunction, BusTools_BC_Init, BusTools_BC_MessageAlloc, BusTools_BC_MessageWrite, BusTools_BC_MessageRead
example_bc_auto_incr.c	This exampe sets up the Bus Controller, Remote Terminal and Bus Monitor and then enables auto-increment on the Bus Controller message buffer 3. BusTools_BC_AutoIncrMessageData is used setup the increment process. Message buffer 3 contains a 32-word receive command to RT4 SA4. The last word in the command, data word word 32 increments each time the message transacts.	BusTools_API_OpenChannel BusTools_SetInternalBus BusTools_BM_Init BusTools_BM_MessageAlloc BusTools_RT_Init BusTools_RT_CbufWrite BusTools_RT_MessageWrite BusTools_RT_AbufWrite BusTools_BC_Init BusTools_BC_MessageAlloc BusTools_BC_MessageWrite BusTools_BC_AutoIncrMessageData BusTools_BM_StartStop BusTools_RT_StartStop BusTools_BC_StartStop BusTools_DumpMemory BusTools_RegisterFunction BusTools_BC_MessageRead BusTools_API_Close

example_bc_branch_on_data.c	This application shows how to setup Message Scheduling, Frame-Start-Timing and Conditional Branch 2. Conditional Branch 2 branches based on the value in an on-board memory location allocated using BusTools_MemoryAlloc. The application writes walues to this location to controlwhich conditional messages transactions. There are 10 conditional messages. Each is run based on a different bit-wise (1,2,4,8...) vaule written to the allocated memory address. By using the data_mask and data_value the application can send any combination of conditional messages by entering a value between 1 and 0x3ff. That value is written to the allocated memory location. The Conditional Branch message evaluate the data at that test address to determine whether to branch. This is an alternate approach to apperiodic messaging, The application can select any combination of messages to run after the periodic messages transact. Unlike aperiodic messages youneed to ensure there is enough time in the frame after the periodic messages for the selected conditional messages to run. All messages run in the current frame. After each conditional message runs the corresponding bit at the test address is cleared so the message only runs once per data update. Note: This example run on both the new V6 firmware and older V4/V5 firmware. In the new V6 firmware allocate 32 bit of memory rather than 16 bits used in the older firmware.	BusTools_API_OpenChannel, BusTools_API_Close, BusTools_SetInternalBus, BusTools_BC_Init, BusTools_BC_MessageAlloc, BusTools_BC_MessageWrite, BusTools_MemoryAlloc32, BusTools_MemoryWrite2, BusTools_MemoryAlloc, BusTools_MemoryWrite, BusTools_BoardIsV6, BusTools_BC_StartStop, BusTools_DumpMemory, BusTools_RegisterFunction, BusTools_BC_MessageRead,
example_bc_branch_on_status.c	This example program demonstrates the use of Conditional Branches. There are several option for branching. In this example there are two branches used, CONDITIONAL BRANCH and CONDITIONAL BRANCH 3. CONDITIONAL BRANCH branches on data in the imeediately preceeding message in the bus list, either command word, status word or data. CONDITIONAL BRANCH 3 branches the same values, but the message buffer is specified in the setup. In this example, 8 message buffers are created. Message 1 is a CONDITIONAL BRANCH that branches when the Status Request bit (SRQ) is set in the status response on message 0, the immediately preceeding message. If the SRQ is set then Msg 2 transact otherwise it is skipped. Message buffer 6 is a CONDITIONAL BRANCH 3 that branches if the data in the last data word of Msg 4 is 0xabcd. If the data is set to 0xabcd then Msg 7 transact, otherwise it is skipped.	BusTools_API_OpenChannel, BusTools_API_Close, BusTools_SetInternalBus, BusTools_BC_Init, BusTools_BC_MessageAlloc, BusTools_BC_MessageWrite, BusTools_BC_StartStop, BusTools_BC_MessageRead, BusTools_TimeGetFmtString, BusTools_RegisterFunction

example_bc_cond_count.c	This application shows how setup a BC messages and use conditional branch on count to run a specific number of frames. There is also a branch on data in a previous message. The BC sets up message buffer 1 as a Conditional Branch 3. That branch looks at the data in the third data word of the previous message. If the third data word equals 0xA002, the branch skips the next message. In this example that condition is always TRUE and the next message in the bus list, RT3 SA3 WC3 TX is never transmitted. You can alter the data and see how that affects the branch. Message buffer 5 is setup as a Conditional Branch 2 that is set to branch on the 10th occurrence. Because of the way the firmware decrements the branch count, it is set to n-1. In this case count is set to 9 (10-1). The count is decremented each time the branch executes. When it hits to zero the branch occurs. In the example the non-branching false condition is a nop message setting the end-of-frame with the next-message pointing to the frame start at message buffer 0. When the branch does executes the message buffer contains a BC LAST command that stops the Bus Controller. This setup executes the frame 10 times then halts the BC. You change the two count values and see how it alters the execution.	BusTools_API_OpenChannel, BusTools_SetInternalBus, BusTools_BC_Init, BusTools_BC_MessageAlloc, BusTools_BC_MessageWrite, BusTools_BC_StartStop, BusTools_BC_MessageRead, BusTools_BC_IsRunning2, BusTools_RegisterFunction
example_bc_data_transfer.c	This example program sets up a Bus Controller to transfer a block of data to an RT. This approach could be used to transfer any amount of data; in this case 1024 words are transferred. An interrupt event signals when the BC message completes so we can write the next buffer of data to the BC message buffer. A Conditional Branch on count is used to transfer 32 messages of 32 words (32 x 32 = 1024). When the transfer is complete the Bus Controller halts and the example exits. This program also uses BusTools_BC_MessageUpdate to update only the data portion of the Bus Controller message buffer.	BusTools_API_OpenChannel, BusTools_SetInternal, Bus, BusTools_BC_Init, BusTools_BC_MessageAlloc, BusTools_BC_MessageWrite, BusTools_BC_StartStop, BusTools_BC_MessageRead, BusTools_BC_MessageUpdate, BusTools_RegisterFunction, BusTools_API_Close

