BufferLib Programmer's Reference Manual

Revision: December 15, 2011

1300 NE Henley Court, Suite 3
Pullman, WA 99163
(509) 334 6306 Voice | (509) 334 6300 Fax

Introduction

This document describes the programming interface to the BufferLib library that is included as part of the PmodLib library. It describes the capabilities of the BufferLib library and all the API functions used to access its features.

The purpose this library is to offer access to an N-way buffering system for the Cerebot 32MX4 or 32MX7 microcontroller. This allows for the smoothing of access to streamed data, such as audio or video.

BufferLib Basic API Functions

uint8_t BufLibInitBuffers(uint8_t BufCount, uint16_t BufSize)

Parameters

uint8_t BufCount - the number of buffers to create
uint16_t BufSize - the size of each buffer in bytes

Return Value

1 – Successfully allocated memory for all the buffers
0 – Unable to allocated memory to the buffers

This function initializes the number of buffers specified by BufCount, each of BufSize size. Allocation is done dynamically so you must have the necessary memory allocated to the heap.

NOTE: BUFLIB_MAX_ARRAY_MEM defines the maximum value of total memory that can be allocated to ALL buffers. Default value is set to 25000 bytes.

uint8_t BufLibReadBuffer(uint16_t *data)

Parameters

uint16_t *data – a memory location to store the value read from the current buffer

Returns

1 – Successfully read a 2-byte word to data
0 – Failed to read a 2-byte word to data

This function attempts to read a 2-byte word out of the current buffer and store it into the memory address passed in via 'data'. If successful, the function returns 1. Otherwise data remains unchanged and a 0 is returned.

uint8_t BufLibWriteBuffer(uint16_t data)

Parameters
uint16_t data – a 2-byte value to store into the current buffer

Returns
1 – Successfully wrote the 2-byte value to the current buffer
0 – Failed to write the 2-byte value to the current buffer

This function attempts to store the 2-byte value passed in via data into the current buffer. If successfully wrote into the buffer, the function returns a 1. Otherwise the buffer remains unchanged and a 0 is returned.

void BufLibResetBuffers(void)

Parameters
None

Returns
None

This function resets all the configuration values to match the initialization values for all buffers.

NOTE: This function must be called if you wish to write to the buffer again after calling BufLibFinishWrite()

void BufLibFinishWrite(void)

Parameters
None

Returns
None

This function frees the write lock on the last buffer written to. Typically this will be called once you have finished writing to the buffer and no more is needed until all data is read from the buffer.

NOTE: You must call BufLibResetBuffers() if you wish to write to the buffer again after calling this function.

BufferLib Additional Information

BufferLib Structs

struct AppBuffer

Fields

uint16_t space_used – tracks the available space in the given buffer
uint16_t *buffer – points to the memory allocated to a buffer
uint8_t read – set to lock the current buffer as read
uint8_t write – set to lock the current buffer as write
uint8_t offset – tracks the read offset in the buffer

Each buffer allocated is defined by the AppBuffer struct. The fields contained in the struct track the order in which each buffer is written/read to. It also contains a pointer to the memory of the actual buffer.

BufferLib Globals

All of these globals are initialized during the call to BufLibInitBuffers()

AppBuffer *gBufLibBufferArray

A pointer to the array of buffers allocated. The amount of buffers in the array is defined by BufCount in BufLibInitBuffers()

uint16_t gBufLibMaxSize

Contains the maximum size of all the buffers allocated

uint8_t gBufLibNumBuffers

Contains the number of buffers allocated

uint16_t gBufLibBufferSize

Contains the size of each buffer

uint8_t gBufLibReadIndex

Tracks which buffer is currently being read from

uint8_t gBufLibWriteIndex

Tracks which buffer is currently being written to