StackShot Communications Interface

Revision 1.2

Table of Contents

1. Programming Interface

An Visual Studio C++ example of the communication protocol over USB (using FTDI D2XX drivers) is available here:
http://www.cognisys-inc.com/downloads/stackshot/StackshotUSB_app.zip

A NetBeans Java example of communicating over WiFi to the StackShot 3X controller is here:
http://www.cognisys-inc.com/downloads/stackshot3x/StackShot3xJava.zip

If using USB to communicate with the StackShot 3X controller, it is recommended to first use standard comm ports to find the controller, and if not found, then revert to the D2XX driver approach. Apple has recently introduced their "own" FTDI driver which fails to implement the D2XX portion of the driver. Using standard comm ports to communicate will function properly with either Apple's or FTDI's driver implementation.

Example code can be provided for using traditional comm ports if needed.

1.1 Functions

Below are the standard functions available via the StackShot API. Additional return values are available but any value other than the ones listed should be considered a failure. This is typically do to the cable being removed and the StackShot handle no longer being valid. In this situation the StackShot_Close function should be called and then re-opened (if required).

1.1.1 StackShot_Open

Summary

Open the specified StackShot device.

Definition

comm_status_t StackShot_Open(FT_HANDLE* handle, comm_rail_axis_t axis)

Parameters

handle Pointer to a variable of type FT_HANDLE where the handle will be stored. This handle must be used to access the device.

axis The desired axis to open. Available options are:

`COMM_RAIL_AXIS_ANY`	Opens the first StackShot device found.
`COMM_RAIL_AXIS_X`	Open the first StackShot configured for the X axis.
`COMM_RAIL_AXIS_Y`	Open the first StackShot configured for the Y axis.
`COMM_RAIL_AXIS_Z`	Open the first Stackshot configured for the Z axis.

Return Value

`COMM_STATUS_NOT_FOUND`	The specified device was not found
`COMM_STATUS_SUCCESS`	The device was opened

1.1.2 StackShot_Close

Summary

Close the specified StackShot device.

Definition

comm_status_t StackShot_Close(FT_HANDLE handle)

Parameters

handle The handle of the StackShot device to close.

Return Value

`COMM_STATUS_SUCCESS`	The device was closed

1.1.3 StackShot_Move

Summary

Move StackShot the specified distance. The move command is relative from the target position of the rail. If two move commands of 10.0 mm are issued back to back, the rail will move 20.0 mm total.

Definition

```
comm_status_t  StackShot_Move(FT_HANDLE handle,
                              comm_rail_dir_t direction,
                              comm_rail_units_t units,
                              bool backlash,
                              float distance)
```

Parameters

handle Handle for the StackShot device.

direction Direction to move the rail. Valid values are:

`COMM_RAIL_DIR_FWD`	Move StackShot forward.
`COMM_RAIL_DIR_BACK`	Move StackShot backward.

units Units that the distance represents. Valid values are:

`COMM_RAIL_UNITS_ENGLISH`	"*distance*" specifies the number of mils to move. (1.0 represents one thousandth of an inch)
`COMM_RAIL_UNITS_METRIC`	"*distance*" specifies the number of mm to move. (1.0 represents one millimeter)
`COMM_RAIL_UNITS_STEPS`	"*distance*" specifies the number of motor steps to move.

backlash Determines if backlash compensation is enabled for direction reversal. This should typically be set to "true".

`true`	Backlash compensation enabled.
`false`	Backlash compensation disabled.

distance A floating point value of the distance to move StackShot. The units is specified by the "*units*" parameter.

Return Value

`COMM_STATUS_BAD_PARAM`	One of the parameters specified is not a valid value.
`COMM_STATUS_BUSY`	StackShot is currently moving. The move request was rejected.
`COMM_STATUS_SUCCESS`	StackShot has started moving the specified distance.

1.1.4 StackShot_MoveAtSpeed

Summary

Move StackShot the specified distance. This move command is relative from the rail's immediate position. If two move commands of 10.0 mm are issued back to back, the rail will move 10.0 mm from the position where the rail is upon receiving the second command.

Definition

```
comm_status_t  StackShot_MoveAtSpeed(FT_HANDLE handle,
                                     comm_rail_dir_t direction,
                                     comm_rail_units_t units,
                                     float distance,
                                    float speedPercent)
```

Parameters

handle Handle for the StackShot device.

direction Direction to move the rail. Valid values are:

`COMM_RAIL_DIR_FWD`	Move StackShot forward.
`COMM_RAIL_DIR_BACK`	Move StackShot backward.

units Units that the distance represents. Valid values are:

`COMM_RAIL_UNITS_ENGLISH`	"*distance*" specifies the number of mils to move. (1.0 represents one thousandth of an inch)
`COMM_RAIL_UNITS_METRIC`	"*distance*" specifies the number of mm to move. (1.0 represents one millimeter)
`COMM_RAIL_UNITS_STEPS`	"*distance*" specifies the number of motor steps to move.

distance A floating point value of the distance to move StackShot. The units is specified by the "*units*" parameter.

speedPct A floating point value that specifies the percent of maximum speed to move the rail. 1.0 would be 100%.

Return Value

`COMM_STATUS_BAD_PARAM`	One of the parameters specified is not a valid value.
`COMM_STATUS_SUCCESS`	StackShot has started moving the specified distance.

1.1.5 StackShot_ShutterFire

Summary

Start the shutter trigger output. If a parameter is out of bounds it will be bounded to the minimum/maximum allowed. Calling StackShot_GetStatus will indicate when the shutter is complete.

Definition

```
comm_status_t  StackShot_ShutterFire(FT_HANDLE handle,
                                     short numPulses,
                                     float pulseDuration,
                                     float pulseOffTime)
```

Parameters

handle	Handle for the StackShot device.
numPulses	The number of pulses to generate on the shutter output.
pulseDuration	The “on” time of each pulse, in seconds.
pulseOffTime	The “off” time of each pulse, in seconds.

Return Value

`COMM_STATUS_SUCCESS`	The shutter trigger output is activated.

1.1.6 StackShot_GetStatus

Summary
Retrieve the current operational status of the StackShot device. Note: The `RAIL_STATUS_USER_ABORTED` bit will clear once read (sticky).

Definition
comm_status_t StackShot_GetStatus(FT_HANDLE handle, unsigned int* railStatus)

Parameters
handle Handle for the StackShot device
railStatus A 32-bit wide bit-field containing the status of the rail:

`#define RAIL_STATUS_IDLE`	`0x00`	The rail is idle.
`#define RAIL_STATUS_MOVING`	`0x01`	The rail is currently moving.
`#define RAIL_STATUS_SHUTTER`	`0x02`	The shutter is currently firing.
`#define RAIL_STATUS_USER_ABORTED`	`0x04`	The user aborted the move by pressing a button.

Return Value

`COMM_STATUS_SUCCESS`	The status was properly retrieved.

1.1.7 StackShot_Stop

Summary
Stop StackShot from moving.

Definition
comm_status_t StackShot_Stop(FT_HANDLE handle)

Parameters
handle The handle for the StackShot device

Return Value

`COMM_STATUS_SUCCESS`	The rail was stopped.

1.1.8 StackShot_GetCurrentPosition

Summary
Get the current position of the StackShot device. The position is completely relative to the initial power-on origin (zero) so the result may be positive or negative.

