

Release Notes for USB2ANY Firmware

v2.8.2.0

General Release

(No changes in the firmware)

v2.8.0.0

General Release

(No changes in the firmware)

v2.7.0.17

1. Added support for SPI streaming (currently disabled by ENABLE_SPI_STREAMING=0).
2. Added support for alternate I2C port.
3. Added MSP430_BlockWrite and MSP430_BlockRead functions.
4. Added Command_MSP430_MemoryWrite and Command_MSP430_MemoryRead commands.
5. Added OneWire mode 5 (pulse with address).
6. Fixed bug in Port_Write function. Port is now updated in one operation instead of two.
7. Added Port_WritePulse function.
8. Added Command_Port_WritePulse command.
9. Added mode to keep MOSI high between transactions (SPI interface).
10. Added support for alternate CS in SPI interface (see MULTI_CS).
11. Added SPI_WriteAndReadEx function that allows CS to be specified.
12. Added support for UART streaming.

v2.7.0.7

13. Enable the SPI_WriteAndReadEx() function, which is essentially the same as the SPI_WriteAndRead() function, except that it has an additional parameter that allows the user to specify a GPIO to be used as the SS signal.

v2.7.0.6

14. Added 1K global memory buffer for support of memory commands
15. Modified MSP430_WordWrite, MSP430_WordRead, MSP430_ByteWrite, and MSP430_ByteRead to use the new global memory buffer.
16. Added support for Command_MSP430_MemoryWrite and Command_MSP430_MemoryRead.
17. Added command codes for Command_Stream_Execute, Command_SPI_StreamOut, and Command_SPI_StreamStop.
18. Added command code for Command_Port_WritePulse.

19. Added globals g_StreamingSPI and g_StreamPreExecute.
20. Added code to support MSP430_BlockWrite and MSP430_BlockRead.
21. Modified Port_Write to modify the port in a single operation, instead of two.
22. Added code to support the Port_WritePulse command.
23. Modified SPI code to allow multiple chip selects.
24. Added code to allow streaming of SPI commands.
25. Added global g_StreamPreExecute flag to control whether commands are pre-executed.
26. Enhanced streaming functions to allow the use of preset data when the STREAM_USE_PRESET_DATA flag is set.
27. Moved all of the main loop code into a single function (mainLoop).

v2.7.0.0

v2.6.4.0

28. Added GPIO_WritePulse command support
29. Added I2C_BlkWriteBlkRead command support
30. Added EasyScale_WriteAndRead command support
31. Added I2C_BlkWriteBlkRead command support
32. Added OneWire_SetMode command support
33. Added UART_SetMode command support
34. Added GPIO_WritePulse command support
35. Added Pegasus_Test command support
36. Added UART_SetMode command support
37. Added DisplayScale_Set command support
38. Added DisplayScale_WriteReg command support
39. Added DisplayScale_ReadReg command support
40. Added DisplayScale_WriteAndRead command support
41. Added streaming mode for I2C_Write
42. Added streaming mode for I2C_Read
43. Added streaming mode for I2C_ReadWithAddress
44. Added streaming mode for I2C_ReadInternal
45. Added streaming mode for I2C_WriteInternal
46. Added streaming mode for I2C_BlkWriteBlkRead
47. Maximum buffer size for ADC changed from 4096 bytes to 3072 bytes
48. Changed I2C_WriteInternal() to return number of bytes written
49. Fixed problem with placement of NACK interrupt enable/disable code (I2C)
50. Improved OneWire functions
51. Modified UART_Write() to not return until all data is sent
52. Improved UART timeout calculation algorithm.
53. Modified UART_Write to implement Special Mode 2
54. Added error codes ERR_DATA_CRC_FAILED, ERR_INVALID_PARAMETER, and ERR_NOT_INITIALIZED
55. Digital Capture is temporarily disabled
56. Improved check for OneDemo to prevent false detections