example_bc_data_wrap.c	This example shows how to wrap transmit and receive buffer so that data from a BC transits command can send the data from a BC receive command. In this example the data buffer from the first message (0) is linked to the second message (1) by reading the data buffer address in buffer 0, the transmit command, and over-writing the data buffer address in the second buffer (1) with the transmit buffer address. That way the data from the transmit command is used by the receive command.	BusTools_API_OpenChannel, BusTools_SetInternalBus, BusTools_TimeTagMode, BusTools_BC_Init, BusTools_BC_MessageAlloc, BusTools_BC_MessageWrite, BusTools_BC_MessageGetaddr, BusTools_BoardIsV6, BusTools_MemoryRead2, BusTools_MemoryWrite2, BusTools_MemoryRead, BusTools_MemoryWrite, BusTools_BC_StartStop, BusTools_RegisterFunction, BusTools_BC_MessageRead, BusTools_TimeGetFmtString, BusTools_API_Close
example_bc_deadman_timer.c	This example shows how to configure a watchdog timer to stop the Bus Controller if the host does not reset the timer. This example uses a Condition Branch on count (CONDITION2) as the timer. The Conditional Branch is setup to branch when the count decrements to 0. The count value is decremented each time the branch executes. The branch command is a BC_CONTROL_HALT that stops the Bus Controller. Otherwise the frame keeps running. To keep the branch from ever executing the halt, the host must reset the counter value to prevent it from reaching 0. To do this, the example gets the address of the BC buffer containing the branch command and resets the count value by writing the initial count value back into the count register thus restarting the down-count. This example runs until the user enters the kill command from the console. Then the count value is no longer reset and the Bus controller halts. Two callback functions are used. One for Bus Controller messages prints the message data to the console. The other is for the conditional branch. This callback resets the counter value. See how altering the count value affects the timer.	BusTools_API_OpenChannel, BusTools_SetInternalBus, BusTools_TimeTagMode, BusTools_BC_Init, BusTools_BC_MessageAlloc, BusTools_BC_MessageWrite, BusTools_BC_MessageGetaddr, BusTools_MemoryRead2, BusTools_BoardIsV6, BusTools_BC_StartStop, BusTools_RegisterFunction, BusTools_BC_MessageRead, BusTools_TimeGetFmtString, BusTools_MemoryWrite2, BusTools_MemoryWrite

example_bc_error_inj.c	This example program sets up a Bus Controller message list containing two messages, 1-R-1-32 and 2-T-2-32, in a 20ms minor frame. This BC list runs until stopped by user input. Error are inject into the command or data words of the two messages. The example injects a PARITY error on the command word either for 1-R-1-32 or 2-T-2-32. It also inject a parity error into the first data of the receive command. Notice the effect of the error by monitoring with an external Bus Analyzer like BusTools/1553. When a parity error is injected on a command word it becomes an invalid command the Analyzer (monitor) ignores the message. If error is injected on the data word then the command is valid and the monitor will show errors like invalid word and parity error.	BusTools_API_OpenChannel, BusTools_SetInternalBus, BusTools_BC_Init, BusTools_BC_MessageAlloc, BusTools_BC_MessageWrite, BusTools_BC_StartStop, BusTools_EI_EbufWriteENH, BusTools_API_Close
example_bc_join.c	Channel Sharing allows multiple applications to run on a single 1553 channel. Channel sharing requires that only one application initialize a channel. That application must share the channel by calling BusTools_API_ShareChannel. Other application can join the shared channel by calling BusTools_API_JoinChannel. There can be only one Bus Controller application, one Bus Monitor application and one Remote Terminal application per channel. This example Shows a BC application joining an already initialized channel.	BusTools_API_JoinChannel, BusTools_SetInternalBus, BusTools_TimeTagMode, BusTools_BC_Init, BusTools_BC_MessageAlloc, BusTools_BC_MessageWrite, BusTools_BC_MessageRead, BusTools_TimeGetString, usTools_RegisterFunction, BusTools_BC_StartStop, BusTools_API_QuitChannel
example_bc_message_read_types.c	This example program shows how to program the Bus Controller to run a multiple minor frames and to demonstrate programming all of the message types available to the Bus Controller. The example initializes the channel and sets it for external bus. Following those initialization steps it initializes the Bus Controller and builds two minor frames. There are examples of each of the BC message in the frames.	BusTools_API_OpenChannel, BusTools_TimeTagMode, BusTools_SetInternalBus, BusTools_SetBroadcast, BusTools_BC_Init, BusTools_BC_MessageAlloc, BusTools_BC_MessageWrite, BusTools_BC_MessageRead, BusTools_BoardIsV6, BusTools_TimeGetFmtString, BusTools_RegisterFunction, BusTools_BC_StartStop, BusTools_API_Close