Definition
comm_status_t StackShot_GetCurrentPosition(FT_HANDLE handle,
comm_rail_units_t units,
float* position)

Parameters

handle The handle for the StackShot device.

units Units that the distance represents. Valid values are:

`COMM_RAIL_UNITS_ENGLISH`	"*distance*" specifies mils. (1.0 represents one thousandth of an inch)
`COMM_RAIL_UNITS_METRIC`	"*distance*" specifies mm (1.0 represents one millimeter)
`COMM_RAIL_UNITS_STEPS`	"*distance*" specifies the number of motor steps

position A pointer to store the current position in the units specified above.

Return Value

`COMM_STATUS_SUCCESS`	The position was properly retrieved.

1.1.9 StackShot_ZeroPosition

Summary
Sets the current position of StackShot to zero. This does not move the rail, it simply sets the internal position (both current and target) to zero. This may be useful to avoid relative (signed) positions from being returned.

Definition
comm_status_t StackShot_ZeroPosition(FT_HANDLE handle)

Parameters

handle The handle for the StackShot device.

Return Value

`COMM_STATUS_SUCCESS`	The current position was set to zero.

1.1.10 StackShot_GetSoftwareID

Summary
Retrieve the software ID of the StackShot device.

Definition
comm_status_t StackShot_GetSoftwareID(FT_HANDLE handle, unsigned int* swID)

Parameters

handle	The handle for the StackShot device
swID	32-bit pointer to a value representing the software version of StackShot

Return Value

`COMM_STATUS_SUCCESS`	The software ID was properly retrieved.

1.1.11 StackShot_GetSoftwareString

Summary
Retrieve the software string of the StackShot device.

Definition
comm_status_t StackShot_GetSoftwareString(FT_HANDLE handle, String** swString)

Parameters

handle	The handle for the StackShot device.
swString	Pointer to a software string to store the retrieved software string.

Return Value

`COMM_STATUS_SUCCESS`	The software string was properly retrieved.

1.1.12 StackShot_GetHardwareID

Summary
Retrieve the hardware ID of the StackShot device. Each hardware revision has a unique identifier associated with it.

Definition
comm_status_t StackShot_GetHardwareID(FT_HANDLE handle, unsigned int* hwID)

Parameters

handle	The handle for the StackShot device
hwID	32-bit pointer to a value representing the hardware version of StackShot

Return Value

`COMM_STATUS_SUCCESS`	The hardware ID was properly retrieved.

1.1.13 StackShot_GetBootloaderID

Summary
Retrieve the boot-loader ID of the StackShot device. Different boot-loader versions may have additional features so the version is tracked seperately from the base software.

Definition
comm_status_t StackShot_GetBootloaderID(FT_HANDLE handle, unsigned int* blID)

Parameters

handle	The handle for the StackShot device
blID	32-bit pointer to a value representing the boot-loader version in StackShot

Return Value

`COMM_STATUS_SUCCESS`	The boot-loader ID was properly retrieved.

1.1.14 StackShot_GetChecksum

Summary
Calculate the flash-memory checksum of the current StackShot software.

Definition
comm_status_t StackShot_GetChecksum(FT_HANDLE handle, unsigned int* checksum)

Parameters
handle The handle for the StackShot device.
checksum 32-bit pointer to a value representing the software checksum in StackShot.

Return Value
`COMM_STATUS_SUCCESS` The checksum was properly retrieved.

1.1.15 StackShot_GetSerialNumber

Summary
Get the stored serial number to the specified StackShot device. A value of 0xFFFFFFFF represents an unprogrammed serial number.

Definition
comm_status_t StackShot_GetSerialNumber(FT_HANDLE handle, unsigned int* serial)

Parameters
handle The handle for the StackShot device.
blID 32-bit pointer to a value representing the serial number for this StackShot device.

Return Value
`COMM_STATUS_SUCCESS` The serial number was properly retrieved.

1.1.16 StackShot_SetSerialNumber

Summary
Store the specified serial number into the StackShot device. This is a write-once register.

Definition
comm_status_t StackShot_SetSerialNumber(FT_HANDLE handle, unsigned int serial)

Parameters

handle	The handle for the StackShot device
serial	32-bit value to store as the device's serial number.

Return Value

`COMM_STATUS_NOT_EMPTY`	The serial number was not stored because one is already saved.
`COMM_STATUS_SUCCESS`	The serial number was properly stored.

1.1.17 StackShot_GetNVMRegister

Summary
Retrieve a 32-bit non-volatile value from StackShot. These registers may be used by developers to store unique identifiers such as keys, additional serialization, etc. A value of 0xFFFFFFFF represents an unprogrammed register. These values may be overwritten as needed but the writes should be limited over the life of the product. Register "0" maps to the axis definition for the device (but you may also use StackShot_GetAxis to access this register).

Definition
comm_status_t StackShot_GetNVMRegister(FT_HANDLE handle,
unsigned char reg,
unsigned int* regData)

Parameters

handle	The handle for the StackShot device.
reg	The register to read. Valid values are 0-31.
regData	32-bit pointer to store the register value read from StackShot.

Return Value

`COMM_STATUS_BAD_PARAM`	The "reg" value is outside of the allowed range.
`COMM_STATUS_SUCCESS`	The register data was properly retrieved.

1.1.18 StackShot_SetNVMRegister

Summary
Store a 32-bit non-volatile value to StackShot. These registers may be used by developers to store unique identifiers such as keys, additional serialization, etc. A value of 0xFFFFFFFF represents an unprogrammed register. These values may be overwritten as needed but the writes should be limited over the life of the product. Register "0" maps to the axis definition for the device (but you may also use StackShot_GetAxis to access this register).

Definition
comm_status_t StackShot_SetNVMRegister(FT_HANDLE handle,
unsigned char reg,
unsigned int regData)

Parameters

handle	The handle for the StackShot device.
reg	The register to write. Valid values are 0-31.
regData	32-bit value to store in the specified register.

Return Value

`COMM_STATUS_BAD_PARAM`	The "reg" value is outside of the allowed range.
`COMM_STATUS_SUCCESS`	The register data was properly stored.

1.1.19 StackShot_GetAxis

Summary
Get the stored axis identifer for this StackShot. An axis value of 0xFFFFFFFF indicates the axis has not been stored.

Definition
comm_status_t StackShot_GetAxis(FT_HANDLE handle, comm_rail_axis_t* axis)

Parameters

handle	The handle for the StackShot device.
axis	32-bit pointer to store the current axis.

Return Value

`COMM_STATUS_SUCCESS`	The serial number was properly retrieved.

1.1.20 StackShot_SetAxis

Summary

Store the specified axis identifier into the StackShot device. This axis identifier may later be used when calling the StackShot_Open function to potentially have multiple StackShot devices opened simultaneously.

Definition

comm_status_t StackShot_SetAxis(FT_HANDLE handle, comm_rail_axis_t axis)

Parameters

handle	The handle for the StackShot device
axis	The axis identifier to store for this StackShot device.

Return Value

`COMM_STATUS_SUCCESS`	The axis identifier was properly stored.