- 57. Improvements to FatalError(): now displays an LED blink code to indicate fatal error
- 58. Improvements to watchdog timer: calls FatalError() on watchdog timeout
- 59. Added DisplayScale protocol
- 60. EasyScale_WriteAndRead() now returns the number of bits read on success
- 61. Fixed bug in EasyScale_WriteAndRead() when partial bytes are read
- 62. Fixed EasyScale_WriteAndRead() bug that prevented some bits from being sent
- 63. Fixed EasyScale_WriteAndRead() timing problems
- 64. Maximum speed for EasyScale_WriteAndRead() is now 100kHz
- 65. Improved speed and efficiency of EasyScale write and read functions

v2.6.3.0

- 66. Improved UART timeout calculation algorithm.
- 67. Added UART_SetMode function.
- 68. Modified UART_Write to implement Mode 2 (RxAAfterTx).
- 69. Fixed bug that caused the Payload Sequence Number to be incremented prematurely.

v2.6.2.25 (beta)

- 70. Improved check for OneDemo to prevent false detections
- 71. Removed code in LocalI2C_Write that sometimes called bit-banged version
- 72. Fixed EasyScale_WriteAndRead() bug that prevented some bits from being sent
- 73. Maximum speed for EasyScale_WriteAndRead() is now 100kHz
- 74. Fixed EasyScale_WriteAndRead() timing problems
- 75. EasyScale_WriteAndRead() now returns the number of bits read on success
- 76. Fixed bug in EasyScale_WriteAndRead() when partial bytes are read
- 77. Improved state machine in TimerB0_ISR
- 78. Increased EASYSCALE_BITBUF_SIZE for new code

v2.6.2.20 (beta)

- 1. Added EasyScale_WriteAndRead().
- 2. Removed some debug code.
- 3. Added I2C_BlkWriteBlkRead()
- 4. Major changes to OneWire interface (touched nearly every line of code)
- 5. Modified UART_Write() to not return until all data is sent.
- 6. Added case for Command_OneWire_SetMode
- 7. Added prototype for DigitalCapture()
- 8. Added prototype for OneWire_Command()

v2.6.2.9 (beta)

1. Fixed bug that caused `hid_sendDataInBackground()` to usually return the wrong value
2. Added code to WDT ISR to monitor stack for overflows
3. Fixed bug in `FatalError` that cause the LED to blink way too fast
4. Increased stack size from 160 bytes to 512 bytes
5. Added code to allow the WDT ISR to be interrupted.
6. Added `Command_I2C_BlkWriteBlkRead` stub
7. Added `ERR_NOT_ENABLED` (-39)
8. Added `GPIO_SET_BOOL` macro and `SET_GPIOxx` macros
9. Added OneWire interface
10. Added support for OneWire Modes 1, 2, and 3
11. Added switch case for `Command_I2C_BlkWriteBlkRead`
12. Fixed bug in `LED_Toggle()` that prevented it from working with USB2ANY
13. Fixed `EasyScale` bug that caused incorrect timing in "extra" bits
14. Fixed bugs in `hid_sendDataInBackground()` that caused an endless loop.
15. Improvements to memory initialization (stack, heap, etc.)
16. Removed obsolete code that checks for "prototype Rev A OneDemo"
17. Several UART-related improvements and bug fixes

v2.6.2.0

1. Added code to initialize the stack (primarily for debugging)
2. Added code to call `SendADCDData()` and `SendDigitalData()` from main loop
3. Added `FatalError()` -- a place to go when mortally wounded
4. Added call to `LED_Blinker()` from watchdog timer interrupt
5. Major re-structuring of ADC code
6. Added conditional code for `BUILD_BP5529`
7. Optimized building of ADC payload
8. FEC functions were essentially rewritten
9. Fixed bug that caused `Command_GPIO_Set` to call an incorrect function
10. Some restructuring of GPIO code for OneDemo build
11. Added `CheckAddress()` function to validate I2C addresses
12. Added `I2C_Init()`
13. Added `needReset` flag to indicate when `I2C_Init()` needs to be called
14. Added code to support LED blink states
15. Added address validation in `MSP430_WordWrite()` and `MSP430_WordRead()`
16. Added address validation in `MSP430_ByteWrite()` and `MSP430_ByteRead()`
17. Minor logic changes to `Firmware_VersionRead()`
18. Changed code in `Power_Set()` to use named constants
19. `SMBUS_Control` did not set error code on failure
20. `SMBUS_Read` failed to read when PEC was enabled
21. All `SMBUS` functions are now emulated by `USB2ANY.DLL`
22. `SPI_Set` now validates all parameters
23. `SetErrorCode` now returns the error code
24. Changed `Command_GetErrorList` to *not* clear the error list.
25. Added handler for `Command_Digital_Capture`