example_bc_message_types.c	This example program shows how to program the Bus Controller to run a multiple minor frames and to demonstrate programming all of the message type available to the Bus Controller. The example initializes the Bus Controller channel. Then it sets the channel for external bus. Following those initialization steps it initializes the Bus Controller and builds two minor frames. There are examples of each of the BC message in the frames.	BusTools_API_OpenChannel, BusTools_SetInternalBus, BusTools_SetBroadcast, BusTools_BC_Init, BusTools_BC_MessageAlloc, BusTools_BC_MessageWrite, BusTools_BoardIsV6, BusTools_BC_StartStop, BusTools_API_Close
example_bc_msg_blk_read.c	This example program shows how to program the Bus Controller to run a simple three message minor-frame. The example initializes the channel the Bus Controller is running on. Then sets the channel for external bus. Following those initialization steps it initializes the Bus Controller and builds a simple 3-message frame. The messages in the frame are linked together and the frame will run continually until stopped by the user. This example also shows how to process the messages using the function BusTools_BC_ReadLastMessageBlock. This function parses the interrupt queue to determine the BC messages that have transacted since the last call. In order for a BC message to record it must have the BC_CONTROL_INTERRUPT set and the BC initialization must define an interrupt condition like BT1553_INT_END_OF_MESS. This processing differs from BusTools_RegisterFunction in that your application must provide a timing loop to call BusTools_BC_ReadLastMessageBlock periodically. The timing must set so the BC messages (and other messages that may be recorded in the interrupt queue) do not overwrite unprocessed entries. This example uses a 50 millisecond delay. This application use the Windows function kbhit to break out of timing loop. If using this example on non-Windows systems, you will need to provide a kbhit function.	BusTools_API_OpenChannel, BusTools_SetInternalBus, BusTools_TimeTagMode, BusTools_BC_Init, BusTools_BC_MessageAlloc, BusTools_BC_MessageWrite, BusTools_BC_StartStop, BusTools_BC_ReadLastMessageBlock, BusTools_BoardIsV6, BusTools_TimeGetFmtString, BusTools_API_Close
example_bc_msg_read.c	This example program shows how to program the Bus Controller to run a simple three message minor-frame. The example initializes the channel and sets the channel for external bus. Following those initialization step it initializes the Bus Controller and builds a simple 3-message frame. The messages in the frame are linked together and the frame will run continually until stopped by the user. A user callback function is set up by using the function BusTools_RegisterFunction. The user function is invoked each time the registered interrupt event is found in the channels interrupt queue. In this example the callback function is registered for callback on EVENT_BC_MESSAGE. Those events occur on BC message that have interrupt enabled (BC_CONTROL_INTERRUPT. The user callback function then process the data in each message.	BusTools_API_OpenChannel BusTools_SetInternalBus BusTools_TimeTagMode BusTools_BC_Init BusTools_BC_MessageAlloc BusTools_BC_MessageWrite BusTools_BC_StartStop BusTools_RegisterFunction BusTools_BC_MessageRead BusTools_BoardIsV6 BusTools_TimeGetFmtString BusTools_API_Close

example_bc_msg_run.c	This example program shows how to program the Bus Controller to run a simple three message minor-frame. The example initializes the channel the Bus Controller is running on. Then sets the channel for external bus. Following those initialization step it initializes the Bus Controller and builds a simple 3-message frame. The message in the frame are linked together and the frame will run continually until stopped by the user.	BusTools_API_OpenChannel BusTools_SetInternalBus BusTools_BC_Init BusTools_BC_MessageAlloc BusTools_BC_MessageWrite BusTools_BC_StartStop BusTools_API_Close
example_bc_msg_sched.c	This application shows how to setup the bus Controller for Message Scheduling. That is a technique that allows the user to schedule message at differing rates in the major frame. In this example, messages going out on every frame, every other frame, and every fifth frame. The frame rate is set at 1 Hz, so the message traffic can be visually seen on an analyzer. When using Message Scheduling the user programs the start-frame and the repeat rate for each message. A message with a start frame of one and repeat rate of one goes in every frame. A start frame of 1 and repeat rate of 2 causes the message to transact in every other frame. By setting the base frame rate and the start frame and repeat rate for each message the user can control the rate at which the messages in the major frame transact. You can vary frame rate and the start-frame and repeat rates of the messages in this example and see how the changes affect message traffic.	BusTools_API_OpenChannel BusTools_GetFWRevision BusTools_SetInternalBus BusTools_RT_Init BusTools_RT_CbufWrite BusTools_RT_MessageWrite BusTools_RT_AbufWrite BusTools_BC_Init BusTools_BC_MessageAlloc BusTools_BC_MessageWrite BusTools_BC_MessageRead BusTools_TimeGetString BusTools_RegisterFunction BusTools_BC_StartStop BusTools_RT_StartStop BusTools_API_Close
example_bc_multi_buffer.c	This example shows how to initialize the Bus Controller with multiple data buffers. This example requires Firmware version 6.0 or greater and BusTools/1553-API version 8.0 or greater. Previous F/W and API versions only support one or two data buffers. When using the Multiple BC buffer option you can create a varying number of data buffers for each BC message. This function initializes the board and steps through the process of enabling, creating, filling, and processing the multiple buffers. Three messages are created with 10, 15, and 5 data buffers. An interrupt callback function is registered using BusTools_RegisterFunction. In that callback the data from the buffer is optionally printed and the receive command (RT4) data is incremented to show how to update the data.	BusTools_API_OpenChannel BusTools_SetInternalBus BusTools_TimeTagMode BusTools_BC_Init BusTools_BC_MessageBlockAlloc BusTools_BC_MessageWrite BusTools_BC_DataBufferWrite BusTools_BC_MessageBufferRead BusTools_BC_DataBufferUpdate BusTools_BC_ReadDataBuffer BusTools_TimeGetString BusTools_RegisterFunction BusTools_BC_StartStop BusTools_API_Close