1.2 Type definitions (typedefs)

```
typedef enum
{
  COMM_STATUS_FAILED,
  COMM_STATUS_SUCCESS,         // The operation was successful
  COMM_STATUS_BUSY,            // The current operation could not begin because it is already in progress
  COMM_STATUS_DATA_MISSING,    // Data was not fully read
  COMM_STATUS_BAD_SYNC,        // The sync-byte was missing in the communiction stream
  COMM_STATUS_BUFFER_OVERRUN, // The controller returned more data than could be processed
  COMM_STATUS_NOT_EMPTY,       // The requested write operation could not be performed because data was already present
  COMM_STATUS_IO_ERROR,        // An input/output error occured (USB unplugged, bad handle, etc)
  COMM_STATUS_BAD_PARAM,       // One of the parameters passed into the function was rejected by the controller
  COMM_STATUS_NOT_FOUND        // The requested controller could not be found via USB
} comm_status_t;

typedef enum
{
  COMM_RAIL_AXIS_ANY,          // Use any available axis when opening
  COMM_RAIL_AXIS_X,            // Only open the X axis
  COMM_RAIL_AXIS_Y,            // Only open the Y axis
  COMM_RAIL_AXIS_Z,            // Only open the Z axis
  COMM_RAIL_AXIS_UNDEFINED     // The axis stored in the controller is undefined
} comm_rail_axis_t;


typedef enum
{
  CC_RAIL_MOVE = 0x1000,              // Move the rail to the specified position
  CC_RAIL_POSITION_TARGET,            // Desired target position for the rail
  CC_RAIL_POSITION_CURRENT,           // Current position of the rail
  CC_RAIL_POSITION_ZERO,              // Zero out the current position of the rail
  CC_RAIL_SHUTTER_FIRE,               // Fire the shutter control
  CC_RAIL_STATUS,                     // Retrieve the current controller status
  CC_RAIL_STOP,                       // Stop the rail from moving
  CC_RAIL_MOVE_AT_SPEED,              // Move the rail at the specified speed
  // StackShot 3X additional commands:
  CC_RAIL_STACK,                      // Run a 1 - 3 axis stack using the passed in parameters
  CC_RAIL_STACK_STATUS,               // Get the current status of a running stack
  CC_RAIL_STACK_POSITION,             // Get the current position of a stack
  CC_RAIL_STOP_RAMP,                  // Stop the motor with a soft ramp

  CC_RAIL_CONFIG_NAME = 0x1080,       // The name for the current configuration
  CC_RAIL_CONFIG_BACKLIGHT,           // Backlighting configuration
  CC_RAIL_CONFIG_MODE,                // Operating mode of the controller
```

```
CC_RAIL_CONFIG_UNITS,               // Units -- mm/mils/steps
CC_RAIL_CONFIG_TORQUE,              // Torque setting for the motor
CC_RAIL_CONFIG_NUM_STEPS,           // Number of steps to use for a stack
CC_RAIL_CONFIG_NUM_PULSES,          // Number of pulses on the shutter per step
CC_RAIL_CONFIG_TOTAL_DISTANCE,      // Total distance config
CC_RAIL_CONFIG_DISTANCE_PER_STEP,   // Distance to travel per step
CC_RAIL_CONFIG_SETTLE_TIME,         // Settling time
CC_RAIL_CONFIG_OFF_TIME,            // Off time between shutter pulses
CC_RAIL_CONFIG_SPEED,               // Speed that the rail will move
CC_RAIL_CONFIG_RAMP_TIME,           // Ramp time for the rail
CC_RAIL_CONFIG_DISTANCE_PER_REV,    // Linear distance per revolution of the motor
CC_RAIL_CONFIG_SHUTTER_DISABLE,     // Shutter disable feature enabled/disabled
CC_RAIL_CONFIG_AUTO_RETURN,         // Auto-return feature enabled/disabled
CC_RAIL_CONFIG_SAVE,                // Save the current configuration
CC_RAIL_CONFIG_LOAD,                // Load the specified configuration
CC_RAIL_CONFIG_AXIS,                // Get/set the controllers configured axis
CC_RAIL_CONFIG_TIMELAPSE,           // Time-lapse feature enabled/disabled
CC_RAIL_CONFIG_PULSE_TIME,          // On time of the shutter pulse
CC_RAIL_CONFIG_BACKLASH,            // Rail backlash configuration
// StackShot 3X additional commands:
CC_RAIL_CONFIG_SPEED_MOVE,          // Speed used by FWD/BACK buttons
CC_RAIL_CONFIG_POLARITY,            // Direction polarity
CC_RAIL_CONFIG_IO_MODE,             // IO mode -- Normal, master, slave
CC_RAIL_CONFIG_IO_DIR,              // IO mode = Master, FWD or BACK
CC_RAIL_CONFIG_ROTARY_BACKLASH,     // Backlash for rotary tables
CC_RAIL_CONFIG_ROTARY_RATIO,        // Rotary table gear ratio
CC_RAIL_CONFIG_ROTARY_DEGREES,      // Degrees to move per step
CC_RAIL_CONFIG_GLOBAL_MODE,         // Global operation mode
CC_RAIL_CONFIG_TOFF_FINAL,          // Whether or not a final Toff is generated if #pulse > 1
CC_RAIL_CONFIG_SHUTTER,             // Configure how the shutter button should operate


CC_RESET = 0x1100,                  // Reset the controller
CC_REFLASH,                         // Start reflash
CC_SOFTWARE_STRING,                 // Software string (human readable)
CC_SOFTWARE_ID,                     // Sftware identifier
CC_HARDWARE_ID,                     // Hardware identifer
CC_BOOTLOADER_ID,                   // Bootloader identifer
CC_SOFTWARE_CHECKSUM,               // Software checksum
CC_SERIAL_NUMBER,                   // The serial number of the device
CC_NVM_ACCESS,                      // NVM register access
// StackShot 3X additional commands:
CC_NAND_ACCESS,                     // Access NAND flash (unimplemented)
CC_NOR_ACCESS,                      // Access NOR flash (unimplemented)
CC_PING,                            // Ping the controller to make sure it is there (Handy for WIFI)
CC_LOG,                             // Retrieve the current log file from the controller
CC_WIFI,
CC_CLOSE,
CC_WAD = 0x1200,                    // Send a composite message (encapsulates several commands in one message for speed)

CC_MOTION_CONFIG = 0x1600,
```

```
  CC_MOTION_START
} comm_cmd_t;


typedef enum
{
  COMM_ACTION_MIN,
  COMM_ACTION_READ = COMM_ACTION_MIN, // Read the specified command
  COMM_ACTION_WRITE,                  // Write the specified command
  COMM_ACTION_RSP_OK,                 // Controller responded with OK
  COMM_ACTION_BAD_PARAM,              // A bad parameter was passed to the controller
  COMM_ACTION_UNSUPPORTED_ACTION,     // The action specified is invalid
  COMM_ACTION_UNSUPPORTED_CMD,        // The command passed in is invalid
  COMM_ACTION_FAILED,                 // The command failed (no further information available)
  COMM_ACTION_NOT_EMPTY,              // The write operation was to a write-once register and it is no longer empty
  COMM_ACTION_BUSY,                   // The controller is already performing the specified action
  COMM_ACTION_MAX
} comm_action_t;


typedef enum
{
  COMM_RAIL_DIR_MIN,
  COMM_RAIL_DIR_FWD = COMM_RAIL_DIR_MIN,     // Move the rail in the forward direction
  COMM_RAIL_DIR_BACK,                        // Move the rail in the backward direction
  COMM_RAIL_DIR_MAX
} comm_rail_dir_t;

typedef enum
{
  COMM_RAIL_UNITS_MIN,
  COMM_RAIL_UNITS_ENGLISH = COMM_RAIL_UNITS_MIN,     // English/mils
  COMM_RAIL_UNITS_METRIC,                            // Metric/mm
  COMM_RAIL_UNITS_STEPS,                             // Motor steps
  COMM_RAIL_UNITS_MAX
} comm_rail_units_t;


typedef enum
{
  COMM_RAIL_MODE_MIN,
  COMM_RAIL_MODE_AUTO_STEP = COMM_RAIL_MODE_MIN,     // Automatic step mode
  COMM_RAIL_MODE_AUTO_DIST,                          // Automatic distance mode
  COMM_RAIL_MODE_TOTAL_DISTANCE,                     // Total distance mode
  COMM_RAIL_MODE_DISTANCE_PER_STEP,                  // distance per step mode
  COMM_RAIL_MODE_MANUAL,                             // Manual mode
  COMM_RAIL_MODE_CONTINUOUS,                         // Continuous mode
  COMM_RAIL_MODE_MAX
} comm_rail_mode_t;
```

```
typedef enum
{
        GLOBAL_MODE_ADVANCED,                // Single axis stacking
        GLOBAL_MODE_STACK_PANO_2X,           // X-Y stacking (scanning)
        GLOBAL_MODE_STACK_PANO_3X,           // X-Y-Z stacking (scanning)
        GLOBAL_MODE_STACK_ROTATE_2X,         // Stack then rotate
        GLOBAL_MODE_STACK_ROTATE_3X,         // Stack, move, then rotate
        GLOBAL_MODE_PANO_1X,                 // Single-axis panoramic
        GLOBAL_MODE_PANO_2X,                 // Dual-axis panoramic
        GLOBAL_MODE_PANO_1X_360,             // 360 degree panoramic
        GLOBAL_MODE_MOTION_DFMOCO,           // DragonFrame DfMoco mode
        GLOBAL_MODE_MOTION_RESERVED,         // Reserved, do not use
        GLOBAL_MODE_MOTION_CONTINUOUS,       // Bezier curve continuous motion (video)
        GLOBAL_MODE_MOTION_CONT_TIMELAPSE,   // Bezier curve continuous motion (timelapse)
        GLOBAL_MODE_MOTION_SMS,              // Bezier curve shoot-move-shoot (timelapse)
        GLOBAL_MODE_MAX
} setting_global_mode_t;


typedef enum
{
        SETTING_SHUTTER_MIN,
        SETTING_SHUTTER_NONE = SETTING_SHUTTER_MIN,  // No shutter is active for this axis
        SETTING_SHUTTER_1,                           // Shutter 1 will be active
        SETTING_SHUTTER_2,                           // Shutter 2 will be active
        SETTING_SHUTTER_BOTH,                        // Both shutter 1 & 2 will be active
        SETTING_SHUTTER_MAX
} setting_shutter_t;


typedef enum
{
        SETTING_AXIS_TYPE_MIN,
        SETTING_AXIS_TYPE_SLIDER = SETTING_AXIS_TYPE_MIN,    // Linear slider
        SETTING_AXIS_TYPE_RAIL,                              // Macro rail
        SETTING_AXIS_TYPE_ROTARY,                            // Rotary/Turn-table
        SETTING_AXIS_TYPE_MAX
} setting_axis_t;
```