26. Fixed bug that prevented Command_GetErrorList from replying
27. Call FatalError() if hid_waitForSendComplete() fails in hid_sendDataInBackground()

v2.6.0.5 (beta)

1. Added FirmwareDebugMode.
2. ADC: Added ADC_SendStatus() function to update ADC conversion status at the DLL level. This allows the status to be polled without incurring the round-trip USB communication delays.
3. ADC: Changed code to asynchronously send ADC data to the DLL when the conversions completes.
4. ADC: Deprecated the ADC_GetData() function. This function is now handled completely by the DLL.
5. EasyScale: Fixed several bugs and other issues.
6. GPIO: Fixed a bug that prevented GPIO12 from being initialized properly when configured as an output.
7. I2C: Added code to allow NAKs to be ignored. This mode is enabled by the Command_FirmwareDebugMode function (0x0100 to disable the feature, 0x101 to enable it).
8. I2C: Improved handling of 10-bit address mode.
9. I2C: Added code to always check slave addresses for validity (00-7F for 7-bit, 000-3FF for 10-bit).
10. I2C: Added code to prevent I2C_Write_Buffer from attempting to write zero bytes (caused problems).
11. Added Command_FirmwareDebugMode to allow special debug modes to be set.
12. SMBus: All SMBus commands have been deprecated. The SMBus functions are now emulated by the DLL using I2C commands.
13. SPI: Modified code to make the CS pulse longer and proportional to the bit rate.
14. SPI: Added SPI_Chip_Select_Type 3 to disable the CS signal.
15. RFFE: Added RFFE commands:
 - Command_RFFE_RegZeroWrite
 - Command_RFFE_RegWrite
 - Command_RFFE_ExtRegWrite
 - Command_RFFE_ExtRegWriteLong
 - Command_RFFE_RegRead
 - Command_RFFE_ExtRegRead
 - Command_RFFE_ExtRegReadLong

NOTE: The following changes are internal and not directly visible to end users.

16. Removed MSC driver code (cleanup).
17. Reduced size of general purpose buffer from 256 bytes to 64 bytes (max packet size).
18. Added code for CDC and BULK mode USB protocols (currently disabled).
19. Changed to initialize all I/O pins as "input with pull-down", instead of "output low".
20. Added hid_waitForSendComplete() function.
21. Added Command_InvokeBSL to put the USB2ANY into BSL mode.
22. Added Command_Restart to allow the USB2ANY to be restarted.
23. SendPacket() now uses a default timeout (currently 10000) when ulTimeout is zero.
24. Added SendPacketWait() function.
25. "Old-style" packets are still supported, but are now handled by DispatchV2Command() function.

KNOWN ISSUES:

26. I2C: There are known problems with 10-bit addresses, in general.

v2.5.1

1. Fixed problems related to GPIO6 and GPIO12. The problems affected all interfaces the used one or both of these signals, or shared the same pins as these signals (GPIO, PWM1, SPI, EFC0, and CLOCK).