example_bc_noop.c	This example shows how to NOOP and un-NOOP a message. When a message is NOOPed, It does not transact. The firmware skips over the message as if it were not in the bus list. A single minor frame is created with three messages in the frame. The Second message in the bus list (message 1) is created as a Noop message (BC_CONTROL_MSG_NOP). That means it created as a BC message buffer but set in the NOOP state. When the frame runs only message 0 and 2 transact. The user can toggle message 1 ON or OFF by entering 'U' or 'N" to Un-NOOP or NOOP the message. The function BusTools_BC_MessageNoop is called change the NOOP setting for message 1. You can use BusTools_BC_MessageNoop on any BC message created with BC_CONTORL_MESSAGE or BC_CONTROL_MSG_NOP.	BusTools_API_OpenChannel BusTools_SetInternalBus BusTools_TimeTagMode BusTools_BC_Init BusTools_BC_MessageBlockAlloc BusTools_BC_MessageWrite BusTools_BC_MessageNoop BusTools_BC_StartStop BusTools_RegisterFunction BusTools_BC_MessageRead BusTools_BoardIsV6 BusTools_TimeGetFmtString BusTools_API_Close
example_bc_options.c	This is anThis is an example program that uses several Bus Controller options to configure a bus list. The options used are Frame-start timing, Message Scheduling, retries and Interrupts. There are two callback functions used. One is for the specific BC message in the bus list excluding the sync mode code. The other callback process the sync mode code. Retries on no-response or busy are enable on several of the messages. Frame start timing use the gap time as the message delay time from the start of the frame. Message scheduling is used to setup the message in the different frames. Some messages transact at 20Hz, some at 10Hz some at 2 Hz and the sync mode code runs at 1Hz. example program that uses several Bus Controller options to configure a bus list. The options used are Frame-start timing, Message Scheduling, retries and Interrupts. There are two callback functions used. One is for the specific BC message in the bus list excluding the sync mode code. The other callback process the sync mode code. Retries on no-response or busy are enable on several of the messages. Frame start timing use the gap time as the message delay time from the start of the frame. Message scheduling is used to setup the message in the different frames. Some messages transact at 20Hz, some at 10Hz some at 2 Hz and the sync mode code runs at 1Hz.	BusTools_API_OpenChannel BusTools_GetFWRevision BusTools_SetInternalBus BusTools_TimeTagMode BusTools_BC_Init BusTools_BC_MessageAlloc BusTools_BC_MessageWrite BusTools_BC_RetryInit BusTools_RegisterFunction BusTools_BC_StartStop BusTools_BC_MessageRead BusTools_BoardIsV6 BusTools_TimeGetFmtString BusTools_API_Close
example_bc_retry.c	This example demonstrates enabling retries on Bus Controller messages. Enabling retries lets the hardware automatically resend a message if the retry condition occurs. In the example the Bus Controller automatically resends the message if a no-response is detected on a retry enabled message or if the RT responds with Busy or Message Error. In order to enable retries you need to configure the Bus Controller to retry by setting the retry condition(s) in BusTools_BC_Init. Once a condition is programmed, each Bus Controller message can then be set to retry. In this example retries are set for message 1 and 2 (0-based) in the bus list. Furthermore, up to eight retries can be programmed by calling BusTools_BC_RetryInit. That function allows programming up to eight retries on either the same or alternate bus. The same and alternate bus designations are from the initial bus setting of the message. For example, if the message transacts Bus A the alternate bus is Bus B. In addition to setting up retries the example also sets up a callback function on a retry.	BusTools_API_OpenChannel BusTools_GetFWRevision BusTools_SetInternalBus BusTools_BC_Init BusTools_BC_MessageAlloc BusTools_BC_MessageWrite BusTools_BC_RetryInit BusTools_BC_StartStop BusTools_RegisterFunction BusTools_BC_MessageRead BusTools_BoardIsV6 BusTools_TimeGetFmtString BusTools_API_Close

example_bc_rt_bm.c	This console application shows how to configure the Remote Terminal, Bus Controller and Bus Monitor and set up interrupt on BC, RT, BM messages. The RT and BC callback function change the data for Transmit and Receive commands. The Bus Monitor callback displays the data.	BusTools_API_OpenChannel BusTools_SetInternalBus BusTools_BM_Init BusTools_BM_MessageAlloc BusTools_RT_Init BusTools_RT_CbufWrite BusTools_RT_MessageWrite BusTools_RT_AbufWrite BusTools_BC_Init BusTools_BC_MessageAlloc BusTools_BC_MessageWrite BusTools_BM_StartStop BusTools_RT_StartStop BusTools_BC_StartStop BusTools_RT_MessageRead BusTools_BM_MessageRead BusTools_BC_MessageReadData BusTools_BC_MessageUpdate BusTools_BC_ControlWordUpdate BusTools_BC_MessageUpdateBuffer BusTools_RegisterFunction BusTools_TimeGetString BusTools_DumpMemory BusTools_ReadBoardTemp BusTools_API_Close
example_bc_rt_broadcast.c	This example shows how to setup broadcast for BC and RT. It also show the Remote Terminal processes Broadcast messages.	BusTools_API_OpenChannel BusTools_SetInternalBus BusTools_SetBroadcast BusTools_RT_Init BusTools_RT_CbufWrite BusTools_RT_CbufbroadWrite BusTools_RT_MessageWrite BusTools_RT_AbufWrite BusTools_BC_Init BusTools_BC_MessageAlloc BusTools_BC_MessageWrite BusTools_RT_StartStop BusTools_BC_StartStop BusTools_RegisterFunction BusTools_RT_MessageRead BusTools_API_Close

example_bc_start_frame.c	This example application shows how to configure the Bus Controller for an initial frame that runs once then run a periodic frame. The application uses BusTools_BC_Start to start the initial frame at message 40. After that the frame starting at message zero (0) runs. This example also shows how to set an interrupt on minor-frame-overflow. A minor-frame-overflow occurs when the messages in a minor-frame take longer than the programmed frame-time to transact. When this occurs, messages exceeding the frame-time are suppressed and the new frame starts. In this example the frame rate is set for 1Hz (1000000). All the messages transact. If you change the frame rate to 1000 Hz (1000), a minor-frame overflow occurs and the last two messages in the frame are suppressed. The minor-frame-overflow interrupt callback increments a counter each time it runs. the number of overflow events are printed out at the end of this example.	BusTools_API_OpenChannel BusTools_SetInternalBus BusTools_BC_Init BusTools_BC_MessageAlloc BusTools_BC_MessageWrite BusTools_BC_Start BusTools_RegisterFunction BusTools_BC_MessageBufferRead BusTools_TimeGetString BusTools_BC_DataBufferUpdate BusTools_API_Close
example_bc_trigger_oneshot.c	This example program demonstrates how to start the Bus Controller using a trigger input. The example sets up a simple BC message list with 1-R-1-32 and 2-T-2-32, in a 500ms minor frame. This BC list runs until stopped by user input. Data is automatically displayed for 1-R-1-32 and 2-T-2-32. The user hits Enter to quit, shutdown the application and exit. BusTools_BC_Trigger is used to setup a 1shot trigger that starts the Bus Controller running. After calling BusTools_BC_StartStop the BC waits for an external trigger input before running. The trigger can be from an external trigger source or generated by the board. If the trigger source in the board you must wrap discrete 7 and 8 together. Note: not all boards have discrete channels or are configure with the discrete 7 and 8. You will need to check the configuration of the board installed to make sure that discrete channels are available.	BusTools_API_OpenChannel BusTools_SetInternalBus BusTools_BC_Init BusTools_BC_MessageAlloc BusTools_BC_MessageWrite BusTools_BC_Trigger BusTools_RegisterFunction BusTools_BoardIsV6 BusTools_SetV6TrigOut BusTools_SetV6TrigIn BusTools_DiscreteWrite BusTools_DiscreteSetIO BusTools_DiscreteTriggerOut BusTools_DiscreteTriggerIn BusTools_BC_StartStop BusTools_BC_MessageRead BusTools_TimeGetFmtString BusTools_API_Close