1.3 Definitions (#defines)

```
#define COMM_BUFFER_SIZE      (40)           // Buffer size used for packet communications
#define STACKSHOT_BAUD_RATE   (38400)        // Baud-rate used for the StackShot interface

#define RAIL_STATUS_IDLE              0x00   // The rail is idle
#define RAIL_STATUS_MOVING            0x01   // The rail is currently moving
#define RAIL_STATUS_SHUTTER           0x02   // The shutter is currently firing
#define RAIL_STATUS_USER_ABORTED      0x04   // The user aborted a move by pressing a button on StackShot
```

2. Serial Communications Interface

One frame consists of a host command transmit and a slave response. The host should set a read timeout sufficient to receive the response and to account for USB latency.

Sync	CMD MSB	CMD LSB	Action/Status	Data Length	Data[0…n]	Checksum
0x55 (PC)						
0xAA (Slave)						

Length specifies the length of the data field. A length of zero would result in a packet of Sync, command (2), Length, and Checksum (sent but not currently verified) .

Mandatory 2ms inter-frame delay for re-syncing to recover from partial packet transmission.

For StackShot 3X, if the command corresponds to a specific axis or shutter output, that value is "or'ed" into the upper-nibble of the Action/Status. This allows compatibility with the current API for single-axis control. If the command applies to an axis or shutter output, it is "or'ed" into bits 4,5 (lowest bits in the upper nibble). If the command references a subset within a shutter output, bit 7 (MSB) clear will reference the first item, and bit 7 set will reference the second.

Example: Reading the current position has a command ID of 0x1002.
- The `COMM_ACTION_READ` has a value of 0x00.
- The length of this command is zero (nothing is being written).
- To read the X-axis, the packet would be:
  - 0x55 10 02 00 00 CK
- To read the Y-axis, the packet would be:
  - 0x55 10 02 10 00 CK
- To read the Z-axis, the packet would be:
  - 0x55 10 02 20 00 CK

Example: Reading the current "On time" for Shutter output 1 (command ID of 1094)
- The `COMM_ACTION_READ` has a value of 0x00.
- The length of this command is zero (nothing is being written).
- To read the Shutter 1 on time for the first pulse the packet would be:
  - 0x55 10 94 00 00 ck
- Shutter 1 2nd and subsequent on times:
  - 0x55 10 94 80 00 ck
- Shutter 2 on time for the first pulse:
  - 0x55 10 94 10 00 ck
- Shutter 2 2nd and subsequent on times:
  - 0x55 10 94 90 00 ck

Note: When parsing the response for the 3-axis controller, be sure to mask off the upper-nibble of the Action/Status if investigating the response.

Byte ordering:

The data field shall use the following for U32's: LSB's first

Example: To send the value 0x12345678 the byte order would be:

Length: 4

Data: 0x78 56 34 12 (least-significant is sent first)

The data field shall use the following for IEEE 32-bit floats: LSB's first

Example: To send the value of 12345678.0 as a float:

Length: 4 (32-bits)

The floating point hexadecimal value is: 0x4b3c614e (conversion to hex is language-specific: Cast as a U8* and reference, or use a `Float.floatToIntBits` function in Java)

Data: 0x4e 61 3c 4b

CC_RAIL_MOVE
ID: 0x1000
Length: 7
Action: Write only
Write:

Cmd Data:

Byte	Data type	Description
0	comm_rail_dir_t	Direction to move the rail. COMM_RAIL_DIR_FWD = 0, COMM_RAIL_DIR_BACK = 1
1	comm_rail_units_t	Units that the data represents
2	bool_t	Backlash compensation used for the move. 1 = True
3-6	F32	Distance in the specified units above to move the rail from the target position

Rsp Data: <none>

CC_RAIL_MOVE_AT_SPEED
ID: 0x1007
Length: 10
Action: Write only
Write:

Cmd Data:

Byte	Data type	Description
0	comm_rail_dir_t	Direction to move the rail. COMM_RAIL_DIR_FWD = 0, COMM_RAIL_DIR_BACK = 1
1	comm_rail_units_t	Units that the data represents
2-5	F32	Distance in the specified units above from the current position to move the rail
6-9	F32	Percent of maximum speed to move the rail (1.0 = 100%)

Rsp Data: <none>

CC_RAIL _POSITION_TARGET
ID: 0x1001
Length: 6
Action: Read Only
Read:

Cmd Data:

Byte	Data type	Description
0	comm_rail_units_t	Units that the data represents

Rsp Data:

Byte	Data type	Description
0	comm_rail_units_t	Units that the data represents
1-5	F32	Absolute target position in the specified units above that represents the target position of the rail.