v2.5.0

1. Fixed ADC bugs that prevented ADC_ConvertAndRead() from working with OneDemo.
2. Added ADC_AcquireAfterTrigger() function to enable delayed triggering of the ADC acquire.
3. Fixed minor bugs and improved error checking in EasyScale functions.
4. Removed code for FEC1, FEC2, and FEC3, since the board has only one FEC port (FEC0).
5. Changed FEC code to return results using an asynchronous packet.
6. Significant changes to all GPIO functions. They now execute much faster and use less code space.
7. Fixed I2C pullup problems with OneDemo.
8. I2C read and write commands now return an ERR_I2C_NO_PULLUP_POWER error code if pullups are enabled and 3.3V power is not enabled.
9. Fixed bugs in I2C read and write commands that caused various problems if the specified data size was too large. Now, they will return an ERR_TOO_MUCH_DATA error code.
10. Changed the maximum size for an internal address in I2C_ReadInternal and I2C_WriteInternal commands from 24-bits to 16-bits (i.e., from three bytes to two bytes). This may be revisited later but, for now, fixing the problems with 24-bit addresses cannot be justified, since no one seems to use them.
11. INT0 pin is no longer available on OneDemo, since it is used for EVM Detect.
12. Added error checking for parameters to Interrupt functions.
13. Command_LED_Set no longer returns ERR_INVALID_FUNCTION_CODE for OneDemo. It can now be use to turn the green LED on/off or toggle it.
14. Command_Power_Set now returns ERR_NO_EVM if no EVM is connected.
15. Command_Power_Set now checks for power faults after changing any power settings.
16. For the Adjustable Power Supply, we now check both the FAULT signal of the TPS2557 power switch and the PG (Power Good) signal of the TPS63020 regulator. If either signal indicates a problem, both devices are disabled to prevent undesirable operation (mainly to prevent the TPS63020 from oscillating).
17. When a power event occurs (i.e., power fault occurs or is cleared), an INTERRUPT_POWER packet is now sent asynchronously, to notify the USB2ANY driver.
18. Added Command_Power_Notify command to enable/disable asynchronous power event notification. It is disabled by default because the USB2ANY driver must setup a handler function before it is enabled.
19. Added additional baud rates for UART functions: 300, 320, 600, 1200, 2400, and 4800.
20. The UART functions are now fully functional.
21. Power status is now checked automatically once in every 256 passes of the main loop.

v2.0.8

1. The name of the FEC Interval enumerated constants has been changed from **FEC_Int** to **FEC_Interval** to prevent conflicts with other constants.
2. Added enumerated constants for **EasyScale_WriteSpeed**, **EasyScale_ReadSpeed**, **EasyScale_WriteACK**, **EasyScale_Status_Received**, **EasyScale_ReceivedDataStatus**, and **EasyScale_Threshold**.
3. Changed the name of a constant from **Command_EasyScale_Set** to **Cmd_EasyScale_Control**.
4. Changed the name of a constant from **Command_EasyScale_Write** to **Cmd_EasyScale_Write**.
5. Added constant **Cmd_EasyScale_Read**.
6. Fixed bug that could prevent **u2aFirmwareVersion_Read** from returning correct value.
7. **u2aGetStatusText** will now use an internal buffer if second parameter is NULL.
8. **u2aReadResponse** will now write received bytes to debug log.
9. **u2aI2C_RegisterWrite** was erroneously exported as **u2aI2C_Register_Write**.
10. **u2aI2C_RegisterRead** was erroneously exported as **u2aI2C_Register_Read**.
11. Fixed bug in **u2aI2C_RegisterRead** that prevented it from returning the correct value.
12. **u2aI2C_MultiRegisterWrite** was erroneously exported as **u2aI2C_Multi_Register_Write**.
13. **u2aI2C_MultiRegisterRead** was erroneously exported as **u2aI2C_Multi_Register_Read**.
14. Fixed bug in **u2aI2C_MultiRegisterRead** that prevented it from returning the correct value.

NOTE: The following changes are internal and not visible to end users.

15. Added class functions **FEC_Control**, **FEC_CountAndRead**, **Interrupt_Control**, **EasyScale_Control**, **EasyScale_Write**, and **EasyScale_Read**.
16. Modifications for FEC functions.
17. Added code for USB plug/unplug detection (WIP).
18. Fixed bug in **u2aI2C_InternalRead** that could cause wrong return value in **POLLED_MODE**.
19. Fixed bug in **u2aI2C_RawRead** that could cause wrong return value in **POLLED_MODE**.
20. Fixed bug in **u2aGPIO_ReadState** that could cause wrong return value in **POLLED_MODE**.
21. Fixed bug in **u2aADC_ConvertAndRead** that could cause wrong return value in **POLLED_MODE**.
22. Fixed bug in **u2aMSP430_WordRead** that could cause wrong return value in **POLLED_MODE**.
23. Fixed bug in **u2aMSP430_ByteRead** that could cause wrong return value in **POLLED_MODE**.

V1.16.0

Known Issues:

1. I2C transactions at 10kHz may have problems (e.g., missing STOP) when attempting to transfer more than 12 bytes of data. It is recommended that only very small packets (i.e., less than 10 bytes) are used when the bit rate is 10kHz. This problem will be addressed in a future release.