example_bc_trigger_user.c	This example application shows BC Triggering using the BC_TRIGGER_USER option. In this option the user controls how the BC frames run by adding BC_CONTROL_LAST to stop the Bus Controller. Each time you stop the Bus Controller using BC_CONTROL_LAST a trigger input is needed to restart. You can have one or more frames transacting off each trigger. This example creates six frames. The first four frames each start on a trigger input. The last two frames combined. The fifth frame starts on a trigger. The sixth frame runs after the normal frame delay, then stop when the BC_CONTROL_LAST buffer transacts. The list loops back to the first frame requiring a trigger to start. The application provides an option for internally generated triggers or an external user-supplied trigger. If you select the internal trigger, you will need to connect discrete7 and 8 together. Not all boards have these discrete channels available so check the board's configuration.	BusTools_API_OpenChannel BusTools_SetInternalBus BusTools_TimeTagMode BusTools_SetV6TrigOut BusTools_SetV6TrigOut BusTools_DiscreteWrite BusTools_DiscreteSetIO BusTools_DiscreteTriggerOut BusTools_DiscreteTriggerIn BusTools_BC_Init BusTools_BC_Trigger BusTools_BC_MessageAlloc BusTools_BC_MessageWrite BusTools_BC_StartStop BusTools_ExtTriggerOut BusTools_DumpMemory BusTools_BC_StartStop BusTools_RegisterFunction BusTools_BC_MessageRead BusTools_BoardIsV6 BusTools_TimeGetFmtString BusTools_API_Close
example_bit.c	This example program that initializes a channel and runs the Internal Built-In-Test and the Cable Wrap Test, then exits.	BusTools_API_OpenChannel BusTools_BIT_InternalBit BusTools_BIT_CableWrap BusTools_API_Close
example_bm_filter_messages.c	This example program sets up a Bus Monitor to record the message traffic to a file. The user is prompted for the number of messages to capture. The program then captures those messages and writes them to a file. This is a version of example_bm_recorder that demonstrates message filtering. This program only captures messages for RT1 SA2 TX. This is a monitor only example. It needs something to generating bus traffic that includes a RT1 SA2 TX message to capture data.	BusTools_API_OpenChannel BusTools_SetInternalBus BusTools_BM_Init BusTools_GetBoardType BusTools_BM_FilterWrite BusTools_BM_MessageAlloc BusTools_BM_StartStop BusTools_BM_MessageReadBlock BusTools_RegisterFunction BusTools_API_Close

example_bm_process_messages.c	This example program sets up a Bus Monitor to process each message transacted on the 1553 bus. The example shows how to use BusTools_RegisterFunction to setup a user-callback function to processes individual Bus Monitor Messages. It also shows how to write a user-callback that processes each Bus Monitor message. In the callback function the Bus Monitor message data is displayed on the console. This is a monitor only example; you will need to connect to a 1553 bus with traffic to capture messages.	BusTools_API_OpenChannel BusTools_BM_Init BusTools_SetInternalBus BusTools_GetBoardType BusTools_BM_MessageAlloc BusTools_BM_StartStop BusTools_RegisterFunction BusTools_BM_MessageRead BusTools_API_Close
example_bm_process_msg_blk.c	This example sets up a Bus Monitor to process each Bus Monitor message transacted. This example also shows how to process the messages using the function BusTools_BM_ReadLastMessageBlock. This function parses the interrupt queue to determine the BM messages that have transacted since the last call. For a BM message to record the BM initialization must define an interrupt condition like BT1553_INT_END_OF_MESS. This processing differs from using BusTools_RegisterFunction in that your application must provide a timing loop to call BusTools_BM_ReadLastMessageBlock periodically. The timing must be set so the BM messages (and other messages that may be recorded in the interrupt queue do not overwrite unprocessed entries. This example uses a 50 millisecond delay. This application use the Windows function kbhit to break out of timing loop. If using this example on non-Windows systems, you will need to provide a kbhit function.	BusTools_API_OpenChannel BusTools_BM_Init BusTools_SetInternalBus BusTools_GetBoardType BusTools_BM_MessageAlloc BusTools_BM_StartStop BusTools_BM_ReadLastMessageBlock BusTools_API_Close
example_bm_recorder.c	This example program sets up a Bus Monitor to record the message traffic to a file. The user is prompted for the number of messages to capture. The program then captures those messages and writes them to a file. Since this is a monitor only example, you will need something else generating bus traffic so there will be messages to capture.	BusTools_API_OpenChannel BusTools_BM_Init BusTools_SetInternalBus BusTools_GetBoardType BusTools_BM_MessageAlloc BusTools_BM_MessageReadBlock BusTools_BM_StartStop BusTools_RegisterFunction BusTools_API_Close