See also **CC_RAIL_ZERO_POSITION.**

CC_RAIL _POSITION_CURRENT
ID: 0x1002
Length: 5
Action: Read only
Read:

Cmd Data:

Byte	Data type	Description
0	comm_rail_units_t	Units that the data represents

Rsp Data:

Byte	Data type	Description
0	comm_rail_units_t	Units that the data represents
1-4	F32	Absolute position in the specified units above to move the rail. May be signed value relative to the origin.

See also **CC_RAIL_ZERO_POSITION.**

CC_RAIL_ POSITION_ZERO
ID: 0x1003
Length: 0
Action: Write only
Write:

Cmd Data: <none>
Rsp Data: <none>

Set's the current position of the rail to the origin. This command will be rejected with COMM_STATUS_BUSY if the motor is moving.

CC_RAIL_SHUTTER_FIRE
ID: 0x1004
Length: 10
Action: Write only
Write:

Cmd Data:

Byte	Data type	Description
0-1	U16	Number of times to fire the shutter
2-5	F32	Duration of shutter pulse (non-zero, non-negative).
6-9	F32	Duration between shutter pulses (non-zero, non-negative).

Rsp Data: <none>

CC_RAIL_STATUS
ID: 0x1005
Length: 4
Action: Read only
Write:

Cmd Data: <none>
Rsp Data:

Byte	Data type	Description
0-3	U32	Bit-field of the current status (see below)

RAIL_STATUS_IDLE	0000	Rail Idle
RAIL_STATUS_MOVING	0001	Rail moving
RAIL_STATUS_SHUTTER	0002	Shutter firing
RAIL_STATUS_USER_ABORTED	0004	User aborted during a move

CC_RAIL_STACK (StackShot 3X only – subject to change!)
ID: 0x1008
Length: 48
Action: Write only
Write:

Cmd Data:

Byte	Data type	Description
0-3	setting_mode_t	The operating mode for this stack
4-7	U32	Number of steps (?)
8-11	U32	Distance
12-15	U32	Pano-frame start (in motor steps).
16-19	U32	Pano-frame stop (in motor steps).
20-23	U32	X axis start position
24-27	U32	X axis end position
28-31	U32	Y axis start position
32-35	U32	Y axis end position
36-39	U32	Z axis start position
40-43	U32	Z axis end position

Rsp Data: <none>

CC_RAIL_CONFIG_NAME
ID: 0x1080
Length: Variable
Action: Read/Write
Read:

Cmd Data: <none>
Rsp Data:

Byte	Data type	Description
0-n	String	Null-terminated string for the current configuration. On read, 0xFF indicates power-on / last saved configuration.

Write:

Cmd Data:

Byte	Data type	Description
0-n	String	Null-terminated string for the current configuration.

Rsp Data: <none>

CC_RAIL_CONFIG_BACKLIGHT
ID: 0x1081
Length: 1
Action: Read/Write
Read:

Cmd Data: <none>
Rsp Data:

Byte	Data type	Description
0	U8	Backlighting level. Valid values are 1-10.

Write:

Cmd Data:

Byte	Data type	Description
0	U8	Backlighting level. Valid values are 1-10.

Rsp Data: <none>

CC_RAIL_CONFIG_MODE
ID: 0x1082
Length: 1
Action: Read/Write
Read:

Cmd Data: <none>
Rsp Data:

Byte	Data type	Description
0	comm_rail_mode_t	Returns the current operating mode of the rail. Applies only to stacking.

Write:

Cmd Data:

Byte	Data type	Description
0	comm_rail_mode_t	Sets the current operating mode of the rail. Applies only to stacking.

Rsp Data: <none>

CC_RAIL_CONFIG_UNITS
ID: 0x1083
Length: 1
Action: Read/Write
Read:

Cmd Data: <none>
Rsp Data:

Byte	Data type	Description
0	comm_rail_units_t	Specifies the measuring units displayed.

Write:

Cmd Data:

Byte	Data type	Description
0	comm_rail_units_t	Specifies the measuring units displayed.

Rsp Data: <none>

CC_RAIL_CONFIG_TORQUE
ID: 0x1084
Length: 1
Action: Read/Write
Read:

Cmd Data: <none>
Rsp data:

Byte	Data type	Description
0	U8	Torque setting. Valid values are 1-10.

Write:

Cmd Data:

Byte	Data type	Description
0	U8	Torque setting. Valid values are 1-10.

Rsp Data: <none>

CC_RAIL_CONFIG_HOLDING_TORQUE
ID: 0x1096
Length: 1
Action: Read/Write
Read:

Cmd Data: <none>
Rsp data:

Byte	Data type	Description
0	U8	Holding torque setting. Valid values are 0-10.

Write:

Cmd Data:

Byte	Data type	Description
0	U8	Holding torque setting. Valid values are 0-10.

Rsp Data: <none>

CC_RAIL_CONFIG_NUM_STEPS
ID: 0x1085
Length: 1
Action: Read/Write
Read:

Cmd Data: <none>
Rsp Data:

Byte	Data type	Description
0	U16	Number of steps to be taken for one series of a capture.

Write:

Cmd Data:

Byte	Data type	Description
0	U16	Number of steps to be taken for one series of a capture.

Rsp Data: <none>

CC_RAIL_CONFIG_NUM_PULSES
ID: 0x1086
Length: 2
Action: Read/Write
Read:

Cmd Data: <none>
Rsp Data:

Byte	Data type	Description
0	U16	Number of pulses (pictures to be taken) applied at each step.

Write:

Cmd Data:

Byte	Data type	Description
0	U16	Number of pulses (pictures to be taken) applied at each step.

Rsp Data: <none>

CC_RAIL_CONFIG_TOTAL_DISTANCE
ID: 0x1087
Length: 5
Action: Read/Write
Read:

Cmd Data:

Byte	Data type	Description
0	comm_rail_units_t	Units that the data represents.

Rsp Data: <none>

Byte	Data type	Description
0	comm_rail_units_t	Units that the data represents.
1-4	F32	Distance in the specified units above for the “Total Distance” configuration.

Write:

Cmd Data:

Byte	Data type	Description
0	comm_rail_units_t	Units that the data represents.
1-4	F32	Distance in the specified units above for the “Total Distance” configuration.

Rsp Data: <none>

CC_RAIL_CONFIG_DISTANCE_PER_STEP
ID: 0x1088
Length: 5
Action: Read/Write
Read:

Cmd Data:

Byte	Data type	Description
0	comm_rail_units_t	Units that the data represents.

Rsp Data:

Byte	Data type	Description
0	comm_rail_units_t	Units that the data represents.
1-4	F32	Distance in the specified units above for the “Distance Per Step” configuration.

