Command:BUILD (BU)

MS2000 syntax

Shortcut	BU
Format	BUILD [X] [Y] [Z]
Units	None

Tiger syntax

Shortcut	BU
Format	[Addr#]BUILD [X] [Y]
Units	None
Type	Card-Addressed (defaults to COMM)

Build Information

BU: (No Arguments) This command returns the firmware build name. An example of a build name is STD_XYZ.

BU X: This command returns various build configuration options, the firmware modules in the build, and the build name.

User String

The BU Y command is used to store a user defined string which can be saved using the SAVESET Z command. The maximum length of the user string is 20 characters. Remember to use the card address if you are storing a string on Tiger.

BU Y=#: writes a character to the user string at the write position. The write position starts at 0 on system power up and increase by 1 every time BU Y=# is sent to the controller. You specify the characters in their decimal ASCII form, for example lower-case 'a' is 97, so you would send BU Y=97. The valid range of ASCII values is 32-126 inclusive.

BU Y-: clears the user string and resets the write position to 0.

BU Y?: reads the user string.

BU Y is available on MS2000 v9.2m+ and Tiger v3.39+ firmware, you will also need Tiger Comm v3.39+ firmware.

To make things easier on the user, here is script that takes a Python string and enters the serial commands for you:

Python User String

asi_user_string.py

```
# /// script
# dependencies = ["pyserial>=3.5"]
# ///
import serial

def main() -> None:
    # the input string to send to the controller
    user_string = "abcdefghij1234567890"
    save_settings = True

    # use an empty string for MS2000 (card_address = "")
    card_address = "2"

    # error checking
    if len(user_string) > 20:
        raise Exception("Max stored string length is 20.")

    # open the serial port and send characters to the controller
    with serial.Serial("COM4", 115200, timeout=1) as serial_port:
        # clear the current stored string
        serial_port.write(bytes(f"{card_address}BU Y-\r",
encoding="ascii"))
        serial_port.readline()

        # send the input string to the controller
        for character in user_string:
            serial_port.write(bytes(f"{card_address}BU
Y={ord(character)}\r", encoding="ascii"))
            serial_port.readline()

        # print the stored string and check to see if it's the
same as the input string
        serial_port.write(bytes(f"{card_address}BU Y?\r",
encoding="ascii"))
        response = serial_port.readline().decode().rstrip("\r\n")
        if response == user_string:
            print(f"Successfully stored the input string =>
{response}")
            if save_settings:
                serial_port.write(bytes(f"{card_address}SAVESET
Z\r", encoding="ascii"))
                serial_port.readline()
                print("Settings saved to the controller using
SAVESET Z.")
        else:
            print(f"Error: expected {user_string} but got
{response} instead!")

if __name__ == "__main__":
    main()
```

Volatile Value

BU Z is used to edit and access an internal integer that is specifically volatile, i.e. the integer is always set to 0 when the controller is powered on or reset. The value ranges between 0 and 65535.

BU Z=<number> sets the value.

BU Z+ and **BU Z-** increment and decrement the value respectively, with wrap-around.

BU Z? queries the value.

Available in MS-2000 as of firmware v9.53.

Example:

```
bu z?
:A 0
BU Z-
:A
BU Z?
:A 65535
BU Z+
:A
BU Z+
:A
BU Z?
:A 1
BU Z=123
:A
BU Z+
:A
BU Z?
:A 124
```

Example Output

This is the response from a Tiger controller, the MS2000 response has less information but the meaning is the same between the two controllers. The MS2000 response does not include `Axis Addr`, `Hex Addr`, or `Axis Props`. Tiger also has an additional `POSITIONS SAVED` or `POSITIONS NOT SAVED` right before the firmware modules. The meaning of each line of the `BU X` command response is as follows:

STD_XY The build name of the firmware.
Motor Axes: X Y The name of each axis.
Axis Types: x x z The type of each axis (see table below).
Axis Addr: 2 2 The card address of each axis. **[*Tiger*]**
Hex Addr: 32 32 The hex address of each axis. **[*Tiger*]**
Axis Props: 10 10 The axis properties for each axis (see table below). **[*Tiger*]**
CMDS: XY The argument names for pseudo-axis commands.