example_bm_share.c	This application demonstrates how to initialize a channel and the share the channel so other applications can use that channel. This application initializes and shares the channel. It then configures and runs a Bus Monitor. This allows separate Bus Controller and Remote Terminal applications to join this channel.	BusTools_API_OpenChannel BusTools_API_ShareChannel BusTools_SetInternalBus BusTools_BM_Init BusTools_BM_MessageAlloc BusTools_RegisterFunction BusTools_BM_StartStop BusTools_BM_MessageRead BusTools_API_QuitChannel BusTools_API_Close
example_bm_trig_start_stop.c	This example shows how to enable Bus Monitor start trigger and stop trigger using the function BusTools_BM_TriggerWrite. A start trigger specifies a condition or set of conditions that must occur before the Bus Monitor starts processing data. The stop trigger defines a condition of set of conditions that must occur to stop the Bus Monitor processing. When a start trigger is set it prevent Bus Monitor data from being recorded in the interrupt queue. A stop trigger disables the Bus Monitor message from going into the interrupt queue. You can still record all the message traffic. In this example the Bus Monitor is configured to start only after a command is sent to RT1. All traffic transacting prior is not processed. Once started the Monitor stops processing messages when a command word with RT22 occurs when armed first by a command word to RT2.	BusTools_API_OpenChannel BusTools_SetInternalBus BusTools_BM_Init BusTools_BM_MessageAlloc BusTools_BM_TriggerWrite BusTools_RT_Init BusTools_RT_CbufWrite BusTools_RT_CbufWrite BusTools_RT_AbufWrite BusTools_BC_Init BusTools_BC_MessageAlloc BusTools_BC_MessageWrite BusTools_RegisterFunction BusTools_BM_MessageRead BusTools_TimeGetFmtString BusTools_BC_StartStop BusTools_RT_StartStop BusTools_BM_StartStop BusTools_RegisterFunction BusTools_API_Close

example_bm_trgout.c	This example shows how to setup the Bus Monitor to generate an external trigger out when it records a specific message. In this example the Bus Monitor and optionally the BC and RT are configured. The Bus Monitor will generate an external trigger (trigger-out on discrete 7) when it records a BC message to RT 8. This example also routes the output trigger back to the input trigger (set up on discrete 8) and sets up an interrupt on external trigger. This setup generates an output trigger and captures it as an input trigger for display. In order for this set up to work you need to physically connect the discrete7 to discrete 8. Discrete and triggers vary between boards and this setup may not run on your board variant. Please refer to the "MIL-STD-1553 Hardware Installation Guide" for details about triggers and discrete channels available on the different 1553 products.	BusTools_API_OpenChannel BusTools_SetInternalBus BusTools_ExtTrigIntEnable BusTools_BM_Init BusTools_BM_MessageAlloc BusTools_BM_TriggerWrite BusTools_SetV6TrigOut BusTools_SetV6TrigIn BusTools_DiscreteWrite BusTools_DiscreteSetIO BusTools_DiscreteTriggerOut BusTools_DiscreteTriggerIn BusTools_RT_Init BusTools_RT_CbufWrite BusTools_RT_MessageWrite BusTools_RT_AbufWrite BusTools_BC_Init BusTools_BC_MessageAlloc BusTools_BC_MessageWrite BusTools_BC_StartStop BusTools_RT_StartStop BusTools_BM_StartStop BusTools_DumpMemory BusTools_RegisterFunction BusTools_API_Close
example_dbca.c	This application shows how to setup Dynamic Bus Controller allocation for both the Remote Terminal and the Bus Controller. This example uses two channels, one as the initial Bus Controller and the other as the initial Remote Terminal. The RT channel also configures a Bus Controller, but it is not started. In this example the active BC is initialized to send a bus list using RTs 2, 4 and 6. There is also an aperiodic message to send a mode code 0 to RT1. If RT1 accepts the DBCA by setting the DBA bit in the Status word, the Bus Controller halts. The inactive BC, configured by the RT channel, has a bus list to RTs 12, 14, and 16. RT1 is setup to accept the DBCA Mode Code 0. It automatically starts the Bus Controller. User input from the console sends the aperiodic Mode Code 0 to RT1. Once that transacts, the initial BC is halted and the RT side BC takes over.	BusTools_API_OpenChannel BusTools_SetInternalBus BusTools_RT_Init BusTools_RT_CbufWrite BusTools_RT_MessageWrite BusTools_RT_AbufWrite BusTools_BC_Init BusTools_BC_MessageAlloc BusTools_BC_MessageWrite BusTools_RT_StartStop BusTools_BC_StartStop BusTools_BC_AperiodicRun BusTools_RT_MessageRead BusTools_RegisterFunction BusTools_API_Close

example_ext_trig.c	This application shows how to setup interrupts on external trigger. In addition it shows how to setup discrete for input and output triggers. This test requires a wrap connector between discrete 7 and discrete 8 on the D50 connector. The test generates a output trigger and wraps output on discrete 7 to the input on discrete 8. The input trigger generates an interrupt.	BusTools_API_OpenChannel BusTools_SetInternalBus BusTools_SetVoltage BusTools_BM_Init BusTools_BM_MessageAlloc BusTools_BM_StartStop BusTools_SetV6TrigOut BusTools_SetV6TrigIn BusTools_DiscreteWrite BusTools_DiscreteSetIO BusTools_DiscreteTriggerOut BusTools_DiscreteTriggerIn BusTools_ExtTrigIntEnable BusTools_RegisterFunction BusTools_BM_StartStop BusTools_ExtTriggerOut BusTools_TimeTagRead BusTools_API_Close
example_irig1.c	This test program demonstrates the setup of the IRIG-B output and input. This program just configures IRIG for EXTERNAL or INTERNAL source, sets the IRIG the current time/date and then loops 20 times reading and displaying the time every second. This REQUIRES a board with the IRIG option. The part number should include a 'W', which indicates IRIG. For example: QPCX-1553-4MW. Use BusTools_BoardHasIRIG to find out if you have an IRIG enabled board.	BusTools_API_OpenChannel BusTools_BoardHasIRIG BusTools_BM_Init BusTools_IRIG_Config BusTools_IRIG_Calibration BusTools_IRIG_Valid BusTools_IRIG_SetTime BusTools_TimeTagMode BusTools_TimeTagRead BusTools_API_Close