Write:

Cmd Data:

Byte	Data type	Description
0	comm_rail_units_t	Units that the data represents.
1-4	F32	Distance in the specified units above for the “Distance Per Step” configuration.

Rsp Data: <none>

CC_RAIL_CONFIG_SETTLE_TIME
ID: 0x1089
Length: 4
Action: Read/Write
Read:

Cmd Data: <none>
Rsp Data:

Byte	Data type	Description
0-3	F32	Time in seconds the rail should pause before activating the shutter.

Write:

Cmd Data:

Byte	Data type	Description
0-3	F32	Time in seconds the rail should pause before activating the shutter.

Rsp Data: <none>

CC_RAIL_CONFIG_PULSE_TIME
ID: 0x1094
Length: 4
Action: Read/Write
Read:

Cmd Data: <none>
Rsp Data:

Byte	Data type	Description
0-3	F32	Time in seconds that the shutter output should be activated.

Write:

Cmd Data:

Byte	Data type	Description
0-3	F32	Time in seconds that the shutter output should be activated.

Rsp Data: <none>

CC_RAIL_CONFIG_OFF_TIME
ID: 0x108A
Length: 4
Action: Read/Write
Read:

Cmd Data: <none>
Rsp Data:

Byte	Data type	Description
0-3	F32	Time in seconds to delay between consecutive shutter firings (only applies if "Num Pulses" is greater than one).

Write:

Cmd Data:

Byte	Data type	Description
0-3	F32	Time in seconds to delay between consecutive shutter firings (only applies if "Num Pulses" is greater than one).

Rsp Data: <none>

CC_RAIL_CONFIG_TIMELAPSE
ID: 0x1093
Length: 4
Action: Read/Write
Read:

Cmd Data: <none>
Rsp Data:

Byte	Data type	Description
0-3	F32	Time in seconds that the time-lapse feature will run.

Write:

Cmd Data:

Byte	Data type	Description
0-3	F32	Time in seconds that the time-lapse feature will run.

Rsp Data: <none>

CC_RAIL_CONFIG_STACK_SPEED
ID: 0x108B
Length: 5
Action: Read/Write
Read:

Cmd Data:

Byte	Data type	Description
0	comm_rail_units_t	Units that the data represents.

Rsp Data:

Byte	Data type	Description
0	comm_rail_units_t	Units that the data represents.
1-4	F32	Maximum stacking speed of the rail in units per second.

Write:

Cmd Data:

Byte	Data type	Description
0	comm_rail_units_t	Units that the data represents.
1-4	F32	Maximum stacking speed of the rail in units per second.

Rsp Data: <none>

CC_RAIL_CONFIG_MOVE_SPEED
ID: 0x1097
Length: 5
Action: Read/Write
Read:

Cmd Data:

Byte	Data type	Description
0	comm_rail_units_t	Units that the data represents.

Rsp Data:

Byte	Data type	Description
0	comm_rail_units_t	Units that the data represents.
1-4	F32	Maximum speed of the fwd/back buttons in units per second.

Write:

Cmd Data:

Byte	Data type	Description
0	comm_rail_units_t	Units that the data represents.
1-4	F32	Maximum speed of the fwd/back buttons in units per second.

Rsp Data: <none>

CC_RAIL_CONFIG_RAMP_TIME
ID: 0x108C
Length: 4
Action: Read/Write

Read:

Cmd Data: <none>

Rsp Data:

Byte	Data type	Description
0-3	F32	Time in seconds the rail will ramp from start to final speed.

Write:

Cmd Data:

Byte	Data type	Description
0-3	F32	Time in seconds the rail will ramp from start to final speed.

Rsp Data: <none>

CC_RAIL_CONFIG_DISTANCE_PER_REV
ID: 0x108D
Length: 5
Action: Read/Write
Read:

Cmd Data: <none>
Rsp Data:

Byte	Data type	Description
0	comm_rail_units_t	Units that the data represents.
1-4	F32	Distance in the specified units above that the rail travels per revolution of the motor.

Write:

Cmd Data:

Byte	Data type	Description
0	comm_rail_units_t	Units that the data represents.
1-4	F32	Distance in the specified units above that the rail travels per revolution of the motor.

Rsp Data: <none>

CC_RAIL_CONFIG_BACKLASH
ID: 0x1095
Length: 5
Action: Read/Write
Read:

Cmd Data:

Byte	Data type	Description
0	comm_rail_units_t	Units that the data represents.

Rsp Data:

Byte	Data type	Description
0	comm_rail_units_t	Units that the data represents.
1-4	F32	Distance in the specified units for backlash compensation.

Write:

Cmd Data:

Byte	Data type	Description
0	comm_rail_units_t	Units that the data represents.
1-4	F32	Distance in the specified units for backlash compensation.

Rsp Data: <none>

CC_RAIL_CONFIG_ROTARY_BACKLASH
ID: 0x109B
Length: 4
Action: Read/Write
Read:

Cmd Data: <none>
Rsp Data:

Byte	Data type	Description
0-3	F32	Distance in stepper motor degrees of the rotary backlash. This backlash is independent of the linear backlash.

Write:

Cmd Data:

Byte	Data type	Description
0-3	F32	Distance in stepper motor degrees of the rotary backlash. This backlash is independent of the linear backlash.

Rsp Data: <none>

CC_RAIL_CONFIG_SHUTTER_DISABLE
ID: 0x108E
Length: 1
Action: Read/Write
Read:

Cmd Data: <none>
Rsp Data:

Byte	Data type	Description
0	bool_t	If “True”, shutter disable is active.

Write:

Cmd Data:

Byte	Data type	Description
0	bool_t	If “True”, shutter disable is active.

Rsp Data: <none>

CC_RAIL_CONFIG_AUTO_RETURN
ID: 0x108F
Length: 1
Action: Read/Write
Read:

Cmd Data:

Byte	Data type	Description
0	bool_t	If “True”, auto-return upon finishing a sequence is enabled.

Rsp Data: <none>

Write:

Cmd Data:

Byte	Data type	Description
0	bool_t	If “True”, auto-return upon finishing a sequence is enabled.

Rsp Data: <none>

CC_RAIL_CONFIG_SAVE
ID: 0x1090
Length: 1
Action: Write Only
Write:

Cmd Data:

Byte	Data type	Description
0	U8	0 – Store the active configuration as the “power on” configuration. 1-10 – Store the active configuration in the specified user slot.

Rsp Data: <none>

This command commits the data to NVM.

CC_RAIL_CONFIG_LOAD
ID: 0x1091
Length: 1
Action: Write only
Write:

Cmd Data:

Byte	Data type	Description
0	U8	0 – Load the active configuration with the “power on” configuration. 1-10 – Load the active configuration from the specified user slot. If the slot is empty, an error will be returned and the load will not complete.

Rsp Data: <none>

CC_RAIL_CONFIG_AXIS
ID: 0x1092
Length: 1
Action: Read/Write
Read:

Cmd Data: <none>
Rsp Data:

Byte	Data type	Description
0	U8	Specifies which axis this controller corresponds to. Can be useful for configuring multi-axis control so that the API can manage multiple controllers. The write immediately commits the change to NVM.

Write:

Cmd Data:

Byte	Data type	Description
0	U8	Specifies which axis this controller corresponds to. Can be useful for configuring multi-axis control so that the API can manage multiple controllers. The write immediately commits the change to NVM.