BootLdr V:0 The version of the bootloader.
Hdwr REV.F The hardware revision of the PCB.
POSITIONS NOT SAVED Were the positions saved on power off? **[*Tiger*]**
RING BUFFER 50 The following entries are firmware modules.
SEARCH INDEX
ARRAY MODULE
IN0_INT
SRVLK_TTL
ZF_KNOB
CLUTCH XYKNOB FASTSLOW
SHUTDOWN_TASK
MOVETASK

Tiger Example Response

```
1BU
STD_XY
```

```
1BU X
STD_XY
Motor Axes: X Y
Axis Types: x x
Axis Addr: 2 2
Hex Addr: 32 32
Axis Props:  10  10
CMDS: XY
BootLdr V:0
Hdwr REV.F
POSITIONS NOT SAVED
RING BUFFER  50
SEARCH INDEX
ARRAY MODULE
IN0_INT
SRVLK_TTL
ZF_KNOB
CLUTCH XYKNOB FASTSLOW
SHUTDOWN_TASK
MOVETASK
```

With an address of 1 given, the specified card #1 replies. This reply just contains axis name and types present on the card. However it goes into more detail, printing all the firmware modules present on the card.

The values listed for axis properties are decimal integer representations of a binary code which represents any special properties of the axis. Usually these could also be identified by doing a `BU X` query of each card and interpreting the response, but they are listed separately on the axis property line for convenience.

MS2000 Example Response

```
BU
STD_XYZ
```

```
BU X
STD_XYZ
Motor Axes: X Y Z
Axis Types: x x z
CMDS: XYZFRTM
BootLdr V:1
Hdwr REV.E
LL COMMANDS
RING BUFFER 50
SEARCH INDEX
IN0_INT
DAC OUT
FS_LED
SHUTDOWN_TASK
```

No card address needed on `MS2000`.

Tiger Comm Card Response

Adding the card address is optional. If no address is given, then Tiger Comm replies. If an address is present, then the specified card replies.

```
BU
TIGER_COMM
```

```
BU X
TIGER_COMM
Motor Axes: X Y A B C C 0 1
Axis Types: x x u u u u w w
Axis Addr: 1 1 2 2 2 2 3 3
Hex Addr: 31 31 32 32 32 32 33 33
Axis Props:   0   0   0   0   0   0   0   0
```

As no address was given, Tiger Comm replies. It replies with its build name, all axis names present in the system (axes will always be A-Z, filterwheels 0-9). For each axis the type is given (see table in section Identifying Controller Configuration) and the card address in both character and hex formats. Finally, an integer is given to indicate the presence of special properties or capabilities of the axis, such as `CRISP` auto-focus or `RING BUFFER` module for TTL positioning; these can be interpreted using the axis properties table below. This command is useful to quickly identify all the axes names and types present in the system.

Tables

Axis Properties

Bit 0:	CRISP auto-focus firmware
Bit 1:	RING BUFFER firmware
Bit 2:	SCAN firmware
Bit 3:	ARRAY firmware or MM_TARGET firmware
Bit 4:	SPIM firmware (v2.81+)
Bit 5:	SINGLEAXIS and/or MULTIAXIS firmware (v2.81+)
Bit 6:	LED illumination (v2.87+)
Bit 7:	Reserved

2016/02/24 15:48 · asiadmin

Axis Type List

Axis Type Short	Axis Type Long	Description
x	XYMotor	XY stage
z	ZMotor	Z focus motor drive. LS50s, Z scopes etc
p	Piezo	Piezo Focus. ASIs ADEPT, Piezo DAC etc
o	Tur	Objective Turret
f	Slider	Filter Changer
t	Theta	Theta Stage
l	Motor	Generic linear motorized stage, TIRF, SISKIYOU etc
a	PiezoL	Generic linear piezo stage
m	Zoom	Zoom magnification motor axis
u	MMirror	Micro Mirror, Scanner 75 etc
w	FW Filter	Wheel
s	Shutter	Shutter
g	Logic	Programmable logic card
i	LED card	Multi LED Driver card
b	Lens	Tunable Lens
d	DAC	Digital to Analog converter(DAC)
u	Unknown	Unknown axis type

2016/02/24 15:48 · asiadmin

commands, tiger, ms2000