example_rev.c	This example program initializes a board and displays version information and general board information.	BusTools_GetDevInfo BusTools_StatusGetString BusTools_API_OpenChannel BusTools_GetBoardType BusTools_ReadBoardTemp BusTools_GetRevision BusTools_GetFWRevision BusTools_BoardIsV6 BusTools_BoardIsMultiFunction BusTools_GetCSCRegs BusTools_GetChannelCount BusTools_BoardHasIRIG BusTools_GetSerialNumber BusTools_MemoryAvailable BusTools_API_Close
example_rt_auto_wrap.c	This application shows how to automatically wrap RT receive and transmit messages buffers. When transmit and receive message buffers are wrapped the RT transmits the data from the previous receive command.	BusTools_API_OpenChannel BusTools_SetInternalBus BusTools_SetVoltage BusTools_BC_Init BusTools_BC_MessageAlloc BusTools_BC_MessageWrite BusTools_RT_Init BusTools_RT_CbufWrite BusTools_RT_MessageWrite BusTools_RT_AbufWrite BusTools_RegisterFunction BusTools_BC_MessageRead BusTools_RT_StartStop BusTools_BC_StartStop BusTools_API_Close

example_rt_buffer_switch.c	The example demonstrates how to setup the Remote Terminal to manually switch between two RT message buffers. The RT is initialized with two message buffer per RT/SA/TX/RX combination. Then, using BusTools_RT_MessageGetaddr and BusTools_MemoryWrite2 the RT message buffer linked list is altered to have each buffer point to its address instead of the next buffer in the list. Instead of looping through the message buffer the RT uses only a single buffer. The user must write the address of the message they want to run into the RT control Buffer. This example sets up a RT3 SA3 TX with 2 buffers. Then manually changes the message buffer linking so only a single buffer is used. By entering 0 or 1 at the command prompt the user can switch between buffer 0 (data = 0x1111) and buffer 1 (data = 0x2222) This example shows this process for both the V6 and V4/5 firmware designs. The high level API hides the underlying design differences in the firmware. When directly accessing memory the user needs to understand the memory layout for the firmware they are using. Refer to the "MIL-STD-1553 Universal Core Architecture Reference Manual" for the V4/5 firmware and the "MIL-STD-1553 Enhanced Universal Core Architecture (UCA32) Local Processing Unit (LPU) Reference Manual for V6 firmware. Also keep in mind that V6 firmware uses 32-bit addressing while V4/5 uses 16-bit addressing. When running this example it you a dump the board's memory by typing 'd' or 'D'. Reviewing the memory dump will help follow the memory manipulations in this example.	BusTools_API_OpenChannel BusTools_SetInternalBus BusTools_BC_Init BusTools_BC_MessageAlloc BusTools_BC_MessageWrite BusTools_RT_Init BusTools_RT_CbufWrite BusTools_RT_MessageWrite BusTools_RT_AbufWrite BusTools_RT_MessageGetaddr BusTools_BoardIsV6 BusTools_RamAddr BusTools_RelAddr BusTools_MemoryWrite2 BusTools_GetAddr BusTools_MemoryRead2 BusTools_RT_MessageRead BusTools_RegisterFunction BusTools_RT_StartStop BusTools_BC_StartStop BusTools_DumpMemory BusTools_API_Close
example_rt_ei_late_rsp.c	This example program sets up RT1 with two subaddresses, SA1 RECEIVE and SA2 TRANSMIT. It also demonstrates how to inject errors into an RT message. In this case, we inject a LATE RESPONSE ERROR. The user can specify the response time from 7-31us. This can be done for SA1 RECEIVE or SA2 TRANSMIT.	BusTools_API_OpenChannel BusTools_SetInternalBus BusTools_RT_Init BusTools_RT_AbufWrite BusTools_RT_CbufWrite BusTools_EI_EbufWriteENH BusTools_RT_MessageWrite BusTools_RT_StartStop BusTools_API_Close
example_rt_ei_parity.c	This example program sets up a simple RT1 with two subaddresses, SA1 RECEIVE and SA2 TRANSMIT. It also demonstrates RT error injection. In this case, it creates a PARITY ERROR. This can be done for SA1 RECEIVE (on STATUS word) or on SA2 TRANSMIT (on STATUS or DATA words).	BusTools_API_OpenChannel BusTools_SetInternalBus BusTools_RT_Init BusTools_RT_AbufWrite BusTools_RT_CbufWrite BusTools_EI_EbufWriteENH BusTools_RT_MessageWrite BusTools_RT_StartStop BusTools_API_Close

example_rt_extended_status.c	This example shows how to configure a Remote Terminal and enable Extended Status updates. Under normal 1553 operation the status returned by the RT is for all Sub-addresses and transmit and receive buffer. BusTools_RT_AbufWrite sets the status response for the RT and enable extended status for a RT. Using Extended Status mode the RT can set the status word for each Sub-address, Transmit, Receive and buffer for the RT. In this example the RTs are set up with two buffers per RT/SA/TX/RX combination. For RT 3 Transmit and RT 4 receive the second buffer is set to respond with updated status. RT 3 responds with Busy (BSY) and RT 4 responds with Message Error (ME). This causes the status word for those two messages to toggle between the RT message status and the extended status programmed by BusTools_RT_MessageWriteStatusWord. To use Extended Status set the Extended Status enable in the 'inhibit terminal flag' parameter in call to BusTools_RT_AbufWrite.	BusTools_API_OpenChannel BusTools_SetInternalBus BusTools_RT_Init BusTools_RT_CbufWrite BusTools_RT_MessageWrite BusTools_RT_AbufWrite BusTools_RT_MessageWriteStatusWord BusTools_BC_Init BusTools_BC_MessageAlloc BusTools_BC_MessageWrite BusTools_RegisterFunction BusTools_RT_StartStop BusTools_BC_StartStop BusTools_BC_MessageRead BusTools_TimeGetFmtString BusTools_API_Close
example_rt_join.c	This example shows how an application joins an already initialized channel. This function demonstrates a simple RT application. RTs 1 - 8 are programmed and a callback function is setup to process RT message in the interrupt queue. This function requires that the channel joined already be initialize and shared by another application. The user can run this example with example_bm_share and example_bc_join. Joining a shared channel allows individual Remote Terminal, Bus Monitor and Bus Controller applications to run off a single channel.	BusTools_API_JoinChannel BusTools_RT_Init BusTools_RT_CbufWrite BusTools_RT_MessageWrite BusTools_RT_AbufWrite BusTools_RegisterFunction BusTools_TimeTagRead BusTools_RT_MessageRead BusTools_TimeGetString BusTools_RT_StartStop BusTools_API_QuitChannel