Rsp Data: <none>

CC_RAIL_CONFIG_POLARITY
ID: 0x1098
Length: 1
Action: Read/Write
Read:

Cmd Data: <none>
Rsp Data:

Byte	Data type	Description
0	bool_t	False – Normal. True – Reverse.

Write:

Cmd Data:

Byte	Data type	Description
0	bool_t	False – Normal. True – Reverse.

Rsp Data: <none>

CC_RAIL_CONFIG_ROTARY_RATIO
ID: 0x109C
Length: 4
Action: Read/Write
Read:

Cmd Data: <none>
Rsp Data:

Byte	Data type	Description
0-3	F32	Gear-ratio of the rotary table. This is used to translate the stepper distance into the rotary distance.

Write:

Cmd Data:

Byte	Data type	Description
0-3	F32	Gear-ratio of the rotary table. This is used to translate the stepper distance into the rotary distance.

Rsp Data: <none>

CC_RAIL_CONFIG_ROTARY_DEGREES
ID: 0x109D
Length: 4
Action: Read/Write
Read:

Cmd Data: <none>
Rsp Data:

Byte	Data type	Description
0-3	F32	Distance to move the rotary table per step.

Write:

Cmd Data:

Byte	Data type	Description
0-3	F32	Distance to move the rotary table per step.

Rsp Data: <none>

CC_RAIL_CONFIG_GLOBAL_MODE (StackShot 3X only)
ID: 0x109E
Length: 1
Action: Read/Write
Read:
Cmd Data: <none>
Rsp Data:

Byte	Data type	Description
0	setting_global_mode_t	The current global mode.

Write:
Cmd Data:

Byte	Data type	Description
0	setting_global_mode_t	The current global mode.

Rsp Data: <none>

CC_RAIL_CONFIG_TOFF_FINAL
ID: 0x109F
Length: 1
Action: Read/Write
Read:
Cmd Data: <none>
Rsp Data:

Byte	Data type	Description
0	bool_t	If “True”, the final “Toff” time will be applied for the shutter output (default).

Write:
Cmd Data:

Byte	Data type	Description
0	bool_t	If “True”, the final “Toff” time will be applied for the shutter output (default).

Rsp Data: <none>

CC_RAIL_CONFIG_SHUTTER (StackShot 3X only)
ID: 0x10A0
Length: 1
Action: Read/Write
Read:
Cmd Data: <none>
Rsp Data:

Byte	Data type	Description
0	setting_shutter_t	For the specified axis, this value indicates which shutter should be active.

Write:
Cmd Data:

Byte	Data type	Description
0	setting_shutter_t	For the specified axis, this value indicates which shutter should be active.

Rsp Data: <none>

CC_CLOSE
ID: 0x110E
Length: 0
Action: Write only
Write:

Cmd Data: <none>
Rsp Data: <none>

Closes the USB mode on the controller. This should be the last command issued prior to closing the port. It allows the controller to clean up any current activities and change the display back to normal.

CC_ RESET
ID: 0x1100
Length: 0
Action: Write only
Write:

Cmd Data: <none>
Rsp Data: <none>

Performs a software reset on the module.

CC_ REFLASH
ID: 0x1101
Length: 0
Action: Write only
Write:

Cmd Data: <none>
Rsp Data: <none>

Launches the reflash bootloader

CC_SOFTWARE_STRING
ID: 0x1102
Length: <variable>
Action: Read only
Read:

Cmd Data: <none>
Rsp Data:

Byte	Data type	Description
0-n	String	Null-terminated string containing the current software version string.

CC_SOFTWARE_ID
ID: 0x1103
Length: 4
Action: Read only
Read:

Cmd Data: <none>
Rsp Data:

Byte	Data type	Description
0-3	U32	Unique software identifier

CC_HARDWARE_ID
ID: 0x1104
Length: 4
Action: Read only
Read:

Cmd Data: <none>
Rsp Data:

Byte	Data type	Description
0-3	U32	Unique hardware identifier

CC_BOOTLOADER_ID
ID: 0x1105
Length: 4
Action: Read only
Read:

Cmd Data: <none>
Rsp Data:

Byte	Data type	Description
0-3	U32	Unique bootloader identifier

CC_SOFTWARE_CHECKSUM
ID: 0x1106
Length: 4
Action: Read only

Read:

Cmd Data: <none>

Rsp Data:

Byte	Data type	Description
0-3	U32	Returns the calculated software checksum. The checksum is performed upon command issue so a delay in the response is to be expected.

CC_SERIAL_NUMBER

ID: 0x1107

Length: 4

Action: Read/Write

Read:

Cmd Data: <none>

Rsp Data:

Byte	Data type	Description
0-3	U32	Serial number of the device. 0xFFFFFFFF indicates an un-programmed unit.

Write:

Cmd Data:

Byte	Data type	Description
0-3	U32	Serial number to save to the device. **This is a one-time programmable burn.** Once the serial number is set it cannot be erased or reprogrammed. Returns COMM_ACTION_FAILED if a second write is attempted.

Rsp Data: <none>

CC_PING

ID: 0x110A

Length: 1 in / 2 out

Action: Read Only

Read:

Cmd Data

Byte	Data type	Description
0	U8	Value sent to controller for “exclusive or” response

Rsp Data:

Byte	Data type	Description
0	U8	“Exclusive or” response to value sent to controller above: out = in ^ 0xFF
1	U8	<Reserved, currently set to zero>

CC_NVM_ACCESS
ID: 0x1108
Length: See below
Action: Read/Write
Provides scratch area that the PC may use to store keys or other information.
Read:
Cmd Data:

Byte	Data type	Description
0	U8	Indicates which NVM register (0-127) that is desired

Rsp Data:

Byte	Data type	Description
0	U32	32-bit NVM register results

Write:
Cmd Data:

Byte	Data type	Description
0	U8	Indicates which NVM register (0-127) that is desired
1-4	U32	32-bit value to store to NVM (re-writes are possible).

Rsp Data: <none>

CC_LOG (StackShot 3X only)
ID: 0x110C
Length: 4
Action: Read Only
Read:
Cmd Data:

Byte	Data type	Description
0	U8	The index to read (increment this value until `COMM_ACTION_BAD_PARAM` is returned to read all entries).

Rsp Data:

Byte	Data type	Description
0-?	String	Returns a null-terminated string log entry of the passed in index.

CC_MOTION_CONFIG (StackShot 3X only)
ID: 0x1600
Length: 132
Action: Write Only
Write:
Cmd Data:

Byte	Data type	Description
0-3	setting_axis_t	Sets what type of device is attached to this axis. This will change internal parameters such as speed, gear ratios, and other settings for the device.
4-7	F32	Curve 0 – P0.x MUST be set to zero since this is the first curve
8-11	F32	Curve 0 – P0.y
12-15	F32	Curve 0 – P1.x
16-19	F32	Curve 0 – P1.y
20-23	F32	Curve 0 – P2.x
24-27	F32	Curve 0 – P2.y
28-31	F32	Curve 0 – P3.x
32-35	F32	Curve 0 – P3.y
36-39	F32	Curve 1 – P0.x MUST be set to zero if not used
40-43	F32	Curve 1 – P0.y
44-47	F32	Curve 1 – P1.x
48-51	F32	Curve 1 – P1.y
52-55	F32	Curve 1 – P2.x
56-59	F32	Curve 1 – P2.y
60-63	F32	Curve 1 – P3.x
64-67	F32	Curve 1 – P3.y
68-71	F32	Curve 2 – P0.x MUST be set to zero if not used
72-75	F32	Curve 2 – P0.y
76-79	F32	Curve 2 – P1.x
80-83	F32	Curve 2 – P1.y
84-87	F32	Curve 2 – P2.x
88-91	F32	Curve 2 – P2.y
92-95	F32	Curve 2 – P3.x
96-99	F32	Curve 2 – P3.y
100-103	F32	Curve 3 – P0.x MUST be set to zero if not used
104-107	F32	Curve 3 – P0.y
108-111	F32	Curve 3 – P1.x
112-115	F32	Curve 3 – P1.y
116-119	F32	Curve 3 – P2.x
120-123	F32	Curve 3 – P2.y
124-127	F32	Curve 3 – P3.x
128-131	F32	Curve 3 – P3.y

Rsp Data: <none>

Motion paths in StackShot 3X are defined using cubic Bézier curves. Up to four curves can be specified to form a Bézier spline yielding five keyframes. Each axis (specified using the upper nibble of the action) can have its own independent spline to follow.

There are some requirements for proper operation:

- Curve 0's starting time (P0.x) must be set to zero since it starts at time 0.
- If subsequent curves are not used then that curve's P0.x point must be set to zero (this will terminate the motion).
- A curve's initial P0.x must match the previous curves P3.x to provide continuity. The exception being Curve 0 since it is the first and at time 0.
- The spline must not have the P1 and P2 control points forcing the curve back upon itself (Curve 1 going backward in time into Curve 0's time)
- Control point P1.x must not be less than P0.x. Likewise, control point P2.x must not be more than P3.x. P0.x and P3.x are the bounds for all timing values.
- All linear distances are specified in mm. 100mm would have the P0.y value of "100", not "0.1". Positive and negative distances are allowed.
- All rotary distance are specified in degrees. Positive and negative distances are allowed but be careful of cable wrap-up or camera/lens damage.
- Limits should be applied to prevent sudden velocity changes which may result in loss of steps.
- The CC_MOTION_START command will begin the motion process.

CC_MOTION_START (StackShot 3X only)
ID: 0x1601
Length: 132
Action: Write Only
Write:
Cmd Data:

Byte	Data type	Description
0	bool_t	Start the X axis (0 – disable, 1 – enable)
1	bool_t	Start the Y axis (0 – disable, 1 – enable)
2	bool_t	Start the Z axis (0 – disable, 1 – enable)

Be sure to issue the CC_MOTION_CONFIG command prior to this. Otherwise the last-used curve will be active.

3. DragonFrame DfMoco interface

StackShot 3X supports DragonFrame's DfMoco interface. Up to three axes can be controlled. To enable the DfMoco interface (which runs over USB) go to Settings->Mode->Motion->DfMoco.

DfMoco allows simple control of your StackShot 3X controller using a terminal program or other method of accessing a serial communications port. The commands are discussed below. The DfMoco interface does not support real-time motion control. It can only be used for shoot-move-shoot applications.

All commands/responses are ASCII. All commands/responses are terminated with "\r\n" (carriage return, followed by a line-feed).

Serial port settings:
57600 BPS
8-bit
No parity
1 Stop bit
(57600 8N1)

Motors are specified as 1, 2, or 3.

Host	StackShot3X action/response
cmd: hi hi, who's out there?	rsp: hi <moco_version> <motor count> <version string> rsp: "hi 1 3 1.2.6 "
cmd: zm <motor> zero motor's position.	rsp: mp <motor> <current position>
cmd: mm <motor> <position> Move motor to a position.	If already at the specified position: rsp: mp <motor> <current position> If NOT at the specified position, this will be transmitted periodically: rsp: mm <motor> <current position>
cmd: mp <motor> Get the motors position. Optionally, a "*" can be sent as the motor to have all positions sent.	rsp: mp <motor> <current position>
cmd: np <motor> <position> Set a new target position. If the motor is currently moving it will go to that position.	rsp: mp <motor> <current position> (yes, that is a "m")
cmd: ms Request if the motors are moving or not.	rsp: ms <motor 1 moving><motor 2 moving><motor 3 moving> ex: ms 101 In this case, motor 1 and motor 3 are moving.
cmd: sm <motor> Stop the specified motor	rsp: sm <motor>
cmd: sa <motor> Stop All.	rsp: sa
cmd: pr <motor> <speed> Set the maximum step speed of the specified motor. The <speed> parameter is in steps/second.	rsp: pr <motor> <speed> If the requested speed is faster than the controller is allowing, then the controller will respond with the maximum speed.
cmd: jm <motor> <position> Jog the motor to the specified position.	rsp: jm <motor>
cmd: im <motor> <position> Inch the motor to the specified position. This is a slow crawl for fine-control of the position.	rsp: im <motor>

4. Wifi Communication

StackShot 3X supports 802.11 a/b/g/n and all the USB commands can also be sent over this transport.

4.1 Establishing a connection

Connecting to the StackShot 3X controller requires both UDP and TCP connections. UDP is used to broadcast a message on the current subnet to discover StackShot, and then TCP is the protocol used for subsequent communication.

The host first starts a TCP server on port 24870. This is for in-bound connections from the StackShot controller.

Secondly, a broadcast message is sent over all available networks via UDP port 24869. All available networks must be used since most devices have multiple networks available. The data for this broadcast message is "FINDSTACK". This process should be done periodically in case there is no controller available. Once a connection is established, these broadcast messages should stop. If the TCP server detects the connection is lost or if the StackShot stops responding to commands (turned off, out of range, etc), the UDP broadcasts should resume.

For the TCP server, the socket timeout should be overridden to something reasonable – 5 seconds for large commands on high-traffic networks would be more than suitable.

Example code written in NetBeans Java is available in the StackShot3xJava.zip file within the network.java class.
http://www.cognisys-inc.com/downloads/stackshot3x/StackShot3xJava.zip

4.2 API commands over WiFi

The commands are identical when sent over WiFi. The maximum size of a single packet is set to 1000 bytes. Any command exceeding this amount will have to be broken into smaller packet transmissions. Example code written in NetBeans Java is available in the StackShot3xJava.zip file in the StackComm.java class.
http://www.cognisys-inc.com/downloads/stackshot3x/StackShot3xJava.zip

A note of caution: Setting the TCP timeout is only useful for sending a command. If the controller is turned off the host's TCP server may not detect that the connection is lost. It may, depending on the host OS take several minutes to detect this lost connection. Therefore, it is recommended to periodically issue the "CC_PING" command to query the controller. Be sure to protect/synchronize the "ping" with other data commands.

Revision 1.1:
Added support for StackShot rev 1.0.7
Config->Mode: Rotary (7)
Config->Polarity: 0=Normal, 1=Reverse
Config->MOVSpeed: Move speed vs. Stack Speed
Config->STKSpeed: Stack Speed
Config->RotaryBacklash: Backlash for rotary tables
Config->RotaryRatio: Gear ratio of the rotary table
Config->RotaryDegrees: Current setting for the number of degrees to rotate the rotary table
Master/slave configuration (setting_io_t, setting_io_dir_t)
Time remaining for an active stack

Revision 1.2:
Added more examples for serial/USB communication
Added commands for StackShot 3X
Added section regarding Wifi