example_rt_mc17.c	Demonstrates how to process Mode Code 17, Sync with Data.	BusTools_API_OpenChannel BusTools_SetInternalBus BusTools_TimeTagMode BusTools_GetRevision BusTools_GetFWRevision BusTools_BC_Init BusTools_BC_MessageAlloc BusTools_BC_MessageWrite BusTools_RT_Init BusTools_RT_CbufWrite BusTools_RT_MessageWrite BusTools_RT_AbufWrite BusTools_RegisterFunction BusTools_BC_StartStop BusTools_RT_StartStop BusTools_DumpMemory BusTools_RT_MessageRead BusTools_BC_MessageRead BusTools_API_Close
example_rt_mode_code.c	This example shows multi-buffering of mode code data. Uses Mode Code 17, synchronize with data and Mode code 19, Transmit Bit Word. This example shows how to setup data for Mode codes if that are in the same bus list.	BusTools_API_OpenChannel BusTools_SetInternalBus BusTools_TimeTagMode BusTools_RT_Init BusTools_RT_CbufWrite BusTools_RT_MessageWrite BusTools_RT_AbufWrite BusTools_BC_Init BusTools_BC_MessageAlloc BusTools_BC_MessageWrite BusTools_RegisterFunction BusTools_RT_MessageRead BusTools_BC_MessageRead BusTools_RT_StartStop BusTools_BC_StartStop BusTools_DumpMemory BusTools_API_Close

example_rt_monitor.c	This example shows how to setup the Remote Terminal in monitor only mode. In this mode the RT capture all messages to the specified RT but will not respond to the RT messages. With the exception of calling BusTools_RT_MonitorInit the Remote terminal is configured identically to a real Remote Terminal.	BusTools_API_OpenChannel BusTools_SetInternalBus BusTools_TimeTagMode BusTools_RT_Init BusTools_RT_CbufWrite BusTools_RT_MessageWrite BusTools_RT_AbufWrite BusTools_RT_MonitorEnable BusTools_RegisterFunction BusTools_RT_StartStop BusTools_RT_MessageRead BusTools_RT_StartStop BusTools_API_Close
example_rt_set_status.c	This example program sets up RT1 with two subaddresses, SA1 RECEIVE and SA2 TRANSMIT. This example program also demonstrates: - DISABLING and ENABLING an RT to respond on the 1553 Bus - toggling of the BUSY, SERVICE REQUEST, and TERMINAL FLAG bits in the RT Status Word - ILLEGALIZING commands to an RTSA An external BC device is required and configured to send a BC->RT command to RT1 SA1 and an RT->BC command to RT1 SA2, at a minimum. Recommended BC setup: - BC->RT Command RT1, SA1, RECEIVE, 32 Data Words, Bus A - RT->BC Command RT1, SA1, TRANSMIT, 32 Data Words, Bus A - BC->RT Command RT1, SA1, RECEIVE, 32 Data Words, Bus B - RT->BC Command RT1, SA1, TRANSMIT, 32 Data Words, Bus B After DISABLING the RT, verify that the RT does not respond with Status to the BC Command on Bus A and Bus B. After ENABLING the RT, verify that the RT responds with Status to the BC Command on Bus A and Bus B. Verify the RT Status Word Response using the BC device as the BUSY, SERVICE REQUEST, and TERMINAL FLAG bits are set and cleared in the RT Status Word. The BC List should execute at a rate to allow for monitoring of the changes to the RT Status Word. When ILLEGALIZING commands to RT1-SA1-RX and RT1-SA2-TX, the RT will respond with the MESSAGE ERROR bit set in the status word. And, for the TRANSMIT command, the RT will not send any data words. Note that alternative coding options are included for setting and clearing the RT Status Word bits.	BusTools_API_OpenChannel BusTools_SetInternalBus BusTools_RT_Init BusTools_RT_AbufWrite BusTools_RT_CbufWrite BusTools_RT_MessageWrite BusTools_RT_StartStop BusTools_API_Close

example_rt_wrap.c	This console example program shows how to manually wrap RT receive and transmit messages buffers. This differ from the automatic way is that the example get the address of the RT buffer and manipulates the buffer address data.	BusTools_API_OpenChannel BusTools_SetInternalBus BusTools_BC_Init BusTools_BC_MessageAlloc BusTools_BC_MessageWrite BusTools_RT_Init BusTools_RT_CbufWrite BusTools_RT_MessageWrite BusTools_RT_AbufWrite BusTools_RT_MessageWrite BusTools_BoardIsV6 BusTools_RT_AbufWrite BusTools_GetAddr BusTools_MemoryRead2 BusTools_RelAddr BusTools_MemoryWrite2 BusTools_MemoryRead BusTools_MemoryWrite BusTools_RT_StartStop BusTools_BC_StartStop BusTools_RegisterFunction BusTools_BC_MessageRead BusTools_API_Close
example_timetag_read.c	This example set the time tag to either zero or the present time and data. Then it reads the time tag register every 500 milliseconds (.5 seconds) and prints the raw nanoseconds or microseconds depending the firmware version. It also converts the time to a string and prints the time string.	BusTools_ListDevices BusTools_API_OpenChannel BusTools_SetInternalBus BusTools_GetRevision BusTools_BoardIsV6 BusTools_TimeTagMode BusTools_RegisterFunction BusTools_TimeTagRead BusTools_API_Close